Journal of Advances in Mathematical & Computational Sciences
An International Pan-African Multidisciplinary Journal of the SMART Research Group
International Centre for IT & Development (ICITD) USA
© Creative Research Publishers
Available online at https://www.isteams.net/ mathematics-computationaljournal.info
DOI: dx.doi.org/10.22624/AIMS/MATHS/V9N1P3
CrossREF Member Listing - https://www.crossref.org/06members/50go-live.html

Intelligent Swarming Networks(iSWAN) for Top-k Query Optimization in Real Time Sensor Databases

Enyindah, P., Onyejegbu, L.N. PhD & Onuodu, F.E. PhD
Department of Computer Science
University of Port Harcourt,
Port Harcourt, River State, Nigeria
E-mails: promise.enyindah@uniport.edu.ng; leaticia.onyejegbu@uniport.edu.ng; g.onuodu@gmail.com
Phone: +2349067011341

ABSTRACT

Real time query processing in distributed databases for sensor networks is an emerging area of research particularly in the area of factory automation and medical diagnostics. This research paper proposes a geno-generative approach involving a co-simulation system including an evolutionary intelligence method based on particle swarms, physical computing hardware, very high-level software systems and thermal sensor devices for solving the Top-k query determination problem. The system has been successfully implemented in real time in a small test bed comprising of physical and non-physical devices. The system has also been compared with an existing model and the results considering the estimated and expected temperatures showed that the proposed technique is more accurate with 70% accuracy over the existing model which reported 40% accuracy.

Keywords: Geno-generative, database, sensor, swarm, Top-k

Enyindah, P., Onyejegbu, L.N. PhD & Onuodu, F.E. PhD (2021): Intelligent Swarming Networks (iSWAN) for Top-k Query Optimization in Real Time Sensor Databases. Journal of Advances in Mathematical & Computational Science. Vol. 9, No. 1. Pp 21-38. DOI: dx.doi.org/10.22624/AIMS/MATHS/V9N1P3
Available online at www.isteams.net/mathematics-computationaljournal.

1. INTRODUCTION

Conventional database systems such as MySQL, SQLite and TinyDB are clearly defined and well standardized/structured using specific language modeling tools such as Structured Query Language, UnityJDBC etc. However, the problem of querying optimally a real time sensor database is rather a different complicated process and traditional querying techniques or database structures cannot be applied to such kinds of database processing requirements. Not that these techniques are ineffective, but that the underlying context for which they are applied to may make them inoperable or inefficient.

One such context is in the real time streaming Top-k query in distributed dynamic optimization systems. The role of a distributed query optimization systems is to map high-level queries on a distributed database (such as a set of global relations) into a sequence of database operations (of relational algebra) on relational fragments. Several important functions characterize this mapping: The calculus query must be decomposed into a sequence of relational operations called an algebraic query. The data accessed by the query must be localized so that the operations on relations are translated to bear on local data (fragments). The algebraic query on fragments must be extended with communication operations and optimized with respect to a cost function to be minimized.

This cost function refers to computing resources such as disk input/output (I/O), central processing unit (CPUs), and communication networks. Dynamic query optimization requires statistics in order to choose the operation that has to be done first. Static query optimization requires statistics to estimate the size of intermediate relations. The accuracy of the statistics can be improved by periodical updating. Most industrial environment systems use centralized decision approach, in which a single site generates the strategy. However, the decision process could be distributed among various sites participating in the elaboration of the best strategy. The centralized approach is simpler but requires the knowledge of the complete distributed database where as the distributed approach requires only local information. Hybrid approach is better where the major decisions are taken at one particular site and other decisions are taken locally. In fact, the problem of dynamic query processing can itself be decomposed into several subprograms, corresponding to various layers of precision and optimizing in these layers represent several degrees of possible (candidate solutions).

The concept of co-simulation modeling approach following the objected oriented paradigm and design patterns based on a geno-generative solution is presented for query processing in real time distributed systems.

The aim of this work is to develop an Intelligent Swarming Network (iSWAN) Geno-Generative Model for an enhanced Top-k query processing in a distributed database system.
The specific objectives of the study are to:

i. Design an Optimal Query Model (OQM) to parse efficient queries on real time data in a distributed co-simulation environment using Intelligent Swarming Network Intelligent (iSWAN) algorithms.
ii. Compare the result obtained from the proposed iSWAN geno-generative system with an existing system model.

2. REVIEW OF RELATED LITERATURE

This section reviews related works and relevant concepts required to gain understanding into the proposed systems approach. The theme is centered on developing more proactive and responsive dynamic systems that are mostly optimal in the real time query processing sense.

2.1 Related Works

Sanjay et al (2011) examined the issues in replicated data for distributed real-time database systems. The authors provided an overview to compare replication techniques available for these database systems. Data consistency and scalability are the issues that were considered in their work. Those issues are maintaining consistency between the actual state of the real-time object of the external environment and its images as reflected by all its replicas distributed over multiple nodes. Furthermore, the authors discussed a frame to create a replicated real-time database and preserve all timing constrains.

In order to enlarge the idea for modeling a large-scale database, they presented a general outline that consider improving the Data consistency and scalability by using an accessible algorithm applied on the both databases, with the goal to lower the degree of replication enables segments to have individual degrees of replication with the purpose of avoiding extreme resource usage, which all together contribute in solving the scalability problem for Distributed Real Time Database Systems. The authors did a good job. However, no hardware device for the capture of real-time unstructured datasets was implemented in order to address the mentioned issues.

Deepali (2013), proposed a dynamic and randomized query optimization algorithm to improve optimality of access plans. In the study, many different dynamic and randomized query algorithms that compute approximate solutions for producing optimal access plan was studied. The goal of database performance tuning is to minimize the response time of your queries and to make the best use of your system's resources by minimizing network traffic, disk I/O, and CPU time. Query processing and optimization is a fundamental, if not critical, part of any DBMS. Queries, in a high level and declarative language e.g. SQL, which require several algebraic operations, could have several alternative compositions and ordering. The author did a good work. However, performance evaluation of their technique showed the absence of a deep generative model on their optimized query technique.

Kakad et al (2013) developed a system for improved query processing in distributed databases involving the use of an arbitrary Top-k Query (TkQ) max-value filtering technique. In their proposed system the child-parent (client-server) topology was used for the processing and coordination of Top-k queries respectively. They used the median-filter paradigm as a basis for max query value identification and control suitable for emergencies or critical infrastructures in industrial setting. They compared their proposed solution to that without filtering and showed substantial improvements in Top-k query value handling.

Parul et al (2014), researched on an overview of distributed databases. The authors presented an overview of Distributed Database System along with their advantages and disadvantages. In addition, they also provided various aspects like replication, fragmentation and various problems that can be faced in distributed database systems. However, they failed to practically implement the discussed issue to a model and a specific distributed database. Anjali et al (2015), researched on Distributed System and its role in Health Care System. The study aimed at giving an overview in this area, evaluating the current status of field and envisioning possible future trends in this field. Distributed System today arises dynamically in terms of new applications, hardware and network components, users, workload changes and in various research applications. A distributed dataset is having its importance to provide the data from various sources. The authors could not implement the discussed issues to a model and further deploy it to a real-life health care scenario. Osegi et al (2015) proposed the GOptimaEMbed which is a smart sms based SQL-like querying system for real plant databases. The proposed system involved the use of embedded system including electronics hardware and software simulator.

Seyedali (2015) presented a new meta-heuristic called Grey Wolf Optimizer (GWO) which is evolved from the hunting behavior of grey wolves. There are mainly four types of grey wolves namely alpha, beta, delta, and omega. The topmost wolf is the alpha wolf in hierarchy. The author mainly implemented steps of hunting, searching, encircling and attacking prey. The results were obtained and then compared with Particle Swarm Optimization (PSO), Gravitational Search Algorithm (GSA), Differential Evolution (DE), Evolutionary Programming (EP), and Evolution Strategy (ES). By comparing the results of all algorithms, it was concluded that grey wolf algorithm gives best performance of all techniques, but the drawback with this algorithm is that it takes a lot of time in query processing.

2.2 Intelligent Swarming Networks

Intelligent Swarming Networks (iSWANs) present an alternative generative model solution that mimics the functions of group behavior found in many natural life systems e.g. social insects and animals. The idea behind the iSWAN approach is to use the principle of evolution basing on exploitation and exploration to model the solution. In this thesis, the iSWAN approach based on Particle Swarm Optimizer (PSO) is described and applied in finding an optimal Geno-Generative Query processing solution.

2.2.1 Particle Swarm Optimizer (PSO)

Particle swarms are a class of optimizer agents that evolve using natural behavior of group exploitation and exploration to find solutions to common problems. PSO is a very useful alternative strategy for solving optimization problems. Idea of swarm particles was first introduced by Kennedy and Eberhart (Kennedy & Eberhart, 1995). It involves the determination of a particle's new solution vector (new velocity state) from a population of particles in a well-defined random manner and after passing through a number of generative steps (generations or evolutions). The randomization step is typically influenced by a generative updates of a weighted velocity vector in conjunction with randomized position states – the states being a sum of the difference between random previous best position states with previous (initial) states at both local and global levels (Sumathi & Paneerselvam, 2010). Typically, the operations that follow are Newtonian and may be described by a velocity update calculation as in (2.2):

$$\begin{aligned} vel_{ij}(new) &= w * vel_{ij}(old) + c_1 rand_1\left(pbest_{ij}(old)\right) \\ &- pos_{ij}(old) + c_2 rand_2\left(pbest_{ij}(old)\right) - pos_{ij}(old) \end{aligned} \tag{2.1}$$

New positions are updated by adding the velocity updates obtained in (2.1) to its old position as in (2.2):

$$pos_{ij}(new) = pos_{ij}(old) + vel_{ij}(new) \tag{2.2}$$

where,

$rand_1$, $rand_2$ = random number between 0 and 1

w = inertia weight

c_1 = coefficient of self-recognition

c_2 = social coefficient

c_1, c_2 = 2.

2.3 Generative Query Processor

A typical Generative Query Processor Network contains the following key stages:

The Probability Distribution Synthesis (or Data Generating) Stage:

This stage includes:

i. A Sensor Feed to acquire real world data
ii. A data generating model to synthesize the learning probabilities
iii. An optimizer such as Genetic Algorithms (GA).

The data generating distribution is implemented as a core probabilistic generative program in software for synthesizing the hidden weights of the acquired sensor data while the GA ensures that certain probabilistic fitness criterion is met.

Inference Stage
The inference stage will use the output probabilities to make decisions on the most likely variables or parameters that influence the data generating stage. This will revolve around:
i. A Bayesian belief network and
ii. A frequency Synthesizer

The Bayesian inference detector will discover novel patterns based on maximum likelihood estimators while the frequency synthesizer will operate based on techniques similar to apriority (Aggrawal, 1998).

Query Stage
This will include a query schema that is well suited for sensor-based application. Standard SQL-like statements will be incorporated with modifications to allow for temporal and approximate probabilistic parameter tuning.
Generally, there are four core operations

i. **Generate the Distribution:** This is done using a pure probabilistic approach. We obtain the mean, variance and mode and randomly generate the training data while genetically comparing against a fitness function.
ii**.** **Form the Query:** Form the query using SQL-like joins and aggregators. More query specific requirements will include time and noise level specification.
iii. **Run the Inference Generator:** This stage is almost automatic requiring that the user explicitly set only a few query parameters. It uses a deep Bayesian Generative Network to learn to classify the hidden weights from the
iv. **Run the Interpreter:** This stage, where necessary, performs a translation from probabilistic world to real world. Though not essential, it gives the user a clear observation of the inference mechanism online. Deep-Generative models apart from being an analytic tool for advanced database research can be applied as a primary information retrieval agent for a vast range of domains including but not limited to health care, traffic monitoring, production workstations, beverages and mining industries, marine, Power Analytics etc. Geno-generative Query Processor models can serve as intermediary systems for a wide range of classical database and mining system; this include but not limited to the Smart and Distributed Network Architectures, Transaction Databases, OLAP Architectures.

2.4 Distributed Query Processing
In this research, we consider a network of interconnected computers. Each computer, known as a node in the network, contains a distributed database management system (DDBMS) and a possibly redundant portion of the database. Data are usually logically viewed in the relational data model; the unit of data distribution is a relation. The DDBMS will maintain system directories so that each query will receive a non-redundant consistent mapping of its required data. Data transmission in the network is via communication links. The data transmission cost between any two nodes is defined as a linear function as:

$$C(X) = c_o + cX \tag{2.3}$$

Where X is the amount of data transmitted. Cost measures may be defined in units of time. The constant co represents an initial start-up time for each separate transmission. An important property of *C(X)* is that if *X* < *Y*, then *C(X)* < *C(Y)*. It is further assumed that the cost of data transmissions between nodes is significantly greater than the cost of local processing and intra-nodal data transfers between local storage devices.

3. MATERIALS AND METHODS

This section presents the proposed approach and how it is developed. It also includes a brief presentation of the comparative existing model used in the simulations.

3.1 Proposed System

The proposed system uses a generative approach to evolve candidate solutions to the Top-k query problem. This problem entails the aspect of finding which sensor node query produces the peak (max) response value at a given instance of query execution time. The following sub-sections details the proposed system concept including the architectural design (see Figure 3.2) the description of the modules that make up the system and the objective function formulation.

Architectural Design

The proposed system expands on the existing system by performing optimization of the Top-k query values as in Figure 1. This is done by carrying out swarming particle optimization and this leads to a more precise and optimal value.

3.2 Modules Description

The primary modules with the exception of Sensor child nodes modules are as developed in Kakad et al (2013). The Sensor child nodes module includes an optimization sub-module that performs optimal query filtering in a real time distributed data processing system. This module is further described in this section.

3.2.1 Sink Query Module

This module requires the development of the Query-Order formulation (QOF) program which accounts for the Query-Order (QO), the QO Representation (QOR) and the QO Encoding (QOE).

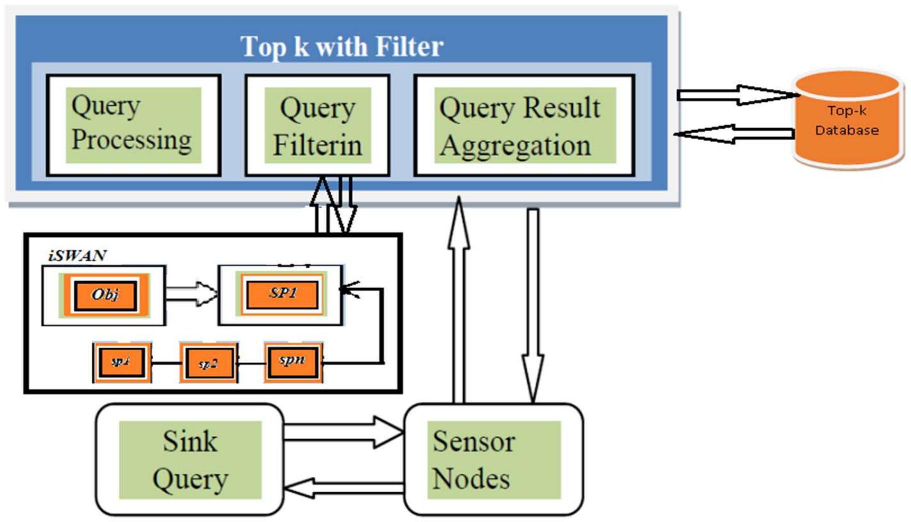

Figure.1. Proposed System Architecture

The emphasis of the QOF is to allow for structured distributed query processing in real time databases. Thus, with the QOF, the aspect of data aggregation, representation and encoding is fully captured in a dynamic context.
The QO, QOR and QOE are described in the following paragraphs:

Query-Order (QO):
In real time big data transmission systems, the query order (QO) is typically a function of the sensor transmission rates in a distributed network of acquisitional sensors (SAN). It defines the number of sensors (or sensor points) that must be activated in addition to some specified transmission rates. Thus sensors transmit data packets to the base station based on the QO in Figure 2. For this particular modeling, we follow the scheme adopted in (Sarode & Nandhini, 2018). In the Figure 2, the labels S1 to SN are sensor nodes, the dotted arrows are base station queries and the boldened arrows the responses generated from the sensors at uncertain orders.

Query-Order Representation (QOR):
The QO representation defines the following key structures (Sarode & Nandhini, 2018):
1) The number of sensors using a particular transmission channel
2) The data transmission capacity or rate

The overall representation is based on the eqns(3.1 - 3.2):

$$S_i = \lfloor z_l \rfloor, \quad 0 \le l \le N_L - 1, \quad S_i \in S \tag{3.1}$$

$$t_{ij} = \| z_l - S_i \| \tag{3.2}$$

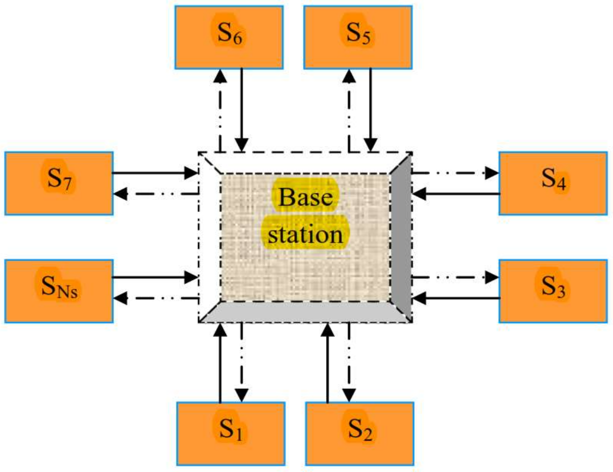

Figure.2. Architectural view of a typical data aggrgation model formed from the QO
(Source: Sarode & Nandhini, 2018).

where,
$\lfloor z_l \rfloor$ = queried sensor label
N_L =number of possible sensor states
L = integer index label, at state *N*
i = integer index label, at state *S*

Query-Order Encoding
In order to aggregate sensor queried data from two or more channels, an encoding scheme is needed. In this thesis, the encoding scheme is attained by forming sensor solution variables using the scheme in Figure 3.3. This scheme assumes a representative variable S_it_j, and constrained by $t_j \geq t_{ij}$,

where:
S_it_j = a solution variable representing *ith* sensor transmitting *j* amounts of data packets
t_j = the amount of data packets transmitted by Sensor *i* at time *j* for a given channel
t_{ij} = the amount of data packets transmitted by Sensor *i* from time *i* to *j* for a given channel

In (Sarode & Nandhini, 2018), the channel is typically defined as 1/4th of its capacity but need not be strictly so in practice – as in reality, the capacities will vary sparsely due to environmental constraints, thresholding etc.

Thus, we propose to describe the channel capacity by a sparsity criterion rule as follows in Algorithm 1.

```
Algorithm 1: Query-Order Encoding
Step 1: Initialize Sensor $S$, Sparse Sensor $S^*$, instance $i$, threshold, $th$, Vj
Step 2: Normalize S: S ← {0, 1}
Step 3: For each Sensor, at instance $i$
Step 4:  If $S_{(i)} > th$
                $S^*_{(i)}$ = Si
         End
Step 5: Obtain the cardinality in S: $n(S) \leftarrow S_{count}$
Step 6: Set Capacity as $C_i$: $S_{count} \leftarrow C_i$
s.t.:
```

$$\sum_{j=1,\, i=1}^{C_j,S_i} V_j^{(i)} > T_i \quad \text{//Channel capacity data availability constraint}$$

where,
T_i = the data to be transmitted to the base station by *ith* sensor
V_j = the data to be transmitted by the *ith* sensor through *jth* channel

3.2.2 Sensor Nodes module

This is the child modules that represent real world physical parameter black boxes that can support query retrieval, response value generation and data parsing. In this node, the following query functions are specifically\accomplished:

1. Child node(s) compute the median value of the top-k query response readings.
2. The computed median value readings of Child node(s) are captured by iSWAN agents.
3. Agent node(s) receives median values of the top-k query readings from all child nodes by swarming particles
    a. Agent node(s) perform adaptive filtering by computing the optimized median index of the received median values using group behavior and an objective (cost) function.
    b. The iSWAN system reports the generated optimal (fitted) median of each child node.

3.2.3 iSWAN module

This represents the intelligent swarming network used in the solution of the Top-k response filter values. The sub-modules *sp1, sp2 … spn* represent swarming agents in the solution search space that scans the sensor nodes via the Top-k with filter module interface. The main optimal solutions are carried basing on the model equations (2.1) and (2.2) with reference to Section 2.2.

The solution points are obtained using the swarming particle optimizer algorithm as shown in Algorithm 2 (Sumathi & Paneerselvam, 2010).

Algorithm 2: iSWAN Main solution Process

```
Step 1: Initialize the size of the swarm particles, n
Step 2: Randomly initialize swarm particle position, x and velocities, v
        While stopping criterion is false do
Step 3:         t = t+1          //Increment iteration counter
Step 4:         Compute initial fitness value of each particle
                $x* = \arg\min_{t-1}^{n}\begin{pmatrix} f(x*(t-1)), f(x_1(t)), \\ f(x_2(t)), \ldots f(x_t(t)), \ldots f(x_n(t)) \end{pmatrix};$
                for i=1 to n
Step 5:             $x_t^{\#}(t) = \arg\min_{t-1}^{n}\left(f\left(x_t^{\#}(t-1)\right), f(x_t(t))\right)$
                    for j = 1 to Dimension
Step 6:                 Update the jth dimension of $x_t$ and $v_t$//$x_t$ = pos, $v_t$ = vel
                            Execute (2.1)
                            Execute(2.2)
                    End for
                End for
        End While
```

3.2.4 Top-k Query Filter Module

This represents the topmost structure of the system and serves as the container module that accommodates the rest processing functions or modules. Thus, it represents the main functional class or module in the OOP paradigm. The following key sub-functions are performed in this module: Query Processing (QP), Query Filtering (QF) and Query Result Aggregation (QRA). In QP, the queries (sensor data calls) are first initiated by generating a subset of QF and QRA commands.

The QF commands follows from the iSWAN and works in tandem with the QRA for each sensor data call. Specifically, the QF sub-module does the job of extracting the queries with the top or maximum reading considering the various sensor nodes. On the other hand, the QRA sub-module performs the specific function of sensor data aggregation of all queries in a streaming data matrix or store-of-value. This is achieved by using relevant function data calls in near real time.

3.3 Objective Function Formulation

The objective function plays a critical role in the design of any optimization system. It defines the fitness or cost criterion that needs to be minimized in order to attain the required solution value (points or positions). In this research study, the Median Error Difference (MED) of the pre-computed medians between the child nodes and their combined iSWAN agent median state is considered as objective function. This difference is computed as follows in Algorithm 3:

Algorithm 3: iSWAN Filter Query Objective Function
Step 1: Initialize Query Optimization Parameters – setting the lower and upper bounds, population size and number of generations
Step 2: Compute Median Error Difference (MED) Calculation:

$$\eta = sum\left(\kappa_{agent(ref)} - \kappa^{i}_{child}\right) \qquad (3.3)$$

where,

$\kappa_{agent(ref)}$ = agent-based median used as reference

κ_{child} = set of child median values at time step *i*

We determine if a data transmission is successfully and correctly (maximally) terminated in a sensor node using eqn(3.4) as:

$$T_1 = \min(\eta) \qquad (3.4)$$

T_1 = a fitness criterion (parameter) for top-k query data transmission validation i.e. if the absolute difference between received child nodes data transmission and a reference parent node standard meets the expected least minimum levels, it is selected; otherwise it is not selected. This is done in an evolutionary manner based on swarming particles.

4. RESULTS AND DISCUSSIONS

In this research, two-stage approach is used in conducting a co-simulation of experiments with a laboratory-type testbed. In the first stage (sub-section 4.1), the query response of the testbed is evaluated on the basis of the existing system model described earlier in Section 3. The second stage employs the proposed iSWAN technique for evaluating the testbed (see sub-sections 4.2 and 4.3). In particular, the second stage describes a dual-tier boundary constraint for each considered sensor at static and dynamic lower bound constraints while comparative results and discussions are presented in sub-sections 4.4 and 4.5 respectively.

4.1. Experiments using the existing system model (Top-k Query Processor)

This experiment applies the Top-k query approach based on median filtering. The results (Top-k query temperature values) using testbed based on the existing model are as shown in Tables 1 to 3 for the child sensor nodes 1, 2 and 3 respectively. The table shows the filtered response locator id (loc_id) and the corresponding Top-k query values (Tkv). The loc_id represents the positions of the Tkv in the child node query list.

Table.1. Top-k Query Response for child sensor node 1

loc_id	Tkv (oC)
1	27
7	32
19	23
25	26
28	38
34	20
49	36

Table.2. Top-k Query Response for child sensor node 2

loc_id	Tkv (oC)
2	28
8	32
17	20
20	23
26	26
29	37
32	20
35	21
50	36

Table.3. Top-k Query Response for child sensor node 3

loc_id	Tkv (oC)
3	24
9	35
18	21
21	23
24	20
27	29
30	36
33	21

4.2. Experiments using the proposed system model with Static Lower Bound Constraints

The results using testbed for the proposed system at Static Lower Bound Constraints (SLB) is as shown in Table 4 for 20 trial runs. The results show the fitted Top-k query temperature values for the child sensor nodes 1, 2 and 3 respectively; it also shows the fitness function value computed by the Particle Swarm Optimizer (PSO) search agents at the attained temperature states.

Table.4. Top-k Query Response for proposed system with SLB

Trial No.	Cost	Sensor 1 (°C)	Sensor 2 (°C)	Sensor 3 (°C)
1	25.0000	25.0000	0.0000	25.0000
2	20.9995	0.0000	21.0005	21.0000
3	18.9995	0.0000	19.0005	19.0000
4	15.4890	15.5072	0.0038	15.5000
5	23.0000	23.0000	23.0000	0.0000
6	19.0000	19.0000	0.0000	19.0000
7	19.9990	20.0000	20.0006	0.0005
8	19.9958	0.0000	20.0042	20.0000
9	19.9994	0.0000	19.9997	20.0000
10	23.0000	23.0000	23.0000	0.0000
11	32.9985	0.0000	33.0000	32.9992
12	38.9771	0.0000	39.0229	39.0000
13	21.0000	21.0000	0.0000	21.0000
14	27.9998	0.0000	27.9999	28.0000
15	16.0000	16.0000	0.0000	16.0000
16	16.9986	17.0014	17.0000	0.0000
17	22.4986	22.5014	22.5000	0.0000
18	41.9784	41.9892	0.0000	42.0000
19	25.9998	0.0000	25.9999	26.0000
20	23.9993	0.0000	24.0007	24.0000

4.3. Experiments using the proposed system model with Dynamic Lower Bound Constraints

Here simulations were performed in order to investigate the nature of using a dynamic lower bound (DLB) in the iSWAN optimization process. In this regard, the *min* function was used adaptively to extract the lower bounds prior to processing. The results of this experiment for 20 trial runs are as shown in Table 5; also shown in Figure 3 is the comparative graphical query response of each sensor node as obtained by the simulation program.

Table.5. Top-k Query Response for proposed system with DLB

Trial No.	Cost	Sensor 1	Sensor 2	Sensor 3
1	1.5000	18.0000	16.5000	18.0000
2	1.0000	25.0000	27.0000	22.0000
3	0.5000	27.0000	26.0000	24.5000
4	0.5026	0.9068	0.1603	0.7849
5	1.5000	15.5000	13.0000	15.0000
6	0.4015	0.8020	0.7046	0.2057
7	1.0000	19.0000	19.0000	18.0000
8	1.0000	30.0000	29.0000	27.0000
9	0.5000	17.5000	15.0000	16.5000
10	0.7509	0.8268	0.8462	0.0565
11	2.5000	25.0000	23.0000	18.5000
12	0.5573	0.8092	0.8871	0.1741
13	0.5250	0.2893	0.8590	0.9038
14	3.0000	17.0000	17.0000	14.0000
15	1.0000	33.0000	40.5000	47.0000
16	0.5000	45.5000	48.0000	42.5000
17	0.7189	0.7944	0.8685	0.0014
18	0.5300	0.8842	0.9118	0.3267
19	2.5000	19.0000	22.0000	22.5000
20	0.6105	0.9395	0.1728	0.8614

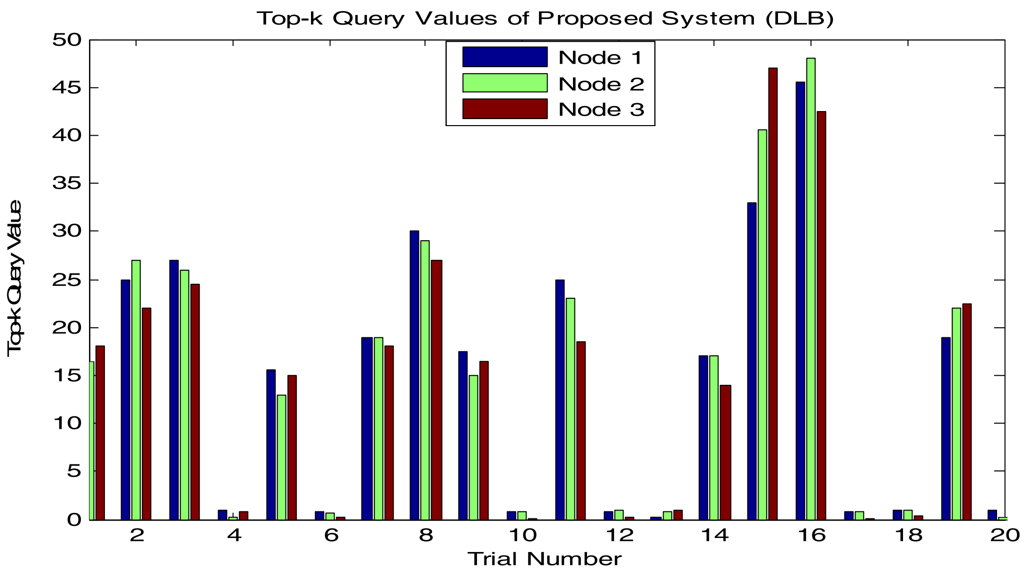

Figure.3. Response of the iSWAN system under dynamic lower bound

4.4. Comparative Experiments

The comparative simulation results for each considered sensor data sources are as shown in Tables 6 to 8 for the Existing vs. Proposed System using the Dynamic Lower Bound (DLB) concept based on the maximum temperature value (Tmax). For the existing model, since multiple values are predicted, we select the first filtered value as the single best solution variable. The classification accuracies (CA) are reported in Table 9.

Table.6. Comparative Top-k Query Response for proposed vs. existing systems Sensor 1

Trial No.	Tmax(°C)_existing	Tmax(°C)_proposed	Tmax(°C)_expected
1	29.0000	30.0000	30.0000
2	17.0000	34.0000	34.0000
3	29.0000	34.0000	34.0000
4	15.0000	34.0000	34.0000
5	36.0000	37.0000	37.0000
6	39.0000	0.7501	37.0000
7	39.0000	39.0000	39.0000
8	36.0000	33.0000	35.0000
9	28.0000	0.0620	36.0000
10	29.0000	40.0000	40.0000

Table.7. Comparative Top-k Query Response for proposed vs. existing systems Sensor 2

Trial No.	Tmax(°C)_existing	Tmax(°C)_proposed	Tmax(°C)_expected
1	32.0000	34.0000	34.0000
2	16.0000	37.0000	37.0000
3	30.0000	37.0000	37.0000
4	38.0000	38.0000	38.0000
5	35.0000	35.0000	35.0000
6	38.0000	0.9047	35.0000
7	36.0000	36.0000	36.0000
8	36.0000	33.0000	35.0000
9	27.0000	0.5575	33.0000
10	42.0000	42.0000	42.0000

Table.8. Comparative Top-k Query Response for proposed vs. existing systems Sensor 3

Trial No.	Tmax(°C)_existing	Tmax(°C)_proposed	Tmax(°C)_expected
1	32.0000	35.0000	35.0000
2	38.0000	38.0000	38.0000
3	31.0000	38.0000	38.0000
4	39.0000	39.0000	39.0000
5	37.0000	37.0000	37.0000
6	26.0000	0.2062	37.0000
7	17.0000	38.0000	38.0000
8	36.0000	33.0000	35.0000
9	29.0000	0.6510	35.0000
10	43.0000	43.0000	43.0000

Table.9. Classification accuracies (CA) of proposed versus existing at different sensor nodes

Sensor Node	CA(exist) %	CA(proposed) %
1	40	70
2	40	70
3	40	70
Average CA	40	**70**

4.5. Discussions of results

The results presented in the aforementioned sub-sections (sub-sections 4.1 – 4.4) are explained succinctly in this subsection under the headings: Experimental discussions of the existing system, Experimental discussions of the proposed system with SLB and DLB and the Comparative discussions.

4.5.1. Experimental discussions of the existing system

The existing system results show some variability in top-k queried temperature values as shown in Table 1-3. For Sensor Node 1 (Table 1), the Top-k value (TkV) is 38^{o}C occurring at location id (loc_id) 28 while for Sensor Nodes 2 and 3, the TkV and corresponding loc_id are 37^{o}C:29 and 36^{o}C:30 respectively. Thus, it is immediately obvious that Sensor Node 1 is the critical one followed by Sensor Nodes 2 and 3. Thus, Sensor Node 1 must be given priority with respect to the other nodes.

4.5.2. Experimental discussions of the proposed system with SLB and DLB

From the simulation results using Geno-Generative iSWAN optimizer with SLB, the Sensor Node 1 TkV is 41.9892^{o}C occurring at loc_id 18; also the least cost (cost) is obtainable at this solution point and is -41.9784. For the Sensor Nodes 2 and 3 Tkv, loc_id and cost are 39.0229:12: -38.9771 and 42.0000: 18: -41.9784 respectively. From these results, it can be clearly seen that Sensor Node 1 is the critical one; this is followed by Sensor Node 3 and then Sensor Node 2. Priority for conditioning should be given following the aforementioned order. This can also be clearly visualized in Figure 6 while the top-most bars are shared between Sensor nodes 1 and 3 only.

From the simulation results using Geno-Generative iSWAN optimizer with DLB, the Sensor Node 1 TkV is 45.5000^{o}C occurring at loc_id 16; also the corresponding least cost (cost) obtainable at this solution point is -0.5000. For the Sensor Nodes 2 and 3 Tkv, loc_id and cost are 48.0000:16: -0.5000 and 47.0000: 15: -1.0000 respectively. From these results, it can be clearly seen that Sensor Node 2 is the critical one; this is followed by Sensor Node 3 and then Sensor Node 1. In particular, there are sub-optimality issues in the DLB as Sensor node 3 gave best cost as against that of Sensor nodes 1 and 2. Thus, priority for conditioning should be given to Sensor Node 2 followed by Sensor Node 3 and then Sensor Node 1. This can be clearly visualized in Figure 7 with the top-most bars shared between Sensor nodes 2 and 3 in that order.

4.5.3. Comparative discussions

The proposed Geno-Generative iSWAN optimizer system is compared with the median based child-parent (client-server) approach considering the expectation maximum (top-k) queried temperature values and at different sensor nodes following the convention utilized in sub-section 4.3. For the Sensor Node 1, apart from the Trials 6, 8 and 9, the Tmax for proposed system is equivalent to the expectation; however, in the case of the existing approach, an exact match occurred only at the 7th trial run. In particular, it can be observed that the proposed system suffered a great deviation from the expectation at the 6th and 9th trials; this great deviation may be attributed to the stalling effect that may occur in the method of swarming particles.

For the Sensor Node 2, apart from the Trials 6, 8 and 9, the T_{max} for proposed system is equivalent to the expectation. In the case of the existing approach, an exact match occurred at the 4th, 5th, 7th and 10th trial runs. Just as in Sensor Node 1, the proposed system suffered a great deviation from the expectation at the 6th and 9th trials. Thus, the competitive nature of the median based child-parent (client-server) approach is evident. For the Sensor Node 3, apart from the Trials 6, 8 and 9, the Tmax for proposed system is equivalent to the expectation. In the case of the existing approach, an exact match occurred at the 2nd, 4th, 5th, and 10th trial runs.

Just as in Sensor Node 1, the proposed system suffered a great deviation from the expectation at the 6th and 9th trials. Thus, the competitive nature of the median based child-parent (client-server) approach is also evident. In Figure 8 to 10 is shown graphically the situation of great deviations at sensor nodes 1-3 respectively query processing between existing, proposed and the expected query response values. As can be seen from the Table 9, the proposed system on the average outperformed the existing model with a CA of 70% over that of the existing which returned a CA of 40% on the average.

5. CONCLUSIONS AND FUTURE WORK

Query processing and optimization is a major function of databases and information systems in large industrial environments. Query processing in distributed systems are a very important problem in statistics and optimization theory. In this research, a new approach towards the query response determination in real time databases has been proposed in this study. The system have been implemented based on a co-simulation systems approach which incorporates a novel intelligence based swarming technique coined iSWAN including both the theory of evolving particles, generative processing and query processing for the Top-k query optimization.

From the simulation results, it have been shown that the proposed system outperformed the existing with a 70% accuracy over the 40% of the existing model. This goes to show that generative optimization models are more suitable for dynamic distributed systems than the existing ones. Thus, future work should be geared towards improving existing infrastructures to support generative modeling solutions.

REFERENCES

1. Aggrawal D. (1998) Bayesian inference detector discover novel patterns based on maximum likelihood estimator. Systems, *Journal of Software Computing (JSC), 3(9), 16 - 43*
2. Anjali S., and P.K Yadav (2015), Distributed System and its role in Health Care System, *International Journal of Computer Science and Mobile Computing (IJCSMC), 4(4), 302 – 308.*
3. Deepali A. (2013), Dynamic and Randomized Query Optimization Algorithms to Improve Optimality of Access Plans, *International Journal of Scientific and Engineering Research*.
4. Kakad, S., Sarode, P., & Bakal, J. W. (2013). Analysis and Implementation of Top K Query Response Time Optimization Approach for Reliable Data Communication in Wireless Sensor Network. *IJEIT*, 3.
5. Eberhart, R., & Kennedy, J. (1995, November). Particle swarm optimization. In *Proceedings of the IEEE international conference on neural networks* (Vol. 4, pp. 1942-1948).
6. Osegi, N. E., & Enyindah, P. (2015). GOEmbed: A Smart SMS-SQL Database Management System for Low-Cost Microcontrollers. *African Journal of Computing & ICT*, *8*(2), 133-144.
7. Parul T. and I. Megha (2014), An Overview of Distributed Databases, *International Journal of Information and Computation Technology, 4(2), 207 – 214.*
8. K., K.Sharma, S.Vishnu (2011), Issues in Replicated Data for Distributed Real-time Database Systems, *International Journal of Computer Science and Information Technologies (IJCSIT), 2(4), 1364 – 1371.*
9. Sarode, P., & Nandhini, R. (2018). Intelligent query-based data aggregation model and optimized query ordering for efficient wireless sensor network. *Wireless Personal Communications*, *100*(4), 1405-1425.
10. Seyedali Mirjalili (2015) presented a new meta-heuristic called Grey Wolf Optimizer (GWO) which is evolved from the hunting behavior of grey wolves. *International Journal of Machine Learning and Computing, 6(5), 372 – 576*.
11. Sumathi, S., & Paneerselvam, S. (2010). *Computational intelligence paradigms: theory & applications using MATLAB*. CRC Press.