Ingénierie des Systèmes d'Information
Vol. 27, No. 1, February, 2022, pp. 101-109
Journal homepage: http://iieta.org/journals/isi

Smart-Approach Based Internet of Things and Skyline Query for Multicriteria Decisions for Travel Services

Oum Elhana Maamra[1,2*], Mohamed-Khireddine Kholladi[1,2], Okba Kazar[3,4], Saad Harous[5]

[1] Faculty of Science Exact, Computer Science Department, University of El-Oued, El Oued 39000, Algeria
[2] MISC Laboratory, Abdelhamid Mehri University of Constantine 2, Constantine 25000, Algeria
[3] Department of Computer Science, Smart Computer Science Laboratory, University of Mohamed Khider, Biskra 07000, Algeria
[4] Department of Information Systems and Security, College of Information Technology, United Arab Emirate University, Al-Aïn 15551, UAE
[5] Computer Science Department, College of Computing and Informatics, University of Sharjah, Sharjah 27272, UAE

Corresponding Author Email: maamra-oumelhana@univ-eloued.dz

https://doi.org/10.18280/isi.270112

Received: 27 September 2021
Accepted: 12 January 2022

Keywords:
Internet of Things, business intelligence and analytics, skyline query, cloud computing, data mining, multicriteria decision

ABSTRACT

The Internet of Things (IoT) and the recent advancements in cloud computing have gained importance with the surge in the amount of data generated globally. Moreover, the rapidly increasing applications of the Internet in many scientific and real-time practical applications have ushered in a new era of complex applications of data flow. Tourism and related services are routinely accessed by millions of customers worldwide. Furthermore, with newer, attractive, rapidly growing services, it has become essential for dealers to promote their services using up-to-date technological tools. The major challenge is to efficiently determine and select the best travel options conforming to the needs and financial requirements of the customers. In this study, the use of a dynamic skyline operator for multicriteria decisions is examined using a time-dependent database to select the best services. Moreover, the impact of implementing the operator on optimizing resource consumption is explored. Results indicate that the implementation of this operator is more efficient than the existing techniques.

1. INTRODUCTION

Nowadays, processing a skyline query has become a crucial topic in database research as it allows extracting interesting objects from multidimensional datasets. Skyline query processing can be employed in many applications requiring multicriteria decisions without using a cumulative functionality to identify the best results based on users' preferences or conditions. The main feature that distinguishes a skyline query from many other preference queries, such as TopK queries, is that it is independent of the definition of a score function while classifying tuples. However, the only limitation to the functioning of the skyline query is a partial relationship among the values of each dimension [1]. The skyline query and its concept can be explained in detail using practical examples. For example, online tourism booking allows customers to carefully select their tourism services based on their needs and financial situation; multiple choices and options are available for suitable booking services and low-cost services.

The Internet of Things (IoT) is concerned with connecting people and physical objects. People can now easily interconnect with physical objects and services using simple gear. For example, interconnection smartphones with sensors deployed in cars and industrial facilities.

This work focuses on applying the principle of skyline queries to booking services to help users select the best available service in an IoT context using a distributed architecture. The selection is performed using a multicriteria decision (selection) among the best elements within a database pack considering two factors, namely, quality and minimum search and selection time.

To fulfill the growing demands of booking and online tourism services, dynamic skyline operators provide active assistance for reducing the search time and cost of the offered services based on the users' requirements. Therefore, the main objective of this study is to propose an efficient approach for searching and selecting the best booking services based on the customers' needs in the shortest possible time. The rest of this paper is organized as follows. In Section 2, an overview of business intelligence and analytics, IoT, cloud computing, and skyline query is provided. In Section 3, the relevant leading research on skyline queries and skyline-join queries in distributed databases is described. In Section 4, the proposed architecture modeling is described along with the motivation for its adoption. In Section 5, the proposed approach is presented. In Section 6, the implementation and result evaluation are described. Finally, in Section 7, conclusions and future research directions are discussed.

2. RELATED WORK

In this section, we first briefly present the purpose of using the cloud computing and business intelligence and present the purpose of using the cloud computing and business

intelligence and analytics in this study. Then, we introduce the success of approaches based on the Internet of Things for reservations systems. Finally, we review related works for distributed databases applications by using skyline query approaches.

2.1 Business intelligence and analytics

Business intelligence and analytics (BI&A) has become a significant topic for many researchers and practitioners. This is attributed to the remarkable impact of data-related problems on the technologies, systems, practices, methodologies, and applications that analyze critical business data, and the inevitability of establishing new approaches and terms for solving problems in contemporary business organizations [2].

Over the years, a huge amount of historical data has been collected various types structured and unstructured data. Such data are crucial for small and large business organizations. Successful companies make strategic business decisions based on the analysis of such data. For further improvement, business organizations must implement integrated data management systems and use BI&A techniques. Big data analytics have several applications related to many academic and industrial fields [3].

2.2 Cloud computing

Cloud computing is a common solution for managing high-speed computing using large-scale databases. This revolutionary concept allows the enhanced use of distributed resources [4]. Moreover, the data -warehousing approach has currently become a popular and efficient solution that can improve decision-making in companies [5].

2.3 Smart-approach based Internet of Things for reservation systems

In an IoT architecture, sensors collect a wide variety of information from their closest environments. The gathered data are sent via gateways from objects (different devices) to the cloud and vice versa. The gateways enable data preprocessing and filtering before sending the data to the cloud. Such data may also be transferred to the cloud for detailed processing and storing. Furthermore, the gateways ensure the transit of the control commands from the cloud to the objects (things) [6]. Subsequently, actuators within these objects (things) execute these control commands [7]. The services provided by the IoT include booking services reliant on distributed and heterogeneous data, where the user seeks to obtain appropriate booking services in a reasonable time. Furthermore, scientific analysis is required for big businesses in this field to make strategic decisions, thereby improving the quality of services provided to the user.

Sagar et al. [8] proposed a vehicle-booking reservation system using IoT. They employed automatic car parking using a microcontroller and adopted the global system for mobile communications to monitor available space for booking via a mobile application.

2.4 Skyline query approaches for distributed databases applications

The skyline query [9] is a popular approach for minimizing the search space to select only few data objects exhibiting certain characteristics.

Let D be a d-dimensional database; d1, ..., dm denote the dimensions of the Skyline; for example, price, distance to the beach, rating, ect. A tuple p = (p1,...,pk, pk+1,...,pl, pl+1, ...,pm, pm+1,...,pn)dominates tuple q = (q1,...,qk, qk+1,...,ql, ql+1,...,qm, qm+1,...,qn) if p is not worse than q in any d dimension and p is better than q in no less than one d dimension. “The skyline query is defined as the process of recovering a set of points where each one is not dominated by another point [10]. It has been applied in several multicriteria-based decision-making applications. Search database systems [11] web service composition [12], and routing in wireless ad hoc networks [13] are examples of such applications.

Skyline queries have been used in many scenarios and configurations. For instance, a skyline query is usually used for handling large or anticorrelated datasets. Moreover, it can be used when the customer is keen to inspect a particular subspace rather than the entire data space or when the users or customers show interest in specific constraints. Each constraint is typically expressed as a range along a dimension of the dataset [14]. In a study by Borzsony et al. [9] the skyline operator in large-scale datasets offers three algorithms: block-nested-loops (BNL), divide-and-conquer, and B-tree-based schemes. Zaman and Morimoto [10] proposed area skyline query for selecting good locations in maps.

The dynamic skyline query [15, 16] is a type of query in which the coordinates of each point are given by a set of distance (dynamic) functions, which consider the distance between a given query/reference point q and a point p of the original dataset [14].

Zou et al. [17] derived a pruning rule that leverages graph properties to determine the candidates for the problem of dynamic skyline queries in a large graph (hereinafter, a DSG-query). To efficiently process a query, they used a filter-and-refine framework. They showed that the DSG-query is answered and short distances between the summits are computed in O(H). Here, H represents the number of maximal hops between any two summits.

Many methods have been developed for skyline-join queries in distributed and no distributed environments, where data are stored in multiple tables requiring the existence of join operations between them to compute the final skyline and then propose efficient approaches for sharing the join processing cost with the skyline computation cost [14]. For example, Vlachou et al. [18] SFSJ (Sort-First-Skyline-Join) can compute the correct skyline set by accessing only a subset of the input tuples, i.e., it exhibits the early termination property. It can be easily implemented in existing database systems, as it relies on a common infrastructure. However, this algorithm is not practical for complex database systems.

Sun et al. [19] proposed a distributed adaptation of SaLSa and an iterative algorithm to calculate the skyline join in a distributed manner. This algorithm is a hybrid of the skyline and joins operations. Raghavan and Rundensteiner [20] established a ProgXe framework for skyline join calculation. This framework supports progressive result generation. ProgXe divides the input relations using a multidimensional grid access approach. It transforms the execution of Multi Criteria Decision Support (MCDS) queries, comprising skyline over joins, to be non-blocking by progressively generating results early and often restricts the framework to joins using base tables; however, in practice, the join may also be calculated over complex relational expressions.

Jin et al. [21] described a skyline operator on multirotational

tables. This operator combines a Trimerge join technique with skyline processing. Given A and B, two relations for computing the skyline over A >< B, where a one-to-many relationship exists between A and B such that the join operation is performed if A has a primary key corresponding to a foreign key in B. Further, the join operation on A and B with a many-to-many relationship can be solved by introducing a third table C that contains the attributes of B and the join attribute of A; then, a join between A and C and one between the resultant join and B can be performed.

Vlachou et al. [22] proposed an algorithm called SKYPEER in peer-to-peer networks, where the dataset is horizontally distributed across the peers. First, in a preprocessing phase each peer Pi computes the local ext-skyline of its dataset Si and sends it to the associated super-peer. The super-peer calculates its own ext-skyline, by merging the local skyline results. This result is a threshold value. In processing request the SKYPEER forwards the skyline query requests among peers according to local results.

Kalyvas et al. [23] used skyline queries and introduced the time factor in traditional skyline algorithms in temporal databases.

Zhang et al. [24] proposed a skyline join algorithm called Skyjog for two or more relations based on group division approach. For each relation, tuples are grouped according to the join attribute. This proposed algorithm divides the tuples into diverse partitions depending on the dominance relationships of inter-group and intra-group. The final result of Skyjog is the tuples generated by some join combinations to be skyline points.

Table 1. Comparison between different approaches

Works	Multi criteria decision		
	skyline query type	Data type	IOT architecture
Sagar et al. (2016) [8]	none	Electric signals	+
Borzsony (2001) [9]	Operator skyline	Relational database	-
Zaman et al. (2016) [10]	Spetial skyline query	Maps data (area)	-
Kalyvas &Tzouramanis (2017) [14]	Temporal skyline query	Temporal Databases	-
Zou et al. (2010) [17]	Dynamic skyline query	Graph properties	-
Vlachou et al. (2011) [18]	Skyline join algorithm	Relational database	-
Sun et al. (2008) [19]	Join skyline query	Distributed data	-
Raghavan et al. (2010) [20]	Join skyline query	Distributed data	-
Jin et al. (2007) [21]	Join skyline query	Multi-relational databases	-
Kertiou et al. (2018) [6]	Dynamic skyline query	Distributed data	+
Zhang et al. (2016) [24]	Skyline join algorithm	Multiple relations	-
Amiruzzaman et al. (2020) [25]	Multi-dimensional Skyline query	Single data set	-

Amiruzzaman and Jamonnak [25] proposed web-based system focusing on customers' stratification with used the multidimensional Skyline query. This system provides ranks of shopping malls based on customers' preferences and helps to find best shopping malls based on users' requirements.

In Table 1, we compare the presented related works of the multicriteria decision such that the comparison lines are skyline query type, data type and distributed architecture.

In brief, the previous studies on skyline were restricted to either single relation or multi-relations in distributed and no distributed environments. Moreover, these approaches did not consider the fact that IoT systems are naturally distributed. The present work focuses on exploiting the distributed nature of IoT to employ a local join dynamic skyline on the gateway level and then proceed to employ a global join dynamic skyline on the server level for a typical example (i.e., booking services). The main goal is to search and select the best available services considering users' requests.

In this work, the skyline queries were used for tourism-condition search and selection of the best booking services. The proposed scheme achieves optimization of the time required to execute queries and reduces resource consumption, thereby improving the quality of decision-making. Further details of our proposed optimization are as presented below:

- The focus is on the dynamic skyline during user-based booking service selection to remove the no dynamic skyline booking services that are dominated by dynamic skyline booking services; in other words, dynamic skyline booking services have better user-based responses with respect to the users' requests and the number of available booking services than no dynamic skyline booking services. The dynamic skyline returns the best services to answer the users.
- This work exploits the parallel nature of IoT architectures, wherein the architecture comprises distributed gateways in the network and is connected to a local server in cloud computing. Each local server manages the data of different travel services before locally responding to users' requests. Thereafter, the global server implements a global dynamic skyline operator to gather the results of all local services; ultimately, the final answer is provided based on the client requirements.
- The server in this architecture is linked to a predefined database comprising services already customized based on the clients' selections. This database is primarily used to analyze different requests related to tourism. This analysis effectively improves access to high-quality services and offers a considerable amount of tourism-service data in a short duration. Moreover, it creates and boosts competition among the businesses that offer similar services to meet the client requirements.

3. MOTIVATION AND ARCHITECTURE MODELING

In this section, the concept of the proposed technology is presented by providing an overview of the proposed architecture for selecting appropriate travel services. Further, a detailed description of the overall implementation flow and action steps is described.

3.1 Booking service detection architecture

In this work, the proposed structure is divided into three

sections. In the first section, the users' request is first introduced in the interface system. In the second section, the global server employs a global booking skyline (GBS) by gathering the results of all local services and subsequently provides the final answer based the clients' specified needs. Moreover, the server of this architecture is linked to a service database established in advance using clients' selections; this database is essentially used to analyze different tourism requirements preferred by the user using methods such as data mining. This approach helps make decisions for improving services to gain as many customers as possible. In the third section, the local booking skyline (LBS) is employed at the local server level. This section consists of gateways dispersed in the network and connected to a local server in cloud computing. Each local server manages a data warehouse of different travel services, and each local server locally responds to users' requests. Thus, in this study, the quantity of data that must be treated during the search is reduced and the best booking services are selected. Figure 1 shows the proposed architecture. Details of the three sections are provided below:

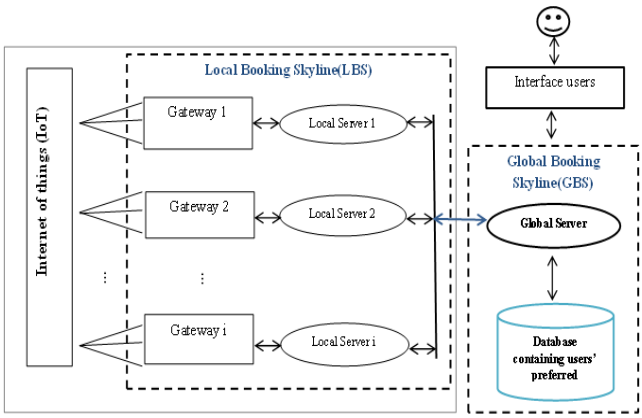

Figure 1. Architecture for detecting booking services

Section 1-the requests and responses of users are introduced in the interface system.

Section 2-The GBS comprises a global server and a database.

- Global server: This server processes the consumer requests, combines the results of all local servers, delivers the final answer to the users, and stores users' preferred selections.
- Database of users' preferred choices: The database is linked to the global server and is used for analyzing different tourism requirements preferred by the users using data mining.

Section 3-The LBS comprises the local servers, gateways, and tourism services.

- Local servers: They are responsible for sending the server requests and collecting the data from the gateways. Additionally, they manage a data warehouse of different tourism services.
- Gateway: The gateway is responsible for navigating data from tourism services to the local server in the cloud and vice versa. It also preprocesses and filters the data before forwarding them to the local server. This step reduces the amount of data to be processed and stored. Furthermore, the gateway handles the sensor updates of travel services.
- Tourism services: They are objects such as hotels, airplanes, cars, and restaurants. A sensor is mounted on each one of these objects for data collection. These objects transfer the data over a network using actuators. The sensors associated with these objects send their measurements to the corresponding gateways.

3.2 System description and modeling

The problem addressed in this study is how to select and book a subset of best various travel services based on the clients' conditions from large groups of tourism services offered on the Internet. The user determines the travel costs that helps in selecting the services that offer prices appropriate to the situation of the customer. Consequently, this problem should be adequately modeled to help represent the problem of research and efficiently select the most effective elements from various travel services. Therefore, we propose an effective technique for solving the problem that has been addressed.

Our model (Figure 2) shows how a user sends a request and how research and selection of the best travel service solution are performed. As for the details of the proposed architecture shown in Figure 3.

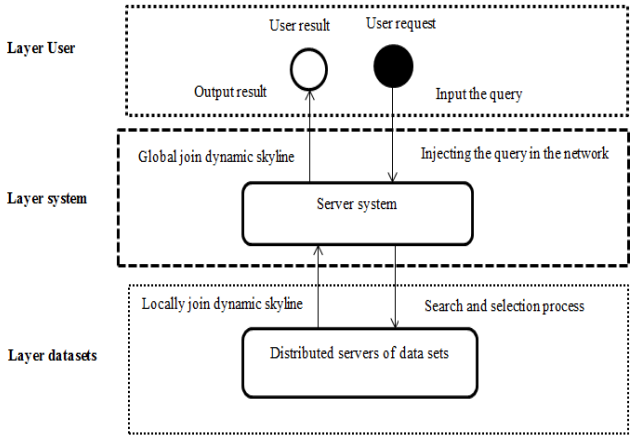

Figure 2. Proposed Processes configuring queries and searching and selecting the best booking services

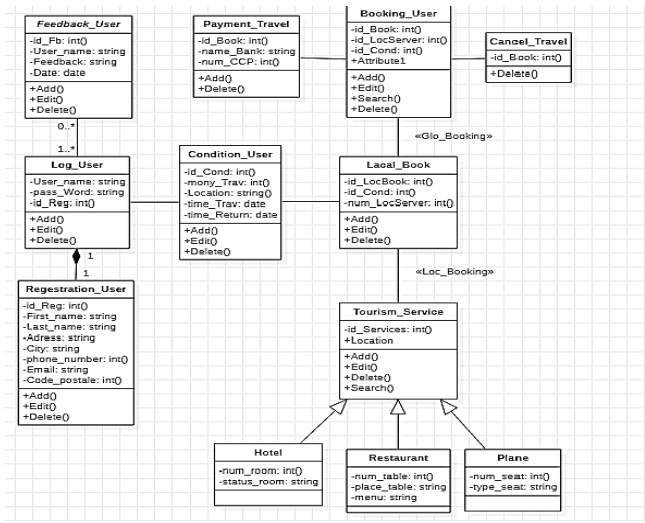

Figure 3. Details of the proposed architecture

This process contains three layers: User layer, System layer and Datasets layer.

- User layer: In this layer the user sends a specific request and implements the basic procedures of process. When all steps of the process are performed, the user receives the best results.
- System Layer: In this layer the global server injects the user's request into the network to access the database servers available to work on. After it receives the results of each server, the global server merges the selected results. Then, it selects the best result and displays it to the user.
- Datasets layer: In this layer the server of each dataset searches and selects the appropriate information for the customer. Each server sends its results to the global server.

3.3 Detailed viewpoint of the proposed architecture

Here, the main constructions of the class diagram are illustrated.

3.4 Functioning viewpoint of the system

Here, a "functional" viewpoint of the system architecture is presented. Through use cases, we will gain sufficient information of the system components to define the limits of the system.

3.4.1 Booking travel services

The following figure (Figure 4) shows our proposed sequence diagram of booking travel services.

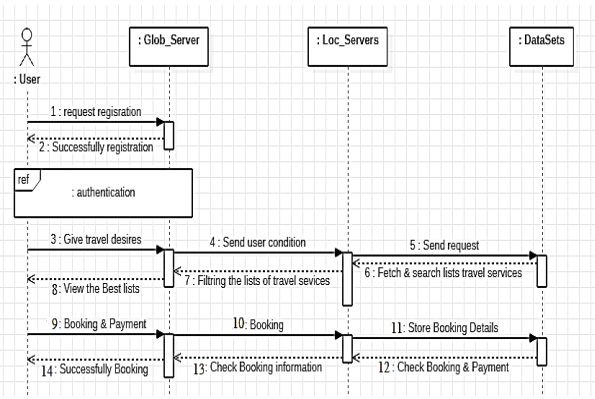

Figure 4. Sequence diagram of booking travel services

3.4.2 Feedback users

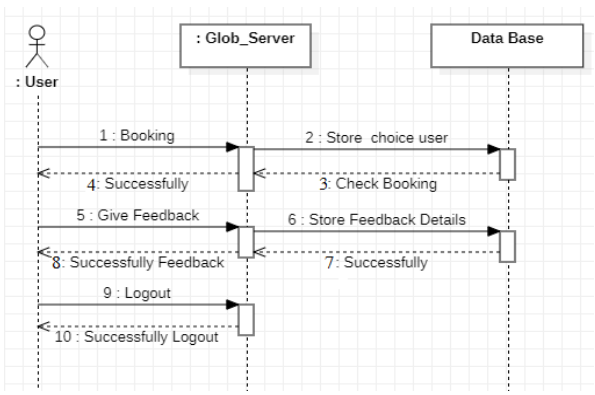

Figure 5. Sequence diagram of answer

With regard to answer, the user selection and comment are stored in the database of clients' preferred selections; this database is used for analyzing different tourism services preferred by the user. Figure 5 shows the proposed sequence diagram of answer.

4. PROPOSED SKYLINE APPROACH FOR TOURISM

In this section, we discuss in detail the proposed approach for travel booking using skyline queries in an IoT environment.

4.1 Problem formulation

Let B be a set of booking services categorized as m services, B={B1, B2, B3, …, Bm}. Each service has a relation in the database. B1={h1, h2, h3, …., hn}∈B represents the relation of hotels, B2={a1, a2, a3, …, ak}∈B represents the relation of airplanes, and B3={r1, r2, r3, …, rd}∈B represents the relation of restaurants, etc. Each booking service type is classified into permanent services, which are effective throughout the period of travel, or temporary services, which are effective at a specific time during travel. The permanent services are annotated using the "+" sign (e.g., B+), whereas the temporary services are annotated using the "−" mark (e.g., B−).

Let C be a set of user conditions, where the common characteristics of travel services represent join attributes J between the relations; let d denote a dynamic scoring attribute between the relations. Let K be a set of numerical attributes in the schema of every relation, where K=(K1 ∪ K2 ∪ K3 ∪ … ∪ Km).

For example, the number of services is m=3(R1=hotels, R2=planes, R3=restaurant), C={Travel_money, Location, start_date, date_retoure}; in this case J={Location, date_travel}, D={Travel_money}, K1={price, nb_star}, K2={price, class}, and K3={price, quality}.

4.1.1 Search space of the booking skyline query

For defining the search space of the LBS, we have a query <K, J, d> and let B be a table. We can define the search space of booking skyline BS by achieving the following conditions:

a) BS = B1 ⊳⊲ B2 ⊳⊲ B3…Bm,$B_i \in$ BS;

b) $\forall \text{att}_i \in (k \cup d) \rightarrow \exists B_i (B_i \in \text{BS}) \wedge (\text{att}_i \in B_i)$;

c) $\forall B_i (B_i \in \text{BS}) \rightarrow \exists \text{att}_i (\text{att}_i \in B_i) \wedge (\text{att}_i = \text{price}_i)$;

d) In this case, d is the total price of booking services. Hence, for the best search space, first, we define the travel time using variable t, where t = date_retoure − start_date; we use the variable t with the permanent services B+.

$$\forall B_i^+ (B_i^+ \in \text{BS}) \rightarrow \exists \text{price}_i^+ (\text{price}_i^+ \in B_i^+) \wedge (\text{price}_i^+ = \text{price}_i \times t)$$

For temporary services B−, we use the following definition:

$$\forall B_i^- (B_i^- \in \text{BS}) \rightarrow \exists \text{price}_i^- (\text{price}_i^- \in B_i^-) \wedge (\text{price}_i^- = \text{price}_i)$$

Then, we cannot determine BS and the total prices are less than or equal to the dynamic attribute:

$$\sum_{i=1}^{m} \text{price}_i^+ + \sum_{i=1}^{m} \text{price}_i^- \leq d$$

e) BS' that has two properties (described above) and fewer tables than BS cannot be found.

$$\nexists \sum_{j=1}^{m} \text{price}_j^{+} + \sum_{j=1}^{m} \text{price}_j^{-} \leq \sum_{i=1}^{m} \text{price}_i^{+} + \sum_{i=1}^{m} \text{price}_i^{-}, j \neq i$$

f) The search space of the distributed booking skyline β for a query <K, J, d> is defined as follows:

$$\beta = \text{BS1} \cup \text{BS2} \cup \text{BS3..} \cup \text{BSn}, \text{BS}_i \in \beta$$

4.1.2 Local booking skyline domination

For a given booking skyline query <K, J, d> and its search space BS, assume t1 = (v0, v1, v2,..., vd) and t2 = (v'0, v'1, v'2,..., v'd) are two tuples of BS. t1 dominates t2 if

$$\forall \text{att}_i(\text{att}_i \in K) \wedge (\text{att}_i = \text{price}_i) \rightarrow \left| d - \sum_{i=1}^{m} v_i \right| \prec \left| d - \sum_{i=1}^{m} v_i^{'} \right| \wedge \exists \text{att}_j(\text{att}_j \in k): \text{v}_j^{'} \leq v_j$$

For clarity, we use ti <dom tj to indicate that the tuple ti booking skyline dominates tj and $\sum_{i=1}^{m} \text{price}_i \prec_d \sum_{j=1}^{m} \text{price}_j$ to indicate that ti booking skyline dominates tj in the sum of price services for d.

4.1.3 Global booking skyline domination

The global result set G for a booking skyline query Q = <K, J, d> in its search space β must have the following properties:

a) $\forall t_i \in G \rightarrow \exists t_j(t_j \in \beta) \wedge t_j \succ_{dom} t_i$;

b) Suppose ti ∈ G and ti = (v0, v1,..., vm); if atti ∈ K, vi must satisfy the corresponding conditions.

4.2 Booking skyline query

In this section, we propose an approach for selecting tourism booking services using distributed databases in an IoT environment.

Our algorithm for computing the booking skyline enables the selection of the best tourism services based on the users' requirements in terms of the cost of travel, location, and travel time. For joining services (tables) in getaways, we determine the location as a join attribute between different tables, leading to the first filtrate in the tables. Tables join is an extension of the original skyline join. The dynamic skyline query is adapted to minimize the search space as much as possible to improve the selection efficiency of the booking services based on the user conditions. The booking skyline is a set of all the services not dynamically dominated by any other service with respect to the distance of a given query service. In particular, we focus on the booking skyline services for user-based service selection, where the ones booked via skyline dominated the no booking skyline services. This is attributed to the fact that the booking skyline services allow better user results than the non booking skyline services.

The steps for executing the proposed skyline procedure are specific in the instructions of Algorithm 1. Table 2 provides the definition of different abbreviations. As noted in the section 4. 1, booking services comprises search and selection processes. It consists of four steps.

Table 2. Different abbreviations used

Abbreviations	Definition
Si	Local server i
Q	User query
GBS	Global booking skyline
GBSO	Ordering booking services of GBS
LBS(Si)	Local booking skyline of Si
BSi	Dataset of the booking server Si
Bi	Table of the travel booking service i
V	Indicates whether service i not temporary during travel
J	Join attribute
Ki	Numerical attributes of Bi
Gi	Group tuples of Bi by J
D	Scoring attribute
LG	Temp table for union selected tuples of different tables

4.2.1 Capturing the user request

Once a consumer logs in to the users interface, he/she enters his/her requirements through a web interface. Then, the interface manager forwards the request to the server, which sends it to the gateways.

Algorithm 1 Best booking service selection at local servers.

Input: Local Servers S, Q;
Output: GBS and GDSR;
1: **Initialize:**
2: Q: capturing the user request;
3: **for** each Si **do**
4: LBS(Si): computing the local booking skyline (Si, Q);
5: **end for**
6: GBS: computing the global booking skyline from LBS;
7: GBSR: ordering booking services of GBS;

4.2.2 Computing the LBS

After the gateways receive the user's request, each local server in the different gateways calculates the LBS. Algorithm 2 illustrates the process of LBS as follows.

Algorithm 2 Local booking skyline

input: BS, v, K, J, d
output: result of the local booking skyline
1: **Initialize:**
2: **for** Bi ∈ BS **do**
3: group tuples of Bi by J
4: **if** v **then**
5: **for** tj of Gi **do**
6: pricej = pricej. t
7: LG = LG join Gi
8: **end for**
9: **for** each tuples ti of LG **do**
10: **if** sum (pricek) ≤ d **then**
11: **for** ki ∈ Ki **do**
12: **if** ∈ k'j ∈ Ki and k'j ≠ pricei: k'j < ki **then**
13: record ti into LBS
14: **end for**
15: **end for**
16: **end for**
17: return LBS

The user's requirements are divided into three parts: part one for joining relations, part two for determining the scoring

attribute between the relations, and part three represents the numerical attributes of every relation. Moreover, the user defines the scoring attribute, which is the main point in the booking skyline algorithms.

After the user's request arrives, the LBS algorithm group begins the identification of tuples using the join attribute. It is the first filter operation of booking services. Then, we check the service type; if it is not temporary during the travel, we multiply the price attribute of table Bi at a period of travel t.

The second filtering process of reserving data starts from the temporary table LG. For each tuple ti of LG, we calculate the total travel cost and compare it with the scoring attribute of user d; if it agrees with the condition, we apply the skyline query at the numerical attributes. If the tuple agrees with all conditions, we add it in the result of computing LBS.

Step 3—Computing the GBS: after concluding the result of the LBS server, the result is transmitted to the global server, which merges the results of all the booking services and computes the GBS. Algorithm 3 illustrates the process of determining the GBS as follows:

Algorithm 3 Global booking skyline

```
input: S, J, K, d
output: result of the global booking skyline
1: Initialize:
2: for Si∈ S do
3:   LBSi = LocalBookingSkyline (Si, J, K, d)
4:   GBS = GBS ∪ LBSi
5: end for
6: for each tuples ti of GBS do
7:   for each tuples tj of GBS do
8:     if tj dominated b ti then
9:       remove tj from GBS
10:    else
11:      remove ti from GBS
12:    end for
13: end for
14: return GBS
```

A GBS algorithm runs right after the results of the local servers are gathered from distributed gateways. This algorithm manages to combine all results into a temporary GBS table. Then, the skyline query is applied to the GBS tuples, where delete dominated tuples by another tuples.

Step 4—Ordering booking services: immediately after calculating the GBS, the final result is integrated within the algorithm of ranking booking results. Then, the result becomes accessible to be displayed to the user.

5. IMPLEMENTATION AND RESULT EVALUATION

In this section, the proposed architecture model and the datasets used are described. Moreover, the results of the analysis of different cases are discussed. Finally, a comparison between the performance of the proposed skyline model and that of the iterative algorithm [19] is presented.

5.1 Implementation

Our proposed algorithms that were implemented in Java Eclipse Helios as a prototype of the proposal with a private network were composed of three computational nodes. The data are saved in MySQL database. The first form allows the users to submit their preferences for multiple booking and travel services (hotels, airplanes, restaurants, etc.) according to the dynamic scoring attribute. In this case, to achieve the finest results with a minimum search time, the payment currency used by the customer is predefined. The hardware environment used for the experiments consists of a personal computer with an Intel Core i5, 3.2-GHz processor with 4-GB RAM.

5.2 Data collection

The IoT is crucial for executing the proposed query; hence, it is considered integration environment for the information related to all travel services. In fact, there are no existing datasets with the data of various tourism services; hence, a synthetic 100,000 tuple dataset is generated on three local servers. The experiences have been repeated ten times, and we have taken the average for analysis the results.

5.3 Performance analysis

For performance analysis, the proposed query is examined based on its multiple parameters. The main factor based on which any query is evaluated is the processing time. Therefore, the impact of changing the number of booking services and the size of the datasets on processing time is analyzed. Finally, the impact of the number of services on the number of reservations based on the clients' preferences is investigated.

5.4 Effect of the number of services on processing time

When evaluating the effect of the number of services on the processing time, LBS and GBS are separately measured. Further, the number of services is changed from 2 to 6. The size of datasets is set constant at 50,000 tuples. The obtained results (Figure 6) show that increasing the number of services increased the processing time for both LBS and GBS. However, the increase in the processing time was higher in the case of LBS than in the case of GBS because of the increasing join operations between different services (tables) within the LBS itself.

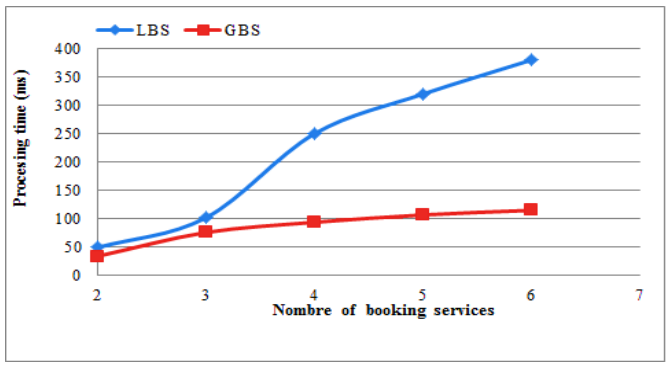

Figure 6. Effect of the number of services on the processing time for local and global booking skylines

5.5 Effect of the size of datasets on processing time

The impact of the size of the datasets on the processing time is examined for both LBS and GBS. The size of the datasets is increased by generating new data, from 1000 tuples to 100,000 tuples.

Figure 7 illustrates the change in the required processing time with an increase in the size of the datasets following the

user's requirement. The analysis findings indicate that when the dataset size was less than 1000 tuples, the difference in the processing times of LBS and GBS significantly decreased. However, immediately after the dataset size exceeded 1000 tuples, the processing time increased until reaching the dataset size of 50,000 tuples.

Beyond 50,000 tuples, the difference in the processing times of LBS and GBS is significantly large.

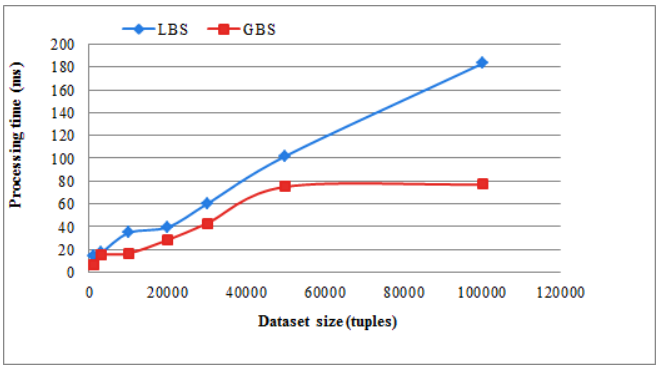

Figure 7. Effect of the dataset size on the processing time for local and global booking skylines

5.6 Effect of the number of services on the number of booking services

Figure 8 shows the impact of the number of tourism services (tables) on the number of booking service results. During these experiments, the principal user's requirement is retained constant in all cases when the number of services is changed. The obtained results clearly reveal that increasing the number of services significantly increased the number of reservation results.

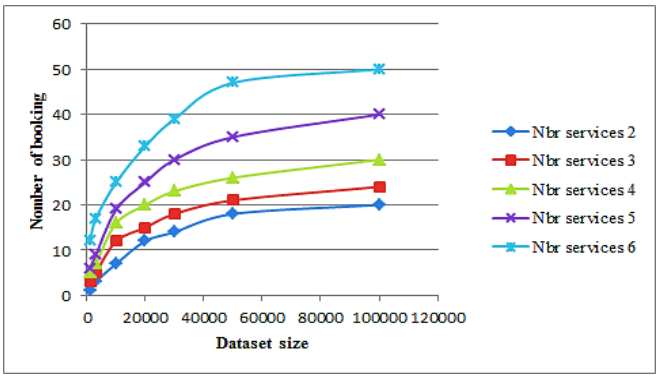

Figure 8. Effect of the number of booking selected for increasing numbers of services and dataset size

5.7 Comparison and results

The steps of the iterative algorithm can be described as follows:

- Compute the local skyline points for each table,
- Calculate the result of partial skyline join,
- Compile the results of the partial skyline join.

In contrast, the proposed skyline model started by computing the local join dynamic skyline, compiling its results, and immediately computing the resulting global join dynamic skyline point. To compare the performance of the proposed skyline model and SaLSa in terms of the processing time, the used datasets size is 500,000 and the number of booking services is 6. Figure 9 shows the results, which clearly indicate that the proposed skyline model achieves better processing time than the iterative algorithm. Because repeated computing results in high-energy consumption with longer processing time, this is why the proposed model outperforms the iterative algorithm.

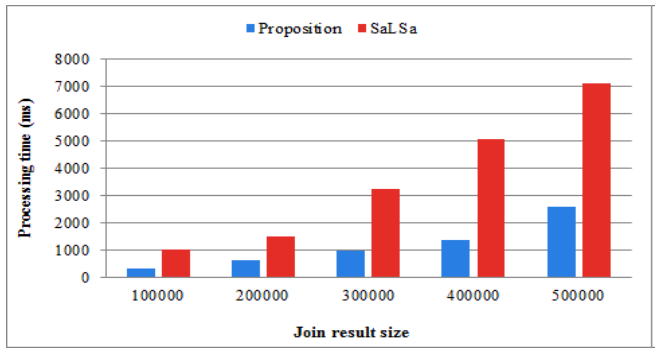

Figure 9. Performance comparison of the proposed model with the iterative algorithm

6. CONCLUSION AND FUTURE WORK

In this study, the problem of the dynamic skyline operator with join in applied in a multicriteria decision-making environment is examined. An IoT architecture is considered. The aim is to establish temporal databases in a particular skyline query processing implemented in tourism booking services. The proposed approach relies on considering distributed gateways on the network connected with a local server within a cloud computing framework. Every local server manages a data warehouse with the data on different tourism services; simultaneously, each local server locally responds to the user's requests. Subsequently, the server runs a global dynamic skyline operator to collect all available local results, thus providing a suitable response based on customized selections of the client. In fact, the server of this architecture is linked to a database containing customized service preferences of multiple clients. A data mining tool is primarily used to analyze the incoming requests regarding travel services of this database. The proposed model achieves high performance; however, to realize less processing time and save resources, the best available traveling services based on the users' requests must be selected. Future works may involve parallel calculation techniques such as MapReduce at distributed local servers.

REFERENCES

[1] Maabout, S., Ordonez, C., Wanko, P.K., Hanusse, N. (2016). Skycube materialization using the topmost skyline or functional dependencies. ACM Transactions on Database Systems (TODS), 41(4): 1-40. https://doi.org/10.1145/2955092

[2] Chen, H., Chiang, R.H., Storey, V.C. (2012). Business intelligence and analytics: From big data to big impact. MIS Quarterly, 36(4): 1165-1188. https://doi.org/10.2307/41703503

[3] Boinepelli, H. (2015). Applications of big data. Big Data,

161-179. https://doi.org/10.1007/978-81-322-2494-5_7
[4] Mouha, R.A. (2021). Internet of Things (IoT). Journal of Data Analysis and Information Processing, 9(2): 77-101. https://doi.org/10.4236/jdaip.2021.92006
[5] Jadeja, Y., Modi, K. (2012). Cloud computing-concepts, architecture and challenges. In 2012 International Conference on Computing, Electronics and Electrical Technologies (ICCEET), pp. 877-880. https://doi.org/10.1109/ICCEET.2012.6203873
[6] Kertiou, I., Benharzallah, S., Kahloul, L., Beggas, M., Euler, R., Laouid, A., Bounceur, A. (2018). A dynamic skyline technique for a context-aware selection of the best sensors in an IoT architecture. Ad Hoc Networks, 81: 183-196. https://doi.org/10.1016/j.adhoc.2018.08.011
[7] Teste, O. (2010). Elaboration d'entrepôts de données complexes. arXiv preprint arXiv:1005.0220.
[8] Sagar, S.V., Balakiruthiga, B., Kumar, A.S. (2016). Novel vehicle booking system using IOT. In 2016 Online International Conference on Green Engineering and Technologies (IC-GET), pp. 1-5. https://doi.org/10.1109/GET.2016.7916811
[9] Borzsony, S., Kossmann, D., Stocker, K. (2001). The skyline operator. In Proceedings 17th International Conference on Data Engineering, pp. 421-430. https://doi.org/10.1109/ICDE.2001.914855
[10] Zaman, A., Morimoto, Y. (2016). Area skyline query for selecting good locations in a map. Journal of Information Processing, 24(6): 946-955. https://doi.org/10.2197/ipsjjip.24.946
[11] Babanejad, G., Ibrahim, H., Udzir, N.I., Sidi, F., Aljuboori, A.A.A. (2014). Finding skyline points over dynamic incomplete database. In Malaysian National Conference of Databases, pp. 60-64. http://dx.doi.org/10.13140/2.1.1270.8162
[12] Wu, J., Chen, L., Yu, Q., Kuang, L., Wang, Y., Wu, Z. (2013). Selecting skyline services for QoS-aware composition by upgrading MapReduce paradigm. Cluster Computing, 16(4): 693-706. https://doi.org/10.1007/s10586-012-0240-9
[13] Abourezq, M., Idrissi, A., Yakine, F. (2016). Routing in wireless Ad Hoc networks using the Skyline operator and an outranking method. In Proceedings of the International Conference on Internet of Things and Cloud Computing, pp. 1-10. https://doi.org/10.1145/2896387.2900333
[14] Kalyvas, C., Tzouramanis, T. (2017). A survey of skyline query processing. arXiv preprint arXiv:1704.01788.
[15] Papadias, D., Tao, Y., Fu, G., Seeger, B. (2003). An optimal and progressive algorithm for skyline queries. In Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, pp. 467-478. https://doi.org/10.1145/872757.872814
[16] Papadias, D., Tao, Y., Fu, G., Seeger, B. (2005). Progressive skyline computation in database systems. ACM Transactions on Database Systems (TODS), 30(1): 41-82. https://doi.org/10.1145/1061318.1061320
[17] Zou, L., Chen, L., Özsu, M.T., Zhao, D. (2010). Dynamic skyline queries in large graphs. In International Conference on Database Systems for Advanced Applications, pp. 62-78. https://doi: 10.1007/978-3-642-12098-5_5
[18] Vlachou, A., Doulkeridis, C., Polyzotis, N. (2011). Skyline query processing over joins. In Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data, pp. 73-84. https://doi.org/10.1145/1989323.1989332
[19] Sun, D., Wu, S., Li, J., Tung, A.K. (2008). Skyline-join in distributed databases. In 2008 IEEE 24th International Conference on Data Engineering Workshop, pp. 176-181. https://doi.org/10.1109/ICDEW.2008.4498313
[20] Raghavan, V., Rundensteiner, E.A. (2010). ProgXe: Progressive result generation framework for multi-criteria decision support queries. In Proceedings of the 2010 ACM SIGMOD International Conference on Management of data, pp. 1135-1138. https://doi.org/10.1145/1807167.1807300
[21] Jin, W., Ester, M., Hu, Z., Han, J. (2007). The multi-relational skyline operator. In 2007 IEEE 23rd International Conference on Data Engineering, pp. 1276-1280. https://doi.org/10.1109/ICDE.2007.368992
[22] Vlachou, A., Doulkeridis, C., Kotidis, Y., Vazirgiannis, M. (2007). Skypeer: Efficient subspace skyline computation over distributed data. In 2007 IEEE 23rd International Conference on Data Engineering, pp. 416-425. https://doi.org/10.1109/ICDE.2007.367887
[23] Kalyvas, C., Tzouramanis, T., Manolopoulos, Y. (2017). Processing skyline queries in temporal databases. In Proceedings of the Symposium on Applied Computing, pp. 893-899. https://doi.org/10.1145/3019612.3019677
[24] Zhang, J., Lin, Z., Li, B., Wang, W., Meng, D. (2016). Efficient skyline query over multiple relations. Procedia Computer Science, 80: 2211-2215. https://doi.org/10.1016/j.procs.2016.05.381
[25] Amiruzzaman, M., Jamonnak, S. (2020). Multi-dimensional skyline query to find best shopping mall for customers. In 2020 6th Conference on Data Science and Machine Learning Applications (CDMA), pp. 71-76. https://doi.org/10.1109/CDMA47397.2020.00018