Int. J. Nonlinear Anal. Appl. 16 (2025) 6, 81–91
ISSN: 2008-6822 (electronic)
http://dx.doi.org/10.22075/ijnaa.2023.32328.4801

Adaptive optimization of multi-objective query in heterogeneous cloud database environment using multi-level cache

Seyyed Kamal Vaezpour, Mohammadreza Mollahoseini Ardakani*, Kamal Mirzaie

Department of Computer Engineering, Maybod Branch, Islamic Azad University, Maybod, Iran

(Communicated by Asadollah Aghajani)

Abstract

Mobile Cloud Computing is one of the most prominent future mobile technology infrastructures, because it accumulates the benefits of mobile computing and cloud computing, and provides optimized services for users. In a distributed database system in the cloud, the connections necessary for a query layout may be stored in multiple sites, increasing the number of equivalent designs possible in the search for an optimal query implementation plan. However, a thorough search of all possible designs in such a large search space is not computationally rational. In this study, our goal is to identify a cost model consisting of a multi-objective function with variable (and possibly conflicting) QoS parameters to solve the query optimization problem in inhomogeneous cloud databases (in terms of pricing models) and mobile. Then, we propose a new strategy for the optimization of queries in these environments using the Learning Based Optimization (TLBO) algorithm. Finally, the results are evaluated in the CloudSim environment and compared with genetic optimization and ant colony optimization (ACO).

Keywords: cloud computing, distributed database, education-learning based optimization, query optimization
2020 MSC: 35Q93, 68Q32

1 Introduction

A mobile cloud database is one of the newest and most widely used cloud technologies to provide access to data at any time and place. This technology is growing significantly, and many companies use it as one of the core principles of their technology structure.

Despite high capacity and easy access to data, mobile cloud databases are facing challenges. One of the challenges is to optimize queries in mobile cloud database environment. Due to the limitations of mobile networks and also communication networks, query optimization in this environment is very important. Optimizing queries in the mobile cloud database will reduce run time, improve database efficiency, reduce battery consumption of mobile devices, improve user experience and reduce data access costs. There are various ways to optimize queries in mobile cloud databases, including reducing the number of queries, using battery charging and unloading, using fast communications networks, numbering data, and prediction queries.

*Corresponding author

Email addresses: `mamentdepx@yahoo.com` (Seyyed Kamal Vaezpour), `managementdepx@yahoo.com` (Mohammadreza Mollahoseini Ardakani), `jkkuyk@yahoo.com` (Kamal Mirzaie)

Received: October 2023 *Accepted:* December 2023

Optimization of query in mobile databases is very important and because of their increasing formation, their efficiency improvement and stability are also on the agenda. By using this, these methods can improve the efficiency of the Mobile Cloud Database and improve the users' experience

2 Mobile cloud computing

Electronic governance has been the most important tool of e-government. This type of monitoring is primarily in the direction of the general purpose of electronic surveillance and is the most important means of achieving its goals. e-surveillance aims to reduce in-person supervision and referrals and increase the use of electronic communications in favor of efficient mechanisms [1].

Mobile devices (e.g. smartphones and tablets), as the most effective and convenient means of communication that are not limited to time and place, have become increasingly an essential part of human life [7]. The rapid development of mobile computing will become a powerful trend in the development of information technology as well as trade and industry. However, mobile devices in their own resources (such as battery life, storage and bandwidth) and communication (such as mobility and security) are faced with many challenges [3, 12]. Limited resources significantly prevent the service from improving. The increasing popularity of mobile applications and the growing demand of users have caused the limitation of resources available on these devices to become more visible. The collaboration between mobile devices and cloud computing has addressed a portion of these issues. Cloud computing offers a new opportunity for the development of mobile applications, allowing mobile devices to maintain a very thin layer of user applications and shift computing overhead toward the virtual environment. Mobile cloud computing is one of the major areas of mobile technology in the future; Because this has both the advantages of mobile processing and cloud computing, leading to the provision of optimal services for users [8, 14].

The MCC Society defines mobile cloud computing as: Mobile cloud computing is, in its simplest form, a substructure, where both storage and processing of data occur outside the mobile device. Mobile cloud applications transfer computing power and data storage away from mobile subscribers to the cloud [1]. In, Aepona describes MCC as a new model for mobile applications, whereby the processing and storage of data is transferred from mobile devices to a powerful and centralized computing platform located in the clouds. These focused applications are then accessed through a wireless connection of mobile devices based on a Thin Client or web browser. On the other hand, MCC can be defined as a combination of mobile web and cloud computing, which is the most popular tool for mobile users accessing Internet applications and services [13]. In short, mobile cloud computing provides the cloud computing and data storage services to mobile users. Mobile devices do not require powerful configuration (such as processor speed and memory capacity) because all complex computing modules can be processed in clouds [3, 6].

3 Mobile cloud architecture

According to the concept of MCC, the general architecture of MCC can be defined as that represented in Figure 1. In Figure 1, mobile devices are connected to mobile networks through base stations (such as the base transmitter/receiver station, access point or satellite). These stations create and control connections and performance interfaces between networks and mobile devices [4, 16]. Requests and information from mobile users (e.g., identifiers and location) are transferred to central processors which are connected to servers serving mobile network services. Here, mobile network operators can provide services such as authentication, authorization, and auditing, based on customer data stored in databases. After that, the customer's requests are transmitted over the Internet to a cloud. In the cloud, cloud controllers process requests to provide cloud services for mobile users. These services are developed based on the concepts of utility computing, virtualization and service-oriented architecture (e.g. web, application, and database servers) [5, 11].

4 Methodology

The goal is to optimize distributed queries in the mobile cloud by using a cost function that takes into account the parametric costs noted above and specifies these values based on user preferences. In addition, learning-based optimization approach is used to achieve the desired results. Figure 2 depicts a simple schematic of the research method.

In mobile cloud computing, the goal is to minimize the cost of implementing a query on mobile devices while using cloud computing infrastructure resources.

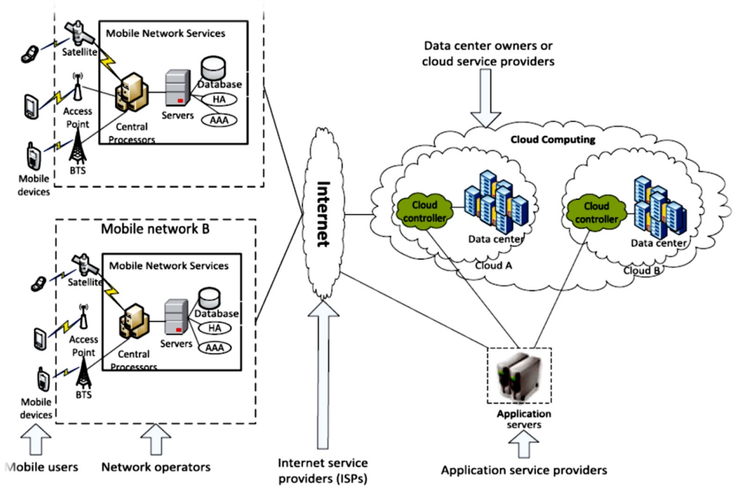

Figure 1: Mobile Cloud Computing Architecture [8]

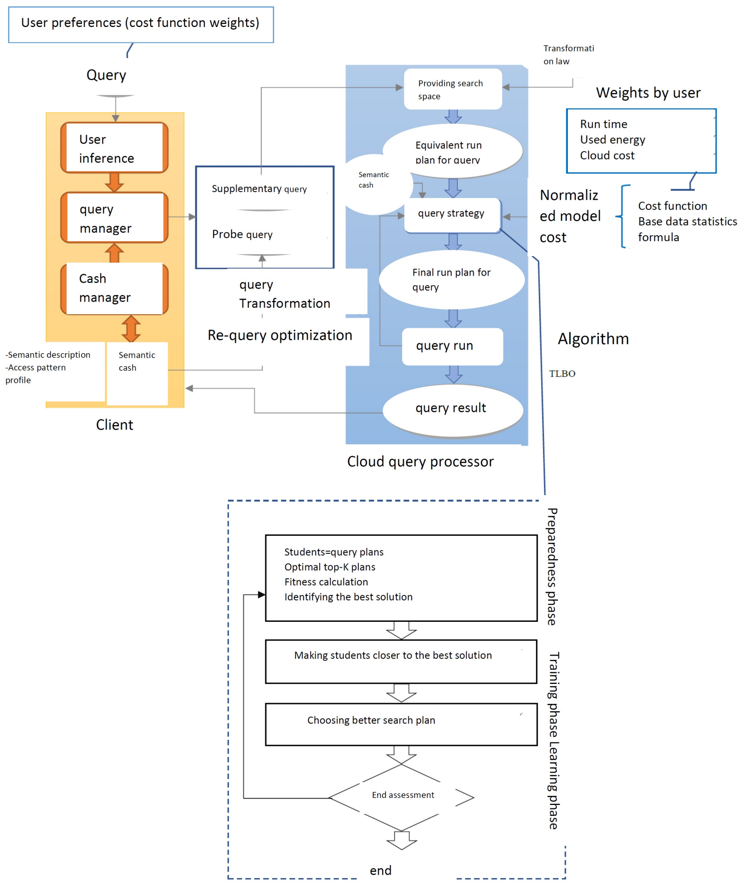

Figure 2: A simple schematic of the research method

The cost function for mobile cloud computing can be defined as a function that takes into account the following parameters:

$$\text{Total Cost} = W1 * \text{Cost\_mobile\_device} + W2 * \text{Cost\_Data\_transfer} + W3 * \text{Cost\_cloud\_computing} + W4 * \text{Latency} \quad (4.1)$$

1. Cost_mobile_device: The cost of using mobile device resources (running time and power consumption of mobile devices)

2. Cost_Data_transfer: Data transfer fee (including data size in transit, distance between source and destination and network bandwidth)
3. Cost_cloud_computing: Cost of Users to Use Cloud Computing Infrastructure
4. Latency delay: This cost can be based on the expected cost of the resources required to perform the necessary calculations or the cost of the loss of results caused by the delay

One of the adaptive methods in optimization, is using preferences and receiving feedback from users. For this purpose, we have considered adaptive high cost function w which one of the four parameters of user preference depends on its preferences.

4.1 How long to run queries on mobile devices

In mobile distributed database environments, as the data is divided into smaller formats and distributed across multiple sites, the multiplication and transformation strategy is used. One of the advantages of this method is improvement of processing power, since each site only focuses on piecewise processing of the data, and processing operations are performed in parallel on different sites, which reduces the response time of the queries.

The total cost formula for how long the queries is carried out in mobile devices based on multiplication and transformation is as follows:

$$TC_{FRS} = TC_{FR} + TC_{RP} + LPS_{AS} \tag{4.2}$$

The cost for a relationship that is fragmented in different sites is obtained from the following formula:

$$TC_{FR} = N_1 * K * T_{cost-att} * n \tag{4.3}$$

where:

N_1: The number of modified relationship tuples

K: The number of traits in relationships that have been replicated and formed in various sites.

$T_{cost-att}$: Transfer cost per adjective

n: Number of selected sites to process query

The cost for all relationships that have been reproduced on $(N_R - 1)$ will be:

$$TC_{RP} = (N_2 * K * T_{Cost-att}) * (N_R - 1) \tag{4.4}$$

N_2: The number of replicated link staples

$(N_R - 1)$: The number of sites where links are duplicated

LPS_{As}: Local processing cost on all sites (including the total cost of the links processed at the cost of the link operation)

$$LPS_{As} = \sum_{j=1}^{p} \cos t \left[\bigcup_{i=1}^{n} (FR)_i \right]_j + [(N_1 * 1 + N_2 * (N_R - 1)) * CT_{comp} + N_3 * CT_{con}] * p \tag{4.5}$$

N_3: Number of link-links tuples

CT_{comp}: Cost of comparing each row

CT_{con}: Cost of each Tapel connection

4.2 How much power mobile devices will use

One of the effective parameters in calculating the costs of mobile devices is the amount of energy consumption. We use the following formula.

$$E_{ij} = P_j * T_{ij} \quad \text{if} \quad x_{ij} = 1 \tag{4.6}$$

$$E_{ij} = 0 \quad \text{if} \quad x_{ij} = 0 \tag{4.7}$$

$$P_j = P_j^{busy} - P_j^{idle} \tag{4.8}$$

$$T_{ij} = \frac{TC_{FRS}}{F_j} \quad \text{if} \quad X_{ij} = 1 \tag{4.9}$$

$$T_{ij} = 0 \quad \text{if} \quad X_{ij} = 0 \tag{4.10}$$

P_j: How much energy consumption of the mobile device

T_{ij}: The length of execution of query i running on the mobile device j

$X_{ij} = 1$: Query i available on j device

$X_{ij} = 0$: No i query available on j device

P_j^{busy}: High Power Consumption of Mobile Device in Busy Mode

P_j^{idle}: Power consumption of mobile device in idle mode

F_j: Mobile device computing capacity (number of operations per unit of multi-core processor time)

TC_{FRS}: Execution Cost based on Duplication and Segmentation Method

4.3 Data transfer fee between mobile devices

To get the data transmission cost, we need to consider several variables such as data size being transmitted, the distance between the data source and the destination, and the available network bandwidth to transmit.

Here are a few steps that can be followed to formulate data transmission costs in the cost model:

1. Calculating the size of the data in transit: This can be done by determining the number of bytes or bits to move.
2. Determining distance between the data source and destination: distance can be measured in network jump, physical distance or delay.
3. Estimation of the available network bandwidth for transmission: This can be based on network infrastructure, type of connection (for example, wired, wireless), and network congestion.
4. Calculating the data transfer cost: This can be done by using a formula that takes into account the size of the data, the distance and the bandwidth of the network.

$$\text{Cost Data Transfer} = \text{Data size} * \left(\frac{\text{Distance}}{\text{Network band width}}\right) \tag{4.11}$$

This formula calculates the cost of data transmission based on the product of data size and data transmission time over the network. Time is calculated by the amount of available bandwidth.

4.4 Cost users pay to use cloud computing infrastructure

Costs paid by users for the use of cloud computing infrastructure vary depending on factors. Some factors that influence the cost of using cloud computing are:

1. Type of service used: The cost of using different cloud computing services varies depending on the type of service. For example, cloud storage services typically cost less than cloud computing services [11].
2. Server placement: The cost of cloud computing depends on the location of servers. For example, cloud computing usage rates may be lower in countries with lower electricity costs.
3. The amount of resources: the cost of using cloud computing depends on the amount of resources consumed by the user. As the user consumes most of their required computing resources from cloud-based servers, the cost of cloud computing will be increased.
4. Type of Contract: The cost of using cloud computing depends on the type of contract; For example, annual contracts typically cost less than monthly contracts.
5. Service level: The cost of cloud computing depends on the level of the desired service. Support for a high level of service may lead to a higher cost than a lower level of service [2].

4.5 Delay

In mobile cloud query optimization, latency is the amount of time that passes until a query from the user's device reaches the cloud server and its response is returned to the user's device. You can use the concept Network delay to formulate latency in optimization of query in cloud.

The network delay consists of three steps:

1. transmission delay: The time is spent moving packets of data from the user's device to the server or from the server to the user's device. For each packet, the delay will be equal to the packet size of Internet bandwidth.
2. processing delay: is the time spent processing data packets on the server. This delay is equal to the expression the server requires to process query.
3. queueing delay: is the time when the data packets are in the waiting queue to be processed on the server. This delay equates to the average time that packets are queued.

More advanced formulas can be used in the context of latency formulation in optimization of query in the mobile cloud. One of these formulae is the Optimal Network Delay Formula:

$$\text{Network Delay} = (T_s + T_w) + \left[\frac{T_r + T_p}{1-\beta}\right]. \tag{4.12}$$

To calculate any of the parameters of the Optical Network Latency Formula, however, one must look at the status and condition of the mobile cloud in which the query operation is performed and their values may change in different conditions. But in general, the way in which each of these parameters is computed can be provided as follows: Server Transmission Time T_s is the time the server decides to send a response to the user's device. This results in the division of response data volumes over the Internet bandwidth. Wireless Transmission Time T_w is the time spent sending a query from the user to the cloud-based server. This time comes from dividing query data over wireless network bandwidth. T_r Round-Trip Time is the time it takes to complete a trip from the user's device to the cloud-based server and back to the user's device. This time is twice as long as the transmission delay. Processing Time is the time spent processing query on the cloud-based server. This time may vary depending on server conditions and processes running on the query. β Coefficient of Variation This parameter represents the variance coefficient of the waiting time in the queue. In other words, it indicates the rate of spread waiting in the queue.

4.6 semantic cache

Semantic memory can be a powerful tool to improve the performance of distributed query processing in mobile cloud computing environments. By storing common data in cache and using an optimizer method for cache management, it is possible to reduce the number of network access to complete queries and speed up the response time to queries [17].

4.7 Semantic cache cost model

The Semantic Cache Cost Model should consider the cost of accessing data from the cache, as well as the cost of accessing data from remote storage. Semantic cache can be incorporated into the cost function formula used to optimize distributed query.

$$\text{Cost}(q.S) = \alpha * \text{Cost\_Remote}(q.S) + \beta * \text{Cost\_cache}(q.S) \tag{4.13}$$

Cost(q.S) is the total cost of executing query q in system S.

Cost_remote(q.S) is the cost of executing query q in system S.

Cost_cache(q.S) is the cost of accessing data from the cache in S system.

α, β are allocated weights that depend on factors such as the rate of cache, cache size, and cost of accessing data via the network.

Cost_remote(q, S) The cost of executing query q in system S, the same time as the whole query execution we explained in the previous contents.

4.8 Cost of access to data in semantic cache

Query Manager will first analyze the exported query to determine if it can be answered locally. If all required data items are available in local cache, query is converted into a local query and is termed fully responsive. If only a part of the query can be locally evaluated, it will be called a partially responsive query and the query will be converted to a probe query and a complete query. A query probe is a local query that retrieves data items from the client cache while sending a full query for evaluation based on the preferences specified by the user for the query to the appropriate cloud query processor. To determine the best cloud query processor, we will use the Cache Optimizer.

Optimization of cache memory is very important in the distributed Mobile Cloud database environment. In this environment, the data is distributed on different servers and it requires a connection with different servers to access the data. Optimization based on cache reduced the number of requests to the servers and as a result, response time is reduced. If it is possible to obtain data from semantic cache, it saves computational time because cache access is faster than database access.

In distributed systems, distributed semantic cache can be used. In this case, the required data for each query is located near the client server. This can reduce data access costs and improve system performance. In this way, each server in the network is responsible for maintaining and updating the semantic cache. For example, each server can update its cache for the data that is most requested. Also, if a server does not have the data required by query, it will receive data from another server on the network. In order to select the best server, we must prioritize all servers based on the following parameters and obtain data from it. We obtain the following values for each i server where I server queries have been copied under the original query.

1. The current queue length Q(Li): The number of processes that are running for execution on the server.
2. Remote SD Server (i): The server's remote location refers to the server's geographic distance from the client. This means that the closer the server is, the less time it takes to fetch the data from the server.
3. Capacity SC(i) Server: The number of processes that can be executed on a server without disrupting normal server operations.
4. Load Server (Li): The ratio of actual processes running on a server to its server capacity.

In the next step, we will prioritize each database that has been copied under query, the main query where it was copied. Suppose that the query is available under two sites i and j, how to prioritize is done by the following relationships.

$$Pr = Pr + \left(Q(li) - \frac{Q(lj)}{10}\right) \tag{4.14}$$

$$Pr = Pr + \frac{SD(i) - SD(j)}{100} \tag{4.15}$$

$$L(i) = Pr + \frac{QL(i)}{SC(i)} \tag{4.16}$$

$$L(j) = Pr + \frac{QL(j)}{SC(j)} \tag{4.17}$$

$$Pr = Pr + (L(i) - L(j)) \tag{4.18}$$

in this algorithm, we first prioritize all servers based on the above parameters and then send a subquery to the server with the highest priority and check the local cache of the server. If the data is present in the cache, we send the data to the client server or it is fetched from the global database.

4.9 Education-learning optimizer

Learning-based optimization is a meta-heuristic optimization algorithm that can be used to solve a wide range of optimization problems. In this study, with the help of education-learning algorithm, the most optimized execution design in the search space which contains the same execution plans as each query is selected and executed. Accordingly, we consider interior search space designs as a population of learners or students, and then consider one of the queries as a teacher at random, and obtain the phase of the teacher based on the following formula [9].

$$X_{newD} = X_{oldD} + r(X_{teacherD} - T_F M_D). \tag{4.19}$$

The index D represents the number of queries (problem variable), the old member X_{oldD}, which still needs to learn from the teacher to increase the level of knowledge, and consists of a vector $1 * D$ which contains the result of any particular subject or lesson. r is a random number in range (1 and 0), $X_{teacher}$ and D is the best member of the population in this iteration, striving to change the mean of the class (population) towards its own position. T_F Teaching Factor, M_D is a vector $1 * D$ that contains the average values of the class results for each subject. The T_F value may be 1 or 2, and it is considered to be an innovative step and is randomly chosen with equal probability. New X_{newD} member is accepted if it is better than older member [15].

Learning Phase: Learners use two different methods, one through the teacher and the other through interaction between themselves, to increase their knowledge. A learner interacts at random with other learners using group discussions, presentations, and more. A learner will learn new things from other students, provided others have a high knowledge of him.

$$X_{new\ i} = X_{old\ i} + r(X_j - X_K). \tag{4.20}$$

The index i varies from one to the total number of members, $X_{old\ i}$ is an old member who has not learned anything from cross-transaction or other students, r_i is a random number between interval (1 and 0) and X_j and X_K are two transactions (students) chosen randomly, with $j \neq k$ condition and condition that X_j's objective function is better than X_k. The new member of $X_{new\ i}$, is accepted if he is better than the older member of $X_{old\ i}$. The TLBO algorithm is as follows:

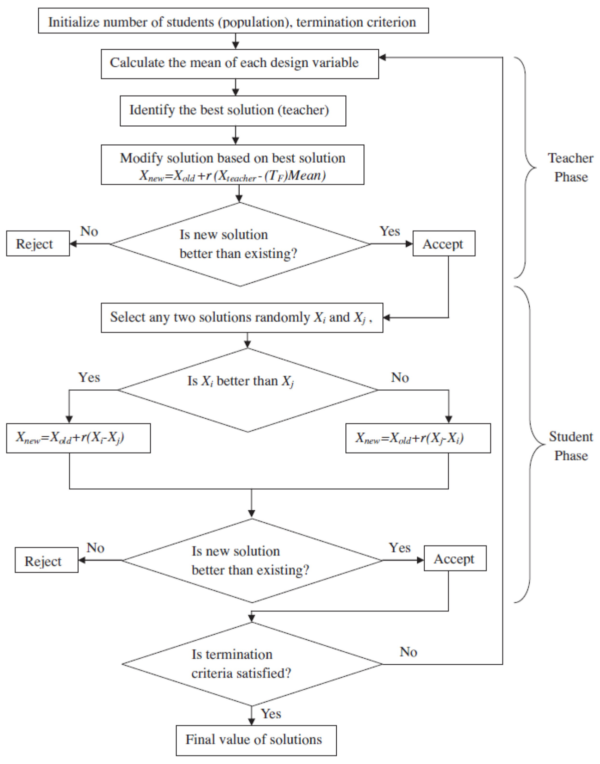
Initialize number of students (population), termination criterion
Calculate the mean of each design variable
Identify the best solution (teacher)
Modify solution based on best solution $X_{new}=X_{old}+r(X_{teacher}-(T_F)Mean)$
Is new solution better than existing?
No
Yes
Reject
Accept
Teacher Phase
Select any two solutions randomly X_i and X_j.
Is X_i better than X_j
Yes
No
$X_{new}=X_{old}+r(X_i-X_j)$
$X_{new}=X_{old}+r(X_j-X_i)$
Student Phase
Is new solution better than existing?
No
Yes
Reject
Accept
Is termination criteria satisfied?
No
Yes
Final value of solutions

Figure 3: Flowchart Algorithm TLBO [10]

4.10 Evaluation and practical outcomes

To assess the proposed approach, the following steps were taken:

The definition of the criteria was considered: first, the criteria that are supposed to be used for evaluation of the performance of the adaptive optimization algorithm of multi-objective query was determined. The criteria include running query time (time required to execute a query) and the amount of power consumption of mobile devices (the amount of energy expended by mobile devices during query execution). Select an evaluation dataset: To perform fair and meaningful comparisons during the validation process, a data set was selected that reflects the characteristics of an inhomogeneous/mobile cloud database environment. In this study an Artificial Business Environment Data Set (TPC-H) was used to produce the evaluated data set. The TPC-H criterion is a widely used benchmark for decision support systems, consisting of a complex set of SQL queries. Figure 4 shows database entities and their relationships, used by SQL queries to the TPC-H dataset.

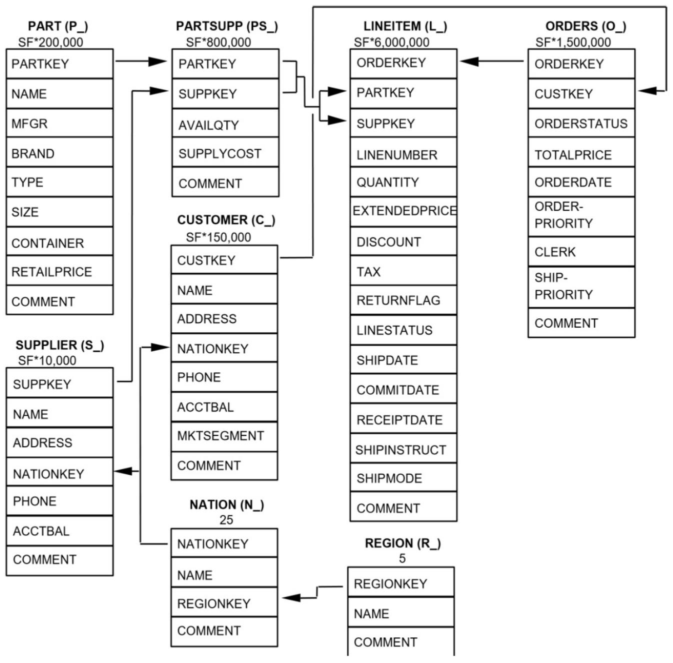

Figure 4: Database entities and the relationships between them used by the TPC-H dataset

5 Simulation results

In this simulation, three optimization algorithms of education-learning, ant colony and genetic were used for cost model optimization and performance of each algorithm was compared and evaluated based on two criteria of query execution and energy consumption of mobile devices. In order to simulate the data, the topology of the hierarchical network has been used for the datacenter. The number of virtual machines (VMs) was considered equal or greater than the number of hosts. Virtual machines were also run on the same number of tasks. Each test was carried out over 10 times. The results obtained from the simulation are given in the following tables with three algorithms.

Table 1: teaching-learning algorithm

Number of queries	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**
Execution time (milliseconds)	12	25	50	74	126	172	249	310	380

Table 2: Genetic algorithm

Number of queries	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**
Execution time (milliseconds)	20	45	73	112	173	248	315	399	481

Table 3: Ant colony algorithm

Number of queries	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**
Execution time (milliseconds)	31	62	101	152	236	299	384	483	599

Table 4: Learning algorithm

Number of queries	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
Execution energy (kilohertz)	80	141	222	301	341	410	538	602	697	771	843

Table 5: Genetic algorithm

Number of queries	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
Execution energy (kilohertz)	98	182	252	305	348	445	579	648	738	837	928

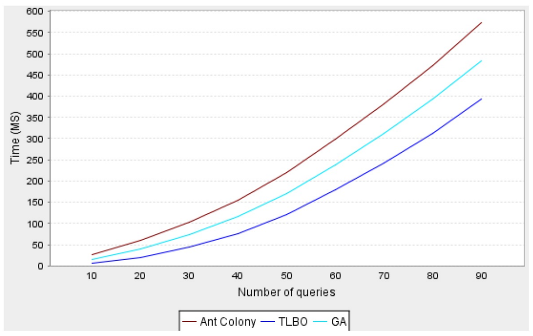

Figure 5: Comparison of the execution time of the queries of the algorithm

Table 6: Ant colony algorithm

Number of queries	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
Execution energy (kilohertz)	105	195	271	312	385	585	748	867	978	1078	1185

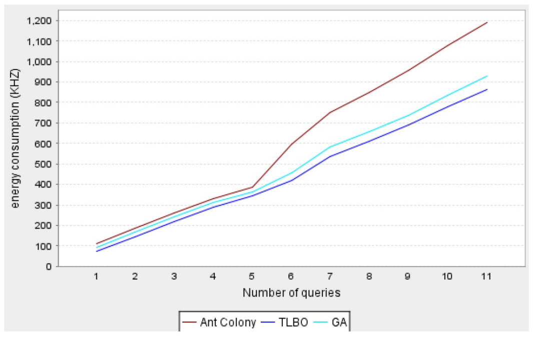

Figure 6: Comparison of energy consumption of different algorithms

6 Conclusion

The simulation results show that with the increase in the number of queries, the query execution time increases, but the increase in the query execution time in the learner training method is less than the two-ant colony and genetic algorithms. Also, with the increase of queries, the amount of energy consumption in the mobile cloud environment increases in three algorithms, but the slope of the increase in the amount of energy consumption in the learner training method is lower than the two methods of ant colony and genetics, and as the number of queries increases, the learner training algorithm performs better than the two methods. Ant colony method and genetics.

References

[1] A.E.J. Akbari, Z. Fathi, and M. Minouei, *Identifying the effective components on promoting tax capacity in e-commerce and providing tools based on identified categories*, Tobacco Regul. Sci. **8** (2022), no. 1, 1114–1142.

[2] T. Alyas, A. Alzahrani, Y. Alsaawy, K. Alissa, Q. Abbas, and N. Tabassum, *Query optimization framework for graph database in cloud dew environment*, Comput. Mater. Contin. **74** (2023), no. 1, 2317–2330.

[3] J.H. Christensen, *Using RESTful web-services and cloud computing to create next generation mobile applications*, Proc. 24th ACM SIGPLAN Conf. Compan. Object Oriented Program. Syst. Lang. Appl., 2009, pp. 627–634.

[4] H.T. Dinh, C. Lee, D. Niyato, and P. Wang, *A survey of mobile cloud computing: architecture, applications, and approaches*, Wireless Commun. Mobile Comput. **13** (2013), no. 18, 1587–1611.

[5] D.B. Gordon and S.L. Mayo, *Branch-and-terminate: A combinatorial optimization algorithm for protein design*, Structure **7** (1999), no. 9, 1089–1098.

[6] S. Gros, *Distributed query optimization*, Master's thesis, https://api.semanticscholar.org/CorpusID:211528213, 2020.

[7] F. Helff, L. Gruenwald, and L. d'Orazio, *Weighted sum model for multi-objective query optimization for mobile-cloud database environments*, EDBT/ICDT Workshops, 2016.

[8] L. Liu, R. Moulic and D. Shea, *Cloud service portal for mobile device management*, IEEE 7th Int. Conf. E-Bus. Engin., IEEE, 2010, pp. 474–478.

[9] P. Michiardi, D. Carra, and S. Migliorini, *Cache-based multi-query optimization for data-intensive scalable computing frameworks*, Inf. Syst. Front. **23** (2021), 35–51.

[10] V. Mishra and V. Singh, *Generating optimal query plans for distributed query processing using teacher-learner based optimization*, Proc. Comput. Sci. **54** (2015), 281–290.

[11] T. Niknam, A. Kavousi, and A. Baziar, *Multi-objective stochastic distribution feeder reconfiguration problem considering hydrogen and thermal energy production by fuel cell power plants*, Energy **42** (2012), no. 1, 563–573.

[12] A. Oludele and O. Oluwabukola, *A survey of mobile cloud computing applications: Perspectives and challenges*, 7th Int. Multi-Conf. Complex. Inf. Cybern. IMCIC 2016 7th Int. Conf. Soc. Inf. Technol. ICSIT 2016 Proc., 2016, pp. 238–243.

[13] M.T. Özsu and P. Valduriez, *Principles of Distributed Database Systems*, Prentice Hall, Englewood Cliffs, 1999.

[14] M. Perrin, J. Mullen, F. Helff, L. Gruenwald, and L. d'Orazio, *Time-, energy-, and monetary cost-aware cache design for a mobile-cloud database system*, Biomedical Data Management and Graph Online Querying: VLDB 2015 Workshops, Big-O (Q) and DMAH, Waikoloa, HI, USA, Springer International Publishing, 2016, pp. 71–85.

[15] R.V. Rao, V.J. Savsani, and D.P. Vakharia, *Teaching-learning-based optimization: A novel method for constrained mechanical design optimization problems*, CAD Comput. Aided Des. **43** (2011), no. 3, 303–315.

[16] M. Satyanarayanan, *Mobile computing: the next decade*, Proc. 1st ACM Workshop on Mobile Cloud Comput. Serv.: Soc. Networks and Beyond (MCS), 2010, no 5.

[17] M. Ul Hassan, A.A. Al-Awady, A. Ali, M.M. Iqbal, M. Akram, J. Khan, and A.A. AbuOdeh, *An efficient dynamic decision-based task optimization and scheduling approach for microservice-based cost management in mobile cloud computing applications*, Pervasive Mob. Comput. **92** (2023), 101785.