Enabling Efficient and Authenticated Trajectory Similarity Retrieval on Blockchain-assisted Cloud

Yiping Teng✉, Haochun Pan, Jiajia Li, Yuyao Tang, Chunlong Fan, and Liang Zhao

School of Computer, Shenyang Aerospace University, Shenyang, China
{typ,lijiajia,FanCHL,lzhao}@sau.edu.cn,phc1059275935@163.com,tangyuyao@stu.sau.edu.cn

Abstract. Blockchain, valued for its decentralized and tamper-proof features, has sprung up as a predominant paradigm in numerous security applications. To facilitate massive trustworthy retrieval services, blockchain-assisted clouds, combining immutable on-chain storage and scalable off-chain storage, offer a data-reliable and storage-flexible platform. Based on the blockchain-assisted cloud model, existing studies have focused on several types of authenticated retrievals including verifiable skyline queries, graph queries, range queries, e.t.c., however, currently, few of them can directly support efficient query processing and result verification for trajectory similarity retrieval. To the best of our knowledge, this paper is the first attempt to address the problem of authenticated trajectory similarity retrieval on the blockchain-assisted cloud. To this end, we first propose novel authenticated data structures, $\mathcal{PMB}$-Tree, to enable authenticated trajectory similarity retrieval, and $\mathcal{SPMB}$-Tree, to save gas consumption on the blockchain. To reduce the client-side verification costs, we further optimized the index as $\mathcal{PMB}^*$-Tree to improve the verification efficiency. In addition, we propose a gas-efficient index maintenance for ADS updates on the blockchain. Finally, we provide a theoretical analysis of the complexity and verification guarantees of the proposed authenticated approaches and conduct extensive experiments on both real and synthetic datasets to demonstrate the efficient performance of our approaches.

Keywords: Trajectory similarity retrieval · Blockchain · Query Authentication

1 Introduction

Background and motivation In recent decades, due to the extensive applications of *Bitcoin* [8] and *Ethereum* [27], blockchain technology has gained global attention in both academic and industrial communities for its decentralized structure, cryptographic hashing, and data integrity guarantees. Such security features have expanded blockchain applications to digital copyright [18], IoT [10], and database systems [33].

With the rapid promotion of Data-as-a-Service (DaaS), to break the storage limitation and enhance trusted retrieval services, blockchain-assisted cloud

environments [1, 13, 30], combining the tamper-proof on-chain storage and scalable off-chain storage, offer a data-reliable and storage-flexible platform, which allows the large-scale data retrieval via cloud service providers and ensures integrity verification via the blockchain. Compared to the traditional verification model, in which third-party auditors are needed, the retrieval and verification on the blockchain-assisted cloud can avoid assuming the existence of trusted third parties, making it implementable in practical applications.

In existing studies, several types of authenticated retrievals based on blockchain-assisted clouds have been developed including verifiable skyline queries [26], authenticated graph queries [28], authenticated range queries [30] and verifiable keyword search [3]. As a critical supplement to the above retrieval techniques, trajectory similarity retrieval is to identify paths satisfying a similarity threshold, benefiting applications like intelligent transportation [17], personalized recommendations [25], e.t.c., whereas the soundness and completeness of retrieval results desperately need to be verified when such services provided in outsourced environments. However, to the best of our knowledge, currently, few existing works can support efficient query processing and result verification for trajectory similarity retrieval. Therefore, it is urgent to devise a solution to efficient and authenticated trajectory similarity retrieval on the blockchain-assisted cloud.

Challenge and Solution To this end, we propose an authenticated trajectory similarity retrieval approach that mainly addresses the following two challenges. The first challenge is how to create an authenticated data structure (ADS) stored on the blockchain that can achieve efficient authenticated trajectory similarity retrieval while saving gas in blockchain operations. To address the problem, we first propose a novel index structure $\mathcal{PMB}$-Tree to enable authenticated trajectory similarity retrieval, based on which we present the trajectory retrieval processing method to efficiently find similar trajectories through a threshold-based pruning and a verification object ($\mathcal{VO}$) generation method to construct $\mathcal{VO}$s on the service provider and blockchain. To reduce gas consumption, we further propose a Suppressed $\mathcal{PMB}$-Tree, named $\mathcal{SPMB}$-Tree, which suppresses the gas-consuming operations on the blockchain using smart contracts. Due to the size of $\mathcal{VO}$s directly affecting the efficiency of client-side verification, the second challenge is optimizing client-side computation, allowing clients to verify the soundness and completeness efficiently without sacrificing the quality of verification. Towards this, based on the proposed verification method for trajectory similarity retrieval on the client side, to effectively reduce the verification costs, we further optimized the index as $\mathcal{PMB}^*$-Tree to scale down the size of $\mathcal{VO}$s thereby improving the efficiency of verification processing. In addition, we further propose a gas-efficient index maintenance method to save gas consumption on the blockchain when incrementally updating the ADS. The verification guarantees are analyzed based on the soundness and completeness of the proposed authenticated approach, and extensive experiments on both real and synthetic datasets demonstrate the efficient performance of our approaches. Our contributions can be summarized as follows:

– To the best of our knowledge, this is the first attempt to define and solve the problem of authenticated trajectory similarity retrieval on the blockchain-assisted cloud.
– To achieve efficient and authenticated trajectory similarity retrieval, we first propose a novel ADS $\mathcal{PMB}$-Tree supporting query processing and $\mathcal{VO}$ generation based on the blockchain as well as a $\mathcal{SPMB}$-Tree to reduce the gas consumption of blockchain operations.
– To optimize the performance, we further propose an optimized index $\mathcal{PMB}^*$-Tree to improve the efficiency of retrieval and verification with lower gas consumption, besides, a gas-efficient index maintenance method is designed to facilitate index updates on the blockchain.
– We analyze the verification guarantees of the proposed approach and conduct extensive experiments on both real and synthetic datasets to evaluate the performance of our approach.

The remainder of the paper is organized as follows: Section 2 defines the problem, and Section 3 covers preliminary concepts. Section 4 introduces our authenticated trajectory similarity retrieval approach, including ADS construction, retrieval processing, $\mathcal{VO}$ generation, and result verification. In Section 5, we introduce the optimization and index maintenance. Section 6 provides analysis, Section 7 presents experimental results, and Section 8 reviews related work. Finally, Section 9 concludes the paper.

2 Problem Definition

2.1 Definitions

Definition 1. ***(Trajectory)*** *A trajectory T_j is defined as a sequence of points generated from a moving object (shown in Fig. 1), denoted as $(t_1^j, t_2^j, ..., t_n^j)$, where each point t_i^j is a d-dimensional tuple.*

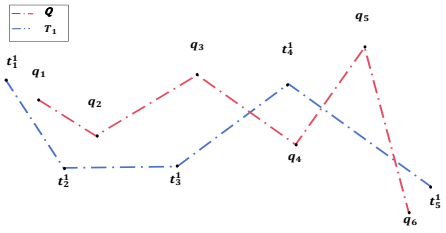


Fig. 1: Sample of Trajectories

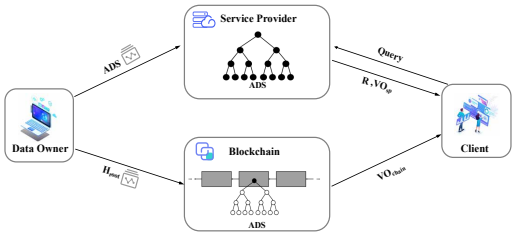


Fig. 2: System Model

In this paper, as one of the most robust and widely adopted similarity functions, we apply Fréchet distance to measure the similarity of trajectories.

Definition 2. ***(Fréchet Distance Similarity)*** *[4, 16, 24] Given trajectories $T = (t_1, ..., t_m)$ and $Q = (q_1, ..., q_n)$, Fréchet distance is defined as Equation 1.*

$$F(T,Q)=\begin{cases}\max\limits_{i=1}^{m} dist(t_i,q_1) & if \quad n=1\\ \max\limits_{j=1}^{m} dist(t_1,q_j) & if \quad m=1\\ \max(dist(t_m,q_n),\min(F(T^{m-1},Q^{n-1}),F(T^{m-1},Q),F(T,Q^{n-1}))) & otherwise\end{cases} \tag{1}$$

where T^{m-1} donates the prefix trajectory of T by removing the last point, and $dist(t_m, q_n)$ is the point-to-point distance between t_m and q_n (Euclidean distance is applied in this paper).

Definition 3. ***(Trajectory Similarity Retrieval)*** *Given query trajectory Q, trajectory set $T = \{T_1, T_2, \ldots, T_i, \ldots\}$ with ID of T_i being i, distance function f and the similarity threshold τ, the trajectory similarity retrieval is to find all trajectories $T_i \in T$, such that $F(T_i, Q) \leq \tau$.*

Note that the proposed approach can support more distance functions than Fréchet distance (e.g., DTW [11], LCSS [16], EDR [2]), of which the parameters and thresholds are analyzed in Section 7.

Definition 4. ***(Authenticated Trajectory Similarity Retrieval)*** *The results obtained through the authenticated trajectory similarity retrieval can be verified by meeting the following requirements:*

- *All results come from the data owner and have not been tampered with.*
- *Results include all the data that meet the query conditions in the database without omission.*

2.2 System Model

Following [30], our system consists of four parties: the data owner ($\mathcal{DO}$), the blockchain, the service provider ($\mathcal{SP}$), and the query client (shown in Fig. 2).

- **DO:** $\mathcal{DO}$ stores all queryable trajectories and constructs an authenticated data structure (ADS) locally. The ADS is sent to $\mathcal{SP}$ for storage, enabling the authenticated retrieval, while the hash values of trajectories H_{root} are stored on the blockchain.
- **SP:** $\mathcal{SP}$ maintains the ADS for all the trajectories. Upon receiving a client's request, $\mathcal{SP}$ processes the retrieval and returns the results along with the verification object ($\mathcal{VO}_{SP}$) to the client.
- **Blockchain:** The blockchain stores the ADS hash values H_{root} and all trajectory data hashes uploaded by $\mathcal{DO}$. During data verification, it provides the client with the generated verification object ($\mathcal{VO}_{chain}$).
- **Client:** Upon receiving results from $\mathcal{SP}$ for request Q, the client uses $\mathcal{VO}_{SP}$ and $\mathcal{VO}_{chain}$ to verify the soundness and completeness of the results. The client only obtains the required trajectory datasets and a $\mathcal{VO}$ with partial trajectory information.

2.3 Threat Model

Our model assumes that $\mathcal{DO}$, the blockchain, and query clients are trusted. At the same time, due to issues like program failures, security vulnerabilities, commercial interests, etc., $\mathcal{SP}$, as a third party, may try to tamper with the data and, therefore, be treated as an untrusted party [30]. Hence, $\mathcal{SP}$ needs to prove that the results of the similarity retrieval should meet the following two conditions:

- **Soundness:** All trajectories in the results originate from $\mathcal{DO}$, have not been tampered with, and meet the query constraints.
- **Completeness:** All trajectories in the results of the authenticated similarity retrieval are returned without invalid answers.

3 Preliminaries

3.1 Cryptographic Hash Function

Cryptographic hash functions [9] map message m of any length to digest $h(m)$ of a fixed size. SHA-256 is one of the common hash algorithms, with its final result being a 256-bit hash value. This paper uses a cryptographic hash function because it is one-way and collision-resistant.

- **One-way Property:** Given a message m, it is easy to calculate $h(m)$, but it is not feasible to get m from $h(m)$.
- **Collision-Resistance Property:** It is difficult to make $h(m_1) = h(m_2)$ when entering two different values m_1 and m_2.

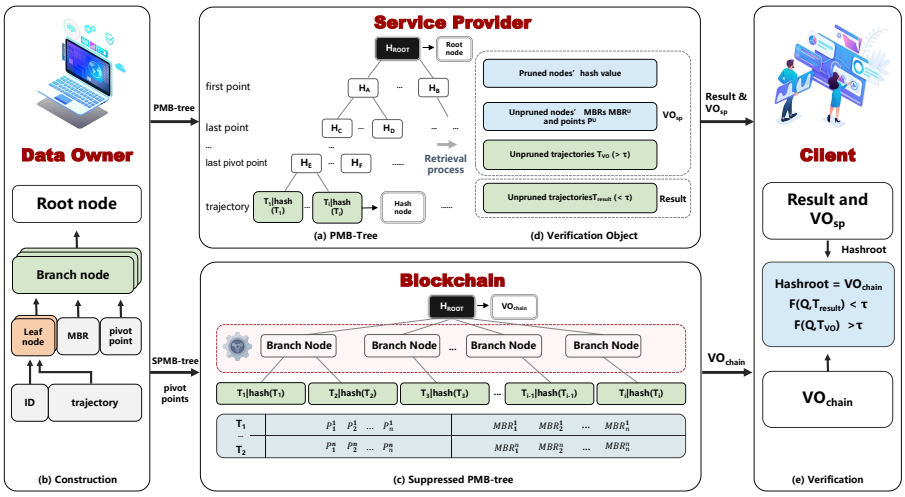


Fig. 3: Complete workflow of authenticated trajectory similarity retrieval, where T_{result} and $T_{\mathcal{VO}}$ represent the trajectories in the results, and the trajectories added to $\mathcal{VO}_{sp}$, respectively.

3.2 Blockchain-based Merkle Hash Tree

Blockchains [8,19,27], e.g., Ethereum, utilize smart contracts and self-executing programs based on consensus protocols to enable automated and trustworthy applications while maintaining blockchain integrity, in which each block contains transaction data and references the previous block's hash, forming an immutable chain that ensures data security, transparency, and immutability. As a key cryptographic structure within each block, Merkle Hash Tree (MHT) [14, 15] is a binary tree where each leaf node holds the hash of a data block and non-leaf nodes contain the hash or concatenation of their child nodes' hashes. The root node of MHT represents the hash of the entire dataset and facilitates efficient data integrity verification, allowing for the validation of individual transactions without recalculating the whole block. Therefore, a blockchain-based MHT is designed as the ADS in this work. Due to space limitations, we only briefly introduce the preliminaries of blockchain and MHT, which can be referred to in the references for details.

4 APPROACH

4.1 $\mathcal{PMB}$-Tree

To achieve efficient authenticated trajectory similarity retrieval, the index structure was designed with two key goals: effective trajectory pruning and integrity verification of all the trajectories. In this context, we propose a novel authenticated index structure called Pivot-based Merkel B-Tree ($\mathcal{PMB}$-Tree), in which *pivot* refers to critical points within a trajectory that are selected for approximate computations due to their significant influence. Such pivots are identified through various analytical methods and are assigned higher weights, reflecting their importance in the overall calculation process.

To achieve the first goal above, inspired by [20], we designed a B-Tree structure that allows for precise index pruning through the pivot calculation and trajectory partition in sequence. Starting with a dummy root node, trajectories in each partition are grouped into N groups based on the first indexing point, and the Minimum Bounding Rectangle (MBR) of each group becomes a child of the root. Within each MBR, trajectories are further grouped by a second indexing point, generating a (K+2)-level tree where K represents the number of pivot points. All trajectories are stored in the leaf MBRs. Fig.3(a) illustrates the final index structure, with a sample shown in Fig. 4.

Table 1: Trajectory Sample

ID	Points	Pivot Points
T_1	$(1,1),(2,2),(3,4),(4,5.5),(6,7.2)$	$(2.0,2.0)$
T_2	$(5.2,4.3),(4,3),(3.5,1.8),(2,1),(1,0)$	$(2.0,1.0)$
T_3	$(5,2),(1.2,4.1),(2,3.2),(3.5,2),(4.2,1)$	$(2.0,3.2)$
T_4	$(7.1,2),(5,3),(4,5.5),(3,6.2),(2,8)$	$(3.0,6.2)$
T_5	$(1,2),(2,3.3),(3.2,3),(4,4.5),(5.5,6)$	$(4.0,4.5)$
Q	$(1,1),(2,2),(3,3),(4,4),(5,5)$	None

To support the authentication, it is imperative to ensure that all the trajectories can be accommodated in the Authenticated Data Structure (ADS).

To achieve this, we connect the trajectory ID with the coordinates and calculate the hash value, i.e., $H(leafnode) = H((Trajectoryid)||(t_i^1)||...||(t_i^j))$, and then connect it with the pointer pointing to the trajectory, which is the leaf node of the hash tree. For the branch node, we connect its child nodes, i.e., MBRs, and pivot points, and calculate the hash value, i.e., $H(Branch) = H(H(child_1)||...||H(child_i)||MBR||(pivotpoints))$. The hash value of the root node is calculated as $H(Root) = H(H(child_1)||..||H(child_i))$ (illustrated in Fig.3(b)). During index construction, trajectories' start and end points are placed on the first and second levels. We create multiple identical indexes to store different trajectories for ensuring efficiency, where these indexes exist within a particular order and their hash values will be stored sequentially in the blockchain.

Only the root hash $H(Root)$ is used during verification. Since the cost of writing or updating operations in the blockchain is much higher than writing in smart contracts, to reduce gas consumption, we suppress all nodes of the $\mathcal{PMB}$-Tree and only materialize the root node in the blockchain. In this way, there is no need for smart contracts to perform data pruning operations repeatedly. We name this index structure Suppressed $\mathcal{PMB}$-Tree ($\mathcal{SPMB}$-Tree). Shown as Fig.3(c), if we need to reconstruct the index to the blockchain storage and update pivot points, the smart contract will compute all nodes of the $\mathcal{SPMB}$-Tree on the fly and only update the root hash, which means that the parts in the red box will not be written into the blockchain. Note that the $\mathcal{PMB}$-Tree in the $\mathcal{SP}$ is maintained similarly but not suppressed.

4.2 Retrieval Processing and $\mathcal{VO}$ Generation

We propose an authenticated trajectory similarity retrieval method based on the approximate distance calculation from query trajectories to achieve the goal of verification of trajectories. It involves pruning and calculation processes of the verifiable trajectory similarity retrieval index by approximate calculation of the distance from query trajectories and the generation process of the verification object by recalculating the distance of trajectories.

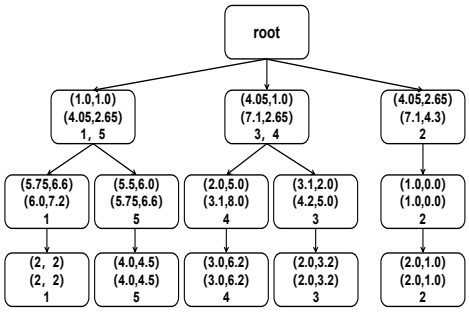


Fig. 4: $\mathcal{PMB}$-Tree sample

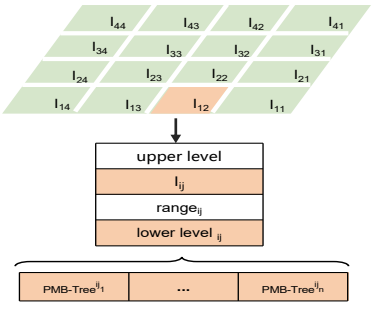


Fig. 5: $\mathcal{PMB}^*$-Tree

The pruning process aims to eliminate trajectories beyond a specified threshold distance from any point on the query trajectory. Suppose the distance between the MBRs covering pivot points on the trajectory and corresponding

points on the query trajectory exceeds this threshold. In that case, the trajectories in these MBR-linked nodes are pruned. Given the ordered nature of trajectory points, calculations start with points in query trajectories aligned to the trajectory's sequence. Unpruned trajectories are further evaluated for similarities by $\mathcal{SP}$. Algorithm 1 details the retrieval processing: The index pruning is covered by lines 1-20, while the result calculation is handled by lines 21-29, with MBR_f, MBR_l, and MBR_p representing the MBRs for the first point, last point and pivot point, respectively. And T_p^U, MBR^U, and P^U respectively represent the unpruned trajectories, MBRs of unpruned trajectories, and pivot points of unpruned trajectories. Regarding the generation of verification objects, the in-

Algorithm 1: Retrieval and Verification Object Generation

Input : Dataset T, Query Q, $\mathcal{PMB}$-Tree, Threshold τ
Output: ResultSet of Trajectory Similarity Retrieval R, Verification Object $\mathcal{VO}_{sp}$

1 **if** *node is a LeafNode* **then**
2 | Insert trajectories T_p^U, MBR^U, and pivot points P^U into $\mathcal{VO}_{sp}$;
3 **if** *node is the RootNode* **then**
4 | **foreach** *child* MBR_f *at the first level* **do**
5 | | **if** $\text{dist}(q_1, MBR_f) \leq \tau$ **then**
6 | | | Insert P^U and MBR_f into VO_{sp};
7 | | **else**
8 | | | Insert `node.Hash` and MBR_f into $\mathcal{VO}_{sp}$;
9 **if** *node is at the FirstLevel* **then**
10 | **foreach** *child* MBR_l *at the second level* **do**
11 | | **if** $\text{dist}(q_n, MBR_l) \leq \tau$ **then**
12 | | | Insert P^U and MBR_l into VO_{sp};
13 | | **else**
14 | | | Insert `node.Hash`, and MBR_l into $\mathcal{VO}_{sp}$;
15 **else if** *node is at levels* $x \geq 3$ **then**
16 | **foreach** *child* MBR_p **do**
17 | | **if** $\text{dist}(Q, MBR_p) \leq \tau$ **then**
18 | | | Insert P^U and MBR^U into VO_{sp};
19 | | **else**
20 | | | Insert `node.Hash` , q_i and MBR^P into $\mathcal{VO}_{sp}$;
21 **foreach** *trajectory* $T_{pi}^U \in T_p^U$ **do**
22 | Compute $F(T_{pi}^U, Q, \tau)$;
23 | **if** $F(T_{pi}^U, Q, \tau) \leq \tau$ **then**
24 | | $T_{result} \leftarrow T_{pi}^U$;
25 | | Insert T_{result} into R;
26 | **else**
27 | | $T_{VO} \leftarrow T_{pi}^U$;
28 | | Insert T_{VO} into $\mathcal{VO}_{sp}$;
29 **return ResultSet** (**R**) and $\mathcal{VO}_{sp}$;

dex nodes that do not meet the threshold requirements will be pruned, and then the hash value and MBRs of pruned trajectories' pivot points will be added to the $\mathcal{VO}_{sp}$ (lines 8,14, and 20). The pivot points P^U and MBR^U, the MBRs of unpruned trajectories T_p^U will be added to the $\mathcal{VO}_{sp}$ (lines 5, 12, and 18). After calculating the trajectory similarity, the trajectories that meet the threshold requirements T_{result} are added to the result set (line 25), while the trajectories that do not meet the criteria T_{VO} are added to $\mathcal{VO}_{sp}$ (line 28). The composition of $\mathcal{VO}_{sp}$ and the results are shown in Fig.3(d).

In Table 1, given threshold $\tau = 1.2$ and query trajectory q, we start by calculating the minimum distance between q's start point and nodes in the first tree layer. T_3, T_4, and T_2 are pruned because their similarities exceed the threshold, with their hash values added to $\mathcal{VO}_{sp}$. In the second layer, we use q's ending point for evaluation. All nodes in this layer fall within the threshold, so no pruning occurs. In the third layer, determining the exact closest point is computationally impractical, so we check whether any point on q, other than the start or end point, meets the threshold. T_1 fails to meet the threshold and is pruned, with its hash value added to $\mathcal{VO}_{sp}$. Nodes that satisfy the threshold have their pivot points and MBRs added to $\mathcal{VO}_{sp}$. Finally, T_5 undergoes further similarity evaluation by $\mathcal{SP}$.

Algorithm 2: Client Verification Process

Input : Threshold τ, Query Trajectory Q, Result Set R, Verification Object VO_{sp}, Hash Value in Blockchain VO_{chain}
Output: Boolean

1 $root.Hash \leftarrow$ `reconstruct`$(\mathcal{VO}_{sp}, R)$;
2 $DO_{root} \leftarrow \mathcal{VO}_{chain}$;
3 **if** $root.Hash \neq DO_{root}$ **then**
4 | **return False**;
5 **if** $\text{dist}(q_1, MBR_f) \geq \tau$ ***or*** $\text{dist}(q_n, MBR_l) \geq \tau$ ***or*** $\text{dist}(q_i, MBR^p) \geq \tau$ **then**
6 | **return False**;
7 **foreach** *trajectory* T_i ***in*** R **do**
8 | **if** $F(T_i, Q) \geq \tau$ **then**
9 | | **return False**;
10 **foreach** *trajectory* T_j ***in*** $\mathcal{VO}_{sp}$ **do**
11 | **if** $F(T_j, Q) \leq \tau$ **then**
12 | | **return False**;
13 **return True**;

4.3 Client Verification

To effectively support the blockchain-assisted authentication of trajectory similarity retrieval, we present a result verification method, outlined in Fig.3(e) and Algorithm 2. During the verification stage, the client utilizes $\mathcal{VO}_{sp}$ and $\mathcal{VO}_{chain}$ to verify the soundness and completeness of the retrieval results. The client reconstructs the $\mathcal{PMB}$-Tree based on the $\mathcal{VO}_{sp}$ and R sent by $\mathcal{SP}$, along with the hash table about the index composition sent by $\mathcal{DO}$ to the client (line 1). Since $\mathcal{PMB}$-Tree does not have the same number of child nodes in each node, the hash table can enable the client to reconstruct the root hash value correctly.

Next, upon receiving the hash value of the reconstructed root node, the client compares $\mathcal{VO}_{chain}$ received from the blockchain with the hash value he/she calculates(line 2). If they are matched, soundness verification is completed, and integrity verification continues. Otherwise, it indicates that the results are tampered with (lines 3 and 4). After that, the client recalculates the distance of pruned trajectories' MBRs and corresponding points in query trajectories to verify whether they are pruned. Then, the client calculates the similarity between each trajectory in the result set and the query trajectory to ensure that all results meet the requirements (lines 7-9). Finally, the client continues calculating the similarities of the trajectories pruned in $\mathcal{VO}_{sp}$ to ensure no missing results (lines 10-12). It is worth noting that when there are multiple $\mathcal{PMB}$-Trees, we will compare them sequentially with the hash values sent by the blockchain. If all the hash values are matched, the result is deemed correct. Otherwise, any inconsistency indicates a potential integrity issue with the results.

5 Optimization and Index Maintenance

5.1 $\mathcal{PMB}^*$-Tree: an Optimized Index

To optimize retrieval efficiency and reduce gas consumption, we further designed $\mathcal{PMB}^*$-Tree. The basic structure of the $\mathcal{PMB}^*$-Tree is a two-level index (as shown in Fig. 5). In the upper level, the domain is split by MBRs that cover the start points of partial trajectories into several regions I_{11}, I_{12}, I_{13}, etc. This split is based on the underlying data distribution to improve performance, aiming for a balanced distribution of keys in each region I_{ij}. In the lower level, multiple $\mathcal{PMB}$-Trees are built for each I_{ij}. The $\mathcal{PMB}$-Tree constructed here differs slightly from a standalone $\mathcal{PMB}$-Tree. In the original $\mathcal{PMB}$-Tree, trajectories are partitioned from their start points. However, since the start points are now used to construct the second-level index, the new $\mathcal{PMB}$-Tree begins partitioning from the ending points of trajectories, thus bypassing the initial partitioning step from the origin. As the pruning efficiency of this index depends on the index height, further improvements require $\mathcal{DO}$ to compute an additional pivot point for each trajectory, which is then integrated into the new $\mathcal{PMB}$-Tree. Moreover, reducing the maximum number of trajectories in each $\mathcal{PMB}$-Tree results in smaller trees, enhancing retrieval and verification efficiency, and reducing gas consumption. Similar to $\mathcal{SPMB}$-Tree, to save gas, we suppress the branch nodes of the lower-level index in the $\mathcal{PMB}^*$-Tree and store only the root hash and leaf nodes on the blockchain sequentially, calling this index $\mathcal{SPMB}^*$-Tree.

5.2 Gas-efficient Index Maintenance

Inspired by [30], the maintenance operations for the $\mathcal{PMB}$-Tree encompass three primary functions: (i) insertion, (ii) update, and (iii) deletion. Deletion is conceptualized as updating the data object with a virtual object, exemplified by changing trajectory points to negative numbers. Moreover, modifying a single point on a trajectory is akin to updating the entire trajectory. Consequently, in index maintenance, we only focus on the insertion and updating operations.

Insertion. When a new trajectory object arrives, the feature points (i.e., the start point, ending point, and pivot point) are calculated. The trajectory is then directed to the most recent index that can still accommodate it, requiring $\mathcal{DO}$ to generate a data count table for each index. Placement within the index depends on whether the feature points can be reasonably inserted into a specific node. Following previous principles, if the trajectory's feature points fall within a node's MBR, they are inserted into that node. This insertion proceeds through child nodes until reaching a leaf, after which hash values along the path are recalculated. If conditions are not met, a full index reconstruction is required. If no suitable index exists, the trajectory serves as the foundation for a new index.

Update. In contrast to insertion, the update operation replaces an existing trajectory with a new one, requiring recalculating the new trajectory's feature points. Similar to insertion, it is necessary to check if these feature points fall within the MBR of a node in the index. If they do, the update proceeds; otherwise, index reconstruction is needed. Unlike insertion, updating does not require locating the most recent index, but rather quickly finding the position of the trajectory to be replaced. For this, $\mathcal{DO}$ generates a hash table mapping each trajectory's position to enable fast localization during updates.

The maintenance of $\mathcal{PMB}^*$-Tree is the same as $\mathcal{PMB}$-Tree, in addition to the need to calculate whether the upper-level index meets the insertion requirements when updating and inserting objects.

6 Analysis

6.1 Computational Complexity

$\mathcal{PMB}$-tree construction and trajectory similarity retrieval. To construct a $\mathcal{PMB}$-Tree, the computational complexity is $O(n \log n)$, where n is the number of leaf nodes, as its construction processing is similar to a B-tree, of which each data point insertion takes $O(\log n)$. For the time complexity of the similarity retrieval, like searching within a B-tree, it takes $O(\log n)$ to retrieve a $\mathcal{PMB}$-Tree. For the similarity calculation, assuming m is the number of candidate trajectories after pruning, where $m \ll n$, the complexity of the similarity calculation is $O(m^3)$. Thus, the overall time complexity for the trajectory similarity retrieval is $O(\log n + m^3)$.

Result verification. In the soundness verification, the client reconstructs an MHT from the $\mathcal{VO}$ with a typical computational complexity of $O(m \log n)$, where m is the number of trajectories in $\mathcal{VO}$ and the search results. To ensure completeness, the client recalculates the similarities between the trajectories in the results and those excluded from $\mathcal{VO}$, which has an average complexity of $O(m^3)$. Thus, the overall time complexity of the result verification is $O(m^3 + m \log n)$.

6.2 Verification Analysis

Definition 5. ***(Verification Security)*** *Assuming that adversary $\mathcal{A}$ who owns $\mathcal{VO}_{chain}$ based on dataset $\mathcal{D}$ outputs a query trajectory Q, result R and $\mathcal{VO}_{sp}$, $\mathcal{A}$ succeeds if $\mathcal{VO}_{sp}$ passes the verification w.r.t. $\mathcal{VO}_{chain}$ and satisfies the following condition: $\{T_i \| T_i \notin \hat{R} \cap T_i \in R\} \neq \emptyset \vee \{T_j \| T_j \in \hat{R} \cap T_j \notin R\} \neq \emptyset$, where $\hat{R}$ donates the real results.*

Definition 5 states that malicious $\mathcal{SP}$ could convince the user of an incorrect or incomplete answer with a negligible probability. Our proposed authenticated retrieval approach now satisfies the desired security requirement.

Theorem 1. *The result of our proposed authenticated retrieval based on the $\mathcal{PMB}$-Tree is secure [30], i.e., sound and complete if the underlying hash function resists collision.*

Proof. We prove this theorem by contradiction.

- **Soundness.** Assuming $\{T_i \| T_i \notin \hat{R} \cap T_i \in R\} \neq \emptyset$, it means that an object in R does not originate from $\mathcal{D}$. Since the client will reconstruct the hash root of the $\mathcal{PMB}$-Tree in which T_i lies and compare it against the hash root in $\mathcal{VO}_{chain}$, such a tampered result means that two different trajectories resulted in the same hash value. Due to the collision resistance property of hash values, this scenario is impossible.
- **Completeness.** Assuming $\{T_i \| T_i \in \hat{R} \cap T_i \notin R\} \neq \emptyset$, it means that a valid answer is missing from R. When the client verifies whether trajectories are pruned by the distance between the MBRs of pruned trajectories' pivot points and their corresponding query points, the trajectory similarity of unpruned trajectories is also recalculated to ensure integrity. Any missing correct results will cause some $\mathcal{VO}$s to fail verification or encounter hash collisions, thereby invalidating the assumption. □

7 Experiments

7.1 Experimental Setups

Setup. We established a private Ethereum-based blockchain using Geth (https://geth.ethereum.org/) and implemented the proposed approach in smart contracts deployed on it. Experiments were conducted on a Dell Precision 7920 Tower Server with dual 40-core Intel Xeon Bronze 3.40 GHz CPUs, 256 GB RAM, and Ubuntu 20.04. SHA-256 was utilized as the hash function, and Fréchet distance served as the default distance metric. The fanout of the $\mathcal{PMB}$-Tree in the proposed ADS was set to 16.

Dataset. We used 3 real trajectory datasets (SHH-Taxi [12], T-drive [31], and GeoLife [32]) and 2 synthetic datasets that followed the Normal distribution ($N(0, 0.2)$) and the range of the walk was determined by the maximum of the distance threshold D_{Smax} and D_{Tmax} in our experiments (Uniform1 and Uniform2), in which the trajectories containing single or no sampling point were removed. The details of the datasets are shown in Table 2.

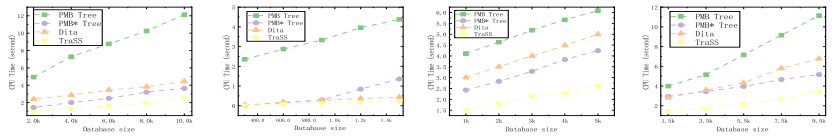
Table 2: Trajectory Datasets

Dataset	# of Traj.	Avg # of Pts	Max # of Pts	Min # of Pts
T-drive	10,000	100	166	52
S-taxi	6,000	100	142	38
Uniform1	1,000	50	79	28
Uniform2	1,000,000	50	85	27
GeoLife	11,000	100	101	33

Parameter Setting. For each trajectory dataset, parameters were varied individually, keeping others at default values. A trajectory similarity threshold of 0.001 (about 111 meters on Google Earth) was chosen. 10 trajectories were extracted from each dataset as the query trajectories, each group of experiments was repeated 20 times and the results were reported on average. Except for Fréchet distance, the proposed approach also supported DTW [11], LCSS [16], and EDR [2], with threshold adjustments impacting pruning strategies. For LCSS, index building considered trajectory length to ensure size differences between trajectories remain within acceptable limits. To evaluate the performance of the proposed approaches, we employed Dita [20] and TraSS [6] as the compared approaches. Note that GEM^2-Tree was presented as a related authenticated approach for range queries in [30]. Since GEM^2-Tree could hardly solve the trajectory retrieval problem studied in this paper directly, we would not compare the performance of GEM^2-Tree with that of our approaches.

7.2 Experimental Evaluation

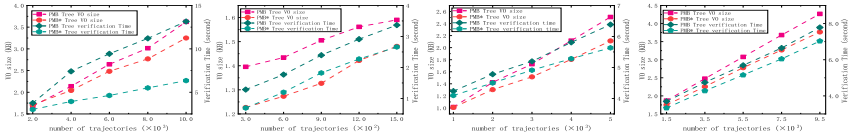
Query performance. Fig. 6 shows the query response time of the four compared approaches, i.e., $\mathcal{PMB}$-Tree, $\mathcal{PMB}^*$-Tree, Dita, and TraSS. As the number of trajectories in the datasets increases, the query response time rises accordingly. Among the four datasets, the proposed approach based on the $\mathcal{PMB}^*$-Tree outperforms both the $\mathcal{PMB}$-Tree and Dita, exhibiting a more stable query response time. This query performance is attributed to the smaller size of $\mathcal{PMB}^*$-Tree index and enhanced pruning efficiency. Compared to TraSS, the query response time of the proposed approaches is slightly longer, since $\mathcal{PMB}^*$-Tree has incurred more costs to achieve verifiable queries than TraSS which cannot support query result verification.



(a) SHH-Taxi (b) T-drive (c) Uniform (d) GeoLife

Fig. 6: Query response time, varying the number of trajectories

$\mathcal{VO}$ Size. Fig. 7 (the red lines) shows the size of $\mathcal{VO}$ generated by our proposed $\mathcal{PMB}$-Tree and $\mathcal{PMB}^*$-Tree. As the size of the dataset grows, the size of $\mathcal{VO}$ increases. This is because a larger dataset will increase the number of trajectories to be verified. The results show that compared to the $\mathcal{PMB}$-Tree, our $\mathcal{PMB}^*$-Tree can significantly reduce the size of $\mathcal{VO}$s.

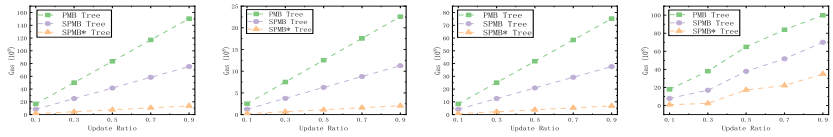


(a) SHH-Taxi (b) T-drive (c) Uniform (d) GeoLife

Fig. 7: $\mathcal{VO}$ size and verification time, varying the number of trajectories

Result Verification Performance. Fig.7 (the green lines) shows the verification time of $\mathcal{PMB}$-Tree and $\mathcal{PMB}^*$-Tree approaches when varying the number of the trajectories in the four datasets. The results show that the verification time becomes longer following the increasing number of trajectories. Compared to $\mathcal{PMB}$-Tree, the $\mathcal{PMB}^*$-Tree solution performs better in terms of verification efficiency since the smaller index leads to a better pruning effect and reduces the amount of content that needs to be verified.

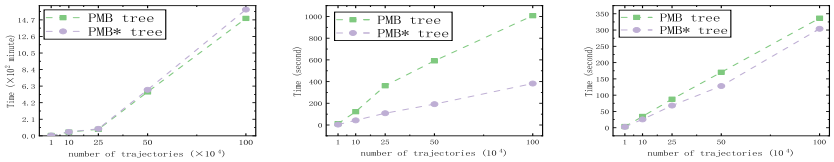
Gas Consumption for ADS Maintenance. In the experiments of ADS maintenance, trajectories were randomly inserted, updated, or deleted, with the update ratio defined as the proportion of updated trajectories in the dataset. Fig. 8 evaluates gas consumption for index maintenance across $\mathcal{PMB}$-Tree, $\mathcal{SPMB}$-Tree and $\mathcal{SPMB}^*$-Tree. As the update ratio increases, gas consumption rises for all datasets. $\mathcal{SPMB}^*$-Tree demonstrates a lower gas consumption than $\mathcal{PMB}$-Tree and $\mathcal{SPMB}$-Tree, attributed to its smaller size, which minimizes costly write or update operations on blockchain compared to memory operations.



(a) SHH-Taxi (b) T-drive (c) Uniform (d) GeoLife

Fig. 8: Gas Consumption, varying update ratio

Scalability. To evaluate the scalability of the proposed approaches, we conducted experiments using the dataset Uniform2, focusing on the scalability of index construction, similarity retrieval, and result verification. As shown in Fig. 9(a), results indicate that as the number of trajectories grows, the index's construction efficiency decreases, leading to approximately linearly increasing query and verification times. Comparing the $\mathcal{PMB}$-Tree and $\mathcal{PMB}^*$-Tree (Fig. 9(b) and Fig. 9(c)), we observed that $\mathcal{PMB}$-Tree had longer query and verification times due to the lower pruning efficiency, whereas $\mathcal{PMB}^*$-Tree achieved better pruning at the expense of higher construction time due to its two-tier structure, enhancing query efficiency. Both structures maintained stable performance up to 250,000 records but showed marked efficiency decline beyond this threshold.



(a) Index construction time (b) Query response time (c) Verification time

Fig. 9: Scalability, varying the number of trajectories

8 Related Work

To our knowledge, this is the first study to address authenticated trajectory similarity retrieval using blockchain. Related work can be categorized into blockchain-based authenticated queries and trajectory similarity computation.

8.1 Blockchain-based Authenticated Queries

Renowned for its tamper resistance and traceability, the blockchain technique has become a cornerstone in privacy protection, storage structures, system stability, and authenticated query mechanisms [26,28–30], gaining significant attention from both academic communities and commercial applications. Regarding blockchain as a fully trusted third-party audit institution, Zhang et al. [30] proposed a blockchain-based hybrid storage architecture. In this architecture, only a small amount of metadata is stored on-chain, while raw data are outsourced to off-chain storage. Wang et al. [23] introduced vChain, a blockchain system designed to ensure the integrity of the query. vChain empowers a lightweight user to authenticate query results returned from a potentially untrusted service provider. Building on this foundation, Wang et al. [26] addressed the authentication of location-based skyline queries concerning privacy preservation, dynamism, and encryption. Xu et al. [29] extended the investigation to the authenticated STK transaction queries under this architecture to improve the verification efficiency.

8.2 Trajectory Similarity Computation

Trajectory similarity computation has been extensively studied over the past several decades with significant advances in accuracy and efficiency. Among these methods, Dynamic Time Warping (DTW) gained attention for its ability to measure similarity by aligning sequences with nonlinear variations in time, as discussed in [11,20]. DTW excels at handling temporal misalignments caused by varying speeds, making it effective for dynamic sequence analysis, though it is sensitive to noise and outliers. Fréchet distance [5] was a pathway space similarity measure proposed by Maurice René Fréchet in 1906. The Longest Common Subsequence (LCSS) method focuses on identifying the longest matching subsequence between two trajectories, providing a similarity measure that is resilient to noise and outliers, as highlighted in [21, 22]. LCSS is particularly robust due to its tolerance for mismatched points, though its performance relies heavily on the choice of spatial and temporal matching thresholds. Edit Distance in Real Sequence (EDR) [2] builds on the edit distance concept by introducing insertion, deletion, and substitution operations to measure trajectory similarity. EDR is well-suited for handling partial mismatches and data with discrete gaps but depends on parameter tuning. Edit Distance with Projections (EDwP) [7], an extension of EDR, incorporates point projection into spatial dimensions, capturing complex geometric alignments effectively. However, EDwP often entails higher computational costs than simpler methods like EDR.

9 Conclusion

In this paper, we investigate the problem of authenticated trajectory similarity retrieval on the blockchain-assisted cloud, and present an authenticated retrieval approach supporting efficient retrieval processing, high-performance result verification, and gas-efficient maintenance. In particular, we first propose $\mathcal{PMB}$-Tree and $\mathcal{SPMB}$-Tree to enable authenticated trajectory similarity retrieval with gas-saving on the blockchain. Furthermore, we design an optimized index $\mathcal{PMB}^*$-Tree to improve the verification efficiency. Besides, we propose a gas-efficient index maintenance on the blockchain. The provided theoretical analysis of complexity and verification guarantees and the reported experimental results

demonstrate the efficiency and viability of the proposed approaches. For future work, we plan to study efficient, verifiable, and secure trajectory similarity retrieval on blockchain-assisted clouds under a fully malicious security model.

10 Acknowledgement

The work was partially supported by the Young and Middle-aged Science and Technology Innovation Talent Support Plan of Shenyang (RC230832), the Scientific Research Project of the Education Department of Liaoning Province (JYTMS20230272), and the Key Research and Development Program of Liaoning Province (2023JH26/10300022). Yiping Teng is the corresponding author.

References

1. Ayoade, G., Karande, V., Khan, L., Hamlen, K.W.: Decentralized iot data management using blockchain and trusted execution environment. In: IRI. pp. 15–22. IEEE (2018)
2. Chen, L., Özsu, M.T., Oria, V.: Robust and fast similarity search for moving object trajectories. In: SIGMOD Conference. pp. 491–502. ACM (2005)
3. Cui, N., Wang, D., Li, J., Zhu, H., Yang, X., Xu, J., Cui, J., Zhong, H.: Enabling efficient, verifiable, and secure conjunctive keyword search in hybrid-storage blockchains. IEEE Trans. Knowl. Data Eng. **36**(6), 2445–2460 (2024)
4. Ding, H., Trajcevski, G., Scheuermann, P., Wang, X., Keogh, E.J.: Querying and mining of time series data: experimental comparison of representations and distance measures. Proc. VLDB Endow. **1**(2), 1542–1552 (2008)
5. Fréchet, M.: Sur quelques points du calcul fonctionnel. Rendiconti del Circolo Matematico di Palermo (1884-1940) **22**, 1–72 (1906)
6. He, H., Li, R., Ruan, S., He, T., Bao, J., Li, T., Zheng, Y.: Trass: Efficient trajectory similarity search based on key-value data stores. In: ICDE. pp. 2306–2318. IEEE (2022)
7. Hu, D., Chen, L., Fang, H., Fang, Z., Li, T., Gao, Y.: Spatio-temporal trajectory similarity measures: A comprehensive survey and quantitative study. IEEE Trans. Knowl. Data Eng. **36**(5), 2191–2212 (2024)
8. Hunt, A.D.: Bitcoin: A peer-to-peer electronic cash system (2008)
9. Kasselman, P.: Analysis and design of cryptographic hash functions (1999)
10. Khan, S., Lee, W., Hwang, S.O.: Aechain: A lightweight blockchain for iot applications. IEEE Consumer Electron. Mag. **11**(2), 64–76 (2022)
11. Kim, S., Park, S., Chu, W.W.: An index-based approach for similarity search supporting time warping in large sequence databases. In: ICDE. pp. 607–614. IEEE Computer Society (2001)
12. L. M. Ni, L. Chen, H. Qu, and et al: Shh-taxi data. https://www.cse.ust.hk/scrg/ (2007)
13. Liu, B., Yu, X.L., Chen, S., Xu, X., Zhu, L.: Blockchain based data integrity service framework for iot data. In: ICWS. pp. 468–475. IEEE (2017)
14. Merkle, R.C.: A digital signature based on a conventional encryption function. In: Conference on the theory and application of cryptographic techniques. pp. 369–378. Springer (1987)
15. Merkle, R.C.: A certified digital signature. In: CRYPTO. Lecture Notes in Computer Science, vol. 435, pp. 218–238. Springer (1989)
16. Myers, C.S., Rabiner, L.R.: A comparative study of several dynamic time-warping algorithms for connected-word recognition. The Bell System Technical Journal **60**, 1389–1409 (1981)

17. Peng, J., Deng, M., Tang, J., Hu, Z., Xia, H., Liu, H., Mei, X.: A movement-aware measure for trajectory similarity and its application for ride-sharing path extraction in a road network. Int. J. Geogr. Inf. Sci. **38**(9), 1703–1727 (2024)
18. Qian, C.: Digital copyright management model of university library based on blockchain technology. 2021 International Conference on Computer, Blockchain and Financial Development pp. 508–513 (2021)
19. Radziwill, N.M.: Blockchain revolution: How the technology behind bitcoin is changing money, business, and the world. Quality Management Journal **25**, 64 – 65 (2018)
20. Shang, Z., Li, G., Bao, Z.: DITA: distributed in-memory trajectory analytics. In: SIGMOD Conference. pp. 725–740. ACM (2018)
21. Toohey, K., Duckham, M.: Trajectory similarity measures. Sigspatial Special **7**(1), 43–50 (2015)
22. Vlachos, M., Gunopulos, D., Kollios, G.: Discovering similar multidimensional trajectories. In: ICDE. pp. 673–684. IEEE Computer Society (2002)
23. Wang, H., Xu, C., Zhang, C., Xu, J., Peng, Z., Pei, J.: vchain+: Optimizing verifiable blockchain boolean range queries. In: ICDE. pp. 1927–1940. IEEE (2022)
24. Wang, H., Su, H., Zheng, K., Sadiq, S.W., Zhou, X.: An effectiveness study on trajectory similarity measures. In: Australasian Database Conference (2013)
25. Wang, W., Chen, J., Wang, J., Chen, J., Gong, Z.: Geography-aware inductive matrix completion for personalized point-of-interest recommendation in smart cities. IEEE Internet Things J. **7**(5), 4361–4370 (2020)
26. Wang, Z., Zhang, L., Ding, X., Choo, K.R., Jin, H.: A dynamic-efficient structure for secure and verifiable location-based skyline queries. IEEE Trans. Inf. Forensics Secur. **18**, 920–935 (2023)
27. Wood, G., et al.: Ethereum: A secure decentralised generalised transaction ledger. Ethereum project yellow paper **151**(2014) (2014)
28. Wu, H., Li, Z., Song, R., Xiao, B.: Enabling privacy-preserving and efficient authenticated graph queries on blockchain-assisted clouds. IEEE Trans. Knowl. Data Eng. **35**(9), 9728–9742 (2023)
29. Xu, H., Xiao, B., Liu, X., Wang, L., Jiang, S., Xue, W., Wang, J., Li, K.: Empowering authenticated and efficient queries for STK transaction-based blockchains. IEEE Trans. Computers **72**(8), 2209–2223 (2023)
30. Zhang, C., Xu, C., Xu, J., Tang, Y.R., Choi, B.: Gemˆ2tree:A gas-efficient structure for authenticated range queries in blockchain. In: ICDE. pp. 842–853. IEEE (2019)
31. Zheng, Y.: T-drive trajectory data sample (August 2011), https://www.microsoft.com/en-us/research/publication/t-drive-trajectory-data-sample/, t-Drive sample dataset
32. Zheng, Y., Xie, X., Ma, W.: Geolife: A collaborative social networking service among user, location and trajectory. IEEE Data Eng. Bull. **33**(2), 32–39 (2010)
33. Zhu, C., Li, J., Zhong, Z., Yue, C., Zhang, M.: A survey on the integration of blockchains and databases. Data Sci. Eng. **8**(2), 196–219 (2023)