SvedbergOpen
DISSEMINATION OF KNOWLEDGE
International Journal of Artificial Intelligence and Machine Learning
Publisher's Home Page: https://www.svedbergopen.com/
Research Paper Open Access

A Transformer-Guided Semantic and Swarm-Based Optimization Model for High Performance Database Query Processing using TSPACO

R. Jeevitha[1], L. Ramesh[2]

[1]Research Scholar Department of Computer Applications, School of Computing Sciences, Vels Institute of Science, Technology & Advanced Studies (VISTAS), Pallavaram, Chennai, Tamilnadu 600 117 – India, Email: jeevitharesearchwork@gmail.com

[2]Assistant Professor, Department of Computer Applications, School of Computing Sciences, Vels Institute of Science, Technology & Advanced Studies (VISTAS), Pallavaram, Chennai, Tamilnadu 600 117 – India, Email: lramesh.scs@vistas.ac.in

Abstract

Query optimization is a crucial operation within a current database system used to enhance performance of the database in terms of speed, accuracy and efficiency of resource usage in a database system when dealing with large scale relational information. The common limitations of the existing query optimization techniques include high execution time, high cost of computation and also inability to adapt to complex SQL queries. This research aims at coming up with an intelligent and dynamic query optimization framework that has the capability of producing the best execution plans in a manner that is both accurate and low in computation costs. A Transformer based Semantic and Hybrid PSO-ACO (TSPACO) query optimization model is proposed as a result of this aim, integrating semantic query analysis, row and table filtering with a hybrid optimization combining Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO). The semantic analysis step determines the tables and rows that are relevant whereas the hybrid optimization step chooses the optimal execution path by balancing between exploration and exploitation strategies. To test the proposed method, Python is used and tested with relational query data to quantify execution time, accuracy, cost of computations and throughput. The proposed TSPACO method is demonstrated to outperform the existing methods with better performance as indicated in experimental results with a query accuracy of 97% at a shorter time and lower cost of computation. The findings affirm that the proposed framework is an efficient and scalable intelligent query optimization solution to large database systems.

Keywords: Ant Colony Optimization, Database Query Optimization, Particle Swarm Optimization, Semantic Analysis, SQL Query Processing, Transformer Model

1. Introduction

Efficient query optimization is a key component of the modern database system that facilitates an improved speed, precision, and use of resources particularly when it involves large scale and heterogeneous data collected through IoT, enterprise applications, and intelligent systems. Conventional query processing systems tend to be slow in responding to the changes in the workload, and this results in a higher rate of execution time and computation overhead. The application of AI-controlled optimization, predictive indexing, and intelligent resource distribution to enhance the performance of databases has been studied recently, with AI-based query planning methods shown to reduce the execution delay [1], autonomous self-tuning database architecture found to yield better throughput and the indexing efficiency [2], and AI-controlled optimization model models like RNN with attention mechanisms have been shown to be more effective in making decisions in a complex environment [3]. Smart city energy systems have also implemented advanced strategies to optimize the use of better computational performance with advanced DL optimization strategies [4], and unified Machine Learning (ML) and data-warehousing platforms to optimize large industrial datasets [5]. Also, supply-chain optimization has been combined with Artificial Intelligence (AI) to enhance system reliability [6], big-data oriented optimization strategies have minimized energy usage in extensive infrastructure [7], and smart AI assistants have been

applied in intelligent decision support systems in education [8]. Knowledge-based optimization of dynamic networks has also led to efficiency in the task allocation [9], secure edge computing optimization has improved the work of IoT healthcare systems [10], and learning-based analytics have been applied to optimize resources efficiently in renewable energy systems [11]. It has also been demonstrated that multi-objective fuzzy optimization is more accurate in complicated prediction problems [12], BiLSTM-based optimized models have a higher semantic extraction rate [13], and Bayesian optimization is utilized to increase the efficiency of models used in intrusion detection systems [14]. Moreover, combined AI-IoT solutions have been shown to have better resource management in various fields [15].

Although there have been developed technologies recently, the current query optimization methods have failed to provide flexibility in real-time, high level of computational complexity, and poor query execution planning when running intricate SQL queries on big relational data. Most of the methods that are currently present are centered on prediction, or optimization of the indexing and do not adopt a standardized approach comprising of semantic analysis and adaptive optimization strategies. In order to address these restrictions, this research proposes a Transformer-based Semantic and Hybrid PSO-ACO (TSPACO) query optimization framework, conducting semantic query analysis, row filtering, table filtering and hybrid optimization via Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO). The method proposed is aimed at decreasing the time required to execute the query, enhancing accuracy, reducing the cost of computation, and increasing the throughput through the selection of the relevant data and the creation of an optimal execution plan. The key contribution of the work is the creation of an adaptive and scalable semantic-aware hybrid optimization framework that is able to efficiently deal with complex queries in big scale intelligent database systems.

Organization: In Section 1, the introduction will be given, in Section 2, the background research and associated research on the current optimization techniques will be given and finally, in Section 3, the proposed TSPACO query optimization approach will be described. Section 4 presents the results of the experiment and the performance analysis and the last section, 5, is the conclusion part of the research that will discuss future research directions.

2. Background Study

V. Panwar (2024) [1] proposed query-level AI-enhanced optimization to optimize database performance, predictive modeling and indexation techniques; query planning based on heuristic algorithms, lack of real-time responsiveness was identified as gap, restricted to medium-scale datasets, a lower time of execution and increased efficiency.

N. O. Oloruntoba (2025) [2] created an independent database management with self-tuning and future query optimization; applied ML to indexing and query execution which reported low testing in heterogeneous enterprise settings, and made better throughput and resource use.

S. A. Alzakari et al. (2025) [3] introduced a smart ransomware detector model based on multi-head attention RNN and optimization algorithms to work with IoT setting; models integrated DL and metaheuristic optimization, despecified gaps in real-time deployment scalability, and based on simulation data, the results revealed excellent threat detection rates.

I. Rojek et al. (2025) [4] investigated the optimization of energy consumption in smart cities with sophisticated DL algorithms; used CNN-LSTM hybrid algorithms to predict energy consumption, discrepancy between model interpretability, and was able to operate under sudden load spikes, demonstrated better energy efficiency indicators.

N. Su et al. (2024) [5] postulated a unified predictive maintenance system consisting of data warehousing, Apache Spark and ML to smart manufacturing; ensemble learning to predict faults, research gap: scalability to multi-factories, lack of historical data variety, led to lower downtimes and maintenance optimization.

M. Riad et al. (2024) [6] designed AI-based framework of supply chain optimization to enhance resilience, used conceptual modeling and optimization of simulation, gap in dynamic adaptation in real-time, limited to theoretical validation, and obtained better risk mitigation and supply chain efficiency.

V. Marinakis (2020) [7] studied the application of big data in energy management in buildings through predictive analytics and optimization methods; gap in real-time adaptive control, constraint in sensor heterogeneity, and proved to be easier in energy consumption and lowered operational cost.

R. Sajja et al. (2024) [8] created an intelligent assistant with AI capabilities to support personalized learning in higher education; utilized adaptive learning algorithms and feedback based on NLP, gap in the research field in large-scale implementation, limited by heterogeneous curriculum datasets, attained greater student engagement and learning.

Table 1: Background Study OnThe Optimization Techniques In AI and Iot Systems					
Author(s) & Year	**Concept**	**Methods Used**	**Research Gap**	**Limitation**	**Key Results**
C.-C. Hu (2026) [9]	Knowledge-based optimization for task offloading in vehicular fog networks	Applied reasoning algorithms with dynamic optimization; used semantic knowledge modeling for decision making	Limited studies on real-time intelligent offloading	Tested mostly in simulations, not large-scale deployment	Improved task allocation efficiency and reduced latency in dynamic networks
R. Gowthamani & S.O. Manoj (2026) [10]	Edge computing optimization in healthcare IoT	Proposed graph-embedded musical chairs optimization with elliptic encryption	Security and optimization rarely combined in edge healthcare systems	Limited validation in real hospital networks	Achieved secure, efficient resource allocation with reduced computation time
D. Behl et al. (2026) [11]	Solar energy optimization via learning-driven analytics	Combined visual intelligence with ML-based optimization for solar panels	Gap in integrating visual analytics with energy prediction models	Limited to specific solar datasets	Enhanced solar output prediction and operational efficiency
S. Mitra et al. (2025) [12]	Multi-objective optimization for protein interactions	Fuzzy multi-objective optimization combined with intelligent models	Need for more accurate, scalable protein interaction prediction	Limited experimental validation	High accuracy in predicting HCV-human protein interactions
R. Khatun & A. Sarkar (2025) [13]	NLP-based triple extraction and RDF generation	Boosted BERT fused attention Bi-LSTM with optimization	Gap in combining DL with semantic optimization	Tested on limited datasets	Improved extraction accuracy and RDF generation efficiency
T.A. Kumari & S. Mishra (2024) [14]	Intrusion detection optimization	Enhanced stacked models using Bayesian optimization and sampling techniques	Limited research on sampling techniques for optimized intrusion detection	Evaluation limited to specific network datasets	Improved detection rate and reduced false positives
A. Alzahrani et al. (2025) [15]	AI + IoT integration for ecological and medical services	Developed AI-driven IoT framework for resource optimization	Gap in unified AI-IoT frameworks for multi-domain applications	Limited real-world implementation	Achieved efficient resource allocation and improved service delivery

The above table 1, highlights the current literature of the AI-driven optimization solutions in the field of IoT, energy, and healthcare to outline the methodologies, gaps in the research, limitations, and significant findings to provide the background of the intelligent database query optimization system.

3. Proposed Methodology

This part outlines the proposed TSPACO model of intelligent database query optimization which covers the general scheme, the workflow and hybrid optimization plan. The section outlines the semantic query analysis based on Transformer embedding as followed by filtering between rows and tables and the hybrid PSO-ACO optimization procedure to come up with the best execution plan. To ensure efficient and accurate query processing, there are also given mathematical equations and pseudocode to formally define the optimization steps, fitness evaluation and final query execution procedure.

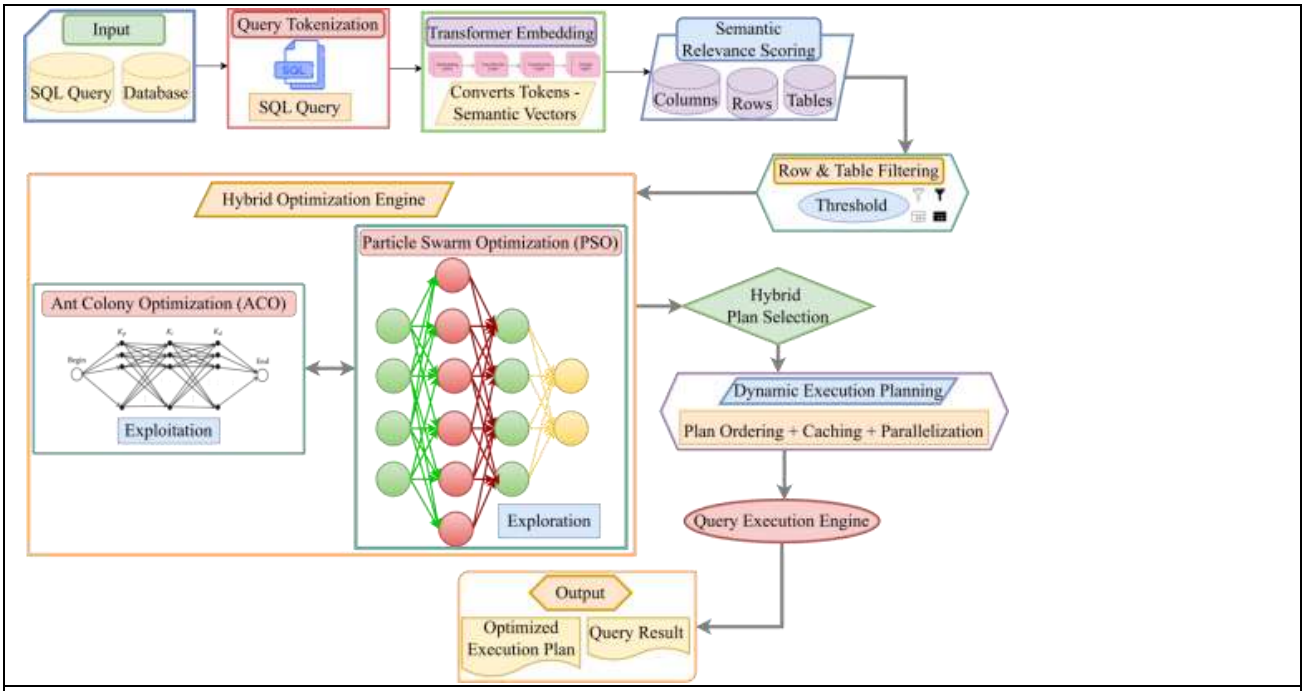

Fig. 1: Semantic Query Optimization Architecture- Hybrid PSO-ACO

The diagram above figure 1 illustrates the end-to-end pipeline of the semantic SQL query optimization which begins with the query tokenization and transformer embedding to calculate the semantic vectors. It demonstrates relevance scoring and a row/table filtering followed by a hybrid optimization engine, which uses PSO to explore the best execution plan and exploitation of the ACO to achieve it. Lastly, query execution engine and dynamic execution planning generate the optimized execution plan and the query result.

3.1 Semantic Query Analysis

The input query in this stage is processed by a Transformer model to derive the semantic context of the input query and get the understanding between various parts of the query like tables, columns and conditions. The query is tokenized and vectors are uncovered into a high-dimensional space, and contextual relationship is learned with the help of self-attention. These embeddings enable the system to determine the most relevant tables, rows and columns to be used and also disregard the irrelevant information. Transformer use several instances of attention and feed-forward networks to extract a semantic form of the query, upon which optimization is applied. Such a representation helps the framework to minimize redundant processing of data thereby enhancing efficiency. Planning queries The system may raise the priority of significant database elements by mapping the query into a semantic vector space so that the query only takes into account the relevant information when planning queries. It leads to a more intelligent and quicker query processing at a lower computational cost.

$$E = Embed(Q) \quad (1)$$

Where in equation (1), E is a semantic embedding vector of query, Q is input query text and Embed () is an embedding function, which transforms tokens into dense vectors. Languages representing semantics in a numerical form by converting raw query tokens.

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (2)$$

In equation (2) denotes that Q is a query matrix (embedding), K principle is a key matrix (embedding), V principle is a value matrix (embedding), d_k is a dimension of key vectors (scaling). Calculate contextual relevance of each token in the query to all other tokens, and records relationships.

$$S_i = \sigma(W \cdot E_i + b) \quad (3)$$

Wherein equation (3) denotes S_i is a relevance score of table/row i, E_i is a semantic embedding of table/row i, W, b is learned weights and bias of mapping embeddings to scores, σ is a sigmoid (scales scores to 0 through 1). Provides relevance probability to each table/row and makes the optimizer select only important data.

3.2 Row and Table Filtering

The system then processes the resultant embeddings and relevance scores after semantic query analysis to filter the irrelevant tables, columns and rows, in effect reducing the search space of the database. A semantic relevance score is given to each table and row and only those with a specified threshold are chosen to be executed. This filtering reduces irrelevant data retrieval and calculation and makes the most significant data of the query. The optimizer is able to discard the irrelevant low-relevance entries thus generating the execution plans more efficiently and the outcome of the query processing is faster. On the whole, semantic-based filtering guarantees that the database engine performs operations on the data only which makes a substantial contribution to the query output and enhances the speed and resource consumption.

$$Selected_i = \begin{cases} 1 \ if S_i \geq \tau \\ 0 \ if S_i < \tau \end{cases} \quad (4)$$

In equation (4) denotes that $Selected_i$ is a binary indicator (1 = selected, 0 = ignored) of table/row i, S_i is a semantic relevance score of table/row i, τ is a selection threshold that is predefined. Eliminate irrelevant data by choosing only tables/rows that have scores of higher than the threshold.

$$\mathcal{D}_{filtered} = \{R_i | Selected_i = 1\} \quad (5)$$

Where in equation (5) is the representation of the fact that $\mathcal{D}_{\text{filtered}}$ is a filtered version of the data following pruning, R_i is a single table/row of the original database. Has the ability to construct a smaller dataset, only high-relevant rows/tables, to minimize computation in terms of optimization.

3.3 TSPACO Optimization

The Step 2 filtered dataset is then run through the TSPACO hybrid framework in this step to decide the best query implementation plan. Particle Swarm Optimization (PSO) considers a variety of possible plans of execution by simulating the movement of particles in the multidimensional space of search, with each particle corresponding to one possible plan. At the same time, the ACO assesses these plans and chooses the best course based on pheromone trails based on the query type, table size and work load which will lead to preferential choices of execution paths that are more efficient. The hybrid method has both the dynamic updating of the particle position and the pheromone concentration to refine the execution strategy adaptively depending on the performance metrics. This enables the system to appropriately deal with variable workloads and non-homogenous sets of data. The joint PSO-ACO procedure ensures that the optimizer determines the execution strategy which minimizes calculation, maximizes speed as well as high accuracy of the intricate queries. In general, TSPACO unites exploration and exploitation planning to offer intelligent and adaptive query execution planning.

$$X_i^{t+1} = X_i^t + V_i^{t+1} \quad (6)$$

In equation (6) means that X_i^t is a position of particle at time t iteration i (is a candidate execution plan), V_i^{t+1} is a velocity vector that changes particle position. Derive the plans of candidate execution by searching the space.

$$V_i^{t+1} = wV_i^t + c_1 r_1 (P_i - X_i^t) + c_2 r_2 (G - X_i^t) (7)$$

Equation (7) above denotes that w is the inertia weight, c_1, c_2 is a acceleration coefficients, r_1, r_2 is a random factors (0-1), P_i is a personal best position of particle i, G is a global best position between all particles. Exploration of the balance followed by exploitation so as to direct particles into superior execution strategies.

$$\tau_{ij} \leftarrow (1 - \rho)\tau_{ij} + \Delta\tau_{ij} \quad (8)$$

In equation (8) indicates that τ_{ij} is the intensity of pheromone on the path ij, ρ is evaporated rate($0 < \rho < 1$), $\Delta\tau_{ij}$ is pheromone deposited depending on the quality of plan. Encourage a good execution course and progressively eliminating the impact of poor execution courses.

$$p_{ij} = \frac{\tau_{ij}^{\alpha} \eta_{ij}^{\beta}}{\sum_{k \epsilon N_i} \tau_{ik}^{\alpha} \eta_{ik}^{\beta}} (9)$$

In equation (9) means that p_{ij} is a probability to choose the path between node i and j, η_{ij} is a desirability of heuristic (by table size, query cost) and α, β is an influence parameters of pheromone and heuristic, N_i is a feasible next nodes i. Lead ants to better successful execution routes with the help of pheromone and heuristic data.

$$F = w_1 T_{exec} + w_2 C_{comp} + w_3 (1 - A_{acc}) (10)$$

In equation (10) signifies that F is a fitness score of the candidate execution plan (low is preferable), T_{exec} is a estimated execution time, C_{comp} is a computation cost, A_{acc} is a projected accuracy of plan execution, w_1, w_2, w_3 is a weight factors of balancing objectives. Measure the quality of every execution plan to optimize PSO and ACO.

3.4 Query Execution

After identifying the best execution plan in TSPACO, a query is implemented on the database according to the plan therefore giving maximum efficacy. First, the system only does the disk accesses and memory use of the filtered tables and rows that have been determined in Steps 1 and 2. The execution engine then goes through with the operations (joins, selections, aggregations) in the order of the optimal order presented by the TSPACO framework. Dynamic scheduling is used to make sure that the operations that have large computational cost are prioritized and are also parallelized where possible. The results can be stored in the middle to eliminate any unnecessary calculations, which are even faster to execute. Lastly, the optimized plan fits well in reduction of time of execution and at the same time accuracy even in complex queries and huge heterogeneous datasets. This phase is needed to make sure that the advantages of semantic analysis and hybrid optimization can be directly applied to practice in terms of performance improvement.

$$T_{exec} = \sum_{i=1}^{n} C_i \cdot R_i \qquad (11)$$

In equation (11) indicates that T_{exec} is a summation of estimated costs of execution of the query, C_i is a cost of operation i (e.g., join, selection), R_i is number of rows handled by operation i. Divides operation cost and related number of relevant rows to determine the total time of execution which is an indication of efficiency of the optimized plan.

$$A_{res} = \frac{|R_{actual} \cap R_{predicted}|}{|R_{actual}|} \qquad (12)$$

The equation (12) is a representation of the A_{res} is an accuracy of query results on execution, R_{actual} a set of actual rows that satisfies query, $R_{predicted}$ is a set of rows returned by optimized execution plan. Check data integrity by checking whether query results are correct.

Algorithm: TSPACO Query Optimization

```
Input:
Q → Input SQL query
D → Database containing tables, rows, and columns
τ → Semantic relevance threshold
MaxIter_PSO → Maximum iterations for PSO
MaxIter_ACO → Maximum iterations for ACO
Weights → Weight factors for execution plan evaluation
 // Step 1: Semantic Query Analysis
Q_tokens = Tokenize(Q)
E_query = TransformerEmbed(Q_tokens)      // Generate semantic embedding
S_tables_rows = ComputeSemanticScores(E_query, D)  // Relevance scores for tables/rows
// Step 2: Row and Table Filtering
Filtered_Data = {}
for each table/row i in D:
   if S_tables_rows[i] >= τ:
      Add i to Filtered_Data
   end if
 end for
// Step 3: TSPACO Optimization (Hybrid PSO + ACO)
Initialize particle swarm positions X_i and velocities V_i
Initialize pheromone matrix τ_ij for ACO
for t = 1 to MaxIter_PSO:
   for each particle i:
      Plan_i = GenerateExecutionPlan(X_i, Filtered_Data)
      F_i = EvaluatePlanFitness(Plan_i, Weights)      // Using execution time, cost, accuracy
      Update personal best P_i if F_i improves
   end for
   Update global best G among all particles
   Update particle velocities V_i and positions X_i
   for iteration = 1 to MaxIter_ACO:
      for each ant k:
         Construct plan based on pheromone τ_ij and heuristic info
         F_k = EvaluatePlanFitness(plan_k, Weights)
         Update pheromone Δτ_ij based on plan quality
      end for
      Evaporate pheromone: τ_ij = (1 - ρ) * τ_ij + Δτ_ij
   end for
```

```
end for
Plan_opt = Global best execution plan G
// Step 4: Query Execution
Execute Plan_opt on Filtered_Data
Store final result in R_opt
return R_opt, Plan_opt
Output:
R_opt → Final query result
Plan_opt → Optimized query execution plan
```

The pseudocode based on the TSPACO framework refers to the initial analysis of semantic queries with the help of the Transformer to score relevance and filter irrelevant tables and rows with the help of which the search space is reduced. It then uses a hybrid PSO-ACO optimization to search and achieve the most optimal execution plan and eventually executed on the filtered dataset to generate the optimized query result in a cost-effective manner.

4. Result and Discussion

This section includes of the proposed TSPACO query optimization framework where performance test is performed using Python programming language by experiment by simulation. The outcome to determine the efficiency of the proposed approach is to execute different optimization methods in Python and compare the execution time, the accuracy, cost of computation and throughput.

4.1 Hardware Setup

The experiments are executed on an architecture with a multi-core processor (Intel i7 or Xeon), 16-32Gb RAM, and massive storage to have the potential of processing large datasets. The deployed database system is a relational database management system (RDBMS) e.g. MySQL or PostgreSQL, and supports the standard query processing. This hardware configuration will be able to perform complex queries besides measuring the performance improvements introduced by TSPACO.

4.2 Experimental Setup

In the analysis of TSPACO, it will be using the Synthetic E-Commerce Relational Dataset https://www.kaggle.com/datasets/naelaqel/synthetic-e-commerce-relational-dataset onKaggle, which is a set of a few inter-relational tables which are the real life e-commerce deals. The questions will be categorized as simple, moderate and complex questions to determine the functionality of the framework under various degrees of computation. Each query is repeated several times to obtain statistical reliability and mean execution time, cost of processing and accuracy is recorded. Such a set-up can make comparisons of TSPACO to the optimization means of the baseline to different and realistic database conditions in a systematic manner.

4.3 Performance Evaluation

To examine the efficiency of the proposed TSPACO method, this section provides the performance analysis of the different query optimization techniques using Python implementation comparing the execution time, accuracy, cost of calculation as well as throughput.

Table 2: Comparison of Query Optimization Techniques

Method	Query Execution Time (T_exec, sec)	Query Accuracy (A_res, %)	Computational Cost (C_comp, CPU units)	Throughput
RNN [3]	12.5	85	120	100
BiLSTM [13]	11.2	88	115	114
Fuzzy Multi-Objective Optimization [12]	10.8	90	110	126
Bayesian Optimization [14]	10.2	92	108	133
Proposed TSPACO	6.5	97	80	185

Table 2 above provides the performance of the different query optimization methods based on the query execution time, query accuracy, the cost and throughput of the computation. The proposed TSPACO method is more appropriate in the view of shorter execution time and lower computation cost and higher accuracy and

throughput of pheromone and probabilistic search optimization process, based on adaptive pheromone and probabilistic searching.

Fig. 2: Comparison of Query execution time of different optimization techniques

Figure 2 above provides the comparison of the query execution time of the different optimization methods namely RNN, BiLSTM, Fuzzy Multi-Objective Optimization, Bayesian Optimization and the proposed TSPACO. The methods of optimization are then shown on X-axis and the time of execution in seconds on Y-axis whereby, the proposed TSPACO method will have a shorter execution time in terms of optimised search process and pheromone-based path selection process.

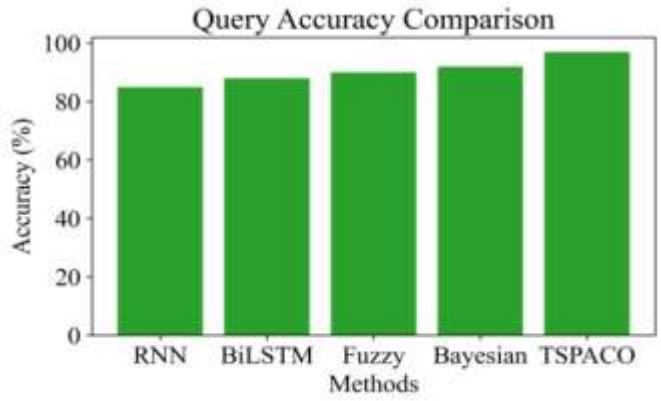

Fig. 3: Comparison of Accuracy of query on the various optimization methods

The above figure 3 shows the comparison of accuracy of query made by various algorithms in the course of query optimization. It should be noted that the X-axis is the optimization techniques and the Y-axis is the percentage of accuracy where the proposed TSPACO is better than the others by adaptive pheromone update and probabilistic selection mechanism to generate more accurate query results.

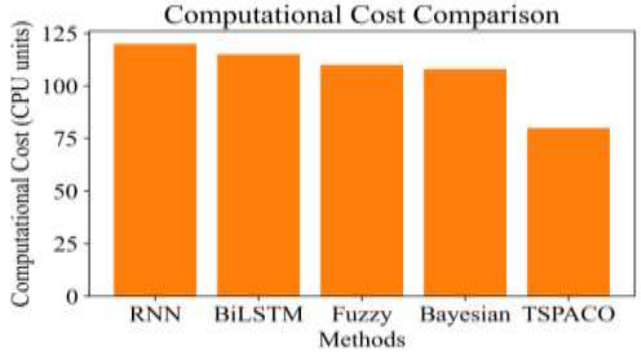

Fig. 4: Computational Comparison Costs of various Optimization Methods

The above figure 4 shows the calculation cost of each optimization method in processing database queries. The X-axis is the optimization methods and the Y-axis is the units of the CPU to compute the queries, whereby the proposed TSPACO can save on cost through efficient path search, less iterations and efficient allocation of resources in processing the queries.

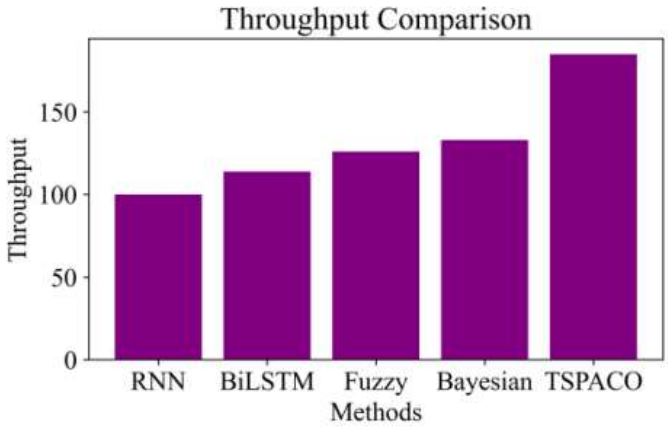

Fig. 5: Comparison of various optimization methods Per throughput

The above figure 5 displays the throughput performance of the various optimization techniques in terms of the number of queries being served per unit time. The X-axis indicates the optimization methods and the Y indicates the throughput in which the recommended TSPACO performs better throughput in terms of parallel search capacity, adaptive pheromone learning and faster convergence process.

5. Conclusion

In this research, a TSPACO query optimization framework is proposed based on the need to enhance the efficiency of SQL query running in large relational database systems. The method proposed combines the analysis of semantic queries, a row and table filtering, hybrid optimization with the help of PSO and ACO to form an optimal execution plan. The framework saves unneeded computations and enhances the overall performance by choosing only the data that is relevant and then executing the optimal paths. The experimental findings that have been conducted through Python illustrate that the proposed TSPACO methodology can deliver a minimized query execution time, high accuracy, low cost of computation, and high throughput in contrast to the current methods of RNN, BiLSTM, Fuzzy multi-objective optimization and Bayesian optimization. The hybrid optimization approach is effective in balancing the exploration and exploitation, which makes the system appropriate in the complex and large-scale query processing environments. The proposed framework can be expanded to include DRL to plan queries dynamically in real-time database systems in the future. It is also possible to apply the model to distribute and cloud based databases to enhance scalability and performance of the model in a large enterprise environment.

Reference

1. V. Panwar, "AI-driven query optimization: Revolutionizing database performance and efficiency," Int. J. Comput. Trends Technol., vol. 72, no. 3, pp. 18–26, 2024. doi: 10.14445/22312803/IJCTT-V72I3P103.
2. N. O. Oloruntoba, "AI-Driven autonomous database management: Self-tuning, predictive query optimization, and intelligent indexing in enterprise IT environments," World J. Adv. Res. Rev., vol. 25, no. 2, pp. 1558–1580, 2025. doi: 10.30574/wjarr.2025.25.2.0534.
3. S. A. Alzakari, M. Aljebreen, N. Ahmad, A. A. Alhashmi, S. Alahmari, O. Alrusaini, et al., "An intelligent ransomware based cyberthreat detection model using multi head attention-based recurrent neural networks with optimization algorithm in IoT environment," Sci. Rep., vol. 15, no. 1, p. 8259, 2025. doi: 10.1038/s41598-025-92711-4.
4. Rojek, D. Mikołajewski, K. Galas, and A. Piszcz, "Advanced deep learning algorithms for energy optimization of smart cities," Energies, vol. 18, no. 2, p. 407, 2025. doi: 10.3390/en18020407.
5. N. Su, S. Huang, and C. Su, "Elevating smart manufacturing with a unified predictive maintenance platform: The synergy between data warehousing, Apache Spark, and machine learning," Sensors, vol. 24, no. 13, p. 4237, 2024. doi: 10.3390/s24134237.

6. M. Riad, M. Naimi, and C. Okar, "Enhancing supply chain resilience through artificial intelligence: Developing a comprehensive conceptual framework for AI implementation and supply chain optimization," Logistics, vol. 8, no. 4, p. 111, 2024. doi: 10.3390/logistics8040111.
7. V. Marinakis, "Big data for energy management and energy-efficient buildings," Energies, vol. 13, no. 7, p. 1555, 2020. doi: 10.3390/en13071555.
8. R. Sajja, Y. Sermet, M. Cikmaz, D. Cwiertny, and I. Demir, "Artificial intelligence-enabled intelligent assistant for personalized and adaptive learning in higher education," Information, vol. 15, no. 10, p. 596, 2024. doi: 10.3390/info15100596.
9. C.-C. Hu, "Knowledge-based optimization and reasoning for intelligent task offloading in dynamic vehicular fog networks," Knowledge-Based Syst., vol. 338, p. 115507, 2026. doi: 10.1016/j.knosys.2026.115507.
10. R. Gowthamani and S. O. Manoj, "A novel graph-embedded musical chairs optimization with secure elliptic encryption framework for intelligent edge computing in healthcare IoT networks," J. Ind. Inf. Integr., vol. 49, p. 101007, 2026. doi: 10.1016/j.jii.2025.101007.
11. D. Behl, A. Tomar, H. Kumar, and S. Sharma, "Solar optimization via learning-driven visual intelligent analytics," Renewable Energy, vol. 260, p. 125171, 2026. doi: 10.1016/j.renene.2025.125171.
12. S. Mitra, A. Chakraborty, M. Bhattacharjee, D. De, and A. J. Pal, "Determining human–hepatitis C virus protein interactions: A synergism of fuzzy multi-objective optimization and machine intelligent models," Next Research, vol. 2, no. 1, p. 100105, 2025. doi: 10.1016/j.nexres.2024.100105.
13. R. Khatun and A. Sarkar, "Classification of triple extraction and RDF generation using boosted BERT fused attention convolutional bi-directional LSTM with optimization," Knowledge-Based Syst., vol. 324, p. 113756, 2025. doi: 10.1016/j.knosys.2025.113756.
14. T. A. Kumari and S. Mishra, "Tachyon: Enhancing stacked models using Bayesian optimization for intrusion detection using different sampling approaches," Egyptian Informatics J., vol. 27, p. 100520, 2024. doi: 10.1016/j.eij.2024.100520.
15. Alzahrani, P. Kostkova, H. Alshammari, S. Habibullah, and A. Alzahrani, "Intelligent integration of AI and IoT for advancing ecological health, medical services, and community prosperity," Alexandria Eng. J., vol. 127, pp. 522–540, 2025. doi: 10.1016/j.aej.2025.05.046.
16. Robbi Rahim. (2026). A Hybrid Statistical–Machine Learning Framework for Robust Sensor Data Analytics in Noisy Environments. *Transactions on Advanced Signal Processing and Analytics*, *1*(1), 1–6.
17. Shaik Sadulla. (2025). Safety-Constrained Multi-Agent Learning Protocols for Reliable and Energy-Aware Control in Connected Vehicular Communication Systems. *Transactions on Secure Communication Networks and Protocol Engineering*, 1-9.
18. Gajraj Singh. (2026). Bifurcation and Stability Analysis of Nonlinear Systems with Computational Validation. *Frontiers in Mathematical and Computational Research*, 11-19.