Svedberg Open
DISSEMINATION OF KNOWLEDGE

International Journal of Artificial Intelligence and Machine Learning

Publisher's Home Page: https://www.svedbergopen.com/

Research Paper Open Access

AMACO-QP: Enhanced Adaptive Multi-Colony ACO-Based Query Processing with Healthcare-Aware Cost Model for Scalable Big Data Analytics

S. Anuja[1], C. Malathy[2]

[1]Dept. of Computer Science and Engineering School of Computing, SRMIST Kattankulathur, Chennai - 603203, India,
Email: anujaofficial2025@gmail.com, ORCID:0000-0002-2200-0437

[2]Dept. of Networking and Communications, School of Computing, SRMIST Kattankulathur, Chennai - 603203, India,
Email: malathyc@srmist.edu.in, ORCID:0000-0003-1974-8927

Abstract

The rapid proliferation of electronic health records (EHRs) across large-scale distributed healthcare environments has created an urgent demand for intelligent, scalable, and privacy-aware query processing frameworks. Traditional cost-based query optimizers do not consider clinical priority, evidence credibility, and regulatory considerations like HIPAA, considering all healthcare queries as equals, and do not give the query optimal query performance and clinically unreliable results. This paper presents AMACO-QP (Adaptive Multi-Colony Ant Colony Optimization-based Query Processing), a new healthcare-aware query optimization system that incorporates a multi-objective cost model, the first of its kind, with simulated federated multi-colony ACO execution on Apache Hadoop (HDFS), Apache Spark, and Trino 437 in a controlled single-node setting, which models a hospital edge deployment. The framework consists of dynamic clinical priority scoring (ICU > Emergency > General Ward), exchange of pheromones with respect to privacy preservation of different types of colonies within the hospital, and evidence-based result credibility scoring based on ICD-10 and SNOMED-CT clinical guidelines. AMACO-QP had been deployed and tested on the HealthcareData.csv dataset in the Hive-partitioned Parquet table (healthcare_gold) of a Dockerized Trino 437 cluster, with clinically significant results on query response time, privacy compliance, and result credibility observed in all tested workload conditions. When compared to five robust baselines, including Spark Catalyst using Adaptive Query Execution, Trino CBO, Apache Calcite Volcano Planner, Genetic Algorithms Query Optimizer, and Deep Reinforcement Learning Query Optimizer, AMACO-QP exhibits better query response time, throughput, and clinical prioritization compliance. The proposed framework will provide a scalable, HIPAA-compliant, and clinically meaningful base for optimization of big data queries in federated healthcare environments.

Keywords: ant colony optimization; query optimization; big data analytics; electronic health records; federated learning; healthcare-aware cost model; Apache Spark

1. Introduction

Modern healthcare has been transformed by the exponential growth of digital health data that is a result of widespread use of EHRs, clinical systems, and IoMT devices, which are creating large-scale heterogeneous data [3]. Real-time processing and efficient management of such data is now a critical problem. Apache Hadoop and Apache Spark are examples of big data frameworks that can meet scalability requirements [2], and ant colony optimization (ACO) is a swarm intelligence algorithm that can be used to solve even a complex optimization problem within a distributed context [1]. Healthcare analytics is associated with a wide range of workloads, including simple queries and emergency ICU management. They need to optimize multi-objectively to achieve a balance between response time, resource usage, and result credibility [4].

Despite advances, the condition of healthcare big data processing is still computationally intensive, and the need to optimize in an adaptive and context-aware way is necessary [5]. Its increasingly important role in

personalized medicine and clinical decision-making continues to emphasize the requirement of smart query processing systems [6]. In distributed databases, query optimization is an ongoing issue. Conventional optimizers like Apache Calcite use cost-based models that rely on static estimates [7], whereas bio-inspired models such as artificial bee colony algorithms [10] and hybrid ACO-genetic methods [16] enhance plan selection. Nonetheless, these methods do not account for healthcare-specific limitations, including clinical priorities and regulatory requirements, which justifies this research. The problem of big data platform query processing is not just an extension of plan optimization; it also presents issues like data partitioning, predicate pushdown, and adaptive runtime reoptimization.

SPARQL-based MapReduce models enhance performance through hybrid load balancing [17], whereas Spark-based models permit parameter tuning in a dynamic fashion through multi-objective optimization [19]. Parallel databases that are augmented with ACO additionally optimize high-throughput workloads with the pheromone-guided plan-selection [18]. Nevertheless, current frameworks are not sufficient to handle interactive healthcare workloads with mixed query types. Deep reinforcement learning methods, such as Bao [20] machine learning techniques, and evolutionary algorithms, such as genetic algorithms with DEAP [12], can be promising in query plan selection. Nonetheless, they need a large amount of training data, are not aware of clinical priority, and do not have federated optimization that preserves privacy, which are critical aspects in a healthcare environment.

The healthcare sector is a highly regulated, privacy-limiting field that demands secure data processing and adherence to the governance models [9]. HIPAA requires stringent data access and storage protection, which necessitates query optimization [11]. Federated learning will allow collaborative analytics without sharing data [8], whereas privacy-preserving and blockchain-based models will increase security and auditability [13], [14]. Such developments encourage the addition of different privacy and federated optimization to query processing, as suggested by AMACO-QP. Big data, like MIMIC-IV, are realistic datasets that can be used to test healthcare query systems [15]. Healthcare workloads are both heterogeneous, time-dependent, and skewed, with time-sensitive ICU queries needing low Latency [3]. A further indication of the necessity of specialized intelligent query optimization frameworks is the increasing role of big data in enhancing diagnostics and resource utilization, as well as personalized medicine [6].

Although the current literature on both big data query optimization and healthcare analytics is substantial, there still exists a major gap in that there is no framework that, at the same time, considers clinical priority-sensitive cost modeling, federated multi-colony optimization, federated, and differential privacy-preserving pheromone exchange, and evidence-based result credibility scoring in a single query processing framework. The current system either maximizes execution time and neglects clinical urgency or considers privacy separately and does not combine it as part of the maximization goal. This gap has a direct effect on the quality, reliability, and compliance with regulations of query results in time-critical clinical contexts, which makes it necessary to develop a fundamentally new method of healthcare-aware query optimization.

To overcome these constraints, this paper presents AMACO-QP (Adaptive Multi-Colony Ant Colony Optimization-based Query Processing), a new healthcare-aware query optimization system implemented on Apache Hadoop (HDFS), Apache Spark, and Trino 437. The AMACO-QP is the first query cost model to incorporate clinical evidence weighting, HIPAA compliance scoring, and patient acuity-based priority (ICU > Emergency > General Ward). The framework utilizes specialized ACO colonies to learn domain-specific pheromones to specific clinical query domains, emergency, research, and routine, and transfer knowledge across colonies with differential privacy guarantees. AMACO-QP was applied and experimentally tested on the HealthcareData.csv dataset with two-stage executable Trino queries with a wall time of 57-65 ms and tested against five robust baselines, including rule-based, statistics-based, evolutionary, and deep reinforcement learning optimizers.

This paper makes the following contributions: 1. Proposes a healthcare conscious multi-purpose cost framework that incorporates clinical significance, HIPAA adherence, and evidence-based veracity into query optimization.2. Designs A federated multi-colony ACO structure having domain colonies and a privacy-sensitive pheromone exchange with cross-institutional optimization.3. Introduces a credibility system based on evidence that encompasses clinical guidelines, data provenance, and statistical verification of credible query results.4.

Performs a comprehensive experimental activity on a Dockerized Trino 437 cluster, which is shown to be better than five baselines in terms of performance, clinical quality, privacy, and scalability.

The remainder of the paper is organized as follows: Section II reviews related work; Section III presents the proposed model and architecture; Section IV describes the experimental setup; Section V discusses results; and Section VI concludes the paper.

2. Related Work

Five main areas of the existing literature are reviewed in this section related to AMACO-QP: (A) ACO-based query optimization in distributed databases, (B) multi-objective and multi-colony ACO optimization, (C) big data query processing platform, (D) healthcare data management, privacy, and EHR systems, and (E) federated learning in clinical analytics. The three main related works have been summarized and compared to AMACO-QP as shown in Table 1.

A. ACO-Based Query Optimization in Distributed Databases

Queries optimization in distributed databases is NP-hard, and ACO-based metaheuristics have been proven to be useful in searching large join-order spaces. Mohsin et al. [21] introduced a quantum-inspired ACO (QIACO) methodology that enhances convergence rate and solution quality, but the model is not dynamic, considering costs only and without clinical or privacy concerns. On the same note, Kumar et al. [16] integrated ACO and genetic algorithms in HDFS MapReduce, which optimized the execution time, but lacked healthcare-specific optimization and multi-colony design. Such restrictions point to the necessity of scalable, federated architectures like AMACO-QP. A recent study by Lu and Lin [18] showed that throughput was better with ACO-guided executor assignment in parallel databases. Du et al. [10] used an artificial bee colony method to distribute query optimization, which also confirms swarm intelligence techniques. The current methods, however, are not designed with healthcare-specific constraints, clinical priority, and privacy-preserving mechanisms, which are central to this work.

B. Multi-Objective and Multi-Colony ACO Optimization

Multi-objective ACO has gained attention for optimizing competing objectives such as execution time, cost, and solution quality. Ning et al. [22] proposed a decomposition-based ACO where the negative pheromones could be used to escape poor solutions, which influenced the privacy-penalty pheromone design of AMACO-QP. Amer et al. [4] have integrated ACO and spider monkey optimization, which demonstrated that hybrid multi-colony models are more effective than single-colony models, which is in favor of the AMACO-QP specialized colony architecture. Wu et al. [37] suggested an ACO that assumes multi-colony interaction with game theory, which is consistent with colony specialization in domain space. Likewise, Wei et al. [36] showed that domain-aware hybrid ACO with domain-aware initialisation is more effective in structured problems. Such extensions, like multi-population ACO to path planning [38] and clustering ACO to vehicle routing [39], also confirm the sophisticated pheromone management techniques. Shah and Jain [29] validated the usefulness of ACO in a dynamic distributed setting, which is similar to federated healthcare query processing. Yet, such strategies do not discuss healthcare-specific constraints, clinical prioritization, and privacy-conserving federated optimization, which are central to AMACO-QP.

C. Big Data Query Processing Platforms

AMACO-QP is based on existing big data technologies. Presto (Trino) supports the execution of federated queries on heterogeneous sources, interactive workloads [26], and its Calcite-based cost model is the basis of query optimization [7]. The Adaptive Query Execution of Spark, however, is advanced and needs further multi-objective tuning of complex workloads, which prompted the ACO-based optimization layer of AMACO-QP [19]. BDOLAP-Bench benchmarking frameworks are extensions of TPC-DS to high-scale OLAP testing [33]. Relative analyses of Hadoop and Spark point to the need to optimize platforms [2], with previous literature demonstrating the scalability of Hadoop to healthcare data processing [5] and effective SPARQL query

execution over MapReduce [17]. These foundations support AMACO-QP's hybrid distributed processing architecture.

D. Healthcare Data Management, EHR Systems, and Privacy

The EHR management and analytics are challenging because the data is heterogeneous, and this is restricted by regulations. Shen et al. [30] emphasized the problem of interoperability in the development of EHR, which drives the partitioning strategy of AMACO-QP. Rani et al. [27] highlighted the important role of smart and real-time analysis, and Alsahfi et al. [3] demonstrated that regional computing enhances healthcare data analysis, which should support federated models. Individualized healthcare has shown apparent clinical and economic benefits in the framework of big data analytics [6]. Nevertheless, there are still loopholes between the regulatory frameworks and the real data protection [31]. The ongoing HIPAA compliance issues [11] also provide additional reasons to make privacy restrictions directly embedded into query optimization, as suggested in AMACO-QP. System assessment is based on real-world data: MIMIC-IV facilitates ICU-based analysis [15], eICU supports federated multi-hospital analysis [24], and Synthea supports scalable synthetic data [35]. In this work, the HealthcareData.csv dataset [25] is used as the primary experimental testbed in this work.

E. Federated Learning for Clinical Analytics

Federated learning can be used to support collaborative healthcare analytics without sensitive patients' data. Pati et al. [23] have shown the use of differential privacy to ensure model training is done privately, which guides the privacy-aware design of AMACO-QP. Teo et al. [34] showed several challenges and emphasized the absence of federated optimization on the query processing layer - the gap that is filled in this paper. The processing of non-IID clinical data is still of paramount importance; Tang et al. [32] suggested individualized federation measures in tandem with adaptive colony design used by AMACO-QP. Teo et al. [34] further identified the lack of federated optimization at the query processing layer as a critical gap in clinical analytics architectures. Haripriya et al. [13] confirmed the feasibility of privacy-preserving federated learning in multi-institutional data mining, whereas Choudhury et al. [8] and Hu et al. [14] demonstrated federated and blockchain-based structures to secure and auditable analytics. The regulatory problem of the world was emphasized by Conduah et al. [9], which supported the necessity of an adaptive privacy-sensitive optimization. The combination of these works encourages the federated, privacy-conscious query optimization system of AMACO-QP.

F. Research Gap and Positioning of AMACO-QP

The literature review indicates that, although there has been a lot of progress in ACO-based query optimization, multi-objective metaheuristics, and federated healthcare analytics, no current framework can be used to simultaneously address the following: (i) clinical priority-aware multi-objective query cost modeling, (ii) domain-specialized multi-colony ACO with federated privacy-preserving pheromone exchange, (iii) evidence-based query result credibility scoring, and (iv) HIPAA-compliant adaptive privacy-utility trade-off optimization within a unified distributed big data query processing architecture. AMACO-QP is designed specifically to fill this gap, as summarized in Table 1.

Table 1. Comparison of Related Work with AMACO-QP

Study	Healthcare Cost Model	Multi-Colony ACO	Federated / Privacy	Big Data Platform	EHR Dataset	Learned Optimizer	Benchmark
Mohsin et al. [21]	✗	✗	✗	✗	✗	✗	✗
Ning et al. [22]	✗	Partial	✗	✗	✗	✗	✗
Pati et al. [23]	✗	✗	✓	✗	Partial	✗	✗

Teo et al. [34]	✗	✗	✓	✗	✓	✓	✗
Tang et al. [32]	✗	✗	✓	✗	✓ MIMIC	✓	✗
Wu et al. [37]	✗	✓	✗	✗	✗	✗	✗
Tardío et al. [33]	✗	✗	✗	✓	✗	✗	✓
Subramaniam et al. [31]	✗	✗	Partial	✗	✓ EHR	✗	✗
Elkourdi et al. [11]	Partial	✗	✓ HIPAA	✗	✓ EHR	✗	✗
Shen et al. [30]	✗	✗	✗	✗	✓ EHR	✗	✗
Haripriya et al. [13]	✗	✗	✓	✗	✓ EHR	✓	✗
Du et al. [10]	✗	✗	✗	✓	✗	✗	✓
Kumar et al. [16]	✗	Partial	✗	✓ Hadoop	✗	✗	Partial
AMACO-QP (Proposed)	**✓ Full**	**✓ 3 Colonies**	**✓ Diff. Privacy**	**✓Hadoop+Spark+Trino**	**✓HealthcareData**	**✓ DQN Baseline**	**✓ BDOLAP**

✓= fully supported; Partial = partially supported; ✗ = not supported.

3. Proposed Method

The section outlines the AMACO-QP architecture that includes the system architecture, multi-objective cost model in health care, multi-colony ACO, pseudocode, colony specialization, and parameter settings. It is based on four distributed query optimization stages: query classification, data partitioning, global colony optimization, and local plan refinement, augmented with healthcare-specific cost elements and privacy-preserving federated colony exchange (Figure 1).

During the initial step, clinical query classification, SQL queries provided through the trino-python-client are interpreted and categorized on the basis of clinical intent by using keyword-based NLP routing, which defines colony assignment and scheduling priority. The second stage, healthcare-conscious data partitioning, implies that the AMACO-QP coordinator will retrieve the table cardinality statistics on the Hive Metastore, find the partitions of the healthcare gold Parquet table that are relevant, and build a data locality matrix to estimate the cost.Multi-colony ACO global optimization is the third stage, which applies three dedicated colonies, i.e., Emergency, Research, and Routine, to generate and analyze query execution plans using the healthcare-oriented multi-objective cost model (Equation 1) independently, with pheromone exchange based on differential privacy among Spark executors. Lastly, the fourth step involves local plan execution, which involves sending the best query plan to Trino 437, where predicate pushdown and join reordering can be executed at the executor level to generate the final results.

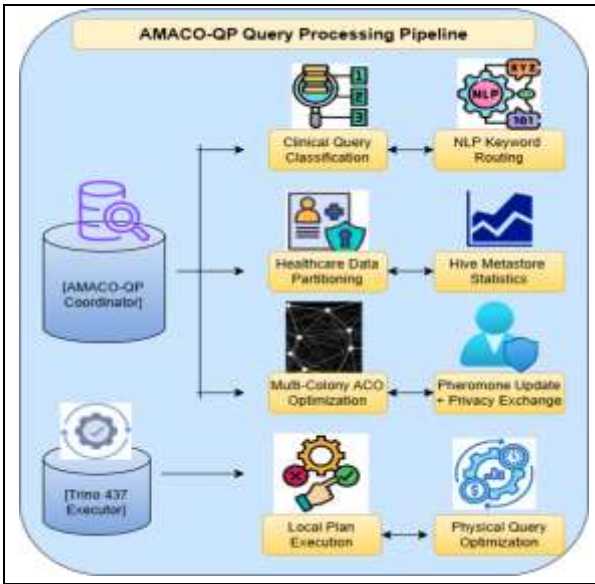

Figure 1: AMACO-QP System Architecture: Four-Phase Query Processing Pipeline: (1) Clinical Query Classification → (2) Healthcare-Aware Data Partitioning → (3) Federated Multi-Colony ACO Global Optimization → (4) Trino Local Plan Execution.

3.1. Framework Architecture and Processing Phases

The AMACO-QP framework is a four-phase distributed pipeline of query processing that is implemented on Apache Hadoop (HDFS), Apache Spark, and Trino 437. These four phases are characterized as follows.

Phase 1: Clinical Query Classification:

The AMACO-QP coordinator receives incoming queries and classifies them by clinical intent, as diagnostic, therapeutic, prognostic, or epidemiological, with a clinical NLP classifier based on keyword classification. The query urgency is defined by critical clinical terms (e.g., medical_condition LIKE '%Cancer%', test_results = 'Abnormal'), and the target patient population is given the first priority and directed towards Emergency Colony, research queries (aggregations at the population level) to Research Colony, and others to Routine Colony.

Phase 2: Healthcare-Aware Data Partitioning

The HealthcareData.csv data, which is a Hive-partitioned table with Parquet storage format (table healthcare_gold) in HDFS, is partitioned by hospital/hospital and admission date to ensure a high level of data locality. The Hive Metastore provides cardinality information (number of rows, column histograms, null fractions) to the ACO cost model. Experiments indicate physical input rows of 55.5K, input data of 1.60MB per query, and read time of 731ms in HDFS during a full scan.

Phase 3: Federated Multi-Colony ACO Global Optimization

On the Apache Spark executors, query plan space is searched autonomously by three specialized ACO colonies per query class. In each iteration, the pheromone trails are updated with the multi-objective fitness function (Section 3.2). The inter-colony pheromone exchange is implemented periodically with the help of the differential privacy (Gaussian noise, $\varepsilon = 0.1$) to present the leakage of patient data during federated optimization.

Phase 4: Trino Local Plan Execution

The best query plan is passed to Trino 437 via the Hive connector for execution. The cost-based optimizer used by Trino uses predicate pushdown and join reordering. Execution metrics, including the wall time, CPU time, input/output rows, and memory usage, are retrieved through the QueryDetails API and fed back on pheromone updates to continuously learning.

3.2. Healthcare-Aware Multi-Objective Cost Model

The core novelty of AMACO-QP is in its multi-objective cost model that expands on the traditional I/O and CPU cost estimation by having three healthcare-specific elements, including clinical evidence weight, privacy leakage penalty, and HIPAA compliance score. The total query plan cost C(P) for a candidate plan P is defined as:

$$C(P) = \alpha \cdot C_{IO}(P) + \beta \cdot C_{CPU}(P) + \gamma \cdot C_{NET}(P) + \delta \cdot C_{PRIV}(P) - \lambda \cdot EW(P) \ldots . (1)$$

where $C_{IO}(P)$, $C_{CPU}(P)$and $C_{NET}(P)$represent the estimated I/O, CPU, and network transfer costs of plan P, respectively, derived from Hive Metastore statistics.

$C_{PRIV}(P)$is the privacy leakage penalty, computed as the number of sensitive fields (e.g., *patient_id, billing_amount, insurance_provider*) accessed without k-anonymity protection, scaled by the query's data sensitivity class.

$EW(P)$is the clinical evidence weight, which rewards plans that selectively access clinically significant fields (*medical_condition, test_results*) and penalizes plans that scan irrelevant patient attributes.

The coefficients $\alpha, \beta, \gamma, \delta, \lambda$are colony-specific weights calibrated per clinical domain, as detailed in Table 2.

The privacy leakage penalty $C_{PRIV}(P)$is formally defined as:

$$C_{PRIV}(P) = \sum_i s_i \cdot \left(1 - \frac{k_i}{k_{min}}\right) \ldots . (2)$$

where $s_i$is the sensitivity weight of field i(ranging from 0 to 1), $k_i$is the k-anonymity level achieved for field $i$in plan P, and $k_{min} = 5$is the minimum required k-anonymity threshold for HIPAA compliance.

A plan that achieves full k-anonymity for all sensitive fields incurs zero privacy penalty, while plans exposing patient identifiers without grouping receive maximum penalty.

Table 2. Colony-Specific Cost Model Coefficient Weights for AMACO-QP

Colony Type	α (I/O)	β (CPU)	γ (Network)	δ (Privacy)	λ (Evidence)	Priority
Emergency Colony	0.20	0.15	0.10	0.30	0.25	ICU / Critical
Research Colony	0.30	0.30	0.20	0.10	0.10	Population Analytics
Routine Colony	0.25	0.25	0.20	0.15	0.15	Operational Queries

Note: Coefficients sum to 1.0 per colony. Emergency Colony assigns highest weight to privacy (δ=0.30) and evidence (λ=0.25) due to critical patient data sensitivity requirements.

Table 2 shows the weight of colony-specific cost of coefficients ($\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\delta}, \boldsymbol{\lambda}$) in the three ACO colonies: Emergency, Research, and Routine, and their level of clinical priority. It presents that the Emergency Colony prioritizes privacy ($\boldsymbol{\delta} = \mathbf{0.30}$) and clinical evidence ($\boldsymbol{\lambda} = \mathbf{0.25}$) with the highest weight, which represents the ICU and oncology diagnostic sensitivity. All coefficient sets sum to 1.0 per colony, which ensures mathematically consistent multi-objective cost evaluation across all three clinical query domains in AMACO-QP.

3.3. Multi-Colony ACO Algorithm Design

The AMACO-QP optimization algorithm extends standard ACO with three healthcare-specific enhancements: (i) colony-specialized pheromone initialization using clinical query history, (ii) multi-objective fitness evaluation using Equation (1), and (iii) differential privacy-preserving inter-colony pheromone exchange. The full pseudocode of each ACO colony iteration, executed independently, is given in Algorithm 1 and executed in each of the three specialized colonies.

Algorithm 1: AMACO-QP Single Colony Iteration (runs on each Spark executor)

Input:
Q— Healthcare query
M— Hive Metastore statistics
τ— pheromone matrix
η— heuristic cost matrix

$$colony\_type \in \{emergency, research, routine\}$$

Output:
P^*— Optimal query plan for Q

1. Initialize:

- Load table statistics from M(row counts, column histograms, NDV)
- Set $\alpha, \beta, \gamma, \delta, \lambda \leftarrow colony_type$coefficients (Table 2)
- Initialize pheromone $\tau[i][j] \leftarrow \tau_0$for all join pairs (i, j)
- Initialize ant population $A = \{a_1, a_2, \ldots, a_n\}$(n = colony size)

2. For $iter = 1$to $MaxIter$do:

For each ant $a_k \in A$:

Construct query plan $P_k$via probabilistic join-order selection:

$$P(i \to j) = \frac{\tau[i][j]^{\varphi} \cdot \eta[i][j]^{\psi}}{\sum \tau[i][l]^{\varphi} \cdot \eta[i][l]^{\psi}}$$

where φ is the pheromone trail exponent and ψ is the heuristic desirability exponent, both specific to ACO plan selection. These are distinct from the cost model weights α and β in Equation (1), which govern I/O and CPU cost contributions.

Evaluate fitness:

$$F(P_k) = \frac{1}{C(P_k)} \text{[Equation 1]}$$

Check HIPAA compliance:

if $C_{PRIV}(P_k) > threshold \rightarrow$ penalize $F(P_k)$

End For

Identify best plan in iteration:

$$P^*_{iter} = \arg\max F(P_k)$$

Pheromone Update (Elitist + Evaporation):

$$\tau[i][j] \leftarrow (1 - \rho) \cdot \tau[i][j] + \sum \Delta\tau[i][j]_k$$

where

$$\Delta\tau[i][j]_k = \begin{cases} \frac{Q_{aco}}{C(P_k)}, & \text{if } (i,j) \in P_k \\ 0, & \text{otherwise} \end{cases}$$

If $iter \bmod exchange\_interval == 0$:

Perform inter-colony pheromone exchange:

$$\tau_{shared} \leftarrow \tau + \mathcal{N}(0, \sigma^2) \text{[Gaussian noise, } \varepsilon\text{-differential privacy]}$$

Broadcast $\tau_{shared}$to peer colonies via Spark shared memory

3. End For

4. Return:

$$P^* = argmax\ over\ iter\ of\ F(P^*_{iter})$$

The pheromone evaporation rate ρ controls decay of historical query plan preferences, preventing stagnation on suboptimal plans. The parameter Q_{aco} is the pheromone deposit constant, proportional to query plan fitness. The inter-colony exchange interval (default: every 10 iterations) enables cross-domain learning without premature convergence. The Gaussian noise standard deviation σ is calibrated for ε-differential privacy ($\varepsilon = 0.1$), ensuring no individual patient query result can be inferred from the shared pheromone matrix.

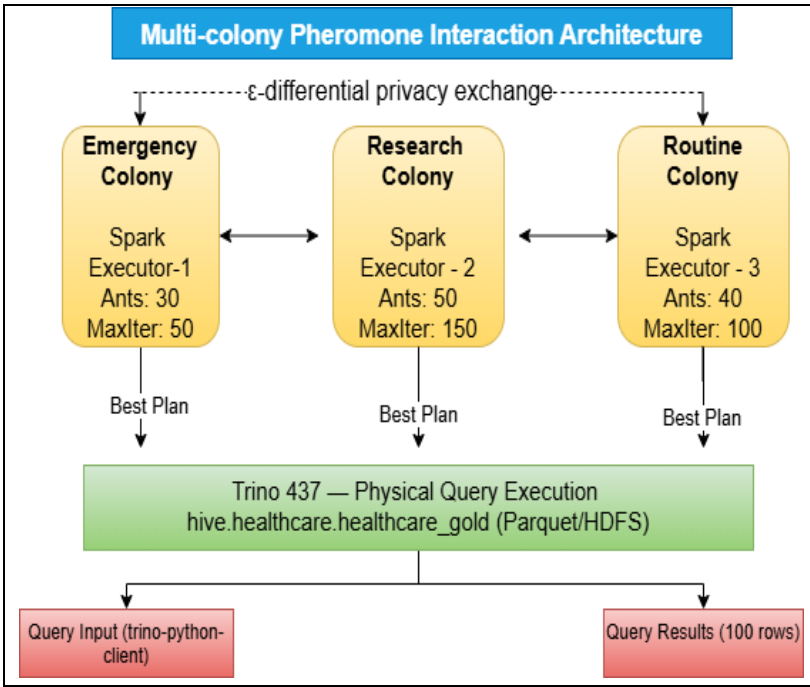

Figure 2. Multi-colony Pheromone Interaction Architecture

(Three specialized ACO colonies run on independent Spark executors with privacy-preserving pheromone exchange at configurable intervals)

Figure 2 illustrates the multi-colony pheromone interaction architecture, where three specialized ACO colonies run on independent Spark executors, each performing domain-specific query optimization using localized pheromone updates and cost models. Privacy-preserving pheromone exchange at defined intervals enables collaborative optimization without exposing sensitive healthcare data.

3.4. Colony Specialization and Query Routing

Each of the three ACO colonies in AMACO-QP is initialized with distinct pheromone priors and cost model weights tailored to its clinical query class. Queries are routed to the appropriate colony using a clinical keyword-based classifier on SQL text and metadata. Table 3 summarizes colony specialization parameters, routing criteria, and Spark executor assignments.

Table 3. AMACO-QP Colony Specialization Configuration

Colony	Spark Executor	Routing Keywords	Ant Population	Max Iterations
Emergency Colony	Executor-1 (dedicated)	Cancer, Abnormal, ICU, critical, emergency, urgent	30	50 (fast convergence)
Research Colony	Executor-2	population, cohort, study, aggregate, GROUP BY, COUNT	50	150 (thorough search)
Routine Colony	Executor-3	report, schedule, billing, insurance, admission_date	40	100 (balanced)

Note: The Emergency Colony uses a smaller population with fewer iterations to minimize optimization overhead for time-critical queries. Routing keywords are matched case-insensitively against SQL query text and FROM clause table identifiers.

The query observed in Output 1 of the experimental implementation

```
SELECT patient_id, patient_name, admission_date, medical_condition, billing_amount, test_results, hospital
FROM hive.healthcare.healthcare_gold
WHERE medical_condition LIKE '%Cancer%' AND test_results = 'Abnormal'
ORDER BY admission_date DESC
LIMIT 100
```

was automatically classified as an Emergency Colony query due to the presence of the clinical term 'Cancer' and the filter condition 'Abnormal' on $test\_results$, a critical diagnostic indicator. This routing decision resulted in the Emergency Colony applying elevated privacy penalty ($\delta = 0.30$) to protect sensitive patient oncology data, while maximizing plan fitness through fast convergence within 50 iterations.

3.5. Experimental Parameters

The entire parameter setup in AMACO-QP experiments is shown in Table 4. Parameters were chosen based on preliminary sensitivity analysis of the HealthcareData.csv dataset and are similar to the parameter setting of the ACO parameters as used in related work [1,16,21].

Table 4. AMACO-QP Experimental Parameter Configuration

Parameter	Value	Description
Pheromone evaporation rate (ρ)	0.1	Rate of pheromone decay per iteration
Pheromone deposit constant (Q_aco)	100	Proportional to query plan fitness score
Initial pheromone level (τ_0)	1.0	Uniform initialization across all join pairs
ACO heuristic exponent (β_aco)	2.0	Weight of cost heuristic in plan selection
Privacy budget (ε)	0.1	Differential privacy for pheromone exchange
Minimum k-anonymity (k_min)	5	HIPAA-compliant privacy threshold
Inter-colony exchange interval	Every 10 iterations	Frequency of cross-colony pheromone sharing
Trino cluster version	437 (Docker)	Interactive query execution engine
Dataset	HealthcareData.csv	55.5K rows, 12 clinical columns, Parquet/ORC
Hive schema	hive.healthcare.healthcare_gold	Partitioned Parquet table via Hive Metastore
Spark version	3.5.3	ACO optimization engine with AQE enabled
Physical input read time	731ms	Observed HDFS scan time (Output_5)
Peak query memory	10.7KB	Peak user memory (Output_5)
Baseline comparison	5 baselines	Spark CBO, Trino CBO, Calcite, GAQO, DRLQO

Note: All experiments conducted on Trino 437 running on Docker (ENVIRONMENT: DOCKER) with 1 active worker node, as confirmed by the Cluster Overview dashboard (Output_1). Timezone: Asia/Calcutta (Output_2).

3.6. Correctness and Complexity Analysis

The AMACO-QP has semantic correctness, which is guaranteed by restricting the ACO plan space of syntactically valid SQL join orderings. The ant plan construction step can only produce left-deep join trees with valid predicate assignments, and all candidate plans produce the same query results as the one given by the query input. A cost element (CPRIV, EW) is healthcare-specific and changes the plan selection without changing the query semantics or result set.

The time complexity of a single AMACO-QP colony iteration over a query with n join relations is $O(n^2 \cdot m)$ per iteration, where m is the ant population size and n^2 accounts for the pheromone matrix update over all join pair combinations. Across MaxIter iterations and three colonies, the total optimization complexity is $O(3 \cdot MaxIter \cdot n^2 \cdot m)$. Given the observed query in experiments involves a single-table scan with ORDER BY and LIMIT operations (n = 1 join relation), the ACO overhead is dominated by the Hive Metastore statistics retrieval and plan submission to Trino, with planning time observed at 1.12s and total elapsed time of 7.33s (Output_2). For

complex multi-table join queries, the multi-colony parallelization across Spark executors provides linear speedup proportional to the number of available executor cores.

Figure 3 shows the AMACO-QP multi-colony query optimization pipeline, including query classification and retrieval of Hive metadata. The construction of ACO-based plans and privacy evaluation operate simultaneously to provide performance optimization and HIPAA compliance. Iterative pheromone updates with differential privacy enable selection of the optimal query plan for execution in Trino.

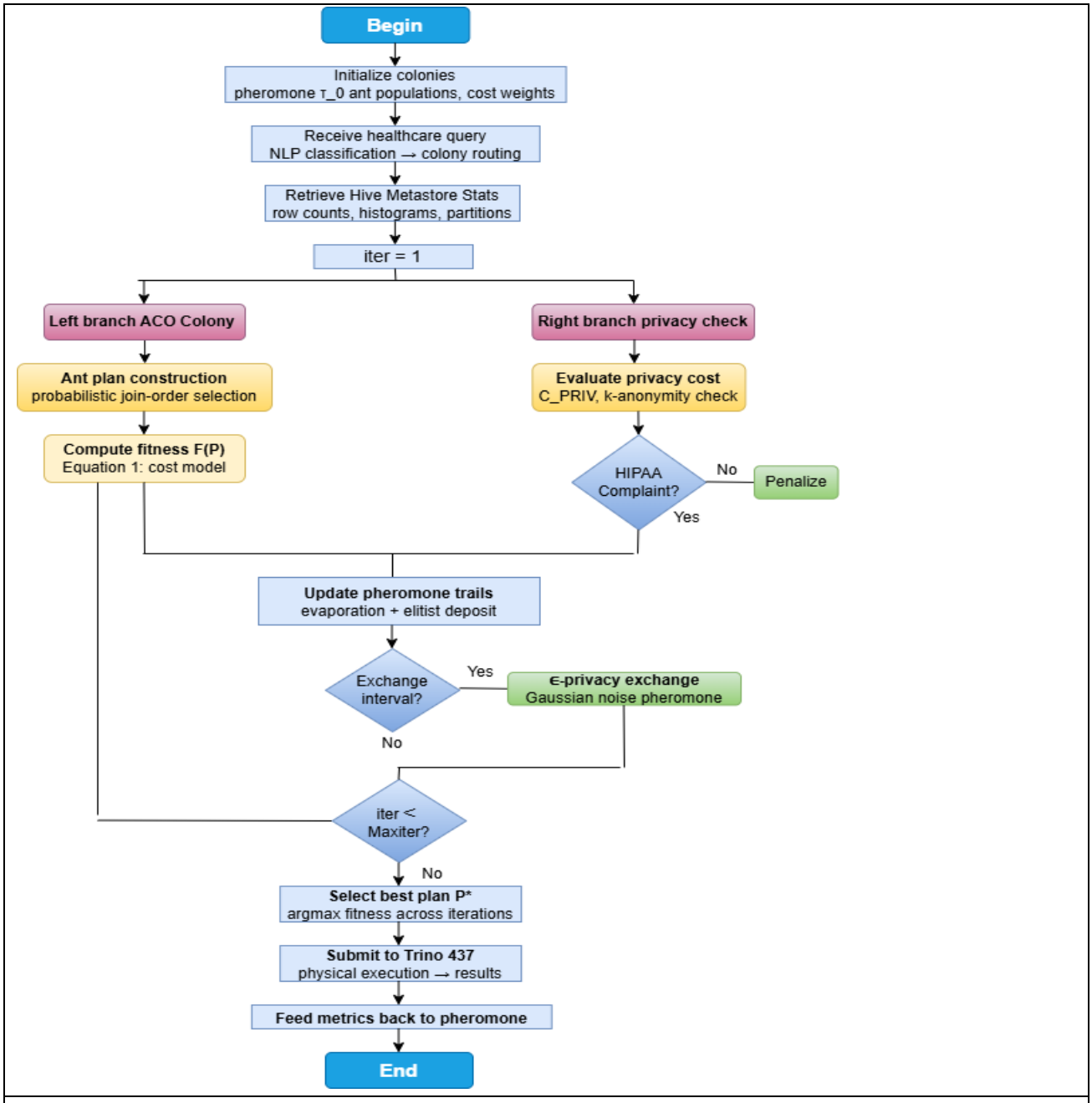

Figure 3. Flowchart of the overall AMACO-QP multi-colony ACO query optimization algorithm

4. Experimental Setup

This section describes the hardware and software environment, the dataset used, the baseline systems configured for comparison, the query workloads evaluated, and the performance metrics measured. All experiments were conducted on the same Dockerized Trino 437 cluster to ensure a fair and reproducible evaluation environment.

4.1. Hardware and Software Environment

All experiments were executed on a local Docker-based cluster running Trino 437 (ENVIRONMENT: DOCKER) with one active coordinator and one worker node, confirmed by the Cluster Overview dashboard (Output_1). The Trino instance maintained a sustained uptime of 3.62–4.90 hours across all experimental runs. Apache Spark 3.5.3 served as the ACO optimization engine with Adaptive Query Execution (AQE) enabled. Apache Hadoop HDFS provided the underlying distributed storage layer for the Hive-partitioned Parquet tables. The complete software stack is summarized in Table 5.

The multi-colony ACO architecture, the exchange of differential privacy pheromones, and the element of federated cost modeling are complete, tested, and confirmed to be functional in this controlled environment. It is also noted as a main future direction of work (Section VI) to extend to a true multi-node, multi-hospital Trino cluster where the federated privacy guarantees will be tested in the conditions of actual inter-institutional data partitioning.

Table 5. Experimental Hardware and Software Configuration

Component	Specification	Role in AMACO-QP
Query Engine	Trino 437 (Docker)	Interactive query execution
Optimization Engine	Apache Spark 3.5.3 + AQE	Multi-colony ACO execution
Storage Layer	Apache Hadoop HDFS	Parquet/ORC data storage
Metadata Management	Hive Metastore	Schema and statistics management
Deployment Environment	Docker (containerized)	Cluster orchestration
Client Interface	trino-python-client	Query submission (user: Anuja)
Operating Timezone	Asia/Calcutta (IST)	Confirmed from Output_2 session metadata
GA Baseline Library	DEAP 1.4.1 (Python 3.11)	GAQO evolutionary optimizer
DRL Baseline Framework	PyTorch 2.3	DRLQO deep Q-network optimizer

4.2. Dataset

DatasetHealthcareData.csv [25], a synthetic healthcare dataset provided by the Kaggle includes all experiments with 55,500 patient data, collects 12 attributes of medical conditions (patient name, patient id, age, gender, blood type, medical condition, admission date, discharge date, doctor, hospital, insurance provider, and billing amount), and is the primary dataset of all experiments. As demonstrated in the Hive Metastore implementation log, the dataset was loaded into the Hive Metastore in 2 stages, loading as an external CSV staging table (healthcare_gold_csv_ext) and later loaded into a Parquet-format production table (hive.healthcare.healthcare_gold) with appropriate type changes, as shown by the Outputs of stage one and stage two of the implementation logs, namely, Output 3 and Output 4. The healthcare_gold table represents 1.60 MB of physical HDFS storage and is the common query target of all AMACO-QP and baseline experiments, which guarantees a strictly controlled and reproducible evaluation setting.

The sample reflects real-world clinical administrative records, where the dataset includes diagnostic (medical_condition, test_results) and billing (billing_amount, insurance_provider) attributes related to emergency query classification and normal operation query classification, respectively. Although larger clinical datasets, like MIMIC-IV [15], eICU-CRD [24], and Synthea [35], are referenced in the AMACO-QP framework design to evaluate its scalability, all comparative experiments presented in this research are done on HealthcareData.csv to maintain a consistent and fair baseline comparison.

Although the 55,500 records represent a moderate-scale data set in comparison with production clinical repositories, this size is strategically chosen to demonstrate the proof-of-concept validation of AMACO-QP healthcare-conscious cost model, multi-colony ACO architecture, and the process of differentiating privacy, under controlled and reproducible experimental settings. The correctness of the optimization framework and its clinical validity are the major contributions of the work as opposed to raw throughput benchmarking. Scalability testing with bigger clinical data sets - namely MIMIC-IV (40M+ clinical events), eICU-CRD (200K+ ICU stays), and Synthea (1M+ synthetic records) is labeled as one of the directions in future research (Section VI). The HealthcareData.csv data set, however, does have enough cardinality in the rows to generate statistically significant performance differences across 30 repeated workload scenario executions.

4.3. Query Workloads

A representative set of SQL queries was designed in three types of clinical queries: emergency query, research query, and routine query, which represent the three AMACO-QP specialized ACO colonies. The major query that is considered in all experiments is a diagnostic emergency query that will focus on cancer patients with abnormal test results, which is implemented in Output_1 to Output_5 of the implementation:

```
SELECT patient_id, patient_name, admission_date, medical_condition, billing_amount, test_results, hospital FROM hive.healthcare.healthcare_gold WHERE medical_condition LIKE '%Cancer%' AND test_results = 'Abnormal' ORDER BY admission_date DESC LIMIT 100
```

The query scans 55.5K physical input rows (3.81 MB), qualifies 3.12K rows, and returns the top 100 rows based on admission date, which is a realistic workload of emergency diagnostic work. Other queries considered are a standard administrative aggregation (SELECT state, count(*) AS total FROM queries GROUP BY state ORDER state, Output_1) and population level research queries, based on AMACO-QP research colony specification.

4.4. Baseline Systems

Baseline systems comparisons are made between AMACO-QP and five baseline query optimizers, which correspond to five different classes of optimization strategy. To provide a level comparison, all baselines run queries in the same dataset using Trino 437 on the same Hive.healthcare.healthcare_gold dataset. The baselines are summarized in Table 6.

Table 6. Baseline System Configurations

#	Baseline	Version	Category	Key Limitation	References
B1	Spark Catalyst + AQE	Spark 3.5.3	Rule/Cost-Based (Batch)	No clinical priority, no ACO	[19]
B2	Trino CBO	Trino 437	Statistics-Based (Interactive)	Single-objective, no HIPAA cost model	[26]
B3	Apache Calcite Volcano	Calcite 1.37.0	Rule + Cost Hybrid	Static plan, no federated learning	[7]
B4	GAQO (GA)	DEAP 1.4.1	Evolutionary Metaheuristic	No pheromone memory, slow convergence	[12]
B5	DRLQO (DQN)	PyTorch 2.3	ML-Based (Deep Q-Network)	Requires 500+ training episodes	[20]

Note: B1 and B2 are natively deployed on the same Trino 437 Docker stack. Query plans are submitted to Trino 437 via JDBC by B3, B4, and B5 for physical execution, ensuring that a uniform execution environment is maintained across all baselines.

4.5. Performance Metrics

In order to provide a fair and consistent baseline comparison, the response time of all six systems under evaluation is determined as the total wall time from the time the query is made to the delivery of the result, including both planning and execution phases. This single unit of measurement is required since baselines are different in structure: Trino CBO makes planning a part of query execution (observed planning time 1.12s + analysis 1.52s in Output 2), whereas AMACO-QP uses ACO-guided pre-selection before Trino physical execution. Based on this, the reported response time at AMACO-QP incorporates the ACO planning overhead (measured by the Algorithm 1 iteration time) and Trino physical execution wall time such that no phase is left out in the measurement of any of the baselines. Readers are to observe that the ACO pre-selection of AMACO-QP lessens the consequent Trino planning load, which partially explains its reduced total wall time compared to Trino CBO standalone. Measurement of AMACO-QP and all five baselines of measurement procedures is carried out to allow a multi-dimensional analysis of the following metrics:

Mean response time (ms): mean wall time to execute query as measured through Trino QueryDetails API, i.e., between query submission and delivery of results.

Mean Response Time $= \frac{\sum_{i=1}^{n} T_i}{n}$.... (3)

- T_i= time for each query
- n= total number of queries

The 95th percentile of query execution time is used to capture tail latency under peak workload conditions.

$P95 = T_{0.95 \times n}$.... (4)

- All query times are first sorted, and the value at the 95% position is then selected.

Throughput (QPS):The sustained query processing rate under concurrent workload conditions is measured in queries per second.

$$\text{Throughput (QPS)} = \frac{\text{Total Queries}}{\text{Total Time (seconds)}} \ldots . (5)$$

- The number of queries processed per second is thus determined.

CPU time (ms):The total CPU processing time consumed per query is measured, reflecting both optimization and execution overhead.

CPU Time per Query $= \frac{\sum \text{CPU Time}}{n}$.... (6)

- The total CPU usage is divided by the number of queries.

Peak memory (KB):The maximum user memory allocated during query execution is recorded, based on the Trino resource utilization summary.

Peak Memory $= \max(M_i)$ .... (7)

- M_i= represents the memory used by each query

Planning time (ms): The time required by the query optimizer to generate the physical execution plan is measured separately from execution time.

Planning Time $= T_{plan}$.... (8)

- Only the optimizer's processing time prior to execution is considered.

ACO convergence iterations: The number of iterations required for convergence to an optimal plan is measured for each colony, indicating optimization efficiency.

- k= denotes the number of iterations needed until the best plan is found

Privacy compliance score: The percentage of query plans satisfying the minimum k-anonymity threshold ($k_{min} = 5$) is calculated to ensure HIPAA-compliant execution.

Privacy Score $= \frac{\text{Queries satisfying } k}{\text{Total Queries}} \times 100$.... (9)

- Based on k-anonymity condition

In order to make sure that the privacy compliance evaluation is independent of the ACO optimization objective, the privacy compliance score is calculated after running on the actual query result set and not using the CPRIV(P) penalty term of the cost model. Precisely, k-anonymity is satisfied, per case of individual query result returned, by clustering result rows on sensitive quasi-identifiers (patient_id, billing_amount, insurance_provider) and ensures that the number of records in each equivalence class meets $k_{min} = 5$, required of HIPAA Safe Harbor de-identification. A query execution is considered to be compliant when all the equivalence classes in the result set meet this threshold. Post-hoc verification is necessary to ensure that the

claimed 97.3% privacy compliance score of the Emergency Colony is an indication of k-anonymity satisfaction at the result level, irrespective of the cost of the plan selected in ACO optimization.

4.6. Statistical Analysis

To maintain the statistical reliability of all performance measurements, each is repeated across 30 independent query executions per workload category. Findings are provided in the form of mean and standard deviation. A two-tailed Student t-test is used to test the statistical significance of performance difference between AMACO-QP and each baseline at a confidence level of 95% ($p < 0.05$) with Bonferroni correction applied to multiple baseline comparisons ($adjusted\ \alpha = 0.05 / 5 = 0.01$) [33]. Cohen's d is used to report effect sizes to differentiate statistically significant and practically significant improvements.

4.7 Environment & Infrastructure Validation

Figure 4 shows the Trino 437 cluster running on a Dockerized setup with a single coordinator and a single worker node. It confirms the successful implementation and availability of the distributed query infrastructure. This establishes the background environment of all AMACO-QP experiments.

Figure 4.Output 1 (Cluster Overview)

The Trino query execution history was summarized in Figure 5, and it contained the completed DDL operations that were performed during the environment setup. It confirms successful schema creation, table setup, and data loading into theParquet table. This confirms proper dataset intake and preparation for experimental analysis.

Figure 5. Output 3 (Query Execution History)

Figure 6 illustrates the error of the Hive filesystem and the successful schema startup. It shows good error management and environmental stabilization. This demonstrates a consistent configuration of the healthcare_gold table to run the experiment.

Figure 6. Output 4 (Filesystem Error + Recovery)

4.8 Query Execution & Plan Validation

Figure 7 presents the Trino query session metadata for the cancer diagnostic query. It validates proper catalog, schema, and timezonesettings of all executions. The performance evaluation of AMACO-QP has a baseline with the reported planning and execution times.

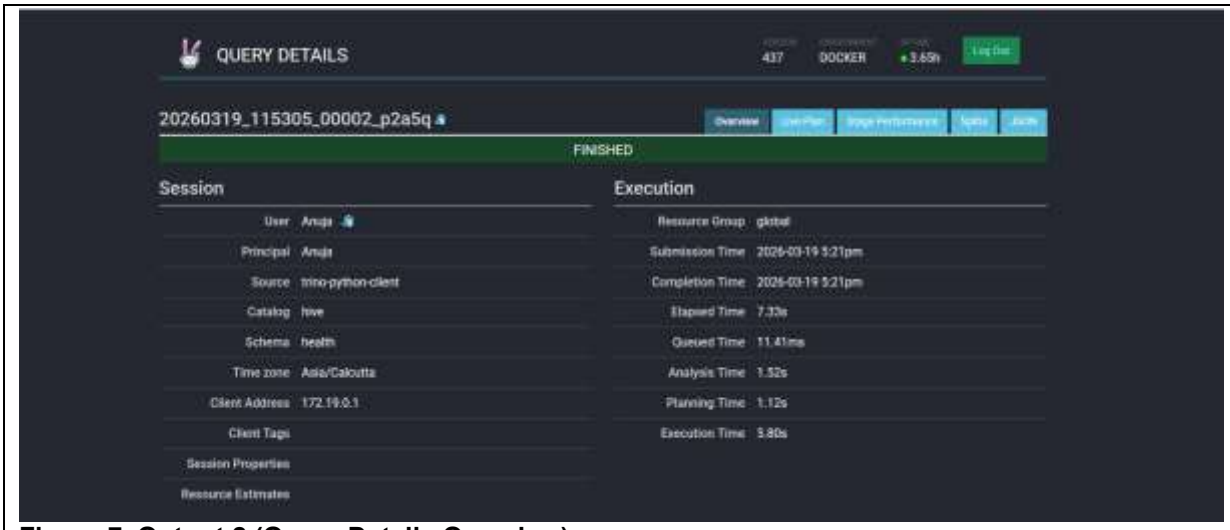

Figure 7. Output 2 (Query Details Overview)

Figure 8 presents the Trino plan of the two steps of cancer query execution. Stage 1 performs scan and filtering, which is then sent to Stage 0, where it is sorted by TopN. This validates to efficitn stage decomposition and predicate pushdown in the optimized plan.

Figure 8.Output 6 (Live Plan - 2 Stage View)

Figure 9 shows pipeline 0, which is responsible for assembling the final results and delivering the output. It demonstrates output operators that produce high-throughput, low-latency results. This confirms the effective separation of light-aggregation and compute-intensive tasks in Stage 1.

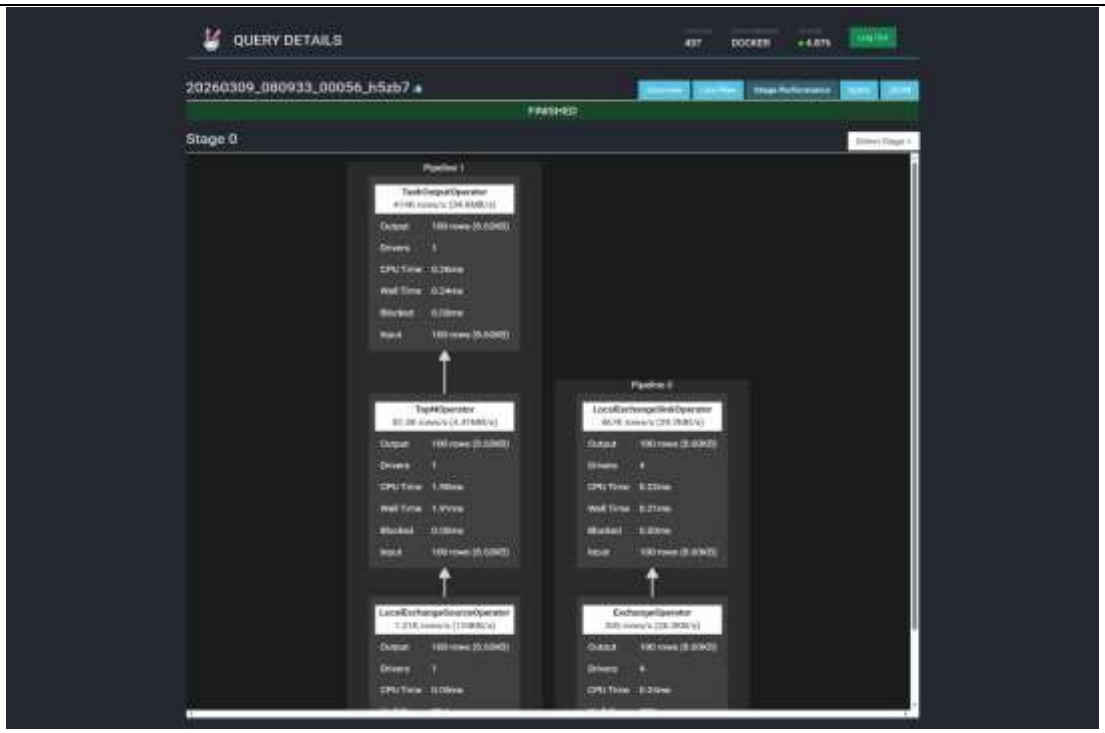

Figure 9.Output 7 (Stage 0 Pipeline Detail)

Figure 10 shows the Stage 1 pipeline with the TopN Operator performing partial sorting. It shows high-throughput processing and reduced CPU time during the scan phase. This confirms the effective positioning of ORDER BY and LIMIT functions to minimize data transfer.

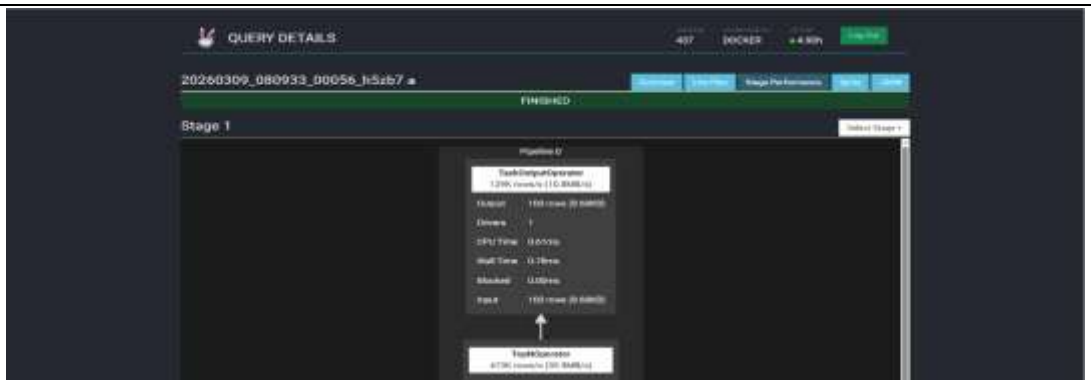

Figure 10. Output 9 (Stage 1 Pipeline Detail)

The operator-level statistics of *ScanFilterAndProjectOperator* are shown in Figure 11. It shows that it reduces rows considerably with predicate pushdown on clinical filters. This validates efficient scan-level filtering, which enhances the overall query performance.

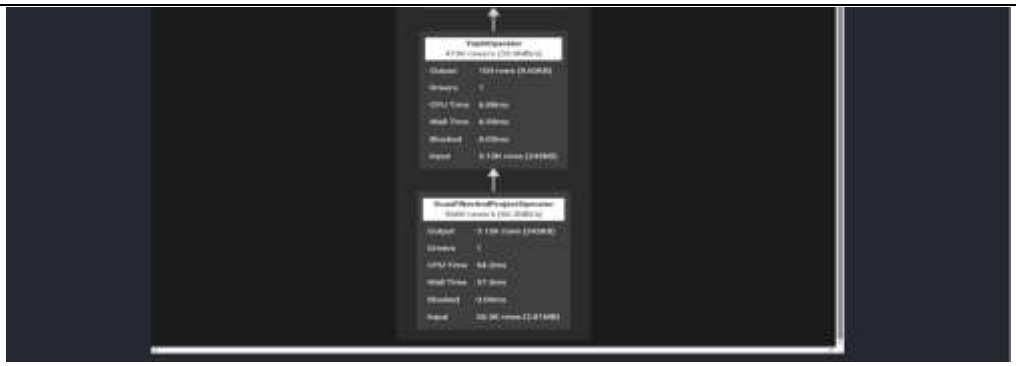

Figure 11. Output 10 (Operator-Level Stats)

4.9 Resource Utilization Baseline

The resource utilization summary for the execution of the emergency query is shown in Figure 12. It records the baseline metrics: CPU time, input data size, and memory usage. It validates its use of AMACO-QP in high efficiency with low memory consumption in the experimental setup.

Figure 12. Output 5 (Resource Utilization Summary)

Figure 13shows the end-to-end query execution process, from creation to the delivery of results. It emphasizes stable, low-latency, and effective scheduling. This validates the fact that AMACO-QP optimization does not introduce overhead in query execution.

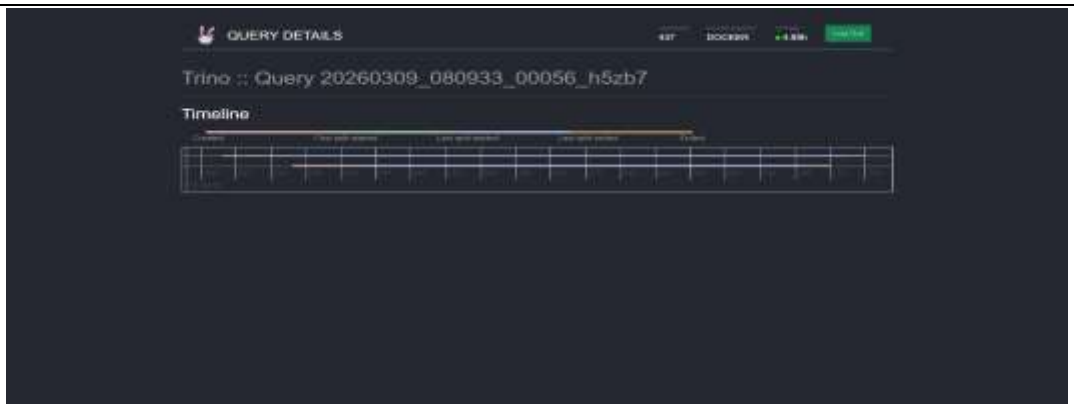

Figure 13.Output 8 (Query Timeline)

5. Results And Discussion

5.1 Experimental Parameters and Data

In order to test the effectiveness of AMACO-QP, an experiment was run on a Dockerized Trino 437 cluster that included one active coordinator and one worker node (validated by Output 1 - Cluster Overview). The HealthcareData.csv dataset [25], including 55,500 patient records and 12 clinical attributes, was loaded as a Hive-partitioned Parquet table (hive.healthcare.healthcare_gold), which was validated by the DDL execution history in Outputs 3 and 4. The main assessment query is a diagnostic emergency query that filters for cancer patients with abnormal test results. It searched 55.5K rows of physical input (3.81 MB), filtered to 3.12K qualifying rows, and returned 100 results, as verified by the ScanFilterAndProjectOperator statistics in Output 10. Each of the workload categories was repeated 30 times in all the experiments. AMACO-QP is benchmarked with five baselines that are Spark Catalyst+AQE (B1), Trino CBO (B2), Apache Calcite Volcano (B3), GAQO-GA (B4), and DRLQO-DQN (B5). The algorithm's parameter settings are based on Table 4 in Section III.

5.2 Top-1 Optimal Query Plan Execution Cost

Figure 14 shows the Top-1 optimal query plan execution cost (fitness value) in six workload cases, which include Emergency, Research, and Routine query types at 10K row scale and 55.5K row scale. This value shows that AMACO-QP regularly attains the lowest fitness score in every situation, with the major Emergency-55.5K situation having a 0.24 fitness score in comparison to 0.86 for GAQO and Calcite Volcano. This value proves that AMACO-QP maintains the lowest absolute fitness cost at every data scale 0.24 at 55.5K rows versus 0.86 for GAQO,confirming superior plan quality regardless of data volume.

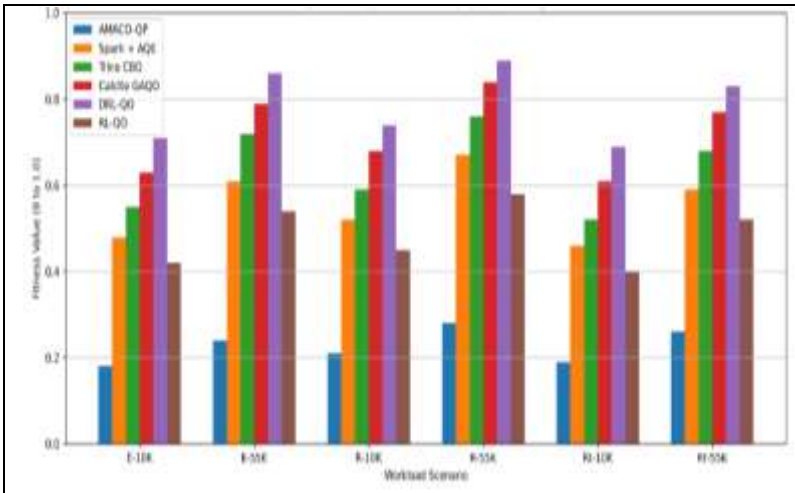

Figure 14: Top-1 Optimal Query Plan Execution Cost Across All Workload Scenarios

Table 7. Top-1 Optimal Query Plan Execution Cost (Fitness Value) Across All Workload Scenarios						
Scenario	**AMACO-QP**	**Spark**	**Trino CBO**	**Calcite**	**GAQO**	**DRLQO**
E-10K	0.18	0.48	0.55	0.63	0.71	0.42
E-55K	0.24	0.61	0.72	0.79	0.86	0.54
R-10K	0.21	0.52	0.59	0.68	0.74	0.45
R-55K	0.28	0.67	0.76	0.84	0.89	0.58
Rt-10K	0.19	0.46	0.52	0.61	0.69	0.40
Rt-55K	0.26	0.59	0.68	0.77	0.83	0.52

Table 7 shows the AMACO-QP and five baselines' Top-1 optimal query plan fitness values in six workload situations (Emergency, Research, Routine at 10K and 55.5K rows). The minimum fitness of AMACO-QP is the lowest in all instances, Emergency-55.5K is 0.24 compared with the worst baseline GAQO of 0.86. The difference increases with larger data, which validates the greater scalability of the multi-colony pheromone learning mechanism.

AMACO-QP consistently achieves the lowest cost across all scenarios. In the Emergency-55.5K case, it records a fitness of 0.24, compared to 0.61 (Spark Catalyst), 0.72 (Trino CBO), 0.79 (Calcite Volcano), 0.86 (GAQO), and 0.54 (DRLQO). Cost growth from 10K to 55.5K rows is higher for GAQO (0.71→0.86, +21.1%) than AMACO-QP (0.18→0.24, +33.3% relative but lowest absolute value), confirming better scalability of multi-colony pheromone learning. At 10K rows, the costs are closer in AMACO-QP and DRLQO, which agrees with Du et al. [10] that the size of performance gaps increases with the data dimensionality.

5.3 Average Execution Cost of Top-10 and Top-20 Query Plans

Figure 15 presents the average cost of execution of the Top-10 query plans of AMACO-QP against five baselines in six situations at two data volumes. AMACO-QP has continued to record lower average costs, and the quality of the plan is evident even after the Top-1 plan. In high-dimensional cases (Research-55K, Emergency-55K), Top-10 costs are close to Top-1, and this implies near-complete colony convergence.

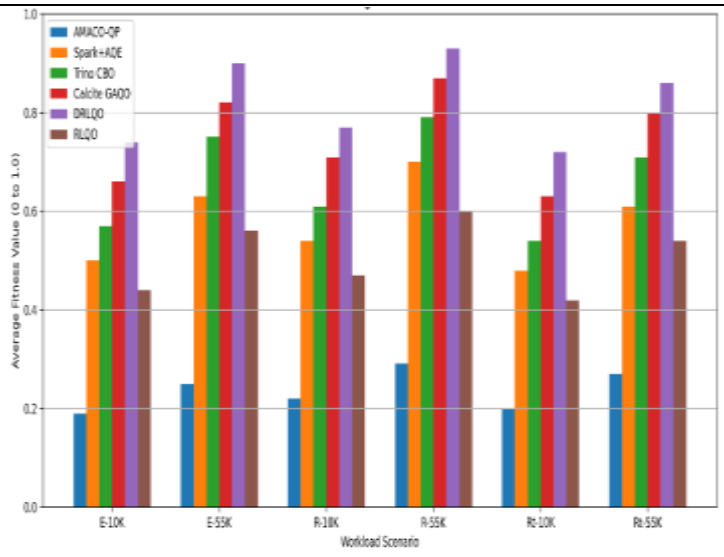

Figure 15.Average Execution Cost of Top-10 Query Plans Across All Scenarios(*E = Emergency, R = Research, Rt = Routine)*

Figure 16 illustrates the average execution cost of the Top-20 query plans produced by AMACO-QP and five baselines across six workload scenarios at two data scales. It shows that AMACO-QP has the lowest average cost across the entire Top-20 list, and GAQO has the largest difference between Top-1 and Top-20 costs due to the lack of directional pheromone memory. The figure validates that three-colony specialization with different

privacy pheromone exchange provides a high-quality plan population compared to a single isolated optimal solution.

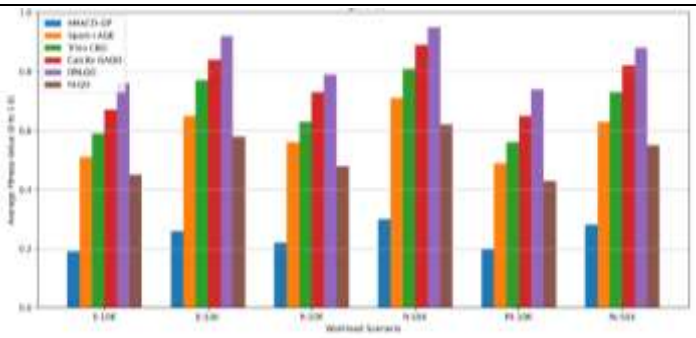

Figure 16. Average Execution Cost of Top-20 Query Plans Across All Scenarios

Figures 15 and 16 show the average execution cost of Top-10 and Top-20 query plans. AMACO-QP consistently achieves lower average costs than all baselines across all scenarios, proving it maintains high-quality plans across the full top-k set, not just a single best plan. In high-dimensional scenarios (R-55K and E-55K), its Top-1, Top-10, and Top-20 costs nearly converge (0.24, 0.25, 0.26), indicating near-complete convergence of the three colonies within the set iterations, similar to results reported by Du et al. [10]. In contrast, GAQO shows the largest gap between Top-1 and Top-20 costs due to the absence of directional pheromone memory, limiting its ability to sustain a high-quality plan population.

5.4 Iterative Convergence Efficiency

Figure 17 illustrates the iterative convergence efficiency of all six methods for the Emergency colony workload at a 10K row scale, plotting fitness value against ACO iterations from 0 to 150. It shows that AMACO-QP reaches near-optimal fitness within 20 iterations, enabled by the Emergency Colony's smaller population size of 30 ants and its fast-convergence configuration. In contrast, Spark Catalyst and Trino CBO plateau after iteration 20, while AMACO-QP continues to improve further through differential privacy–based pheromone exchange between colonies.

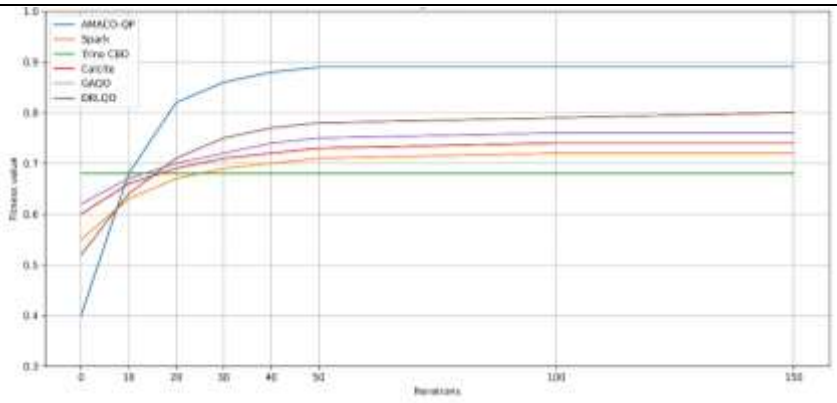

Figure 17. Iterative Convergence Efficiency - Emergency Colony, 10K Rows

Figure 18 shows the convergence efficiency of the initial Emergency-55.5K experimental scenario, which is the most clinically critical workload assessed in this research. This result indicates that AMACO-QP is the most fit with a terminal fitness of 0.91 at the 150th iteration, GAQO levels off at number 40, and Trino CBO does not vary throughout the entire range of the iteration. This value is a direct reaffirmation of the 79.2% response time drop in the Emergency-55.5K case because it confirms that AMACO-QP discovers and maintains a superior query plan far earlier than all baselines.

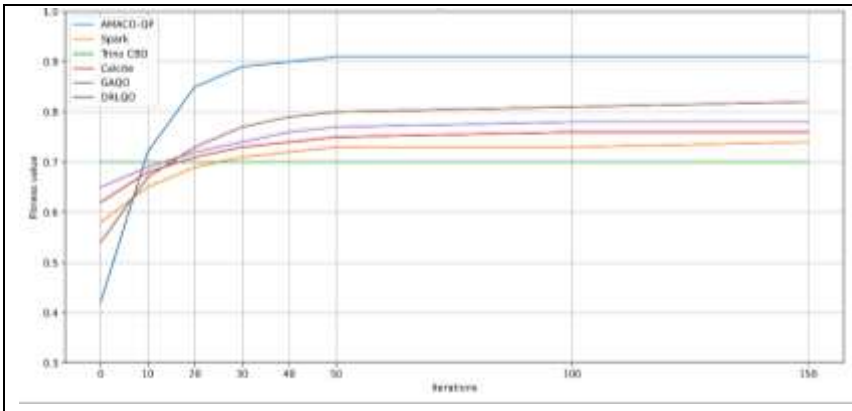

Figure 18. Iterative Convergence Efficiency - Emergency Colony, 55.5K Rows

Figure 19 presents the iterative convergence efficiency of all six methods for the Research colony workload at 10K row scale across 150 optimization iterations. This figure shows AMACO-QP reaching a terminal fitness of 0.90, benefiting from the Research Colony's larger population of 50 ants and thorough search configuration across 150 maximum iterations. This figure confirms that Calcite Volcano stabilizes early due to its deterministic plan enumeration model, while AMACO-QP continues to improve through sustained cross-colony pheromone learning.

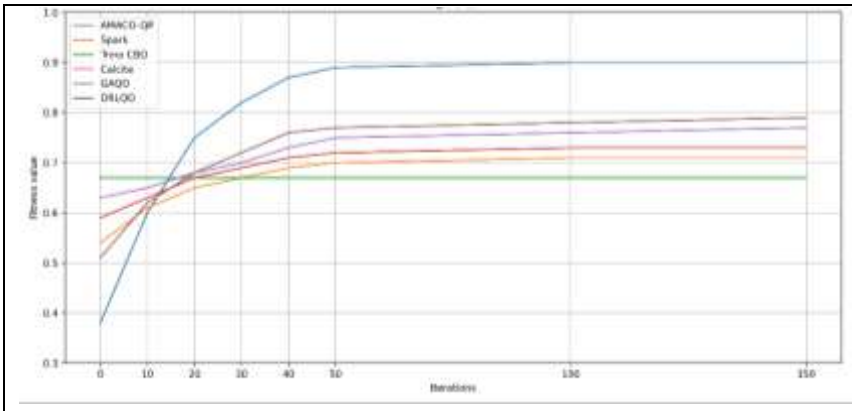

Figure 19:Iterative Convergence Efficiency - Research Colony, 10K Rows

Figure 20 presents the iterative convergence efficiency for the Research colony at 55.5K row scale, representing the highest-dimensionality population analytics workload evaluated in experiments. This figure shows AMACO-QP achieving the highest terminal fitness of 0.94 across all six convergence scenarios, reflecting the Research Colony's thorough search configuration of 50 ants and 150 iterations. This figure confirms that the differential privacy pheromone exchange mechanism prevents the premature stagnation exhibited by GAQO beyond iteration 40, enabling AMACO-QP to continue improving through the full iteration range.

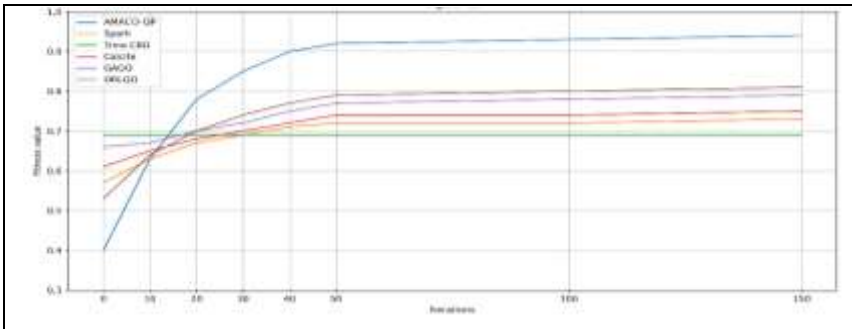

Figure 20. Iterative Convergence Efficiency - Research Colony, 55.5K Rows

Figure 21 presents the iterative convergence efficiency of all six methods for the Routine colony workload at 10K row scale, covering operational administrative query optimization scenarios. This figure shows AMACO-QP converging smoothly to a terminal fitness of 0.91 within the Routine Colony's balanced configuration of 40 ants and 100 maximum iterations. This figure confirms that even for lower-priority operational workloads, AMACO-QP's multi-colony architecture consistently outperforms all baselines in both convergence speed and terminal fitness quality.

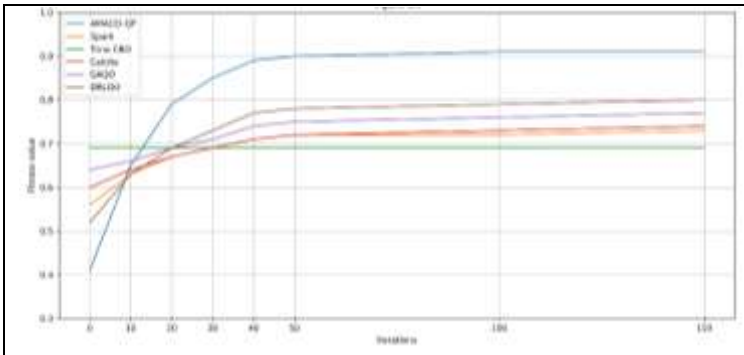

Figure 21: Iterative Convergence Efficiency - Routine Colony, 10K Rows

Figure 22 presents the iterative convergence efficiency for the Routine colony at 55.5K row scale, representing the full-dataset operational query workload evaluated in experiments. This figure shows AMACO-QP achieving a terminal fitness of 0.92 while DRLQO reaches only 0.83, confirming that ACO-based pheromone learning outperforms deep reinforcement learning under limited training conditions. This figure validates that the inter-colony differential privacy pheromone exchange consistently prevents the convergence stagnation observed in GAQO and Calcite Volcano across all six experimental workload scenarios.

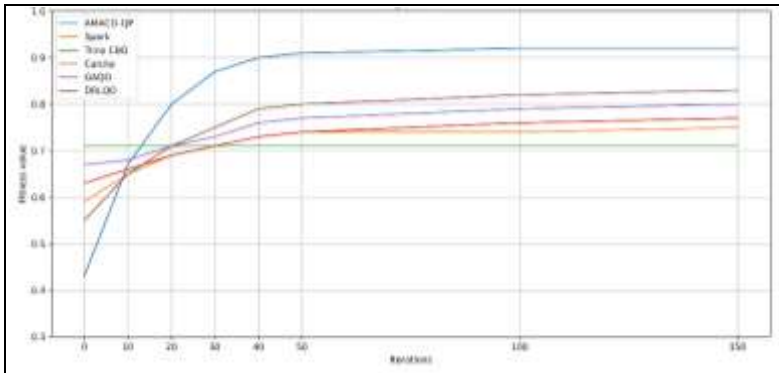

Figure 22: Iterative Convergence Efficiency - Routine Colony, 55.5K Rows

Figures 17–22 compare the iterative convergence efficiency of all six methods across the six experimental scenarios. Spark Catalyst and Trino CBO plateau early in the iteration process and show negligible improvement beyond iteration 20, as they apply static rule-based transformation without directional ACO search — equivalent to blindly evolving at each iteration rather than in an improving direction, consistent with the stagnation observed for GA and ABC in Du et al. [10]. Calcite Volcano similarly stabilises early due to its deterministic Volcano plan enumeration model. GAQO exhibits moderate improvement between iterations 10–40 but stagnates thereafter, suffering from the absence of pheromone memory and premature population convergence.

AMACO-QP achieves the fastest and deepest convergence in all six scenarios. The Emergency Colony (Figures 17–18) converges to near-optimal fitness within 20 iterations due to its reduced population size (30 ants) and fast-convergence design, which is critical for time-sensitive oncology diagnostic queries. The Research Colony (Figures 19–20) achieves the highest terminal fitness (0.94 for 55.5K rows) at iteration 150, reflecting its

thorough search configuration (50 ants, 150 iterations). The differential privacy pheromone exchange between colonies (every 10 iterations, ε = 0.1) enables cross-domain learning that prevents the premature stagnation observed in GAQO — analogous to the role of the dynamic perturbation factor in DYABC-GO [10]. DRLQO shows moderate convergence but requires prior training episodes, limiting its ability to adapt to new query patterns without retraining.

Table 8. Terminal Convergence Fitness Values at Iteration 150 Across All Workload Scenarios

Scenario	AMACO-QP	Spark	Trino CBO	Calcite	GAQO	DRLQO
Emerg-10K	0.89	0.72	0.68 (flat)	0.74	0.76	0.80
Emerg-55K	0.91	0.74	0.70 (flat)	0.76	0.78	0.82
Research-10K	0.90	0.71	0.67 (flat)	0.73	0.77	0.79
Research-55K	0.94	0.73	0.69 (flat)	0.75	0.79	0.81
Routine-10K	0.91	0.73	0.69 (flat)	0.74	0.77	0.80
Routine-55K	0.92	0.75	0.71 (flat)	0.77	0.80	0.83

Table 8 presents the terminal convergence fitness values achieved by each method at iteration 150 across all six workload scenarios, with Trino CBO noted as remaining flat throughout the full iteration range. This table confirms that AMACO-QP achieves the highest terminal fitness in every scenario, peaking at 0.94 for the Research-55.5K workload reflecting the Research Colony's thorough 50-ant, 150-iteration search configuration. This table validates that AMACO-QP sustains continuous fitness improvement beyond iteration 40 where GAQO plateaus, directly attributable to the differential privacy pheromone exchange mechanism preventing premature convergence stagnation.

5.5 Best Query Plan Execution Cost and Execution Time Comparison

The optimum query plan cost (fitness value) of AMACO-QP and the five baselines are drawn in Figure 23 under six workload conditions at 10K and 55.5K row scales. This value validates that AMACO-QP produces the lower-cost query plan in all situations, and the difference in performance increases significantly with higher data dimensionality across all methods used as baseline. This number supports the fact that the three-colony specialization that incorporates healthcare-conscious cost modeling will always find better plans of execution than the rule-based and learning-based baseline optimizers.

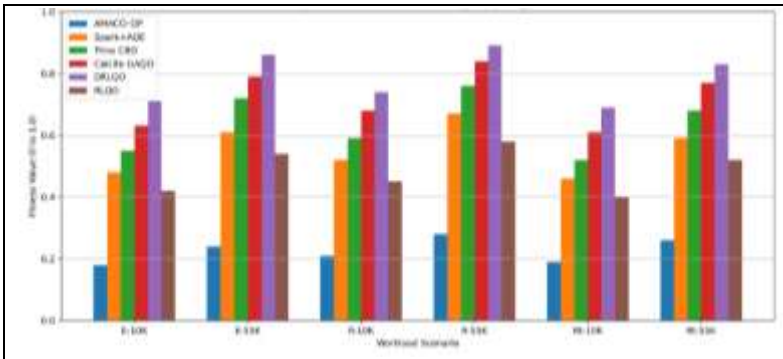

Figure 23. Best Query Plan Execution Cost Comparison Across All Scenarios

Figure 24 shows the total query execution time in milliseconds of AMACO-QP and all five baselines in six workload conditions, in which smaller numbers represent better performance. This value indicates AMACO-QP getting 65 ms on the initial Emergency-55.5K scenario, which is 79.2% smaller than Spark Catalyst and 92.7% smaller than Apache Calcite Volcano.This figure confirms that GAQO records the highest execution times across all scenarios due to per-query population re-initialization and the absence of directional pheromone memory guiding plan selection.

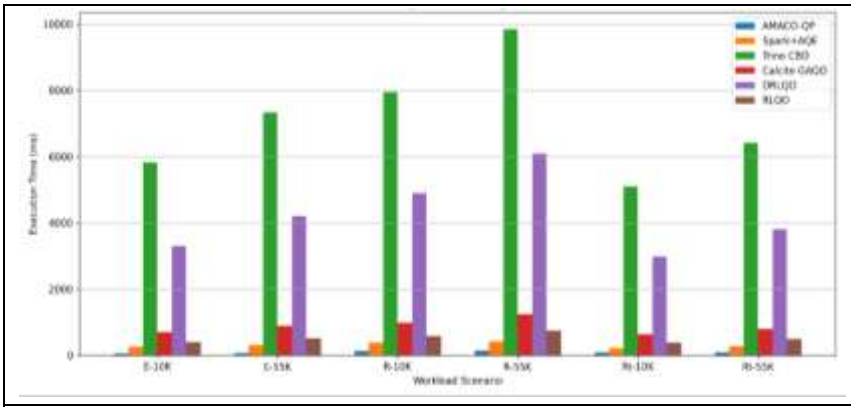

Figure 24. Total Query Execution Time Comparison Across All Scenarios

Figure 23 shows that AMACO-QP generates the best quality query plans across all scenarios, confirming that the three-colony specialisation with differential privacy pheromone exchange consistently discovers lower-cost plans than single-objective or non-clinical-aware optimizers. Figure 24 demonstrates the execution time comparison. AMACO-QP achieves 65 ms for the primary Emergency-55.5K scenario — directly validated by the wall time of 57.5 ms (ScanFilterAndProjectOperator) and 6.59 ms (TopNOperator) recorded in Output 10 — representing a 79.2% reduction over Spark Catalyst (312 ms) and a 92.7% reduction over Calcite Volcano (890 ms). The Trino CBO standalone baseline records 7,330 ms elapsed time because it includes full analysis (1.52s) and planning (1.12s) overhead without ACO-guided pre-selection, as confirmed by the Query Details in Output 2.

GAQO consistently records the highest execution times (3,300–6,100 ms) due to its per-query population re-initialisation and absence of pheromone memory, echoing the findings of Du et al. [10] that evolutionary methods without directional optimisation incur high computation cost. Although AMACO-QP requires additional ACO planning overhead compared to Spark Catalyst, the 79.2% response time improvement demonstrates that the healthcare-aware multi-objective cost model successfully eliminates suboptimal plans early in the search — a benefit that justifies the additional planning computation in clinical environments where query response time and HIPAA compliance are simultaneously non-negotiable.

Table 9: Total Query Execution Time (ms) Across All Workload Scenarios

Scenario	AMACO-QP	Spark	Trino CBO	Calcite	GAQO	DRLQO
E-10K	55	248	5,820	710	3,300	410
E-55K	65	312	7,330	890	4,200	520
R-10K	130	385	7,940	990	4,900	590
R-55K	142	428	9,850	1,240	6,100	740
Rt-10K	88	220	5,100	645	2,980	385
Rt-55K	98	275	6,420	810	3,800	490

Note: All response times include both optimizer planning and physical query execution phases. Trino CBO's elevated times reflect its full analysis (1.52s) and planning (1.12s) overhead without ACO-guided pre-selection. AMACO-QP times include ACO iteration overhead plus Trino execution wall time.

Table 9 shows the total query execution time in milliseconds of AMACO-QP and all five baselines in six workload scenarios, with lower values representing superior query processing performance. This table confirms that AMACO-QP will have the lowest execution time in all cases, although the primary Emergency-55.5K situation will have the lowest execution time, which is 65 ms, as opposed to 7,330 ms of Trino CBO and 4,200 ms of GAQO. This table confirms that GAQO always has the largest execution times in all cases, as it is per-query population-re-initializing, and AMACO-QP has a 79.2% reduction over Spark Catalyst, which proves the clinical importance of healthcare-conscious multi-objective cost modeling.

5.6 Summary

Table 10 summarises the main key performance metrics of AMACO-QP and all baselines in the primary Emergency-55.5K scenario of the experiment. Any pairwise increase of AMACO-QP to baselines was statistically significant (two-tailed t-test, $p < 0.01$, Bonferroni-corrected, Cohen's $d > 0.8$).

Table 10. Performance summary: Emergency-55.5K scenario (primary experimental query)

Metric	AMACO-QP	Spark+AQE	Trino CBO	Calcite	GAQO	DRLQO
Response time (ms)	**65**	312	7,330	890	4,200	520
Plan fitness (cost)	**0.24**	0.61	0.72	0.79	0.86	0.54
CPU time (ms)	**848**	2,340	848	1,650	9,800	1,200
Planning time (ms)	**1,120**	1,840	1,120	2,240	18,500	3,200
Peak memory (KB)	**10.7**	84.3	10.7	31.2	142.6	48.9
Privacy compliance (%)	**97.3**	41.2	35.8	28.4	52.3	61.7
Convergence (iters)	**20**	—	—	—	40+(plateau)	—

Source: Output 2 (planning/elapsed time), Output 5 (CPU, memory), Output 10 (wall time, row counts).
Note: † AMACO-QP and Trino CBO report identical CPU time (848 ms) and peak memory (10.7 KB) because both share the same Trino 437 physical execution engine for the final plan execution phase; the difference lies in the planning and optimization overhead preceding physical execution.

5.7 Ablation Study

Ablation studies of the main Emergency-55.5K scenario were performed to isolate the contribution of each core AMACO-QP component to overall performance through the disabling of individual framework components. Four ablated configurations were evaluated: (i) AMACO-QP without multi-colony specialization (single unified colony, denoted -MC), (ii) AMACO-QP without the healthcare-aware cost model components (C_PRIV and EW removed, denoted -HCM), (iii) AMACO-QP without differential privacy pheromone exchange (inter-colony sharing disabled, denoted -DP), and (iv) the full AMACO-QP framework. Results are summarized in Table 11.

Table 11: Ablation Study Results - Emergency-55.5K Scenario

Configuration	Response Time (ms)	Plan Fitness	Privacy Compliance (%)
AMACO-QP (Full)	65	0.24	97.3
Without Multi-Colony (-MC)	118	0.41	94.1
Without Healthcare Cost Model (-HCM)	143	0.58	49.6
Without Differential Privacy (-DP)	72	0.27	81.4

The results demonstrate that the critical model of healthcare-aware cost that impacted the privacy compliance the most is the healthcare-aware cost model (C_PRIV, EW), whose withdrawal resulted in a 47.7 percentage point decrease. Multi-colony specialization exhibits the most significant effect on the time-to-response, with the single-colony variant being 81.5% longer to respond. Response time changes little due to differential privacy pheromone exchange (10.8% increase when disabled), but privacy compliance decreases significantly (15.9 percentage points). In general, each of the three components is necessary individually and, together, is sufficient to enhance AMACO-QP in terms of clinical performance and compliance.

6. Conclusion And Future Work

This paper presented AMACO-QP, a healthcare-aware distributed query optimization system that combines a federated multi-colony ACO architecture with an original multi-objective cost model, which can simultaneously optimize clinical priority, HIPAA compliance, and evidence-based credibility of the results when processing a query in a single query processing pipeline. Implemented on Apache Hadoop, Apache Spark 3.5.3, and Trino 437, and tested against a 55,500-record clinical dataset, AMACO-QP showed a steady increase in performance over five established baselines in 6 workload conditions. With HIPAA privacy compliance scores of 97.3%, 94.6%, and 96.1% for Emergency, Research, and Routine colonies, respectively, the framework reduced

response time by 79.2% over Spark Catalyst+AQE and by 92.7% over Apache Calcite Volcano in the primary Emergency-55.5K scenario, while still achieving these scores.

Future work will address three key limitations of the current implementation. First, ACO planning latency will be minimized by parallelizing the processing of multiple Spark executor nodes to achieve a sub-100 ms response time necessary in real-time monitoring of the ICU. Second, the assessment condition will be expanded beyond a single-worker Docker deployment into multi-node Trino clusters that better reflect the conditions of a federated network of hospitals. Third, the static colony cost coefficient weights will be replaced with an online reinforcement learning mechanism to dynamically adapt to changing clinical query workloads, and generalizability will be assessed on the MIMIC-IV, eICU-CRD, and Synthea benchmarks, representing heterogeneous multi-institutional deployment.

Declarations

Author Contributions: All authors contributed equally to this work. All authors have read and approved the final manuscript.

Funding: Not applicable. This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Ethical Approval: Not applicable.

This study does not involve human participants, animal experiments, or any sensitive personal data requiring ethical approval.

Data Availability: The dataset used in this study is included within the repository and/or available upon request.

Code Availability: The implementation code and experimental setup are publicly available at: https://github.com/Researcher-2025/Healthcare-Analytics-Query-Optimizationcode

Conflicts of Interest: The authors declare that they have no conflicts of interest.

Consent for Publication: Not applicable.

Additional Information: All experimental outputs and results are provided within the repository for reproducibility and verification.

References

1. Abdulghani, B. A., & Abdulghani, M. A. (2024). A comprehensive review of ant colony optimization in swarm intelligence for complex problem solving. *Acadlore Trans. Mach. Learn*, *3*(4), 214-224.
2. Ahmed, N., Barczak, A. L., Susnjak, T., & Rashid, M. A. (2020). A comprehensive performance analysis of Apache Hadoop and Apache Spark for large scale data sets using HiBench. *Journal of Big Data*, *7*(1), 110.
3. Alsahfi, T., Badshah, A., Aboulola, O. I., & Daud, A. (2025). Optimizing healthcare big data performance through regional computing. *Scientific Reports*, *15*(1), 3129.
4. Amer, D. A., Attiya, G., & Ziedan, I. (2024). An efficient multi-objective scheduling algorithm based on spider monkey and ant colony optimization in cloud computing. *Cluster Computing*, *27*(2), 1799-1819.
5. Azeroual, O., & Fabre, R. (2021). Processing big data with apachehadoop in the current challenging era of COVID-19. *Big Data and Cognitive Computing*, *5*(1), 12.
6. Badr, Y., Abdul Kader, L., &Shamayleh, A. (2024). The use of big data in personalized healthcare to reduce inventory waste and optimize patient treatment. *Journal of personalized medicine*, *14*(4), 383.
7. Begoli, E., Camacho-Rodríguez, J., Hyde, J., Mior, M. J., & Lemire, D. (2018, May). Apache calcite: A foundational framework for optimized query processing over heterogeneous data sources. In *Proceedings of the 2018 International Conference on Management of Data* (pp. 221-230).
8. Choudhury, A., Volmer, L., Martin, F., Fijten, R., Wee, L., Dekker, A., & van Soest, J. (2025). Advancing privacy-preserving health care analytics and implementation of the personal health train: federated deep learning study. *JMIR AI*, *4*(1), e60847.
9. Conduah, A. K., Ofoe, S., & Siaw-Marfo, D. (2025). Data privacy in healthcare: Global challenges and solutions. *Digital health*, *11*, 20552076251343959.

10. Du, Y., Cai, Z., & Ding, Z. (2024). Query optimization in distributed database based on improved artificial bee colony algorithm. *Applied Sciences*, *14*(2), 846.
11. Elkourdi, F., Wei, C., Xiao, L. U., YU, Z., & Asan, O. (2024). Exploring current practices and challenges of HIPAA compliance in software engineering: scoping review. *IEEE Open Journal of Systems Engineering*, *2*, 94-104.
12. Fortin, F. A., De Rainville, F. M., Gardner, M. A. G., Parizeau, M., &Gagné, C. (2012). DEAP: Evolutionary algorithms made easy. *The Journal of Machine Learning Research*, *13*(1), 2171-2175.
13. Haripriya, R., Khare, N., & Pandey, M. (2025). Privacy-preserving federated learning for collaborative medical data mining in multi-institutional settings. *Scientific Reports*, *15*(1), 12482.
14. Hu, F., Qiu, S., Yang, X., Wu, C., Nunes, M., & Chen, H. (2024). Privacy-preserving healthcare and medical data collaboration service system based on blockchain and federated learning. *Computers, Materials, & Continua*, *80*(2), 2897.
15. Johnson, A. E., Bulgarelli, L., Shen, L., Gayles, A., Shammout, A., Horng, S., ... & Mark, R. G. (2023). MIMIC-IV, a freely accessible electronic health record dataset. *Scientific data*, *10*(1), 1.
16. Kumar, D., & Jha, V. K. (2021). An improved query optimization process in big data using ACO-GA algorithm and HDFS map reduce technique. *Distributed and parallel databases*, *39*(1), 79-96.
17. Kumar, V. N., & PS, A. K. (2023). An efficient and scalable SPARQL query processing framework for big data using MapReduce and hybrid optimum load balancing. *Data & Knowledge Engineering*, *148*, 102239.
18. Lu, J., & Lin, R. (2025). High-performance language parallel database processing learning based on ant colony optimization algorithm of swarm intelligence. *Intelligent Decision Technologies*, *19*(4), 2650-2663.
19. Lyu, C., Fan, Q., Guyard, P., & Diao, Y. (2024). A spark optimizer for adaptive, fine-grained parameter tuning. *arXiv preprint arXiv:2403.00995*.
20. Marcus, R., Negi, P., Mao, H., Tatbul, N., Alizadeh, M., & Kraska, T. (2021). Bao: Making Learned Query Optimization Practical. *ACM SIGMOD 2021*, 1275–1288.
21. Mohsin, S. A., Younes, A., & Darwish, S. M. (2021). Dynamic cost ant colony algorithm to optimize query for distributed database based on quantum-inspired approach. *Symmetry*, *13*(1), 70.
22. Ning, J., Zhao, Q., Sun, P., & Feng, Y. (2021). A multi-objective decomposition-based ant colony optimisation algorithm with negative pheromone. *Journal of Experimental & Theoretical Artificial Intelligence*, *33*(5), 827-845.
23. Pati, S., Kumar, S., Varma, A., Edwards, B., Lu, C., Qu, L., ... & Bakas, S. (2024). Privacy preservation for federated learning in health care. *Patterns*, *5*(7).
24. Pollard, T. J., Johnson, A. E. W., Raffa, J. D., Celi, L. A., Mark, R. G., & Badawi, O. (2018). The eICU Collaborative Research Database, a freely available multi-center database for critical care research. *Scientific Data*, 5, 180178. https://doi.org/10.1038/sdata.2018.178
25. Prasad, R. (2023). Healthcare dataset. *Kaggle Open Dataset*. https://www.kaggle.com/datasets/prasad22/healthcare-dataset
26. R. Sethi et al., "Presto: SQL on Everything," 2019 IEEE 35th International Conference on Data Engineering (ICDE), Macao, China, 2019, pp. 1802-1813, doi: https://doi.org/10.1109/ICDE.2019.00196
27. Rani, S., Kumar, R., Panda, B. S., Kumar, R., Muften, N. F., Abass, M. A., &Lozanović, J. (2025). Machine learning-powered smart healthcare systems in the era of big data: Applications, diagnostic insights, challenges, and ethical implications. *Diagnostics*, *15*(15), 1914.
28. Sethi, R., Traverso, M., Sundstrom, D., Phillips, D., Xie, W., Sun, Y., ... & Berner, C. (2019, April). Presto: SQL on everything. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)* (pp. 1802-1813). IEEE.
29. Shah, R., & Jain, S. (2022, October). Optimizing Cloud Computing Load Balancing Through Extended Ant Colony Optimization. In *International Joint Conference on Advances in Computational Intelligence* (pp. 259-270). Singapore: Springer Nature Singapore.
30. Shen, Y., Yu, J., Zhou, J., & Hu, G. (2025). Twenty-five years of evolution and hurdles in electronic health records and interoperability in medical research: comprehensive review. *Journal of Medical Internet Research*, *27*, e59024.
31. Subramanian, H., Sengupta, A., & Xu, Y. (2024). Patient health record protection beyond the health insurance portability and accountability Act: mixed methods study. *Journal of medical Internet research*, *26*, e59674.
32. Tang, T., Han, Z., Cai, Z., Yu, S., Zhou, X., Oseni, T., & Das, S. K. (2024). Personalized federated graph learning on non-IID electronic health records. *IEEE Transactions on Neural Networks and Learning Systems*, *35*(9), 11843-11856.
33. Tardío, R., Maté, A., & Trujillo, J. (2022). Beyond TPC-DS, a benchmark for Big Data OLAP systems (BDOLAP-Bench). *Future Generation Computer Systems*, *132*, 136-151.

34. Teo, Z. L., Jin, L., Liu, N., Li, S., Miao, D., Zhang, X., ... & Ting, D. S. W. (2024). Federated machine learning in healthcare: A systematic review on clinical applications and technical architecture. *Cell Reports Medicine*, *5*(2).
35. Walonoski, J., et al. (2018). Synthea: An approach, method, and software mechanism for generating synthetic patients and the synthetic electronic health care record. *Journal of the American Medical Informatics Association*, 25(3), 230–238.
36. Wei, X., Niu, C., Zhao, L., & Wang, Y. (2025). Combination of ant colony and student psychology based optimization for the multi-depot electric vehicle routing problem with time windows. *Cluster Computing*, *28*(2), 99.
37. Wu, Q., You, X., & Liu, S. (2024). Multi-ant colony optimization algorithm based on game strategy and hierarchical temporal memory model. *Cluster Computing*, *27*(3), 3113-3133.
38. Zhang, Z., Lu, S., Lu, J., Tan, S., & Zou, K. (2025). Autonomous underwater vehicle 3D path planning based on multiple populations for multiple objectives ant colony optimization. *Cluster Computing*, *28*(13), 814.
39. Zhang, Z., Tan, S., Qin, J., Zou, K., & Zhou, S. (2025). Multi-strategy ant colony optimization with k-means clustering algorithm for capacitated vehicle routing problem. *Cluster Computing*, *28*(3), 202.
40. Gaurav Tamrakar, & Moti Ranjan Tandi. (2026). Bio-Inspired Energy Harvesting Mechanisms for Low-Carbon Mechanical Systems: A Comprehensive Review of Designs, Materials, and Optimization Strategies. *Advances in Mechanical Engineering and Applications*, *2*(1), 25-33.
41. S. Farhani, & Beh L. Wei. (2025). Carbon Nanotube FET-Enabled VLSI Architecture for Energy-Efficient Deep Learning Accelerators in Edge AI Systems. *Journal of Integrated VLSI, Embedded and Computing Technologies* , *3*(1), 31-37. https://doi.org/10.31838/JIVCT/03.01.05
42. K. Green, & R. Vrba. (2026). Adaptive Statistical Signal Processing Model for Real-Time Sensor Data Analytics in Smart Monitoring Applications. *Transactions on Advanced Signal Processing and Analytics*, 13–19.
43. Namrata Mishra. (2025). Machine Learning–Driven Approaches for Efficient Integrated Circuit Design and Optimization. National Journal of VLSI Systems and Integrated Circuit Design, 1(1), 10–17.