



OPEN ACCESS

EDITED BY

Erick Antezana,
United Nations - International Computing
Center, Spain

REVIEWED BY

Hiromi Kajiya-Kanegae,
National Agriculture and Food Research
Organization (NARO), Japan
Cláudia Brito,
Campus da Faculdade de Engenharia da
Universidade do Porto, Portugal

*CORRESPONDENCE

Jia Liu

✉ lsegily@126.com

Xiaoli Zhu

✉ zhuxiaoli14@mails.ucas.ac.cn

[†]These authors have contributed
equally to this work

RECEIVED 08 September 2025

REVISED 11 November 2025

ACCEPTED 12 November 2025

PUBLISHED 03 December 2025

CITATION

Liu J, Xu Y, Gera A, Li X, Zhao L and Zhu X
(2025) A framework for privacy-preserving
similarity search of massive multi-party
genomic data.
Front. Plant Sci. 16:1684243.
doi: 10.3389/fpls.2025.1684243

COPYRIGHT

© 2025 Liu, Xu, Gera, Li, Zhao and Zhu. This is
an open-access article distributed under the
terms of the [Creative Commons Attribution
License \(CC BY\)](#). The use, distribution or
reproduction in other forums is permitted,
provided the original author(s) and the
copyright owner(s) are credited and that the
original publication in this journal is cited, in
accordance with accepted academic
practice. No use, distribution or reproduction
is permitted which does not comply with
these terms.

A framework for privacy-preserving similarity search of massive multi-party genomic data

Jia Liu^{1,2*†}, Yanping Xu^{1,2}, Abdullah Gera^{1,2}, Xiaoning Li^{1,2},
Liping Zhao^{1,2} and Xiaoli Zhu^{1,2*†}

¹Office of Academic Research, Weifang Institute of Technology, Weifang, China, ²Office of Academic Research, Nilai University, Nilai, Negeri Sembilan, Malaysia

To address the challenges of data silos across different institutions, privacy concerns, and the multi-party genomic data matching problem in crop breeding, this paper proposes a Fed-LSH framework. This framework is a collaborative framework integrating privacy-enhanced Locality-Sensitive Hashing (LSH) algorithm with Federated Learning. It enables participants can conduct cross-institutional similar genomic association analysis and elite allele identification. And they do not need to share raw genomic data with each other. This framework utilizes distributed hash index construction, outsourced computation, and encrypted similarity search to accomplish this task. Experiments show that Fed-LSH can achieve a hit rate of $60.72\% \pm 1.2\%$ when recommending 4 candidates, using a 40×3 hash size on 3072-dimensional data (implemented on standard personal computers). It can select 4 candidates from 10,000 genomic fragments (each 3072-dimensional) in less than 0.5 seconds. These performance metrics indicate that Fed-LSH provides foundational technical support for privacy-preserving collaborative tomato breeding.

KEYWORDS

privacy protection, federated learning, genomic selection, locality-sensitive hashing, plant breeding

1 Introduction

The similarity matching technology of crop genes is one of the core technologies supporting modern agricultural genomics and crop improvement (Gao, 2021). Gene sequence similarity matching technology can help reveal the intrinsic association mechanism between genetic diversity, structural variation and crop traits by analyzing the genetic similarity among different crops and different varieties of the same crop, and then select suitable breeding plants, providing technical and theoretical support for precision molecular breeding and precision agriculture of crops (Rasmussen, 2020). However, genomic data is often huge in volume, and it is frequently confronted with the problem of high computational complexity. In cases where the length of the genome

approaches millions of base pairs, the existence of polyploid genomes also makes the matching of similar genes in crops more complex. All these have put forward requirements and challenges for the rapid detection of massive gene data in crops (Y. Chen et al., 2020).

Rapid similarity matching of massive gene sequences is a key technology in modern crop genomics and genetic research, which can significantly enhance the efficiency of breeding. Currently, various efficient and fast computing models exist, including high-throughput SNP chips (Ganal et al., 2012), k-mer counting algorithms (Mahmood et al., 2012; Wilbur and Lipman, 1983), parallel similarity matching algorithms, and improved fast sequence alignment tools (Yonia et al., 2024). With the development of high-throughput sequencing technology and the establishment of massive genomic databases, numerous efficient algorithms have emerged, which can support the rapid comparison and similarity analysis of tens of thousands of genes (Hahn et al., 2024). One of the key improvements of these algorithms is the ability to complete the rapid similarity matching of large-scale gene data within a short time. For example, in the application of high-throughput SNP chips, large-scale gene similarity assessment and gene mapping in crop sample databases can be completed through the rapid genotyping of thousands or even tens of thousands of markers across the entire genome. In the application of k-mer counting algorithms, through the extraction of short sequence gene features, comprehensive matching and acceleration of large-scale protein/nucleic acid gene sequences can be achieved (such as the *afree* tool), which can significantly improve the comprehensive retrieval efficiency of homologous genes. The rapid gene matching based on k-mer counting algorithms is applicable to the matching of similar gene fragments in complex genomic environments. The new generation of DNA pattern matching algorithms, such as EPAPM and EFLPM (Ibrahim et al., 2023) enhance matching speed through parallel processing technology and improve similarity matching performance by providing error tolerance mechanisms, enabling the rapid completion of gene sequence similarity matching in complex and massive genomic similarity matching environments at the expense of certain accuracy.

In fact, multi-party gene matching, especially that among different regions and countries, has demonstrated multiple values in crop genetic breeding and trait improvement (Tao et al., 2019). Due to the differences in environmental adaptability and stress resistance of multi-party gene matching, it can significantly promote the genetic diversity of crop genetic breeding, the expression of heterosis, and enhance stress resistance, etc. At the same time, it can provide a rich genetic resource pool for coping with different climates and ecological environments, and support complex gene matching and genetic breeding (Snowdon et al., 2021; Jeynes-Cupper and Catoni, 2023; Legarra et al., 2023). However, important germplasm resources globally are scattered among different institutions (such as the Consultative Group on International Agricultural Research (CGIAR), national gene banks, and private breeding companies). Adopting multi-party gene matching would face the problem of local genomic data

leakage. Due to competition and privacy concerns, original data is often not directly shared (Harmanci and Gerstein, 2016). The core concept of federated learning technology is distributed machine learning, which allows multiple parties to collaboratively train models without sharing data. It is a solution to this problem (Li et al., 2023). Through federated learning technology, each institution retains the original genomic data and only exchanges encrypted model parameters (gradients or hash indexes).

Traditional centralized databases, such as NCBI (Schoch et al., 2020) and SRA (Katz et al., 2022), require public data uploads, which cannot meet the privacy protection needs of institutions. Traditional encrypted retrieval methods [such as homomorphic encryption BLAST (Rupa et al., 2023)] have huge computational costs (retrieval of TB-scale data requires response times on the order of hours), making them difficult to be practically applied. Federated learning (FL) is a potential solution to this problem. However, although FL algorithms support distributed training, they lack corresponding algorithm and application framework support, especially for distributed algorithms for rapid and massive matching of similar crop genes in a multi-party privacy protection environment (Wu et al., 2025).

The Locally Sensitive Hash (LSH) algorithm, as an efficient approximate search tool in gene matching (Baker and Langmead, 2023), works by mapping similar gene sequences into the same hash bucket, thereby significantly accelerating large-scale searches by reducing the number of exact sequence comparisons (K. Chen and Shao, 2023). LSH supports various similarity measures such as Euclidean distance, Jaccard similarity, and Hamming distance, and can be adapted to different data types by designing corresponding hash function families (Martínez et al., 2022), demonstrating multi-metric adaptability. For instance, some improved LSH-based algorithms can achieve performance acceleration by several orders of magnitude while maintaining a recall rate of over 0.95 and keeping resource consumption low (Jia et al., 2021; Satuluri and Parthasarathy, 2012; Sundaram et al., 2013; Yu et al., 2017, 2019), thus offering significant performance advantages. For example, the LSH-ALL-PAIRS algorithm demonstrates high sensitivity in gene detection, capable of identifying local sequence similarities as low as 63% in massive genomic datasets of tens of megabases (Buhler, 2001). The core advantage of LSH in the field of gene similarity matching lies in its ability to perform rapid similarity matching detection on ultra-large-scale gene data (Buhler, 2001). Moreover, LSH is often combined with other algorithms such as deep learning and generative models, expanding the complexity of biological gene analysis dimensions (Lall et al., 2022; Tavakoli, 2019). Therefore, LSH provides high application and research value in offering efficient and scalable approximate search capabilities for massive gene data, integrating similar gene sequence comparisons, maintaining algorithm sensitivity and accuracy, and integrating with other feature extraction algorithms.

To break through the above-mentioned bottlenecks, this paper proposes the Fed-LSH framework (Federated Locality-Sensitive Hashing), which deeply integrates Locality-Sensitive Hashing (LSH) with federated learning to achieve privacy-protected

similarity search for tomato genomic data. The innovation of this framework lies in its ability to conduct similarity search without sharing local raw data through the improved Fed-LSH framework. Based on the data, rapid similarity matching is carried out. Meanwhile, cloud-assisted machine learning technology is adopted for batch rapid local gene operations, reducing the local computing burden and meeting the needs of multiple institutions for rapid similarity gene matching.

2 Method design

2.1 Federated LSH architecture

Cloud Server: The cloud server maintains the consistency of the global projection matrix of the LSH algorithm, generates the LSH algorithm projection matrix w , and transmits it to each institution (client) to complete the cloud-assisted machine learning of the LSH algorithm. Through the data x' and w' transmitted by the local node, the encrypted $HashTable'$ is generated.

Clients: Each institution (Client) holds private genomic data, generates encryption keys, encrypts local data x and projection matrix w to form x' and w' . Meanwhile, based on the $HashTable'$ returned by the cloud server, it verifies the correctness of $HashTable'$ and generates the real $HashTable$.

Data Server: Data server aggregates the global hash table and responds to cross-node similarity queries.

Clients can generate $HashTables$ locally using the projection matrix generated by the Cloud Server from their stored private genetic data and send it to the Data Server. Alternatively, they can use the proposed cloud-assisted machine learning technology to outsource part of the LSH algorithm's computations to the Cloud Server, which will batch-generate encrypted $HashTable'$, and return it to the Clients. The Clients then decrypt the batch $HashTable'$ to obtain the original $HashTable$ and send it to the Data Server. For the specific architecture, please refer to Figures 1, 2.

2.2 Framework operation steps

2.2.1 Global projection matrix generation

The Cloud Server generates a global projection matrix W , where $W \in \mathbb{R}^{t \times d}$, and transmits W to each Client. t is the number of rows of the global projection matrix W , the size of the hash. The larger the t , the higher the hash discrimination degree, which can better distinguish similar and dissimilar data. A shorter hash size improves the query speed, a longer hash size improves the accuracy rate, but the computational cost increases. d represents the feature dimension of the original data, the larger the d , the higher the data dimension, and the more complex the represented data is.

2.2.2 Local key generation

Each institution (Client) acts as a data provider holding private data. Client generates its own key. The data of each Client is $X \in \mathbb{R}^{m \times d}$. With the parameters $\lambda = (m, d)$, the Client selects $2l$ Givens

matrices. $P_1, P_2, \dots, P_l, Q_1, Q_2, \dots, Q_l$ and a random real number α as the secret key $SK = \{\alpha, P_1, P_2, \dots, P_l, Q_1, Q_2, \dots, Q_l\}$.

The Givens matrix is also known as the elementary rotation matrix, as shown in Equation 1. It is a sparse matrix with most of its elements being 0 and is orthogonal, which can reduce the computational load in matrix multiplication. The client randomly selects an integer $\theta \in (0, 2\pi)$, and constructs the Givens matrix T_{ij} as follows:

- Let $c = \cos\theta$, $s = \sin\theta$.
- Set the i -th and j -th main diagonal elements to c .
- Set the other diagonal elements to 1.
- Set the element at the i -th row and j -th column to s .
- Set the element at the j -th row and i -th column to $-s$.

Set all other elements to 0.

$$T(i, j) = \begin{bmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & & c & & s \\ & & & & 1 & \\ & & \vdots & & \ddots & \\ & & & & & 1 \\ & & -s & & c & \\ & & & & & 1 \\ & & \vdots & & & \ddots \\ & & & & & & 1 \end{bmatrix} \quad (1)$$

2.2.3 Local data encryption

Client encrypts the data into X' and W' using SK , as shown in Equations 2, 3.

$$X' = P_1 P_2 \dots P_l (\alpha X) Q_1 Q_2 \dots Q_l \quad (2)$$

$$W' = (\alpha W) Q_1 Q_2 \dots Q_l \quad (3)$$

2.2.4 Cloud-assisted machine learning

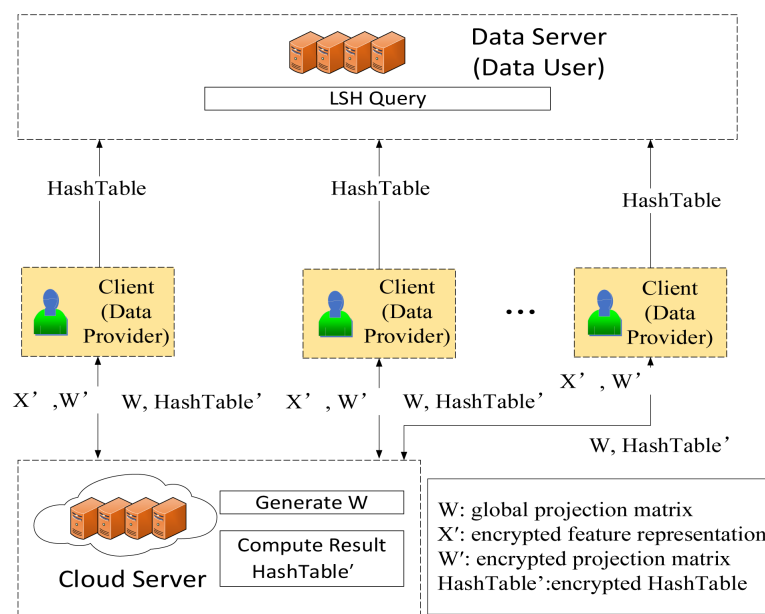
The Cloud Server computes the encrypted hash code and sends the encrypted hash table $HashTable'$ back to the Client, as shown in Equation 4.

$$HashTable' = W' X' T, \quad W \in \mathbb{R}^{t \times d} \quad (4)$$

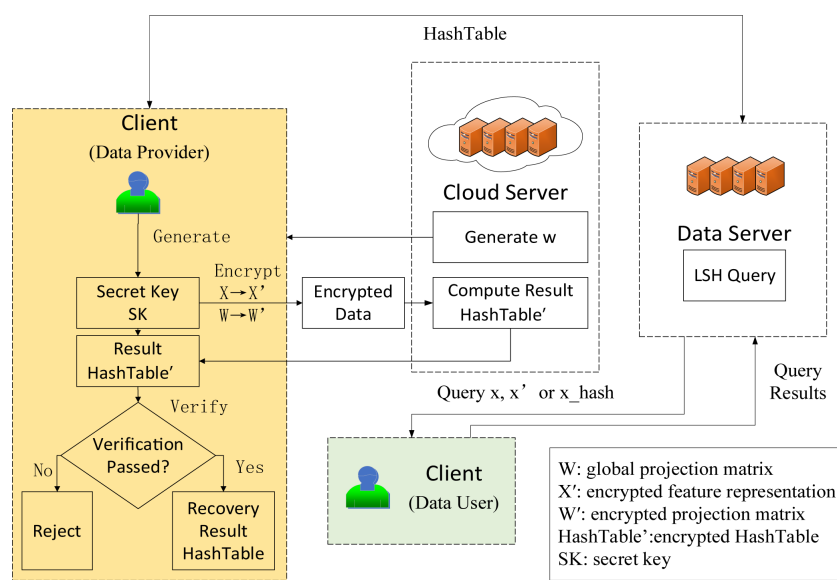
The Client verifies the correctness of the encrypted hash table $HashTable'$. The Client randomly selects a vector $r \in \mathbb{R}^{1 \times t}$ to calculate V_1 and V_2 , as shown in Equations 5, 6. If the value of V_1 and V_2 are equal, then the $HashTable'$ is correct.

$$V_1 = (r W') X' T \quad (5)$$

$$V_2 = r HashTable' \quad (6)$$



a



b

FIGURE 1

(A) System architecture overview. (B) Operation flow and data exchange protocol.

2.2.5 Restore the original hash table

The Client restores the *HashTable*, as shown in Equation 7.

$$HashTable = \frac{1}{\alpha^2} (HashTable' P_1 P_2 \dots P_l) \quad (7)$$

2.2.6 Send the hash table to the data server

The Client will either use cloud-assisted machine learning to batch-generate or directly adopt the *W* – computed *HashTable* and send it to Data server.

2.2.7 Fast query of multiple similar gene data

If the genetic data does not require privacy protection, the Client can choose to send the raw data to the Data Server. The Data Server then computes WX^T to form the hash value of the query data, enabling a fast LSH query on the Data Server side.

For a small amount of genetic data that requires privacy protection, the Client can calculate WX^T and send the hash value of the data to the Data Server for similarity queries.


```
[160]: #3072*[3072,1000] = 1000
v=np.dot(v,np.transpose(train_x_cy))

[165]: print(v)

[ 7.72935238e+08 -4.14150123e+09  2.89871819e+09  2.73706853e+09
 1.71202381e+08  5.81518923e+08 -7.98199242e+08 -3.70809529e+09
 3.09323573e+09  1.49789795e+09  6.01681548e+08  2.55868589e+09
-5.35189057e+09  1.17138663e+09 -1.68791851e+09 -9.97495194e+08
 5.25253573e+08 -4.33962192e+08  9.26872789e+08  2.66605896e+09
-5.67363378e+08  1.05800587e+09 -1.59975471e+09  1.27130429e+09
 2.81708156e+09 -9.86591865e+08 -1.12028442e+07 -9.41940743e+08
-3.49078698e+08 -6.84771843e+08 -4.34107380e+08 -4.63616310e+08
-3.02815274e+07  1.80050829e+08 -1.63787477e+09 -3.78373829e+08
 1.27029316e+08  1.65876098e+08 -6.37192425e+08  2.55798980e+09
-9.29096753e+08  4.70804682e+08  3.37894212e+08 -3.67128005e+09
 2.48043421e+09  2.16760719e+09 -1.87902017e+09 -1.57466536e+08
-2.07602392e+09 -2.00133164e+09 -8.08400682e+07  1.56459369e+09
-1.44067548e+08  4.11488681e+09  1.21838228e+09 -8.72682601e+08
 3.02475661e+08 -1.56402443e+09  1.70089114e+09  1.28039722e+09
-2.27481807e+09  3.05790377e+08 -1.14895394e+09  2.07288775e+07
 1.43765770e+09 -1.09644734e+09 -2.56993221e+09  2.28437514e+09
-2.32286559e+09 -2.24519595e+08  2.34780969e+09 -2.56527406e+09
 6.36029407e+09  1.12231878e+09 -1.48185215e+09 -3.34052072e+08]

[163]: v2=np.dot(np.array(r),np.transpose(np.array(list(hash_table_cy[0].keys()))))

[164]: print(v2)

[ 7.72935238e+08 -4.14150123e+09  2.89871819e+09  2.73706853e+09
 1.71202381e+08  5.81518923e+08 -7.98199242e+08 -3.70809529e+09
 3.09323573e+09  1.49789795e+09  6.01681548e+08  2.55868589e+09
-5.35189057e+09  1.17138663e+09 -1.68791851e+09 -9.97495194e+08
 5.25253573e+08 -4.33962192e+08  9.26872789e+08  2.66605896e+09
-5.67363378e+08  1.05800587e+09 -1.59975471e+09  1.27130429e+09
 2.81708156e+09 -9.86591865e+08 -1.12028442e+07 -9.41940743e+08
-3.49078698e+08 -6.84771843e+08 -4.34107380e+08 -4.63616310e+08
-3.02815274e+07  1.80050829e+08 -1.63787477e+09 -3.78373829e+08
 1.27029316e+08  1.65876098e+08 -6.37192425e+08  2.55798980e+09
-9.29096753e+08  4.70804682e+08  3.37894212e+08 -3.67128005e+09
 2.48043421e+09  2.16760719e+09 -1.87902017e+09 -1.57466536e+08
-2.07602392e+09 -2.00133164e+09 -8.08400682e+07  1.56459369e+09
-1.44067548e+08  4.11488681e+09  1.21838228e+09 -8.72682601e+08
 3.02475661e+08 -1.56402443e+09  1.70089114e+09  1.28039722e+09
-2.27481807e+09  3.05790377e+08 -1.14895394e+09  2.07288775e+07
 1.43765770e+09 -1.09644734e+09 -2.56993221e+09  2.28437514e+09
-2.32286559e+09 -2.24519595e+08  2.34780969e+09 -2.56527406e+09
 6.36029407e+09  1.12231878e+09 -1.48185215e+09 -3.34052072e+08]
```

FIGURE 2

Python experimental results of the verification algorithm.

To conduct batch queries of similar data, the Client can leverage the Cloud Server to generate a cloud-assisted batch query data *HashTable*. The Cloud Server generates *HashTable'* through the privacy-protected cloud-assisted machine learning algorithm proposed above. The Client then verifies and restores the original

HashTable and sends it to the Data Server for rapid LSH-based similar data queries.

The Data Server generates the LSH algorithm hash table *LshHashTable* based on Equation 8 and the transmitted *HashTable*, and simultaneously selects one of the distance algorithms such as

Euclidean distance, cosine distance, or L1 norm distance for the LSH algorithm similarity query, returning num_results results.

$$\text{LshHashTable} == \left\lceil \frac{\text{HashTable} + b}{\text{num_hash}} \right\rceil \quad (8)$$

3 Performance verification

3.1 Correctness verification

We denote P and Q as $P = P_1 P_2 \cdots P_l$, $Q = Q_1 Q_2 \cdots Q_l$ respectively. Each Client recovers the HashTable from HashTable' through $\text{HashTable} = \frac{1}{\alpha^2} (\text{HashTable}' P)$, as shown in Equation 9.

$$\begin{aligned} \text{HashTable} &= \frac{1}{\alpha^2} (\text{HashTable}' P) \\ &= \frac{1}{\alpha^2} (W' X' T P) \\ &= \frac{1}{\alpha^2} ((\alpha W) Q (P(\alpha X) Q)^T P) \\ &= \frac{1}{\alpha^2} ((\alpha W) Q Q^T (\alpha X)^T P^T P) \\ &= \frac{1}{\alpha^2} (\alpha^2 W X^T) \\ &= W X^T \end{aligned} \quad (9)$$

Therefore, the recovered HashTable is the output of the original computation. Meanwhile, we prove the correctness of the verification algorithm. The Client verifies the correctness of the value computed by the Cloud Server by checking if $V_1 = V_2$.

$$\begin{aligned} V_1 &= (r W') X' T \\ &= r W' X' T \\ &= r \text{HashTable}' \\ &= V_2 \end{aligned} \quad (10)$$

Therefore, if $V_1 = V_2$, the result is correct. Meanwhile, the code result implemented through Python code is as follows, which also

proves the correctness of the verification algorithm, as shown in Equation 10.

3.2 Security verification

The CIFAR-10 dataset was used to perform the encryption operation of the Client-side key. We tested the blinding effect of the original data X multiplied by different numbers of l Givens matrices, namely $P_1 P_2 \cdots P_l X$. It can be seen that the Givens matrix has a good blinding effect on the original data, as shown in Table 1.

3.2.1 The proposed algorithm has input/output privacy

In the proposed scheme, the original data of the Client, i. e., X , is hidden from the Data Server and the Cloud Server. In our scheme, the Client transforms X into X' by multiplying a real number with a series of basic rotation matrices. The selected Givens matrices are randomly generated. These basic rotation matrices are randomly chosen. The $P_1 P_2 \cdots P_l$ in the key SK is only used locally and will not be transmitted over the network. Therefore, for the Cloud Server, if it needs to recover the original X , it has to guess the values of $P_1 P_2 \cdots P_l$. Assuming the probability of successfully guessing an elementary rotation matrix is $\frac{1}{\delta}$. Then the probability of successfully guessing l elementary rotation matrices is $\text{negli}(l) = \frac{1}{\delta^l}$, which is a non-polynomial time function.

In fact, since the values in the elementary rotation matrix are real numbers with a very wide range, the probability of successful guessing is extremely low. If one needs to guess the value of X , the probability of correctly guessing X is $\text{negli}(md) = \frac{1}{\delta^{md}}$. For the Cloud Server, the probability of successfully guessing the correct HashTable is $\text{negli}(tm) = \frac{1}{\delta^{tm}}$. All of these are non-polynomial time. Therefore, the probability that Cloud Server successfully guesses the series of Givens matrices, X and HashTable is relatively low. Similarly, for Data Server, since the Client only transmits H

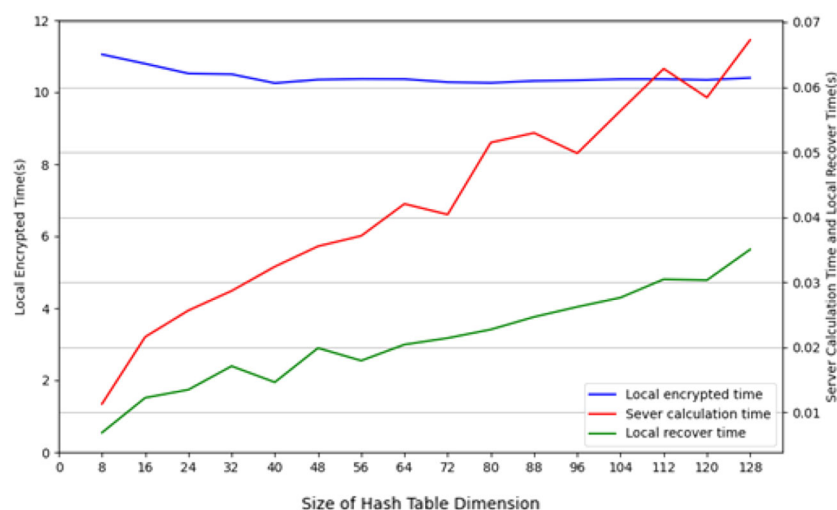


FIGURE 3

The running time of different-sized hash table on the client and cloud server on CIFAR-10 dataset.

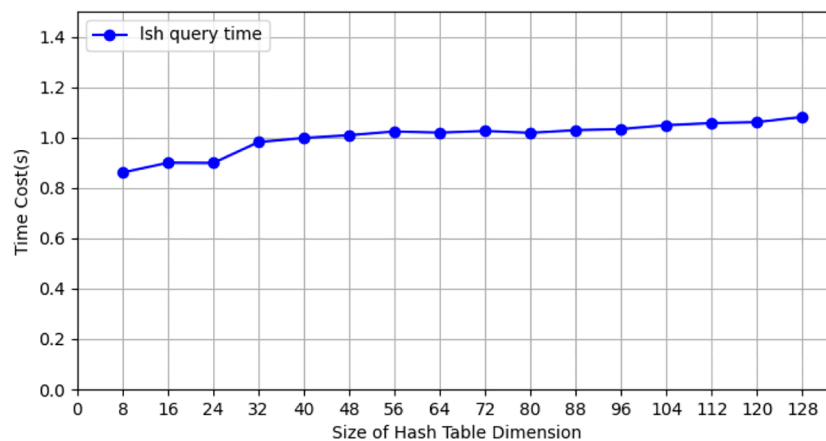


FIGURE 4

The query efficiency of data server for hash tables of different sizes on CIFAR-10 dataset.

ashTable to Data Server, the original data X is hidden from Data Server.

3.2.2 The efficiency of the proposed algorithm on the client side is $((2md + 2td + 2tm + 4ml + 4dl)/(tdm))$ efficient

We define scalar multiplication as SM and ignore linear computations such as scalar addition. For a Client, it performs $md + td + 2ml + 4dl$ SM . During the encryption phase on the Client side, the Client executes $md + td + 2ml + 4dl$ SM . The Cloud Server performs tdm SM in the computation phase. The Client recovers the *HashTable* by performing $tm + 2ml$ SM . Therefore, a single Client performs a total of $2md + 2td + 2tm + 4ml + 4dl$ SM . The Cloud Server performs tdm SM in total. The computational cost of the algorithm without cloud-assisted machine learning for the Client is mtd SM , which is as high as the computational complexity of the LSH algorithm using cloud assistance. Thus, the efficiency of the proposed method is $((2md + 2td + 2tm + 4ml + 4dl)/(tdm))$.

4 Experimental verification and evaluation

4.1 Algorithm verification on CIFAR-10 dataset

- The verification experiment was conducted on a regular computer with a 2.5GHz CPU and 16GB of memory running the Ubuntu 22.04 operating system. Python 3.11 was used as the programming language. To verify the correctness of the algorithm, the CIFAR-10 image dataset was employed for simulation. To test the algorithm's performance on crop gene data, the NCBI tomato gene dataset was utilized for application and performance evaluation.
- The CIFAR-10 dataset contains 60,000 image data across 10 categories. Each category consists of 6,000 32x32 images.
- The NCBI tomato gene data contains 5,156 tomato gene records.

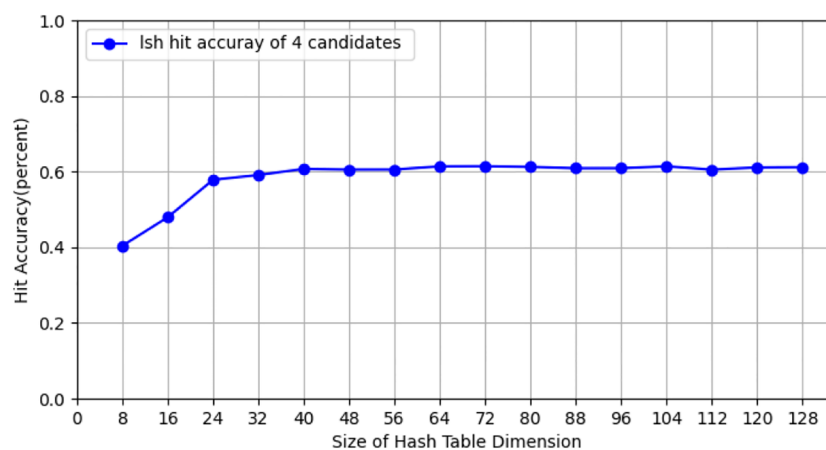


FIGURE 5

The hit rate of providing four candidate members for hash tables of different sizes on CIFAR-10 dataset.

During the testing process using the CIFAR-10 dataset, we conducted tests with the CIFAR-10 training dataset (50, 000 images), employing 10 client nodes for federated learning LSH algorithm testing, with each client containing 5,000 images. Performance tests were carried out on both the client and Cloud Server ends using different hash sizes. All experiments were repeated 10 times with different random seeds, and results were averaged for final reporting. The specific results are shown in Figure 3. The query efficiency at the Data Server end is presented in Figure 4, and the performance of hit rate providing 4 candidate results is shown in Figure 5.

Fed-LSH achieves a hit rate of $60.72 \pm 1.2\%$ (averaged over 10 runs) on CIFAR-10 dataset.

4.2 Performance evaluation on NCBI tomato dataset

We applied it on the tomato gene data from NCBI. When processing the tomato gene data from NCBI, due to the inconsistent

lengths of the gene data, we performed slicing of the gene data. The specific algorithm is Algorithm 1.

- Numericize the NCBI tomato gene data as follows: {'A': 0, 'T': 1, 'C': 2, 'G': 3}.
- Set the sequence length `sequence_length` and the overlap for the gene data slices.
- For each gene data, slice the data based on `sequence_length` and overlap.
- For each kmer slice, if there is a part that is shorter than `sequence_length`, fill it with -1.
- Encapsulate each kmer of the slice and the corresponding count-th gene data to form a (kmer, count) pair.
- For each gene data, a total of `n_chunks` pairs of (kmer, count) are obtained.
- Gene data slicing and k-mer extraction for genomic hashing.

Algorithm 1. Gene data slicing and k-mer extraction for genomic hashing.

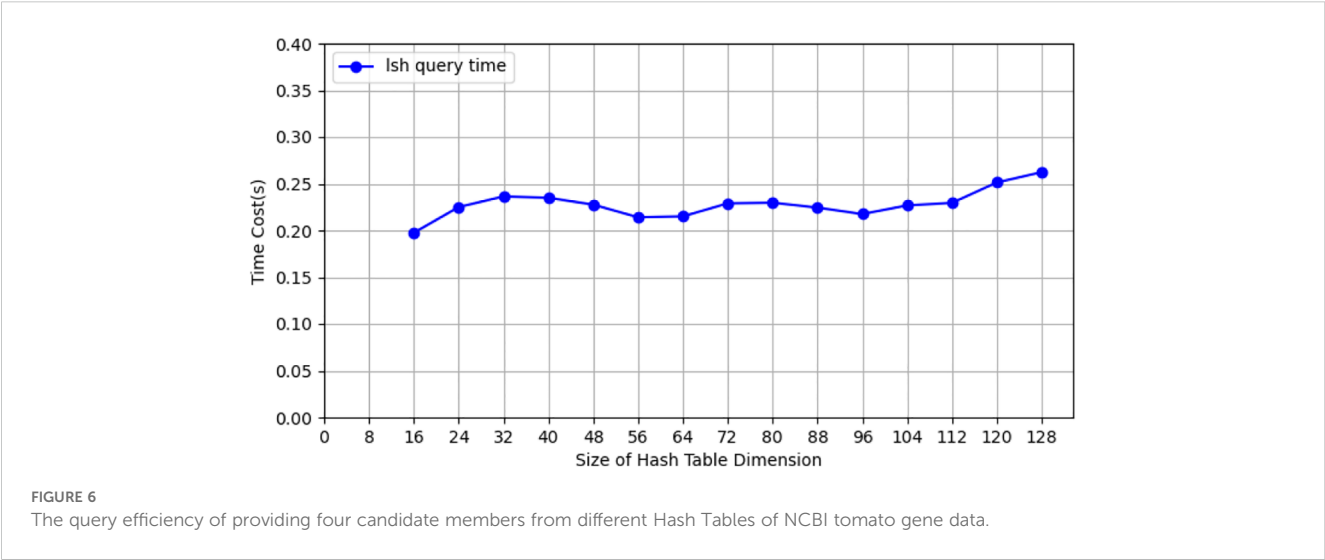


TABLE 1 Blinding effects of givens matrices with different *l* values on image data.


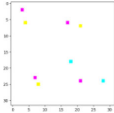
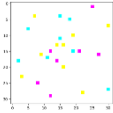
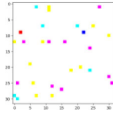
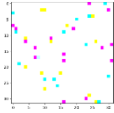
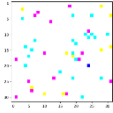
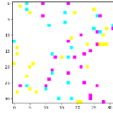
Cell Type	T-bet* hTregs	GATA3* Tregs	RORyt* Tregs
Characteristic			
			
	51	61	71
			

TABLE 2 The time consumption of each calculation step of the client.

Phase	Encryption	Verification	Recovery
SM	$md + td + 2ml + 4dl$	$td + dm + tm$	$tm + 2ml$

TABLE 3 The dataset in the experiments.

Dataset	Dimension d	Volume of data
CIFAR-10	32×32	60000
NCBI tomato gene data	1206-81465427	5156

With a sequence length of 3072 and an overlap of 100, after processing the gene data, we obtained a total of 298,613 gene slice data.

To prevent data duplication, in practical applications, a unique identifier (UUID) can be assigned to each gene fragment. During the process of building the hash index, UUID conflict detection is carried out first. Once duplicate gene items are detected, the system can skip them. This ensures that the hash index only contains unique gene sequences and avoids duplicate analyses.

We selected 10,000 gene data records for performance testing. Each client had 1,000 gene data records. Four candidate results were selected to test the performance of multi-party queries. The specific performance is shown in Figure 6. From the performance test, it can be seen that the performance of similar gene queries is relatively high. On an ordinary computer without GPU acceleration, the time required to select four candidate sets from 10,000 gene fragments of 3072 in length is less than 0.5 seconds.

Tables 2-4 compares the performance of the Proposed Fed-LSH with other privacy-preserving methods (HELib, DP, SMPC) in terms of computational cost, query latency, and privacy level. The Proposed Fed-LSH achieves low computational cost and low query latency (<0.5s) while maintaining a high privacy level. In contrast, HELib has very high computational cost and long query latency (>3.6s) but high privacy; DP offers low computational cost and low latency (2.67s) but only low privacy; SMPC has high computational cost, long latency (>2.8s), and high privacy. Thus, the Proposed Fed-LSH demonstrates a better balance among computational efficiency, query speed, and strong privacy protection.

TABLE 4 Performance comparison of Fed-LSH with other privacy-preserving methods.

Method	Computational cost	Query latency (s)	Privacy level
HELib (Lee et al., 2025; Tsuji and Oguchi, 2022)	Very High	>3.6	Very High
DP (Aziz et al., 2022)	Low	2.67	Low
SMPC (Nakagawa et al., 2022)	High	>2.8	High
Proposed Fed-LSH	Low	<0.5	High

5 Conclusion

Genomic data is on the scale of terabytes. To address the issue of rapid similarity matching of massive genomic data from multiple parties, we propose a federated LSH algorithm solution and framework. At the same time, we adopt cloud-assisted machine learning technology to alleviate the computational pressure on the Client side. To the best of our knowledge, there is currently no universal secure federated learning method applicable to all machine learning algorithms. Fully homomorphic encryption (FHE) is a feasible solution for achieving universal cloud-assisted machine learning, but the low computational efficiency of FHE makes FHE-based methods inapplicable to practical scenarios (Li et al., 2023). Differential privacy can also protect client data. By adding controllable noise, the original data cannot be identified, but the original data features are affected, which disrupts the characteristic signals on the genes (Wei et al., 2021). In the architecture proposed in this paper, the data on the Client side is all confidential, ensuring the privacy protection of Client data. Meanwhile, the Client can use optional edge computing and cloud-assisted machine learning to generate the HashTable of genomic data. With edge computing, the Client can directly use W to generate the HashTable. Alternatively, the cloud-assisted machine learning technology proposed above can be used to generate cloud-assisted batch HashTable for massive genomic data. The Data Server only stores the hash values of the genomic data, ensuring the privacy of the genomic data. At the same time, it can perform rapid similarity matching of massive genomic data from multiple parties.

The Fed-LSH framework proposed in this paper integrates federated learning, cloud-assisted machine learning, and LSH algorithm to solve the problem of rapid similarity matching of massive multi-party genomic data. The algorithm and framework have been tested for performance using tomato genome data and can also be applied to other genomic and whole-genome data, providing a standardized similarity matching tool for multi-institutional joint breeding. Through the deep coupling of federated learning and LSH, secure and efficient collaborative analysis of tomato genome data has been achieved.

Although we only conducted this test on the tomato genome, the Fed-LSH algorithm can be readily extended to fields such as crop gene matching (including rice, wheat, corn, etc.) and animal genetics. For PB-level genomic datasets, the algorithm demonstrates strong parallel computing potential and scalability, with performance significantly improved through parallel hash operations. The algorithm is particularly well-suited for high-performance computing and distributed computing environments. Cross-organizational genomic data sharing has raised critical ethical and privacy concerns. The proposed Fed-LSH framework supports compliance with GDPR and FAIR principles through fine-grained access control policies, comprehensive data usage audit logs, and dynamic consent mechanisms.

6 Challenges and prospects

However, the sharing of multi-party genetic data also faces many challenges. The differences in sequencing depth and

annotation among different institutions can affect the consistency of gene hash values. Due to the long length and large volume of whole-genome data, creating a HashTable takes a considerable amount of time. GPU acceleration and distributed computing clusters can be adopted on Cloud Servers to speed up the calculation. Meanwhile, the standards for cross-border transmission of genetic data vary among different countries, and a more legal and compliant global privacy protection framework for cross-border transmission of genetic data needs to be designed. Combining blockchain technology with Fed-LSH technology can provide feasible solutions for data traceability and incentivizing multiple parties to provide genetic data.

Data availability statement

The datasets presenting in this study can be found here: CIFAR-10: <https://www.cs.toronto.edu/~kriz/cifar.html>; NCBI tomato gene data: https://www.ncbi.nlm.nih.gov/bioproject/?linkname=assembly_bioproject&from_uid=15321541.

Author contributions

JL: Conceptualization, Methodology, Software, Writing – original draft. YX: Writing – review & editing. AG: Conceptualization, Writing – review & editing. XL: Conceptualization, Writing – review & editing. LZ: Conceptualization, Writing – review & editing. XZ: Writing-review & editing.

Funding

The author(s) declare financial support was received for the research and/or publication of this article. This work was supported by Shandong Province Key Research and Development Project (Action Plan for Promoting Rural Revitalization through

Scientific and Technological Innovation) (grant numbers 2024TZXD061, 2023TZXD028); Weifang Science and Technology Development Project (grant number 2023GX076), China University Industry-University-Research Innovation Fund (grant number 2024ZC035), Weifang Institute of Technology Scientific Research Project (grant numbers KY-X202430, KY-X202431, KY-X202306), Shandong Province Undergraduate Teaching Reform Project (grant number M2024314), Shandong Provincial Natural Science Foundation (grant number ZR2021QB183).

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Generative AI statement

The author(s) declare that no Generative AI was used in the creation of this manuscript.

Any alternative text (alt text) provided alongside figures in this article has been generated by Frontiers with the support of artificial intelligence and reasonable efforts have been made to ensure accuracy, including review by the authors wherever possible. If you identify any issues, please contact us.

Publisher's note

All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

References

- Aziz, M.M. Al, Anjum, M. M., Mohammed, N., and Jiang, X. (2022). Generalized genomic data sharing for differentially private federated learning. *J. BioMed. Inform.* 132, 104113. doi: 10.1016/j.jbi.2022.104113
- Baker, D. N., and Langmead, B. (2023). Genomic sketching with multiplicities and locality-sensitive hashing using Dashing 2. *Genome Res.* 33, 1218–1227. doi: 10.1101/gr.277655.123
- Buhler, J. (2001). Efficient large-scale sequence comparison by locality-sensitive hashing. *Bioinformatics* 17, 419–428. doi: 10.1093/bioinformatics/17.5.419
- Chen, K., and Shao, M. (2023). Locality-sensitive bucketing functions for the edit distance. *Algorithms Mol. Biol.* 18, 7. doi: 10.1186/s13015-023-00234-2
- Chen, Y., Song, W., Xie, X., Wang, Z., Guan, P., Peng, H., et al. (2020). A collinearity-incorporating homology inference strategy for connecting emerging assemblies in the triticeae tribe as a pilot practice in the plant pangenomic era. *Mol. Plant* 13, 1697–1708. doi: 10.1016/j.molp.2020.09.019
- Ganal, M. W., Polley, A., Graner, E.-M., Plieske, J., Wieseke, R., Luerssen, H., et al. (2012). Large SNP arrays for genotyping in crop plants. *J. Biosci.* 37, 821–828. doi: 10.1007/s12038-012-9225-3
- Gao, C. (2021). Genome engineering for crop improvement and future agriculture. *Cell* 184, 1621–1635. doi: 10.1016/j.cell.2021.01.005
- Hahn, G., Lutz, S. M., Hecker, J., Prokopenko, D., Cho, M. H., Silverman, E. K., et al. (2024). Fast computation of the eigensystem of genomic similarity matrices. *BMC Bioinf.* 25, 43. doi: 10.1186/s12859-024-05650-8
- Harmanci, A., and Gerstein, M. (2016). Quantification of private information leakage from phenotype-genotype data: linking attacks. *Nat. Methods* 13, 251–256. doi: 10.1038/nmeth.3746
- Ibrahim, O. A. S., Hamed, B. A., and El-Hafeez, T. A. (2023). A new fast technique for pattern matching in biological sequences. *J. Supercomput* 79, 367–388. doi: 10.1007/s11227-022-04673-3
- Jeynes-Cupper, K., and Catoni, M. (2023). Long distance signalling and epigenetic changes in crop grafting. *Front. Plant Sci.* 14. doi: 10.3389/fpls.2023.1121704
- Jia, P., Wang, P., Zhao, J., Zhang, S., Qi, Y., Hu, M., et al. (2021). “Bidirectionally densifying LSH sketches with empty bins,” in *Proceedings of the 2021 International Conference on Management of Data*. (New York, NY, USA: Association for Computing Machinery), 830–842. doi: 10.1145/3448016.3452833

- Katz, K., Shutov, O., Lapoint, R., Kimelman, M., Brister, J. R., and O'Sullivan, C. (2022). The Sequence Read Archive: a decade more of explosive growth. *Nucleic Acids Res.* 50, D387–D390. doi: 10.1093/nar/gkab1053
- Lall, S., Ray, S., and Bandyopadhyay, S. (2022). LSH-GAN enables in-silico generation of cells for small sample high dimensional scRNA-seq data. *Commun. Biol.* 5, 577. doi: 10.1038/s42003-022-03473-y
- Lee, C. H., Lim, K. H., and Eswaran, S. (2025). A comprehensive survey on secure healthcare data processing with homomorphic encryption: attacks and defenses. *Discov. Public Health* 22, 137. doi: 10.1186/s12982-025-00505-w
- Legarra, A., Gonzalez-Dieguez, D. O., Charcosset, A., and Vitezica, Z. G. (2023). Impact of interpopulation distance on dominance variance and average heterosis in hybrid populations within species. *Genetics* 224, iyad059. doi: 10.1093/genetics/iyad059
- Li, Q., Wen, Z., Wu, Z., Hu, S., Wang, N., Li, Y., et al. (2023). A survey on federated learning systems: vision, hype and reality for data privacy and protection. *IEEE Trans. Knowl. Data Eng.* 35, 3347–3366. doi: 10.1109/TKDE.2021.3124599
- Mahmood, K., Webb, G. I., Song, J., Whisstock, J. C., and Konagurthu, A. S. (2012). Efficient large-scale protein sequence comparison and gene matching to identify orthologs and co-orthologs. *Nucleic Acids Res.* 40, e44–e44. doi: 10.1093/nar/gkr1261
- Martinez, S., Gérard, S., and Cabot, J. (2022). Efficient model similarity estimation with robust hashing. *Softw. Syst. Model.* 21, 337–361. doi: 10.1007/s10270-021-00915-9
- Nakagawa, Y., Ohata, S., and Shimizu, K. (2022). Efficient privacy-preserving variable-length substring match for genome sequence. *Algorithms Mol. Biol.* 17, 9. doi: 10.1186/s13015-022-00211-1
- Rasmussen, S. K. (2020). Molecular genetics, genomics, and biotechnology in crop plant breeding. *Agronomy* 10, 439. doi: 10.3390/agronomy10030439
- Rupa, C. h., Greeshmanth, and Shah, M. A. (2023). Novel secure data protection scheme using Martino homomorphic encryption. *J. Cloud Comput.* 12, 47. doi: 10.1186/s13677-023-00425-7
- Satuluri, V., and Parthasarathy, S. (2012). Bayesian locality sensitive hashing for fast similarity search. *Proc. VLDB Endowment* 5, 430–441. doi: 10.14778/2140436.2140440
- Schoch, C. L., Ciufu, S., Domrachev, M., Hotton, C. L., Kannan, S., Khovanskaya, R., et al. (2020). NCBI Taxonomy: a comprehensive update on curation, resources and tools. *Database* 2020, baaa062. doi: 10.1093/database/baaa062
- Snowdon, R. J., Wittkop, B., Chen, T.-W., and Stahl, A. (2021). Crop adaptation to climate change as a consequence of long-term breeding. *Theor. Appl. Genet.* 134, 1613–1623. doi: 10.1007/s00122-020-03729-3
- Sundaram, N., Turmukhametova, A., Satish, N., Mostak, T., Indyk, P., Madden, S., et al. (2013). Streaming similarity search over one billion tweets using parallel locality-sensitive hashing. *Proc. VLDB Endowment* 6, 1930–1941. doi: 10.14778/2556549.2556574
- Tao, Y., Zhao, X., Mace, E., Henry, R., and Jordan, D. (2019). Exploring and exploiting pan-genomics for crop improvement. *Mol. Plant* 12, 156–169. doi: 10.1016/j.molp.2018.12.016
- Tavakoli, N. (2019). “Modeling genome data using bidirectional LSTM,” in *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*. 183–188 (Milwaukee, WI, USA: IEEE). doi: 10.1109/COMPSAC.2019.10204
- Tsuji, A., and Oguchi, M. (2022). “Performance analysis of HELib on a privacy preserving search for genome information with fully homomorphic encryption,” in *2022 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*. 129–136 (Bangkok, Thailand: IEEE). doi: 10.1109/CloudCom55334.2022.00028
- Wei, J., Lin, Y., Yao, X., Zhang, J., and Liu, X. (2021). Differential privacy-based genetic matching in personalized medicine. *IEEE Trans. Emerg. Top. Comput.* 9, 1109–1125. doi: 10.1109/TETC.2020.2970094
- Wilbur, W. J., and Lipman, D. J. (1983). Rapid similarity searches of nucleic acid and protein data banks. *Proc. Natl. Acad. Sci.* 80, 726–730. doi: 10.1073/pnas.80.3.726
- Wu, X., Zhang, Y.-T., Lai, K.-W., Yang, M.-Z., Yang, G.-L., and Wang, H.-H. (2025). A novel centralized federated deep fuzzy neural network with multi-objectives neural architecture search for epistatic detection. *IEEE Trans. Fuzzy Syst.* 33, 94–107. doi: 10.1109/TFUZZ.2024.3369944
- Yonia, D. L., Hidayati, S. C., Sarno, R., and Septiyanto, A. F. (2024). “DNA Pattern Matching Algorithms within Sorghum bicolor Genome: A Comparative Study,” in *2024 7th International Conference on Informatics and Computational Sciences (ICICoS)*. 36–41 (Semarang, Indonesia: IEEE). doi: 10.1109/ICICoS62600.2024.10636821
- Yu, C., Luo, L., Chan, L. L.-H., Rakthanmanon, T., and Nutanong, S. (2019). A fast LSH-based similarity search method for multivariate time series. *Inf. Sci. (N. Y.)* 476, 337–356. doi: 10.1016/j.ins.2018.10.026
- Yu, C., Nutanong, S., Li, H., Wang, C., and Yuan, X. (2017). A generic method for accelerating LSH-based similarity join processing. *IEEE Trans. Knowl. Data Eng.* 29, 712–726. doi: 10.1109/TKDE.2016.2638838