Chapter 17
IoT Application for Smart Cities Data Storage and Processing Based on Triangulation Method

Muzafer Saračević, Šemsudin Plojović, and Senad Bušatlić

Abstract This chapter presents the possibilities of applying the triangulation method in the IoT application for smart cities data storage and processing. The chapter includes a method in field of computational geometry (specifically polygon triangulation as a fundamental algorithm in computational geometry). The implementation of the application was done with two modules. Proposed first module has a storage function and collecting data in a smart city space, while the second module has a function of data processing and finding the minimum triangulation in the smart city space. The method is derived using triangulation and properties of Catalan numbers. The process of storing the coordinates of points using the Ballot method (walk on the integer lattice) is given. Due to its exceptional efficiency in terms of launching programs on various computer architectures and operating systems, Java programming language (Net Beans environment) enables an efficient implementation of our method. In experimental research, we tested a proposed solution based on the application of the triangulation method using the Ballot notation (for data storage) and minimal triangulation (for data processing).

Keywords Smart cities · IoT · Optimal triangulation · Data storage and processing · Java programming · GIS

M. Saračević (✉) · Š. Plojović
Department of Computer Sciences, University of Novi Pazar, Novi Pazar, Serbia
e-mail: muzafers@uninp.edu.rs; s.plojovic@uninp.edu.rs

S. Bušatlić
International University of Sarajevo, Sarajevo, Bosnia and Herzegovina
e-mail: sbusatlic@ius.edu.ba

M. Alam et al. (eds.), *Internet of Things (IoT)*, S.M.A.R.T. Environments,
https://doi.org/10.1007/978-3-030-37468-6_17

17.1 Introduction

Navigating within smart cities is directly linked to sustainable, safe and efficient transport systems and infrastructure, such as their rejection at both local and national and international levels. So for a city to become smart, it is imperative that it innovates in transportation that will sustain it. These steps are used not only for the economy but also for the environment and provide a better quality of life for all citizens. Previous research and implementation of e-City services in the field of map usage and mapping uses GIS (*Geo-information Systems*) commercial solutions. Also, you can use Open Source GIS solutions as well as free Web Mapping Services, which in the last year have achieved the usable value of GIS commercial solutions, are overwhelming in a large segment. The possibility of combining GIS resources and free Web map services was explored, as well as presenting numerous information with external services within the projected maps, which enables a very rich presentation of the requested information. Research combining e-City services with the concept of creating "smart cities" (*Smart City*) is very current today. Superior analyzes and comparisons between GIS systems and free public ticketing services, towards finding the best solution for creating a centralized electronic service at the city level.

A modern technique for collecting data in the field is a system of terrestrial orbiting satellites that transmit precise weather signals to special electronic devices to record the position of an object on the ground. This system is referred to as "*The Global Positioning System (GPS)*". The receivers provide direct measurements of the position on the Earth's surface, with the location indicated by a standard system, using coordination using a triangle system. Triangulation is a process that is very important in computer geometry and graphics. Triangulation allows the representation of three-dimensional objects from a set of points. This process is very important for the speed, quality and resolution of the objects displayed. Polygon triangulation finds its application in geo-information systems as well as in the process of digital terrain modeling.

IoT-enabled smart city use cases span multiple areas: from contributing to a healthier environment and improving traffic to enhancing public safety and optimizing street lighting. Smart cities are servicing new digital technologies and are related to traffic management through smart traffic lights, parking space design, gathering information on their waste storage, rational use of refreshments, to build energy-efficient communications in the future with sufficient local government. All of these services involve reducing pollution, moving to cleanup transportation clean up and heating. The use of IoT (*Internet of Things*) traffic regulation is used to reduce costs and increase passenger satisfaction while reducing the number of traffic accidents. Future solutions will be based on main memory and environmentally sound vehicles and their connection to infrastructure facilities such as gas stations, parking lots, garages and the like. Wider use of advanced information technology, except vehicle communications with vehicle infrastructure, capability and communication.

IoT applications fot smart cities is just another solution that the modern age has to deal with the problems that are in line with achievements and to provide citizens with a healthy and safe lifestyle as it may have once been, or might even get better. Some examples of implementation of intelligent transport systems is: integrated traffic systems (traffic flow management, traffic lights, variable traffic messages, highway access control, speed checking, parking management, etc.), transportation management (traffic routing, incident management, identification of breaches, maintenance of transport infrastructure) and passenger information (providing information). Other appropriate sustainable transport practices that characterize smart cities may be linked to the introduction of an initiative such as car sharing, that is, the sharing of private passenger vehicles with the same final destination (using GIS and triangle methods).

The main motivation of this chapter is presents presents the possibilities of applying the triangulation method in the IoT application for smart cities. The chapter includes a method in field of computational geometry (specifically optimal polygon triangulation as a fundamental algorithm in computational geometry). The main contribution of this chapter is concrete solutions for storing and processing smart city data based on the optimal triangulation method.

This chapter consists of seven sections. In Sect. 17.2, similar research is presented in the field of the GIS technology in the realization of the concept of smart cities. In addition, similar research has been reported in the field of the main examples of triangles in the application of geo-information systems. In Sect. 17.3 of this chapter, some of the features of the Java NetBeans environment for working with data processing and storage are outlined. Section 17.4 describes the proposed method (first module) for storing and collecting data in a smart city space. In Sect. 17.5 has been presented the second module of the proposed method relating to data processing and finding minimum triangles in smart city space. Give specific examples of how to apply the proposed methods in a *Java NetBeans environment*. Sections 17.6 and 17.7 provides a discussion and concluding consideration.

17.2 Overview of Related Research

GIS offers user-friendly and advanced capabilities for smart city applications. The successful implementation of a smart city application requires the development of a system that can manage and visualize the geospatial data in a user-friendly environment. In paper (Yamamura et al. 2017) authors present a new method based on GIS for smart city planning. The research proposes a GIS based urban energy planning system and 3D visualization with a user-friendly interface. Authors in paper (Lella et al. 2017) discuss possible collection methods for waste management and presents methods for transportation of waste using GIS techniques through analysis of network.

Authors in paper (Cosido et al. 2013) construct a model for Smart City by using GIS techniques. Experimental results show that approach performs very well and that the presented methodology is promising for further application in other concrete scenarios. Paper (Tan and Wong 2006) presents applications for smart cities with concrete Google maps and GIS tools. The release of mapping API like Google maps has changed the way location systems work. Also, authors apply visualization tools as a study into the usefulness of functions of Google maps and GIS.

A multi-criteria GIS-based methodology for smart cities site selection presents in paper (Fashal et al. 2019). This research contributes to a site selection method that satisfies the decision maker's criteria. Layers corresponding to these criteria were built in GIS. In paper (Shahrour 2018) authors give concrete concept based on the use of geospatial data concerning the urban built environment and urban services. Authors show how a GIS could help in concrete implementation of smart city and describes its use in the construction of a model of the smart city. Also, authors in paper (Chen and He 2017) state that GIS plays a very important and fundamental role in supporting the smart city construction. This paper expounds the concrete model of the smart city construction with GIS technique.

The paper (Pradhan et al. 2017) proposes a new computational code for GIS using triangulated irregular network and Delaunay triangulation methods. The quality of surface (GIS) representation after using proposed model is compared with the original map, and results show that this model can be used for significant reduction of data set.

Authors in paper (Ledoux et al. 2014) present a novel approach to automatically repair GIS polygons, based on the use of a constrained triangulation. Their approach is simple to implement as it is mostly based on triangles and authors have implemented algorithms and show that this model is faster and more efficient than alternative methods. The paper (Roy and Mandal 2012) proposed a time-efficient graph-based spatial clustering for large scale GIS data. As the volume of GIS data is large, data is preprocessed using Delaunay Triangulation to reduce both: space and time complexities.

Implementing security in an IoT solution involves multi-layered access and observation of application security from multiple aspects: network, server, code, database, user, etc. For some concrete and specific examples of attacks on smart cities applications see paper (Saračević et al. 2019). The device must be safe at all times, from design to use in an operating environment. This includes the following measures: Safe Startup, Access Control, Device Authentication, Intrusion Prevention Systems, etc.

The security of an IoT solution should not be seen as additional functionality, but rather as a necessary component for the reliable functioning of the device. IoT techniques are connecting more devices in the Cloud Computing environment. In papers (Alam et al. 2013c; Alam and Shakil 2013a, b, 2016; Ali et al. 2019; Ali and Alam 2016) the authors discuss about management techniques for the Cloud Computing environment. In papers (Alam et al. 2013a, b; Alam 2012a, b; Alam and Alam 2013; Alam et al. 2014; Alam and Sethi 2013; Malhotra et al. 2015, 2018), the authors emphasize importance of cloud database management system and data integration of cloud-based and relational databases. Also, authors in papers (Shakil and Alam 2014; Shakil et al. 2018; Shakil and Mansaf 2017) describe some techniques for exploiting data reduction principles in cloud-based data management and data management in cloud based environment. Additionally, papers (Imran et al. 2015; Khan et al. 2016, 2017, 2019; Kumar et al. 2017; Kumari et al. 2015; Samiya et al. 2017) describe cloud based big data analytics, data model and computing.

17.3 Data Processing and Storing in Java NetBeans Environments

In this part of the chapter, we have outlined the key packages and classes in Java that we used in the implementation process of the triangulation method for processing and storing data in a smart city space. The two essential packages we used to implement Java 3D solution are:

com.sun.j3d.utils.geometry (*GeometryInfo, Triangulator*),
org.j3d.geom (*TriangulationUtilis*).

The *Java Open Geometry Library (JOGL)* is actually a library for working with geometric shapes and contains packages and classes that are essential for programming in computer geometry. *JOpenGL* is a java link for the *OpenGL 3D Graphics API*. It provides full access to the OpenGL 2.0 specification and supports integration with Swing components. *JOpenGL* maintains its focus on 3D rendering with assistive libraries that make it easier to create geometric primitives.

JOpenGL to some extent offers an object layer that invokes the functions of this library. The implementation of the Java 3D version consists of three packages, with *OpenGL, DirectX* and *JOpenGL* backend (Davis 2004; Sowizral and Deering 1999). Access to the OpenGL interface was achieved through direct calls through the *Java Native Interface (JNI)*. Some important classes for the problem of polygon triangulation from the Java 3D API are described below (Chen and Chen 2008).

The *GeometryInfo* class contains ready-made methods for working with string arrays for geometric objects:

TRIANGLE_ARRAY (takes a set of three vertices forming a triangle), using the GL_TRIANGLES method to create triangles,
POLYGON_ARRAY (a set of vertices for non-convex and convex polygons) creates a polygon using the GL_POLYGON method,
QUAD_ARRAY (takes a set of four vertices forming a quadrilateral) using the GL_QUADS method to create quadrilaterals,
TRIANGLE_FAN_ARRAY (set of scopes that forms a triangulation in the shape of a fan), using the GL_TRIANGLE_FAN method, creates a triangulation of a series of triangles around a common central spine,

TRIANGLE_STRIP_ARRAY (a set of vertices forming a ribbon-shaped triangulation), using the GL_TRIANGLE_STRIP method, creates a triangulation of a series of bound triangles.

To store the data, we used the JAVA API for working with databases (*JDBC – Java Data Base Connection*). This API is based on SQL language, that is, JDBC calls SQL interface for Java: *java.sql.Connection* and *javax.sql.DataSource*.

The JDBC API contains a number of abstract Java interfaces that allow you to connect to a specific database, execute a SQL command, and process the results.

Setting up a database of *JavaDB-Sun's Apache Derby* distribution was used to implement the method. In this way, it is possible to use any other database that provides the JDBC driver. The NetBeans environment provides an easy interface for creating databases and establishing connections (Heffelfinger 2011).

The corresponding Java DB (JDBC) driver is available under NetBeans. Figure 17.1 shows how to create a new connection (Fig.17.1, left) and start a Java DB server (Fig. 17.1, right).

After connecting to the server, the database is created for the records of triangulations in the smart city space (Fig. 17.2, left) and the creation of connections to the created database (Fig. 17.2, right).

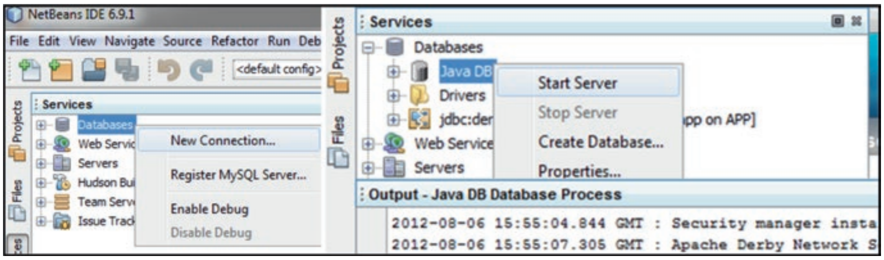
The created database has tables of the form T_n where n is the number of selected points in the smart city space (polygon). Before generating triangulations, it is necessary to check that a JDBC form connection is established:

JDBC: <subprotocol>: <subname>, where *<subprotocol>* is a type of database access mechanism supported by one or more drivers (Fig. 17.3).

Figure 17.4 (left part) shows the *BlockTriangulation* database tables. Each table contains columns which is equivalent to the number of points of given polygon in a smart city space (Fig. 17.4, right).

The implementation of the Java Test application was done in a NetBeans environment. Below, we describe two modules of the proposed application. The fourth part of the chapter describes the proposed first module for storing and collecting data in a smart city space.

In the fifth part of the chapter, the second module is given, which deals with the process of data processing and finding the minimum triangulation in the smart city space.



```
2012-08-06 15:55:04.844 GMT : Security manager insta
2012-08-06 15:55:07.305 GMT : Apache Derby Network S
```

Fig. 17.1 Creating a new connection and starting the server

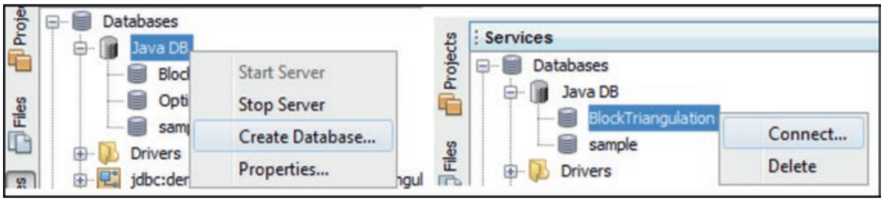


Fig. 17.2 Creating a database for triangulation blocks and connection to the created database

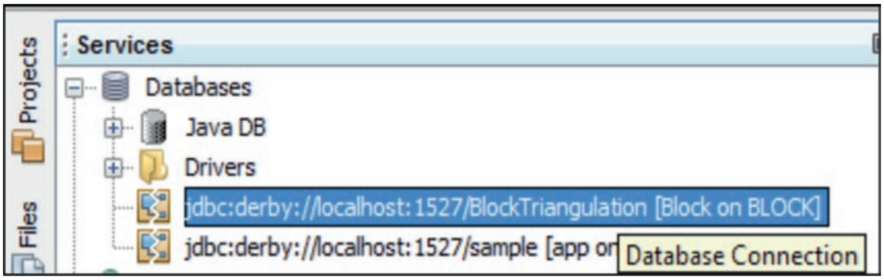

Projects
Files
Services
Databases
Java DB
Drivers
jdbc:derby://localhost:1527/BlockTriangulation [Block on BLOCK]
jdbc:derby://localhost:1527/sample [app on
Database Connection

Fig. 17.3 Database access mechanism supported by the appropriate Java driver

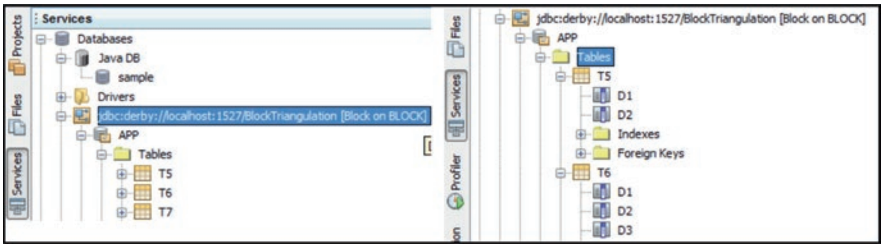

Services
Databases
Java DB
sample
Drivers
jdbc:derby://localhost:1527/BlockTriangulation [Block on BLOCK]
APP
Tables
T5
T6
T7
jdbc:derby://localhost:1527/BlockTriangulation [Block on BLOCK]
APP
Tables
T5
D1
D2
Indexes
Foreign Keys
T6
D1
D2
D3

Fig. 17.4 The generated database tables BlockTriangulation and details

17.4 Triangulation Method for Data Storage in Smart City Space

This section describes the technique for storing a particular record in the Smart city space. We will apply the Ballot technique of recording movement through triangulation, which we have described in detail in previous work (Saračević et al. 2014; Saračević and Selimi 2019).

Notation of the Ballot problem can be represented graphically in an integer network consisting of a certain number of points in the Cartesian coordinate system. The problem relates to the number of calculations of paths through an integer network. The paths consist of n steps with some starting point and ending point m. As shown in Fig. 17.5 (left), we can encode each path in an integer network in a specific

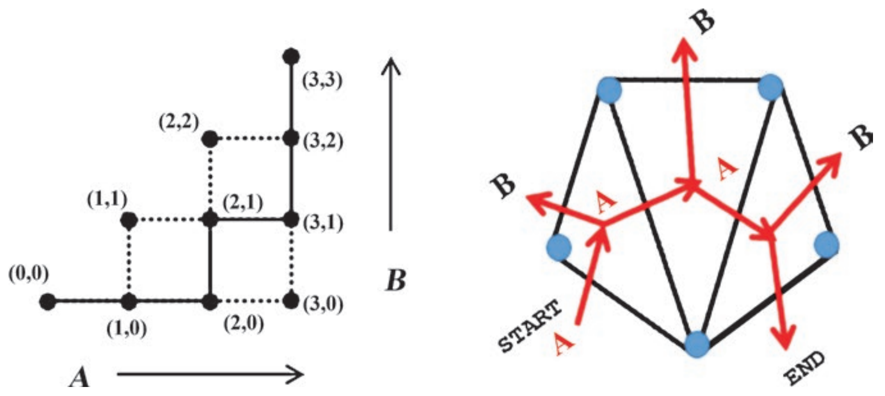


Fig. 17.5 Movement in the integer network (left) and Ballot notation of ABABAB triangulation (right)

order of the right-shift vector (1, 0) and the up-shift vector (0, 1). Combined with the Ballot problem, the choice of upward displacement positions (*mark B*) uniquely determines the path in the integer network because the remaining positions represent rightward shifts (*mark A*).

The number of possible movements through triangulation, analogous to the problem of the number of roads in an integer network, is equal to the Catalan number (see previous papers (Saračević et al. 2014; Saračević and Selimi 2019; Stanimirović et al. 2014)). The presented process of moving over the entire network in combination with the Ballot record served as a good idea for forming a system of recording (notation) of movement through triangulation in the smart city space (Fig. 17.5, right).

In Java, the Graphics2D class with the *Point.Cartesian* class makes it possible to work with the Cartesian coordinate system, where on the basis of roads in the entire network it is possible to realize movement through the triangulation of polygons based on Ballot records.

The storage of triangulations, following the model from previous paper (Stanimirović et al. 2012), can be performed by setting the diagonals of triangulation to be marked with *A* and the outer sides with *B*. The movement is clockwise. This direction of motion, as well as the start and end points of motion (*START / END*), must be the same for all triangulations because of the uniqueness of the record. The condition to complete the movement is to visit all pages (see Fig. 17.5, right).

The implementation of this movement through triangulation was done with following Java classes: *Triangulations, BallotPath, Graphics2D, Nodes, LNodes, ListBallot*.

The main class *BallotPath* contains a method *notation()* that realizes movement records through triangulation. This movement record is backed by *Node* and *LeafNode* classes, which are standard and used to work with polygon page tags and page markers.

Class *Triangulations* is responsible for generating triangulations in the smart city space based on the record produced by the *BallotPath* class. Navigating through Ballot-based triangulation in the BallotPath class was realized based on the described concept of Cartesian coordinates and the problems of paths in an integer network. Java 2D allows points on a flat surface to be accessed using the horizontal and vertical axes.

Class *Graphics2D* provides more sophisticated control over geometry, coordinate transformation, color management, and text layout. Generic points are immutable and can always report their Cartesian coordinates: *Point.OnSegment* and *Point.Cartesian*. The coordinates that deviate from the axis are called Cartesian coordinates, which are also used in Java 2D objects (Fig. 17.6). Java 2D defines coordinates in units and rendering occurs in a hypothetical plane called user space.

Figure 17.7 shows the contents of an output file containing Ballot records for various triangulations in smart city space.

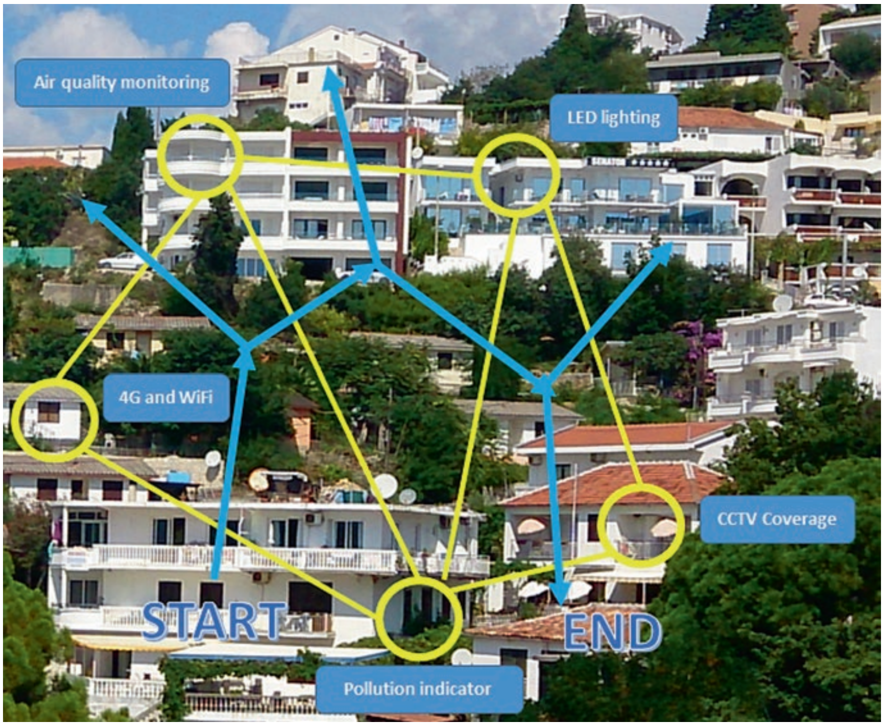
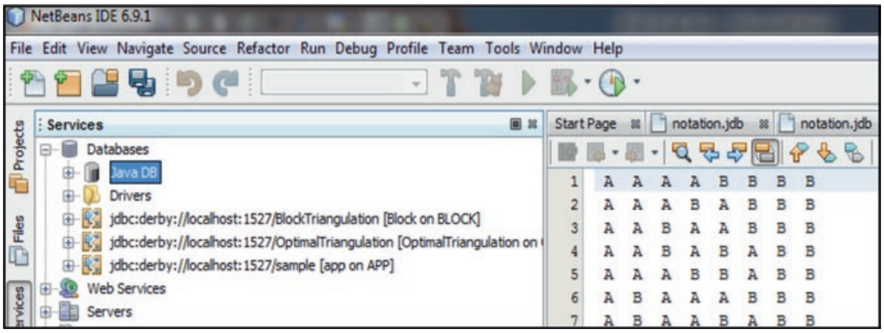


Fig. 17.6 Formation of triangulation in Smart city space


NetBeans IDE 6.9.1
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Projects
Files
Services
Start Page
notation.jdb
notation.jdb

```
Databases
  Java DB
  Drivers
  jdbc:derby://localhost:1527/BlockTriangulation [Block on BLOCK]
  jdbc:derby://localhost:1527/OptimalTriangulation [OptimalTriangulation on
  jdbc:derby://localhost:1527/sample [app on APP]
Web Services
Servers
```

```
1  A A A A B B B B
2  A A A B A B B B
3  A A B A A B B B
4  A A B A B A B B
5  A A A B B A B B
6  A B A A A B B B
7  A B A A B A B B
```

Fig. 17.7 The contents of the output file as a result of testing a Java application

17.5 Triangulation Method for Data Processing in Smart City Space

In this part of the chapter, a second module is presented that deals with data processing and finding the optimal (minimum) triangulation in a smart city space. Papers (Aybeyan et al. 2019; Saračević et al. 2018) deals with the problem of finding the minimum weight triangulation. The default role in storing weights for triangulations is the *Java* package *DefaultTableCellRenderer*, which allows multiple values to be stored in one field of a table (offering efficient division of fields into multiple columns and rows).

Let je V_0, V_1, …, V_{n-1} be the selected polygon in the smart city space. The given polygon is divided into (n-2) triangles with (n-3) diagonals that do not intersect. Generally, we can associate triangles with weight (i.e. measure: the length of the longest center of gravity line). For the chosen measure, triangulation can be determined such that the sum of these measures of triangles, into which the polygon is divided, is minimal. Each side of a polygon *(i, j)* in a given triangulation of a convex polygon, with k vertices, can form triangles, and for them the weights are calculated:

$$m[i,j] = m[i,k] + m[k,j] + w\left(\Delta v_i v_j v_k\right) \tag{17.1}$$

The method of storage weight triangulations is based on the formation of a square matrix of dimensions *(i = j),* where is also the number of rows, j is the number of columns. The matrix M is used to organize the vertices *(k)* of triangles *(i, j, k)* but also to represent their weights *(w)*. The matrix-based storage method *M* is used in the calculation. Matrix *M* can be permanently switched to table form using the *JDBC API*. The matrix is divided diagonally into two parts (the symmetry divides the matrix into the left and right parts), where the positions *(i, j)*, for which *i = j* apply, are filled by zeros. On the one side of that symmetry (left part of the matrix), the *k*-vertices of the triangles are recorded and on the other side of the symmetry (right part), the corresponding weights are recorded: k [i, j] = w [j, i]. First, the left part of the matrix that defines triangles of (i, j, k) is filled. For (i, j) corresponding to the row and column of the matrix, the corresponding allowed topics k are added. All allowances are those between (i, j).

The method for data processing in smart city space consists of 4 phases:

- In the first step, a square matrix is formed and fields are filled diagonally with values *0* (where $i = j$). Then the first diagonal line of the field of symmetry is filled. These are adjacent vertices and for them there are no *k* values with which a (*i, j, k*) triangle can be formed. After that, all values of *k* on (*i, j*) are added.
- The second step is to calculate all the weights (*w*) for the rows and columns of the matrix for which, by *Eq. (17.1)*. The additional values of *k* at position (*i, j*) add the corresponding weights to (*j, i*).
- In the third step, the corresponding triangulation of the polygons is joined to the obtained values in the matrix. From the obtained values, (*n-2*) triangles are determined which determine the triangulation.
- The fourth step is to show the optimal triangulation in the smart city space (see Fig. 17.8).

Example 1
In the specific case, it is shown how the scenario described works. The four steps explain how optimal triangulation occurs in a smart city space.

Step 1: If the number of selected points in the smart city space is equal to 5, then a matrix of dimensions 5x5 is formed. Fill in the fields diagonally in the zero matrix, as well as the values for adjacent vertices.

Step 2: After that, fill in for all *(i, j, k)* triangles. In this case, there are a total of 10. There are as many calculated weights (*w)* and each triangle at the (*i, j*) position corresponds to the calculated weight at *(j, i)* position. The resulting matrix after 1 and 2 steps looks like this:

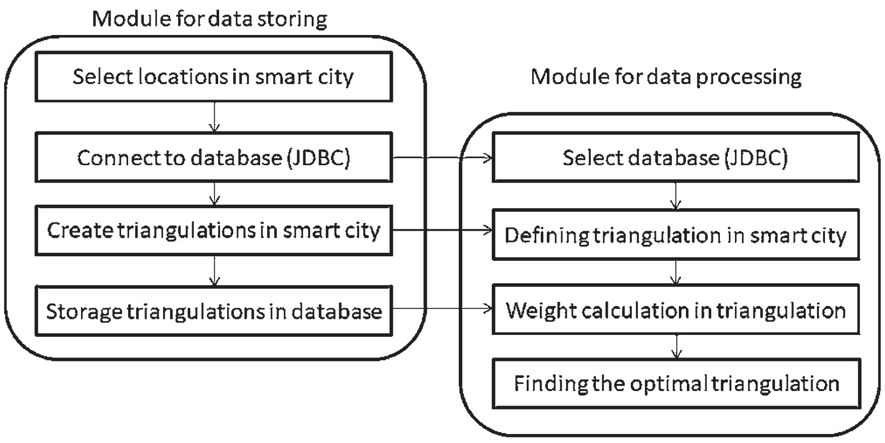

Module for data storing
Select locations in smart city
Connect to database (JDBC)
Create triangulations in smart city
Storage triangulations in database
Module for data processing
Select database (JDBC)
Defining triangulation in smart city
Weight calculation in triangulation
Finding the optimal triangulation

Fig. 17.8 Flowchart of proposed approach (for two modules)

$$opt5=\begin{bmatrix} 0 & \{0\} & \{35\} & \{42,40\} & \{31,43,\langle 25\rangle\} \\ \{0\} & 0 & \{0\} & \{35\} & \{38,41\} \\ \{2\} & \{0\} & 0 & \{0\} & \{36\} \\ \{2,3\} & \{3\} & \{0\} & 0 & \{0\} \\ \{2,3,\langle 4\rangle\} & \{3,4\} & \{4\} & \{0\} & 0 \end{bmatrix}$$

Step 3: Summarize the weights *(w)*. A new column *W* is added to the table containing the sum of all triangulation weights in the current row. The triangulation storage table is expanded with a new column W (see Table 17.1):

Step 4: Find the lowest weight or minimum (optimal) triangulation:

$$OptT_w = \min\left\{sumAllW\left[T_1\right],\ldots,sumAllW\left[T_i\right]\right\}$$

In Table 17.1, the third row is the minimum weight triangulation *OptTw = 100.* It is determined by the diagonals *δ (2,4)* and *δ (1,4),* see Fig. 17.9.

In the Java NetBeans environment, the *OptimalSmartTriangulation* application has been implemented that implements the above steps. The main class has the *compute ()* method, which is in charge of calculating all the weights in *table T*. To work with the cells of the table, the *DefaultTableCellRenderer* class was used. In addition to the class specified, *JTable, TableCellRenderer, BasicTableUI* were used. An *OptimalTriangulation* database has been formed in a *Java* database service that contains tables with persistent triangulation weights.

The application works according to the scenario described:

1. The JDBC connection is first checked, followed by the weight calculation, by clicking on the "Weight Calculation" button (Fig.17.10)

After the corresponding table T_n is created, a new weight column *(w)* will be filled in order for each triangulation (i.e. for each row in the table).

Filling in the new column refers to recognizing *(i, j, k)* values from the matrix based on the diagonal *δ (i, j)* in table T_n (Fig.17.11).

After calculating the weights for all triangulations, the user will be shown the standard output *„status ok*"and the lowest weight triangulation will be plotted on *JPanel*. Figure 17.12 shows an application for finding the optimum (minimum) triangulation.

Table 17.1 Weight table

Calculation (i)	D1 (first)	D2 (second)	Weight
1	δ (1,3)	δ (1,4)	124
2	δ (1,3)	δ (3,5)	136
3	δ (2,4)	δ (1,4)	100
4	δ (2,4)	δ (2,5)	109
5	δ (2,5)	δ (3,5)	112

$$
\begin{array}{c|ccccc}
 & 1 & 2 & & & \\
\hline
 & 0 & \{0\} & \{35\} & \{\boxed{40},42\} & \{31,43,\boxed{25}\} \\
 & \{0\} & 0 & \{0\} & \{\boxed{35}\} & \{38,41\} \\
 & \{2\} & \{0\} & 0 & \{0\} & \{36\} \\
4 & \{\boxed{2},3\} & \{\boxed{3}\} & \{0\} & 0 & \{0\} \\
5 & \{2,3,\boxed{4}\} & \{3,4\} & \{4\} & \{0\} & 0
\end{array}
\rightarrow
$$

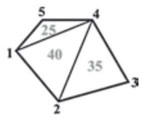
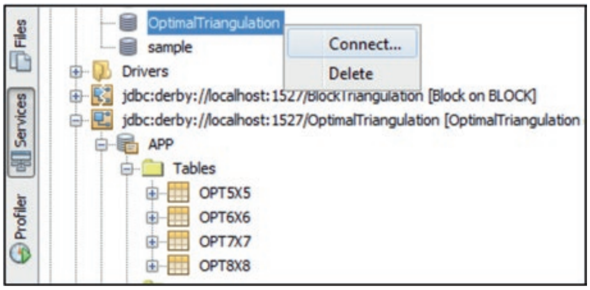
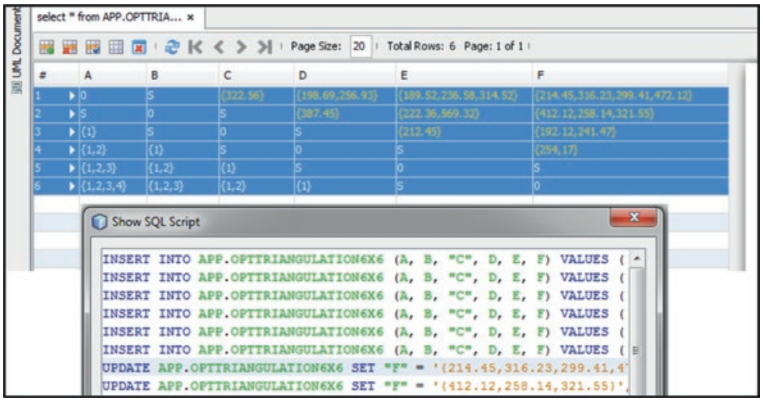


Fig. 17.9 Appropriate values in the matrix and graphical representation of optimal triangulation



OptimalTriangulation
sample
Drivers
jdbc:derby://localhost:1527/BlockTriangulation [Block on BLOCK]
jdbc:derby://localhost:1527/OptimalTriangulation [OptimalTriangulation
APP
Tables
OPT5X5
OPT6X6
OPT7X7
OPT8X8

Fig. 17.10 Java service for connecting to the OptimalTriangulation



select * from APP.OPTTRIA...

Page Size: 20 Total Rows: 6 Page: 1 of 1

#	A	B	C	D	E	F
1	0	S	{322.56}	{198.69,256.93}	{189.52,236.58,314.52}	{214.45,316.23,299.41,472.12}
2	S	0	S	{387.45}	{222.36,569.32}	{412.12,258.14,321.55}
3	{1}	S	0	S	{212.45}	{192.12,241.47}
4	{1,2}	{1}	S	0	S	{254,17}
5	{1,2,3}	{1,2}	{1}	S	0	S
6	{1,2,3,4}	{1,2,3}	{1,2}	{1}	S	0

```
INSERT INTO APP.OPTTRIANGULATION6X6 (A, B, "C", D, E, F) VALUES (
INSERT INTO APP.OPTTRIANGULATION6X6 (A, B, "C", D, E, F) VALUES (
INSERT INTO APP.OPTTRIANGULATION6X6 (A, B, "C", D, E, F) VALUES (
INSERT INTO APP.OPTTRIANGULATION6X6 (A, B, "C", D, E, F) VALUES (
INSERT INTO APP.OPTTRIANGULATION6X6 (A, B, "C", D, E, F) VALUES (
INSERT INTO APP.OPTTRIANGULATION6X6 (A, B, "C", D, E, F) VALUES (
UPDATE APP.OPTTRIANGULATION6X6 SET "F" = '{214.45,316.23,299.41,4
UPDATE APP.OPTTRIANGULATION6X6 SET "F" = '{412.12,258.14,321.55}',
```

Fig. 17.11 Joining weights in triangulations in JTable

Figure 17.13 shows a standard output with generated parameters at each step of the proposed method.

17.6 Discussion

Utilizing technology and modern scientific advances, civilization is moving to a higher level and thus generating "smart" ideas to keep up with the ongoing challenges, risks and problems it faces. Smart cities are a concept of the future that will

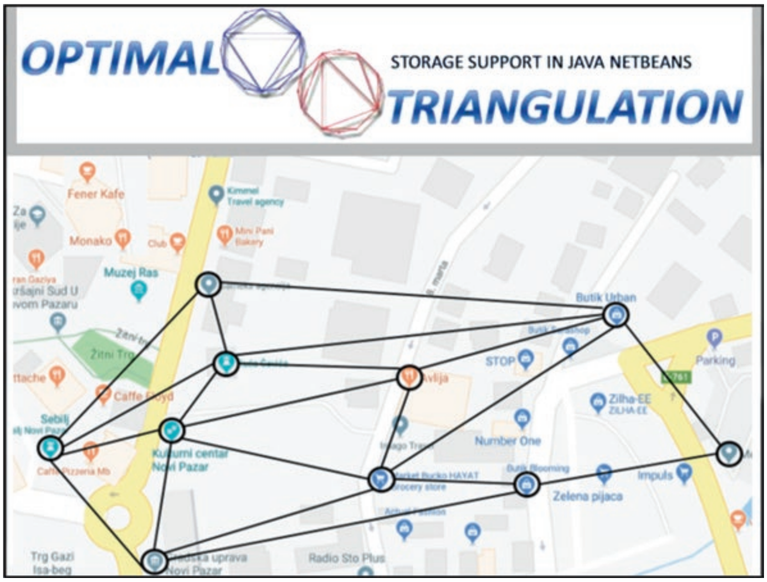


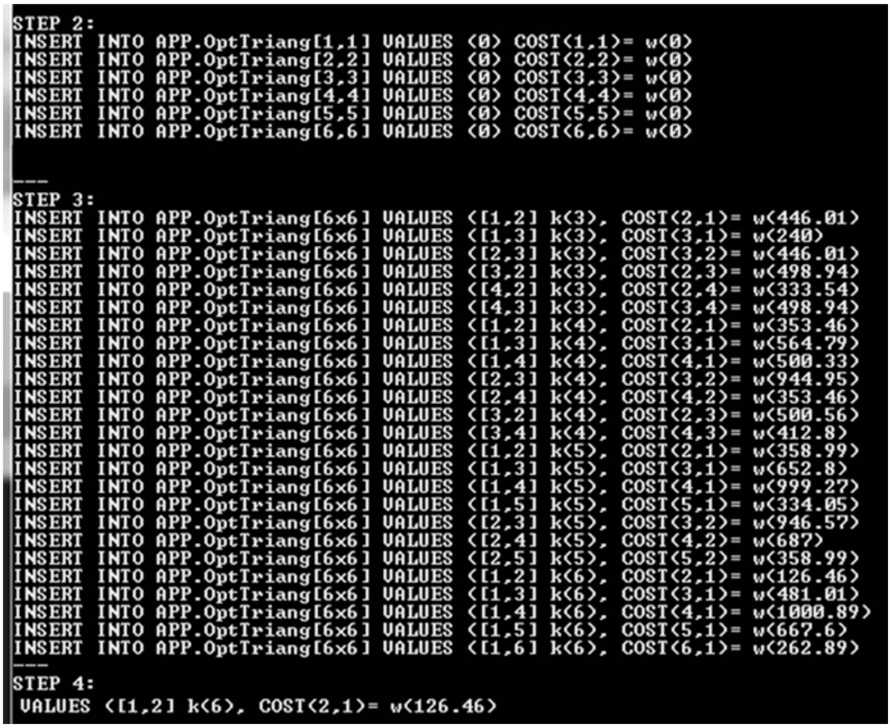
Fig. 17.12. Java application for finding optimal triangulations

need more and more attention and that becomes inevitable in the coming times. The aim of this concept is to ensure environmental, energy and social sustainability through the use of technology-information and other smart systems, with the city covered with various sensors, software and devices to monitor developments and changes in it. The city would not change its appearance, only its organization and way of functioning.

A smart city is just another solution that the modern age has to deal with the problems that are in line with achievements and to provide citizens with a healthy and safe lifestyle as it may have once been, or might even get better.

However, there are challenges and obstacles in creating the concept of smart cities in most cities, one of which is the complexity of existing infrastructure and the need for a large amount of human and material resources, and a particular challenge is the issue of security. When integrating different intelligent devices, it is common for protocols to be incompatible, which generally differ in the amount of data they exchange and the speed of their processing.

New applications and online services in IoT place high demands on scalability, reliability and flexibility. Traditional computer networks with a hierarchical architecture model are finding it increasingly difficult to fill them. The solution is software-defined networks that bring a high degree of programmability. As further directions for research in this field, we can say that it is useful to examine the possibility of applying triangulation methods in the field of software defined networks and openflow protocols.



```
STEP 2:
INSERT  INTO APP.OptTriang[1,1] VALUES (0) COST(1,1)= w(0)
INSERT  INTO APP.OptTriang[2,2] VALUES (0) COST(2,2)= w(0)
INSERT  INTO APP.OptTriang[3,3] VALUES (0) COST(3,3)= w(0)
INSERT  INTO APP.OptTriang[4,4] VALUES (0) COST(4,4)= w(0)
INSERT  INTO APP.OptTriang[5,5] VALUES (0) COST(5,5)= w(0)
INSERT  INTO APP.OptTriang[6,6] VALUES (0) COST(6,6)= w(0)


---
STEP 3:
INSERT  INTO APP.OptTriang[6x6] VALUES ([1,2] k(3), COST(2,1)= w(446.01)
INSERT  INTO APP.OptTriang[6x6] VALUES ([1,3] k(3), COST(3,1)= w(240)
INSERT  INTO APP.OptTriang[6x6] VALUES ([2,3] k(3), COST(3,2)= w(446.01)
INSERT  INTO APP.OptTriang[6x6] VALUES ([3,2] k(3), COST(2,3)= w(498.94)
INSERT  INTO APP.OptTriang[6x6] VALUES ([4,2] k(3), COST(2,4)= w(333.54)
INSERT  INTO APP.OptTriang[6x6] VALUES ([4,3] k(3), COST(3,4)= w(498.94)
INSERT  INTO APP.OptTriang[6x6] VALUES ([1,2] k(4), COST(2,1)= w(353.46)
INSERT  INTO APP.OptTriang[6x6] VALUES ([1,3] k(4), COST(3,1)= w(564.79)
INSERT  INTO APP.OptTriang[6x6] VALUES ([1,4] k(4), COST(4,1)= w(500.33)
INSERT  INTO APP.OptTriang[6x6] VALUES ([2,3] k(4), COST(3,2)= w(944.95)
INSERT  INTO APP.OptTriang[6x6] VALUES ([2,4] k(4), COST(4,2)= w(353.46)
INSERT  INTO APP.OptTriang[6x6] VALUES ([3,2] k(4), COST(2,3)= w(500.56)
INSERT  INTO APP.OptTriang[6x6] VALUES ([3,4] k(4), COST(4,3)= w(412.8)
INSERT  INTO APP.OptTriang[6x6] VALUES ([1,2] k(5), COST(2,1)= w(358.99)
INSERT  INTO APP.OptTriang[6x6] VALUES ([1,3] k(5), COST(3,1)= w(652.8)
INSERT  INTO APP.OptTriang[6x6] VALUES ([1,4] k(5), COST(4,1)= w(999.27)
INSERT  INTO APP.OptTriang[6x6] VALUES ([1,5] k(5), COST(5,1)= w(334.05)
INSERT  INTO APP.OptTriang[6x6] VALUES ([2,3] k(5), COST(3,2)= w(946.57)
INSERT  INTO APP.OptTriang[6x6] VALUES ([2,4] k(5), COST(4,2)= w(687)
INSERT  INTO APP.OptTriang[6x6] VALUES ([2,5] k(5), COST(5,2)= w(358.99)
INSERT  INTO APP.OptTriang[6x6] VALUES ([1,2] k(6), COST(2,1)= w(126.46)
INSERT  INTO APP.OptTriang[6x6] VALUES ([1,3] k(6), COST(3,1)= w(481.01)
INSERT  INTO APP.OptTriang[6x6] VALUES ([1,4] k(6), COST(4,1)= w(1000.89)
INSERT  INTO APP.OptTriang[6x6] VALUES ([1,5] k(6), COST(5,1)= w(667.6)
INSERT  INTO APP.OptTriang[6x6] VALUES ([1,6] k(6), COST(6,1)= w(262.89)
---
STEP 4:
 VALUES ([1,2] k(6), COST(2,1)= w(126.46)
```

Fig. 17.13. The parameters obtained in the phases of the proposed method

17.7 Conclusion

In this chapter, we introduced one solution for smart cities data storage and processing based on the triangulation method. The proposed method has two modules: the first for storing and collecting data in a smart city space, while the second module relates to the data processing process and finding the minimum triangulation in a smart city space. We have provided specific examples of implementing and testing the proposed method in a Java NetBeans environment.

In particular, the proposed solution can find its application in the concept of realizing smart traffic or transportation. For traffic congestion, solutions are mainly found through projects based on the use of computer systems and simulations of different traffic cases, ie in the integration of IT and traffic infrastructures. The use of modern information technologies encourages the establishment of new infrastructure consisting of networks of roads, railways, airports, stations and ports connected by internet-based systems.

Efficiency and quality are significantly influenced by intelligent systems that improve the mobility and safety of road users, as they provide proactive maintenance and faster and better diagnostics. These advanced solutions, by the way, increase the productivity of business operations, shorten travel time and reduce environmental pollution.

References

Alam, M. (2012a). *Cloud algebra for cloud database management system*, The second international conference on Computational Science, Engineering and Information Technology (CCSEIT-2012), October 26–28, Coimbatore, India, Proceeding published by ACM.

Alam, M. (2012b). Cloud Algebra for handling unstructured data in cloud database management System. *International Journal on Cloud Computing: Services and Architecture (IJCCSA)*, *2*(6), Taiwan.

Alam, M, & Alam, B. (2013). Cloud query language for cloud database. In *Proceedings of the international conference on recent trends in computing and communication engineering – RTCCE 2013* (pp. 108–112), Hamirpur..

Alam, M., & Sethi, S. (2013). Security risks & migration strategy for Cloudsourcing: A government perspective. *International Journal of Engineering and Innovative Technology, 2*(7), 205–209.

Alam, M., & Shakil, K. A. (2013a). A decision matrix and monitoring based framework for infrastructure performance enhancement in a cloud based environment, international conference – Hyderabad, Nov 08–09, India, Elsevier.

Alam, M., & Shakil, K. A. (2013b). Cloud database management system architecture. *International Journal of Advances in Computer Science and Its Applications, 3*(1), 27–31, Australia.

Alam, M., & Shakil, K. A. (2016). *Big data analytics in cloud environment using Hadoop*. In International conferences on Mathematics, Physics & Allied Sciences-2016, March 03–05.

Alam, B., Doja, M. N., Alam, M., & Malhotra, S. (2013a). 5-layered architecture of cloud database management system. *AASRI Procedia Journal, 5*, 194–199, Elsevier.

Alam, B., Doja, M. N., Alam, M., & Malhotra, S. (2013b). Security issues analysis for Cloud Computing. *International Journal of Computer Science and Information Security, 11*(9), 117–125.

Alam, M., Sethi, S., & Sethi, S. (2013c). Covert channel detection techniques in cloud, confluence 2013: The next generation information technology Summit (4th International Conference), (pp. 127–132), IEEE.

Alam, M., Ara, K., Javed, M. S., & Ansari, M. (2014). *Detect and filter traffic attack through cloud Trace back and neural network, Imperial College*. The 2014 International Conference of Data Mining and Knowledge Engineering (ICDMKE) London, U.K., 2–4 July.

Ali, S. A, & Alam, M. (2016). *A relative study of task scheduling algorithms in Cloud Computing environment.* In Proceedings of the 2016 2nd international conference on contemporary computing and informatics, 7917943, (pp. 105–111).

Ali, S. A, Affan, M., & Alam, M. (2019). *A study of efficient energy management techniques for Cloud Computing Environment*. 9th International conference on Cloud Computing, data science & engineering (Confluence), (pp. 13–18), Noida, India.

Aybeyan, S., Krrabaj, S., Saracevic, M., & Pepic, S. (2019). Memoization method for storing of minimum-weight triangulation of a convex polygon. *Computer Science – AGH, 20*(2), 195–211.

Chen, J., & Chen, C. (2008). *Foundations of 3D graphics programming: Using JOGL and Java3D*. London: Springer.

Chen, X., & He, K. (2017). The function of GIS in the Smart City construction, smart computing and communication. *Book Series: Lecture Notes in Computer Science, 10135*, 374–380.

Cosido, O., Loucera, C., & Iglesias, A. (2013). *Automatic calculation of bicycle routes by combining meta-heuristics and GIS techniques within the framework of smart cities*. 2013 International conference on new concepts in smart cities: Fostering public and private alliances (SMARTMILE).

Davis, G. (2004). *Learning Java bindings for OpenGL*. Bloomington: Author-house publisher.

Fashal, N., El Khayat, G., Salem, B., & El Kaffas, S. (2019). A multi-criteria GIS based methodology for smart cities site selection, electronic governance and open society: Challenges in Eurasia. *Book Series: Communications in Computer and Information Science, 947*, 38–51.

Heffelfinger, D. (2011). *Java EE 6 development with NetBeans 7*. Birmingham: Pack publishing.

Imran, K., Naqvi, S. K., & Alam, M. (2015). *Data model for Big Data in cloud environment, computing for sustainable global development (INDIACom)*. 2015 2nd International Conference on, 11–13 March 2015, (pp. 582–585), New Delhi, India, IEEE.

Khan, S., Shakil, K. A, & Alam, M. (2016). *Educational intelligence: Applying cloud-based big data analytics to the Indian education sector*. In Proceedings of the 2016 2nd International conference on contemporary computing and informatics, IC3I 2016, 7917930, (pp. 29–34).

Khan, S., Shakil, K. A., & Alam, M. (2017). Big Data computing using cloud-based technologies: Challenges and future perspectives. In M. Elkhodr, Q. F. Hassan, & S. Shahrestani (Eds.), *Networks of the Future: Architectures, Technologies and Implementations* (Book series: Computer and information science series). Boca Raton: Chapman and Hall/CRC Press.

Khan, S., Arshad Ali, S., Hasan, N., Ara Shakil, K., & Alam, M. (2019). *Cloud Computing for geospatial Big data analytics* (pp. 1–28). Cham: Springer Book.

Kumar, V., Kumar, R., Kumar Pandey, S., & Alam, M. (2017). *Fully homomorphic encryption scheme with probabilistic encryption based on Euler's theorem and application in Cloud Computing, big data analytics* (pp. 605–611). Springer.

Kumari, M. Y., Kumar, A. V., & Alam, M. (2015). Design flaws and cryptanalysis of a standard mutual authentication protocol for Cloud Computing based healthcare system, Springer lecture notes in Electrical Engineering.

Ledoux, H., Ohori, K., & Meijers, M. (2014). A triangulation-based approach to automatically repair GIS polygons. *Computers & Geosciences, 66*, 121–131.9.

Lella, J., Mandlab, V., & Zhuc, X. (2017). Solid waste collection/transport optimization and vegetation land cover estimation using Geographic Information System (GIS): A case study of a proposed smart-city. *Sustainable Cities and Society, 35*, 336–349.2.

Malhotra, S., Doja, M. N., Alam, B., & Alam, M. (2015). Data integration of cloud-based and relational databases. *International Conference on Soft Computing Techniques and Implementations, ICSCTI, 7489542*, 83–86.

Malhotra, S., Najmud Doja, M., Alam, B., & Alam, M. (2018). Generalized query processing mechanism in cloud database management system. In V. Aggarwal, V. Bhatnagar, & D. Mishra (Eds.), *Big data analytics. Advances in intelligent systems and computing* (Vol. 654, pp. 641–648). Springer.

Pradhan, B., Sandeep, K., Mansor, S., Ramli, A., & Sharif, A. (2017). Second generation wavelets based GIS terrain data compression using Delaunay triangulation. *Engineering Computations, 24*(2), 200–213.

Roy, P., & Mandal, J. (2012). *A delaunay triangulation preprocessing based Fuzzy-Encroachment graph clustering for large scale GIS data*. In: 2012 international symposium on Electronic System Design (ISED 2012), (pp. 300–305).

Samiya, K., Shakil, K. A., & Alam, M. (2017). *Cloud based Big data analytics: A survey of current research and future directions, Big data analytics* (pp. 629–640). Cham: Springer.

Saračević, M., & Selimi, A. (2019). Convex polygon triangulation based on ballot problem and planted trivalent binary tree. *Turkish Journal of Electrical Engineering and Computer Sciences, 27*(1), 346–361.

Saračević, M., Stanimirović, P., Krtolica, P., & Mašović, S. (2014). Construction and notation of convex polygon triangulation based on ballot problem. *ROMJIST – Journal of Information Science and Technology, 17*(3), 237–251.

Saračević, M., Masovic, S., Stanimirovic, P., & Krtolica, P. (2018). Method for finding and storing optimal triangulations based on square matrix. *Applied Sciences – Geometry Balkan Press, 20*, 167–180.

Saračević, M., Selimi, A., & Plojovic, S. (2019). *Some specific examples of attacks on information systems and smart cities applications*. Advanced sciences and technologies for security applications (in Book: Cybersecurity and Secure Information Systems). Springer.

Shahrour, I. (2018). Use of GIS in smart city projects. *GIM international-the worldwide magazine for geomatics, 32*(5), 21–23.

Shakil, K. A., & Alam, M. (2014). Data Management in Cloud Based Environment using k-median clustering technique. *International Journal of Computer Applications, 3*, 8–13.

Shakil, K. A., & Mansaf, A. (2017). *Cloud Computing in bioinformatics and Big data analytics: Current status and future research, Big data analytics* (pp. 629–640). Berlin: Springer.

Shakil, K. A, Alam, M., Shakeel, S., Ora, A., & Khan, S. (2018). *Exploiting data reduction principles in cloud-based data management for cryo-image data.* In Proceedings of the 2018 international conference on computers in management and business (pp. 61–66), ACM.

Sowizral, H., & Deering, M. (1999). The Java 3D API and virtual reality. *IEEE Computer Graphics and Applications, 19*(3), 12–15.

Stanimirović, P., Krtolica, P., Saračević, M., & Mašović, S. (2012). Block method for triangulation convex polygon. *ROMJIST – Journal of Information Science and Technology, 15*(4), 344–354.

Stanimirović, P., Krtolica, P., Saračević, M., & Mašović, S. (2014). Decomposition of Catalan numbers and convex polygon triangulations. *International Journal of Computer Mathematics, 91*(6), 1315–1328, Taylor and Francis.

Tan, S., & Wong, O. (2006). Location aware applications for smart cities with Google maps and GIS tools. In *Advances in intelligent IT: Active media technology* (Book series: Frontiers in Artificial Intelligence and applications) (Vol. 138, pp. 223–228).

Yamamura, S., Fan, L., & Suzuki, Y. (2017) *Assessment of urban energy performance through integration of BIM and GIS for smart city planning*. International High-Performance Built Environment Conference, in Book Series: Procedia Engineering 180 (pp. 1462–1472).