Evolutionary Optimization for Prioritized Materialized View Selection:
An Exploratory Analysis

Heena Madaan, Guru Gobind Singh Indraprastha University, India*
Anjana Gosain, Guru Gobind Singh Indraprastha University, India

ABSTRACT

Selecting appropriate views that provide faster query response time is a critical decision in data warehouse design. Top-level users expect quick results from a data warehouse for faster decision-making to gain a competitive edge in business. Prioritizing a view can distinguish views required to answer top-level users’ queries from regular users and provide a better selection chance. The prioritized materialized view selection (PMVS) problem addresses how to utilize the given space to materialize prioritized views more relevant to users. Particle swarm optimization algorithm has been used to achieve minimized query processing costs. Evolutionary algorithms are widely known to solve complex optimization problems quickly by reaching a semi-optimal solution. This paper explores the performance of six evolutionary algorithms: particle swarm optimization, coral reef optimization, cuckoo search, ant colony optimization, grey wolf optimization, and artificial bee colony. The results of empirical and statistical analysis show that PSO, CRO, and GWO algorithms are best suited to solve PMVS.

KEYWORDS

Ant Colony Optimization, Artificial Bee Colony, Coral Reef Optimization, Cube Selection, Cuckoo Search, Grey Wolf Optimization, Particle Swarm Optimization, Query Priority, View Priority

INTRODUCTION

Materializing views is a widely practiced strategy to improve online analytical processing (OLAP) query performance significantly. Since storing all possible views is not a cost-effective approach, the materialized view selection problem (MVS) focuses on selecting an optimal set of views to materialize. Such views help attain reasonable query processing time within adequate storage space (Harinarayan et al., 1996; Lin & Kuo, 2004).

Making the right information available at the right time to the right user is the critical need of the hour. All of the existing materialized view selection approaches treat each of the queries and views...
equally. However, in the real world, users have varying importance in an organization and have different query performance expectations (Gosain & Madaan, 2018a; Kimball & Caserta, 2004; Sauter, 2014). Thus, queries hold priority values depending on the users. The prioritized materialized view selection problem (PMVS) (Gosain & Madaan, 2018b) leverages this concept to prioritize cubes based on the query priority and provides a higher chance of selecting the cubes/views required by the top users. Furthermore, the authors proposed a priority and frequency-based cost model (PFBCM) by adding view priority into the fitness function for optimization. It has been proven to select a more suitable set of views within the specified space constraints and increase user satisfaction by achieving better query performance.

The search space for the view selection problem increases exponentially with the dimensionality, making it an NP-hard problem (Harinarayan et al., 1996; Lin & Kuo, 2004). In such cases, deterministic methods exhaust the acceptable time limits for searching, whereas evolutionary algorithms, although they may not find an optimal solution, arrive quickly at a near-optimal solution. Evolutionary optimization algorithms are stochastic search algorithms on a random population that follow some species’ natural evolution or social behavior for an optimal solution search (Sanz et al., 2014, Mirjalili et al., 2014, Karaboga & Basturk, 2007). Evolutionary algorithms have been used aggressively to find optimal results in various domains. Many of them, such as the genetic algorithm (GA), particle swarm optimization (PSO), ant colony optimization (ACO), artificial bee colony (ABC) optimization, and others have been explored to obtain an optimal materialized view set (Arun & Kumar, 2017; Gosain & Heena, 2016; Lin & Kuo, 2004; Loureiro & Belo, 2006; Song & Gao, 2010).

The particle swarm optimization (PSO) algorithm has been utilized to solve PMVS and has produced satisfactory results (Gosain & Madaan, 2018b). Recently, naive evolutionary algorithms such as coral reef optimization (CRO) (Sanz et al., 2014), grey wolf optimizer (GWO) (Mirjalili et al., 2014), artificial bee colony (ABC) (Karaboga & Basturk, 2007), ant colony optimization (ACO) (Dorigo et al., 2006), and cuckoo search (CS) (Yang & Deb, 2009) are rapidly adopted in diverse areas for efficient optimization. These algorithms have shown better results than PSO in various problems, but an algorithm working well for one class of problems might not perform well in another. Therefore, this study paper aims to investigate different evolutionary algorithms for solving PMVS. Performance was assessed based on optimized cost objective, convergence speed, and view selection specific metrics, such as detailed cost savings ratio, percentage of views materialized for high priority queries, and query processing cost. Empirical and statistical analysis to discover algorithms best suited for the PMVS problem is presented.

The paper is structured as follows. First, it briefly provides the background of different evolutionary algorithms adopted in MVS, followed by introducing the PMVS problem and presenting pseudo-codes of the evolutionary algorithms. Next, it provides detailed experimental results and analysis and concludes the work at the end.

**BACKGROUND**

Research to solve MVS began from work (Harinarayan et al., 1996), where authors introduced the cost model and used greedy heuristics to select views by minimizing the cost model. However, greedy approaches fail to provide optimal solutions for high-dimensional datasets and big data. After that, evolutionary algorithms being robust, efficient, scalable, and capable of finding semi-optimal solutions quickly, began to be explored in choosing materialized views. For the first time, the authors in (Zhang & Yang; 1999) applied the genetic algorithm to choose views based on multiple view processing plans and explored the practical effectiveness against heuristic approaches. The authors (Lin & Kuo, 2004) used a lattice framework to design views for better suitability in the OLAP query system and implemented a Genetic algorithm for optimization. In the last decade, with the popularity of evolutionary algorithms for optimization, several works have been proposed by applying variants of evolutionary algorithms to solve MVS. Song & Gao (2010) presented the application of the ACO
algorithm to achieve a sub-optimal set of views and achieved better query response cost than GA. They used a lattice framework for constructing views and optimized the space-constrained cost function. The authors applied a simulated annealing algorithm in (Kumar & Kumar, 2012) and ABC algorithm (Arun & Kumar, 2017) with N-point random insertion operators to select top-k views in a lattice and reduce the overall query response time.

Azgomi & Sohrabi (2019) proved the coral reef optimization (CRO) efficiency in selecting materialized views in a data warehouse by increasing the coverage rate of queries compared to other methods. Work proposed by Gosain & Sachdeva (2020a) simulated a random-walk GWO algorithm in finding optimal views. They optimized the cost function within space constraints and proved the scalability and efficiency of the algorithm. In (Gosain & Sachdeva, 2020b), the authors solved MVS with the constraint handling technique using a cuckoo search (CS) algorithm for optimization and stochastic ranking for constraint handling. Some authors also utilized particle swarm optimization (PSO) (Gosain & Heena, 2016) to find a compelling set of materialized views.

Recently, Gosain & Madaan (2018a, 2018b) introduced the idea of prioritizing cubes for selection. They took advantage of query priority based on different users in an organization and leveraged it further to assign importance to views. Giving a cube priority value induces the notion of providing a varying chance to views, based on weights, competing for their candidature in materialized view selection. They designed a priority-frequency-based cost model (PFBCM), used the PSO algorithm to optimize the cost function, and proved the purpose of the priority introduction in materialized view selection. Observing such a comprehensive exploration of different evolutionary algorithms on MVS motivated the authors of this paper to inspect their performance to solve PMVS and to determine the most suitable algorithm for this problem. This study assesses PSO, CRO, CS, ACO, GWO, and ABC algorithms for solving PMVS.

**PRIORITIZED MATERIALIZED VIEW SELECTION (PMVS)**

PMVS proposed by the authors in Gosain & Madaan (2018b) is a problem to choose an optimal set of prioritized views in a data warehouse under storage space limits. Cube priority assigns an importance value to each cube/view during the selection process. The cube priority is based on the query priorities answerable by the view. Query priority is a combination of the user-assigned local priority and the system-defined global priority, based on the user level, the department level, and the query type. The authors presented a notion that top user’s queries have a higher priority value; then, a cube priority answering these queries must reflect the user expectations by obtaining a high priority value and preference for view selection.

**Cube Priority Methodology**

Each query is answered from a cube/view aggregated on the same attributes as required by the query, and one view can answer multiple queries. Therefore, all these queries demanding the aggregated view play a role in defining its priority value. Cube priority proposed by Gosain & Madaan (2018b) is calculated as the weighted sum of query priority answered by the cube (Equation 1). Each query’s weight depends on its priority interval, and weight decreases exponentially with query priority. For instance, if a cube c6 answers queries Q7 (low priority), Q10 (high priority), Q18 (high priority) having priority values of 0.02, 0.24, 0.3 and weights of 0.0625, 0.25, 0.25, respectively, its cube priority is calculated as $(0.02*0.0625 + 0.24*0.25 + 0.1*0.125) = 0.073$. Similarly, if cube c7 answers query Q3 (low priority), Q5 (low priority), Q11 (low priority) with priorities of 0.02, 0.005, 0.01 and weights of 0.0625, 0.03125, 0.0625, respectively, then c7 gets a priority value of 0.002. Thus, a cube with low priority queries has a lower impact than a cube having high priority queries.
where cube \( c_j \) answers \( p \) queries, gets its priority \( c_{p_j} \); \( q_p \) is the query priority, and \( q_w \) is the query weight.

**Cost Model: Priority-Frequency-Based Cost Model (PFBCM)**

Gosain & Madaan (2018b) designed a cost model PFBCM that integrates cube priority in query processing cost, which acts as a preference for each cube during the selection process. Their approach presented a higher chance of selecting a cube with a high priority, indirectly reflecting a top-user query requirement. The cost model comprised query processing cost and the view maintenance cost, and the optimization algorithm tried to balance the two costs. Since all cubes cannot be stored, the cost model was constrained by some space limitations, able to store only a few cubes. The authors of this study used the cost model PFBCM as their fitness function during the optimization performed by evolutionary algorithms. Each of the algorithms targets to minimize the fitness function value to achieve an optimal set of prioritized views chosen for materialization in a data warehouse.

Given a lattice framework \( L \) of \( n \) cubes \( C = \{c_1, c_2, ..., c_n\} \) with cube priority \( c_p = \{c_{p_1}, c_{p_2}, ..., c_{p_n}\} \), cube frequency \( c_f = \{c_{f_1}, c_{f_2}, ..., c_{f_n}\} \), and cube update frequency \( c_u = \{c_{u_1}, c_{u_2}, ..., c_{u_n}\} \), a materialized cube set \( M \) where \( M \subseteq C \), the objective was to minimize the cost function \( F \) comprising of query processing cost \( q(c, M) \) and view maintenance cost \( u(c, M) \) under space constraint \( S \). PFBCM is defined as follows-

\[
F(M) = \sum_{i=1}^{n} \left((\alpha * c_{p_i} + (1 - \alpha) * c_{f_i}) * q(c_i, M)\right) + \sum_{c \in M} c_{u_c} * u(c, M) \leq S
\]

where \( \alpha \) is the tradeoff factor between cube frequency and cube priority weightage chosen in the range 0 to 1.

**EVOLUTIONARY ALGORITHMS**

This section provides a brief description, pseudo-code, and parameter settings for each of the algorithms under investigation. Each algorithm received the same input comprising a lattice framework ‘\( L \)’, cube priority set ‘\( c_p \)’, cube frequency set ‘\( c_f \)’, and cube update frequency set ‘\( c_u \)’.

**PSO—Particle Swarm Optimization Algorithm**

Kennedy & Eberhart (1997) proposed the PSO algorithm that simulates bird flocking behavior to find a place to land with the maximum possibility of food and minimize predator’s risk. It begins with a randomly initialized swarm population and searches in the direction of the optimal solutions dictated by findings of swarm particles. Each particle in the swarm updates its location determined by a velocity vector adjusted based on the individual particle’s best value and the swarm’s best value found so far. The authors of this study implemented the improved version of PSO known as NBPSO, proposed by Nezamabadi et al. (2008), to overcome the original PSO premature convergence problem. It updates the velocity vector based on a sigmoid function value, and the particle position is updated if a random value is less than the sigmoid function value. Figure 1 provides the pseudo-code for the NBPSO applied for solving PMVS. PSO has the advantage of setting few parameters compared to other optimization algorithms.
CRo - Coral Reef Optimization Algorithm

CRo is a novel bio-inspired meta-heuristic algorithm (Salcedo et al., 2014) introduced in 2014 and mimics the corals and reef formation process for solving optimization problems. Corals in the reef grow and reproduce sexually and asexually, but they need to fight their reef space. New individuals might replace the weaker solutions, or weak solutions might exist if the reef has empty rooms.

We adopted the CRo approach applied by Azgomi & Sohrabi (2019) for materialized view selection. The process initializes an NxM square reef R with some squares allotted to few corals based on the free/occupied ratio (r0). After the initialization phase, reproduction and reef creation begin. It applies broadcast spawning on a fraction Fb of reef corals where each pair of corals then creates a larva through a 3-point crossover, as shown in Figure 2. Then it applies a brooding operation to perform one-bit random mutations. Finally, all the larvae fight to seize space where each larva gets a fixed number of attempts (i.e., k). The budding operation selects the top Fa fraction of corals, and these new solutions fight similarly to find a position in the reef. Lastly, it selects a fraction Fd of weak solutions for deletion to allocate more space for new larvae in the depredation phase by considering a probability parameter Pd. Figure 3 presents its pseudo-code.

Being naive, CRo is being explored in various applications and proven to solve complex optimization problems and MVS (Azgomi & Sohrabi, 2019) successfully. Therefore, the authors chose to check its efficacy for the PMVS problem.

CS - Cuckoo Search Algorithm

CS is another recent meta-heuristic population-based algorithm (Yang & Deb, 2009) inspired by cuckoo bird breeding behavior and birds’ levy flight characteristics. It has been explored in image
Figure 2. An example of three-point crossover

![Three-point crossover example](image)

Figure 3. Pseudocode for CRO algorithm (Azgomi & Sohrabi, 2019; Salcedo et al., 2014)

---

**Parameters settings:** N=10, r=10, r0=0.3, Fp=0.7, Fc=0.3, F=0, ϕ=0, k=4

**Code:**

1. **Initialization**
   - a. Create N x N reef matrix
   - b. Select r0 random cells for initial coral assignment
   - c. Assign a random bit vector of length n as X=(x1,x2,...,xn) to each cell and add X to populationList ‘pL’
   - d. Calculate fitness value of assigned coral using eq 2

2. For t iterations:
   - a. Perform Broadcast spawning operation
      - i. Select Fp corals from the reef randomly and create pairs
      - ii. For each pair,
         1. Generate 3 random numbers n1,n2,n3 from 1 to n-1
         2. Create a larva Xnew by applying 3-point crossover as displayed in Fig 2.
         3. Add larva Xnew to LarvaeList L, L = L ∪ Xnew
   - b. Perform Brooding operation
      - i. Select randomly (1-Fp) remaining coral solutions from the reef
      - ii. For each solution, choose a random bit between 1 to n and flip bit from 0 to 1 or 1 to 0, to create a larva Xnew, and add it to LarvaeList, L = L ∪ Xnew
   - c. Perform Larvae setting operation - For each Larva Xnew ∈ L,
      - i. Select a random cell c in R
      - ii. If c is empty, then set Xnew in c and pL = pL ∪ Xnew
      - Else, get fitness (X) of solution present in reef, if fitness(Xnew)>fitness(X), then
         1. Remove X from c and pl=pl-X
         2. Set Xnew in c and pl = pl ∪ Xnew
   - d. Perform Budding operation
      - i. Sort pl in descending order of fitness
      - ii. Select top Fp coral solutions from pl
      - iii. For each selected coral solution X,
         1. Replicate X to Xdup as a larva
         2. Perform larva setting operation (as in step 2c)
         3. If placed in R, pl = pl ∪ Xdup
   - e. Perform Depredation
      - i. Select Fc weakest solutions as list W5
      - ii. For each X ∈ W5,
         1. Generate a random number r
         2. If r=1, then remove X from R and pl = pl - X
         3. Pr=Pr + 0.1/t
   - f. Perform Selection -
      - i. Select Fp strongest solutions as list W3
   - g. For each X ∈ W3,
      1. Generate a random number r
      2. If r=1, then remove X from R and pl = pl - X
      3. Pr=Pr + 0.1/t

3. Return the best fitness achieved in the reef
processing, scheduling, and agriculture and proved to be efficient and robust in finding optimal solutions. The authors adopted its binary version, proposed by Gherboudj et al. (2012), wherein each solution \( X \) is transformed to its binary solution \( X' \) using a sigmoid function. For a solution \( X = (x_1,x_2,..x_n) \), each dimension’s real value is inputted into a sigmoid function. Each dimension gets a flipping chance by generating a random number \( r \) and setting the corresponding bit in \( X' \) to 1 if \( r \) is less than the sigmoid value, else 0 is assigned, as per Equation 3.

\[
S(x_i) = \frac{1}{1+e^{-x_i}}, \quad x_i = \begin{cases} 
1 & \text{if rand()} < S(x_i) \\
0 & \text{otherwise}
\end{cases} 
\] (3)

The searching functionality works on the following rules: In the initialization phase, each cuckoo (solution \( X \)) selects a host nest to lay an egg. Each cuckoo randomly follows a Levy flight during the generations, consisting of random steps fulfilling the search exploration objective. The authors (Nasa-ngium et al., 2013) followed Mantegna’s algorithm to calculate step length. Cuckoos select nests randomly, and the best fitness ones remain in the nest. Also, a fraction (\( P_a \)) of worst nests creates a new solution at a new location, taking a Levy flight. The best quality nest passes to the next generation. Figure 4 represents the pseudo-code for CS.

**ACO—Ant Colony Optimization Algorithm**

ACO (Dorigo et al., 2006) is an artificial simulation of the foraging behavior of ants. It works on the interactions between ants via pheromones to find short paths between nests and food. ACO performs well for solving complex discrete optimization problems (Eldem & Ulker, 2017; Song & Gao, 2010; Wu, 2017). It starts by constructing the ants and move them in the search space according to state transition probability and pheromone. Next, it builds the binary conversion of the solution using roulette wheel selection for our bit representation. It then computes ants’ fitness as per the different paths taken by ants and updates the global fitness with the minimum fitness achieved. Finally, it updates the pheromone considering the evaporation rate. Figure 5 illustrates the ACO pseudo-code.

**GWO — Grey Wolf Optimization Algorithm**

The leadership hierarchy and the hunting strategies in grey wolf packs inspire the GWO (Mirjalili et al., 2014) modeling. It starts by initializing wolves as random n-bit length solutions. Then, it mimics

![Figure 4. Pseudocode for CS (Gherboudj et al., 2012; Nasa-ngium et al., 2013)](image-url)
their social behavior by assigning the best solution as the alpha (α) wolf (i.e., leader wolf), second-best solution as the beta (β) wolf (i.e., subordinate wolf), third-best solution as the delta (δ) wolf (i.e., caretaker wolf), the rest as the omega (ω) wolves. Omega wolves follow their leader wolves, α, β, and δ to improve their position iteratively. The authors of this study followed the binary version of GWO proposed by Emary et al. (2016) to update wolves’ position using the random walk performed by Gosain & Sachdeva (2020a) instead of the linear step. Figure 6 represents the pseudo-code for GWO used for this study.

ABC – Artificial Bee Colony Algorithm

ABC (Karaboga & Basturk, 2007) is another intelligent swarm algorithm that mimics the foraging behavior of honeybees. A honey swarm consists of three types of bees: employees, onlookers, and scout bees interacting with one another to find high-quality food sources. Each bee is an n-bit vector solution X and commences the search as shown in Figure 7.

For this study, the N-point random insertion technique was adopted (Arun & Kumar, 2017) by randomly choosing N views with bits as 1 and N views with bits as 0 and flips these bits to get a food source. Each employed bee executes the search by performing N-point random insertions, while onlooker bees perform roulette wheel-based probabilistic solutions for the search. Lastly, scout bees, whose trial limit has reached, search to find a better replacement. They explore the search space using 1-point and 2-point random insertions by choosing views present in the global best solution. The global best solution improves iteratively.

RESULTS

This section provides a detailed analysis of the six evolutionary algorithms for solving PMVS.

The experiments were run using MATLAB R2016b, and statistical tests were performed on IBM SPSS Statistics 24. The authors used Microsoft datasets—six-dimensional fact ‘FactOnlineSales’ of
Contoso Data Warehouse (Microsoft Contoso BI Demo Dataset for Retail Industry, 2010) and seven-dimensional fact ‘Order’ of World Wide Importers (WWI) data warehouse (Wide World Importers sample database v1.0, 2016). These are publicly available sample datasets for big data and analytics. The authors used 200 queries for the six-dimensional dataset and 400 queries for the seven-dimensional dataset generated by Gaussian distribution. The local query priority was assigned values ranging from 1 to 5, and user, department, and query type ranging from 1 to 4. The methodology proposed by Gosain & Madaan (2018b) was adopted to assign weights and priority to queries and cubes. Lastly, cube priority and frequency values were normalized in the range of 0 to 1.

The population size for each algorithm was kept at 100 and search iterations at 150. The authors performed 50 trials on each dataset for each algorithm and compared the performance using all of the evaluation metrics. Table 1 lists the parameter settings applied for each evolutionary algorithm in this study’s experiments.

The performance of all of the algorithms was evaluated and compared based on the following metrics:

1. Fitness cost (FC) (Gosain & Madaan, 2018b): The global minimum fitness cost computed using Equation 2. This incorporates both query processing cost and view maintenance cost.
2. Execution time (ET) (Azgomi & Sohrabi, 2019): ET provides an estimate of each algorithm’s convergence speed to the minimum fitness cost. The authors selected the first iteration, after which fitness did not improve as the algorithm’s convergence point.

\[
ET = \frac{\text{Total Time Taken}}{\text{Number of Iterations}} \times \text{Convergence Point}
\]  

3. Detailed Cost Savings Ratio (DCSR) (Kotidis & Roussopoulos, 1999): This is the most widely used metric to evaluate the optimal selection of materialized views. It measures the cost savings achieved in answering the query using either the direct view, the smallest parent view, or the base tables. DCSR lies in the range of 1 to 100, with a higher value indicating more savings. Thus, ‘ci’ represents the query execution cost using the base tables, cv is the cost of answering the query from either the required materialized view (cv = 0) or the smallest materialized parent view v (cv = size(v)).

\[
DCSR = \frac{\sum_i c_i - c_v}{\sum_i c_i}
\]
4.  High Priority% (HP%) (Gosain & Madaan, 2018b): This metric evaluates the percentage of high-priority queries answered by the materialized view set: higher the ratio, better the selection.

\[
HP\% = \frac{\text{Number of High Priority Queries answered from materialized views}}{\text{Total Number of High Priority Queries}}
\]  

(6)

5.  Query processing cost (QPC) (Lin & Kuo, 2004): Computes the number of rows read to answer all queries, i.e., the size of view used to process the query. QPC is inversely proportional to DCSR.

**Empirical Analysis**

Results obtained from solving PMVS on Contoso and WWI datasets applying the six evolutionary algorithms are presented in Table 2 and Table 3. Fitness cost results from the tables show that all the algorithms reach near to the minimum fitness. However, it can be analyzed that PSO, CRO, GWO, and ABC attained minimum fitness costs. On the other hand, ACO and CS failed to reach the minimum possible fitness and became stuck at a semi-optimal solution worse than others.

Although ACO and CS appeared to have high HP%, they did not perform well in other metrics. Overall fitness is a tradeoff between query processing cost and view maintenance cost; thus, a semi-optimal solution compromises one of the costs. Therefore, the authors consider fitness cost a more valuable criterion than others to judge an algorithm’s effectiveness in this study’s experiments. Figure 8 and Figure 9 display the average convergence of fitness towards the minimum value during the search iterations. As shown in these figures, ACO and CS searching capability worsens with the increase in the search space. Their inability to converge to better fitness confirms that CS and ACO cannot explore the search space efficiently; on the other hand, PSO, CRO, GWO, and ABC succeed.

**Table 1. Parameter settings for the evolutionary algorithms**

| Algorithm | Parameter settings |
|-----------|--------------------|
| PSO       | population size pp=100, particle length = n, constants c1=c2=2, vmax=6, weight decreasing linearly from 0.6 to 0.2 |
| CRO       | M=10, N=10, r0=0.3, F_r=0.7, F_c=0.3, F_p=0.3, P_p=0, k=4 |
| CS        | Discovery rate P_a = 0.25, search domain bounds [-6,6], population size pp=100 |
| ACO       | Initial pheromone τ0=0.05, pheromone exponential weight α=1, heuristic exponential weight β=0.02, evaporation rate ρ=0.3, n=100 ants |
| GWO       | Settings based on equations 5 and 6 |
| ABC       | pp=100, Trial limit L=30, Number of employer bees= onlooker bees= pp |

**Table 2. Results of five performance metrics achieved by six evolutionary algorithms for Contoso dataset**

|               | PSO   | CRO   | CS    | ACO   | GWO   | ABC   |
|---------------|-------|-------|-------|-------|-------|-------|
| Fitness cost  | 17.26 M | 17.28 M | 17.41 M | 18.13 M | 17.26 M | 17.26 M |
| ET            | 1.69  | 1.53  | 6.45  | 2.27  | 1.83  | 22.40 |
| DCSR          | 74.49%| 74.49%| 74.43%| 73.12%| 74.49%| 74.46%|
| QPC           | 168.09 M | 168.15 M | 168.53 M | 177.11 M | 168.12 M | 168.30 M |
| HP%           | 30.64%| 30.55%| 43.51%| 39.94%| 30.49%| 28.98%|
The authors performed a more profound analysis to compare PSO, CRO, GWO, and ABC. These algorithms achieved the same high-cost savings and better query processing cost as presented in Tables 2 and 3. Figure 8 and Figure 9 show that the ABC converged faster to the minimum value than others. This behavior occurred due to N-point random insertions operations applied to each iteration solution. It proves that ABC searches diversely per iteration and reaches minimum values in fewer iterations. However, at the same time, it was observed from the ET results in both tables, that ABC consumes a significant amount of execution time, as these operations create many solutions for fitness evaluation per iteration. Therefore, ABC did not suit the problem well compared to the other competitive algorithms producing the same results in significantly less time.

PSO, CRO, and GWO appeared to produce similar results by achieving minimum fitness cost, high DCSR, low query processing cost, and considerate HP%. Hence, this study’s authors believe that PSO, CRO, and GWO are the best performers, delivering quick results for PMVS based on their experimental analysis.

### Statistical Analysis

In this subsection, the authors’ goal was to understand whether or not there is a statistical difference in the results obtained by different algorithms. Statistical tests were conducted on two metrics—fitness cost and DCSR, as these are highly relevant measures for judging materialized view selection. ANOVA test is the most suitable test to judge the statistically significant difference for two or more groups. The ANOVA test’s null hypothesis states that all groups’ mean is equal with no statistical

![Figure 8. Convergence curve of the six evolutionary algorithms on Contoso dataset](image)
difference. The authors had six groups under consideration, and hence the null (H₀) and the alternative hypothesis (Hₐ) were:

\[ H_0: \frac{1}{2} = \frac{1}{3} = \frac{1}{4} = \frac{1}{5} = \frac{1}{6} \]

\[ H_a: \text{means of any two groups is not equal} \]  

ANOVA test assumes that each group has an equal variance and is tested using the Levene statistic. Considering 95% confidence interval, Figure 10 proves homogeneity of variances for both metrics in Contoso and WWI datasets, respectively, as sig> = 0.05. Hence, the assumption for conducting ANOVA stood true.

Figure 11 presents the descriptive statistics and ANOVA test result for fitness cost and DCSR values achieved on the Contoso dataset. Based on a 5% significance level, the null hypothesis was accepted (Sig. 0.415> α 0.05) for fitness cost, proving no significant difference in the fitness cost attained by different algorithms. In contrast, the alternative hypothesis was accepted for DCSR values (Sig. 0.005 < α 0.05), implying DCSR values are different for some groups. A post hoc Tuckey HSD test was conducted to examine the cause for the significant difference. Results (partial results shown because of space constraints) in Figure 12 indicated that ACO has different DCSR values, and from Table 2, it can be noted that ACO has less DCSR than others. Therefore, a higher DCSR demonstrated a better solution. All other algorithms showed no significant difference in DCSR.

Figure 13 displays the descriptive statistics and ANOVA test results for the WWI dataset and proves a significant difference in fitness cost and DCSR values with significance value .001<0.05 and 0.000<0.05, respectively. The post-hoc Tuckey HSD results (partially shown in image) in Figure

| Test of Homogeneity of Variances | Test of Homogeneity of Variances |
|----------------------------------|----------------------------------|
| Leverene Statistic | df1 | df2 | Sig | Leverene Statistic | df1 | df2 | Sig |
| sixDimFitness | .923 | 5 | 294 | 1.000 | sevenDimFitness | .210 | 5 | 294 | .958 |
| sixDimDCSR | .270 | 5 | 294 | .929 | sevenDimDCSR | .998 | 5 | 294 | .419 |

Figure 10. Levene statistic results for fitness cost and DCSR results on a Contoso dataset and b) WWI dataset
14 show that ACO has a semi-optimal solution with lesser DCSR (i.e., an average of 3% from Table 3) than other algorithms.

The statistical tests showed that CS had no significant difference for fitness cost and DCSR in both datasets. However, from the experimental results, it was noted that CS could not reach the optimal solution and took more time than its competitors. Its significance value was drastically decreasing from six-dimensional to seven-dimensional data, showing decreasing performance in large datasets. Similarly, although ABC reached an optimal solution, it performed time-intensive iterations, leading

Figure 11. ANOVA test results for fitness cost and DCSR on Contoso dataset

|                | N | Mean   | Std Deviation | Std Error | 95% Confidence Interval | Lower Bound | Upper Bound | Minimum | Maximum |
|----------------|---|--------|---------------|-----------|-------------------------|-------------|-------------|---------|---------|
| sixDimFitness | PSO| 50     | 17291026.1    | 2413277.29| 341288.948              | 16575780.2  | 17947471.9  | 1.28E+7  | 2.23E+7 |
| DRO           | 50 | 17278208.1 | 2406229.89   | 340575.137 | 16593796.7              | 1796219.6   | 1.28E+7    | 2.26E+7 |
| CS            | 60 | 17411948.7 | 2455098.32   | 346520.492 | 16714485.9              | 18109814.6  | 1.28E+7    | 2.26E+7 |
| ACO           | 50 | 18127965.3 | 2491835.34   | 352397.037 | 17449635.2              | 18833073.3  | 1.35E+7    | 2.34E+7 |
| GWO           | 60 | 17262308.1 | 2412705.59   | 341216.766 | 16576807.3              | 17948008.9  | 1.28E+7    | 2.23E+7 |
| ABC           | 50 | 17261990.5 | 2412399.07   | 341292.027 | 16576147.5              | 17947851.0  | 1.28E+7    | 2.23E+7 |
| Total         | 300|        | 17433932.6    | 2432390.30 | 140494.120              | 17157566.1  | 1.28E+7    | 2.34E+7 |

|                | N | Mean   | Std Deviation | Std Error | 95% Confidence Interval | Lower Bound | Upper Bound | Minimum | Maximum |
|----------------|---|--------|---------------|-----------|-------------------------|-------------|-------------|---------|---------|
| sixDimDCSR    | PSO| 50     | 74.4942       | 2.01518   | 28.498                   | 73.9216     | 75.0509     | 70.44   | 70.22   |
| DRO           | 50 | 74.4857 | 2.00186      | 28.3111   | 73.9168                 | 75.0256     | 70.44       | 73.24   |
| CS            | 50 | 74.4262 | 2.15553      | 30.520    | 73.6148                 | 75.0416     | 69.16       | 70.62   |
| ACO           | 50 | 73.1250 | 2.36523      | 33.449    | 72.4528                 | 73.7972     | 67.84       | 78.20   |
| GWO           | 50 | 74.4882 | 2.01018      | 28.428    | 73.9179                 | 75.0502     | 70.44       | 73.22   |
| ABC           | 50 | 74.4619 | 2.02959      | 28.703    | 73.8861                 | 75.0397     | 70.44       | 79.22   |
| Total         | 300|        | 74.2474       | 2.14316   | 12.374                  | 74.0039     | 74.4099     | 67.84   | 73.62   |

ANOVA

|                | Sum of Squares | df | Mean Square | F     | Sig. |
|----------------|----------------|----|-------------|-------|------|
| between Groups | 2.975E+13      | 5  | 6.049E+12   | 1.006 | 0.416|
| within Groups  | 1.736E+16      | 284| 6.016E+12   |       |      |
| Total          | 1.761E+16      | 289|             |       |      |

|                | Sum of Squares | df | Mean Square | F     | Sig. |
|----------------|----------------|----|-------------|-------|------|
| between Groups | 75.733         | 5  | 15.147      | 3.432 | 0.006|
| within Groups  | 1297.611       | 284| 4.414       |       |      |
| Total          | 1373.343       | 289|             |       |      |

Figure 12. Post Hoc Tuckey HSD test results for DCSR in Contoso dataset

|                | Mean Difference (I-J) | Std. Error | Sig. | 95% Confidence Interval |
|----------------|-----------------------|------------|------|-------------------------|
|                |                       |            |      | Lower Bound             | Upper Bound  |
| CRO            | 0.0853                | 42017      | 1.000| -1.1968                 | 1.2138      |
| CS             | 0.0902                | 42017      | 1.000| -1.1392                 | 1.2713      |
| ACO            | 1.36925               | 42017      | 0.016| -1.1230                 | 2.5745      |
| GWO            | 0.0056                | 42017      | 1.000| -1.2003                 | 1.2104      |
| ABC            | 0.0323                | 42017      | 1.000| -1.1730                 | 1.2376      |
Figure 13. ANOVA test results for fitness cost and DCSR on WWI dataset

| Descriptives          | N    | Mean  | Std. Deviation | Std. Error | 95% Confidence Interval for Mean | Lower Bound | Upper Bound | Minimum | Maximum |
|-----------------------|------|-------|----------------|------------|---------------------------------|-------------|-------------|---------|---------|
| sevenDimFitness       | PSO  | 50    | 312773.66      | 307824.94  | 52018.2177                      | 323200.14   | 332269.19   | 2.25E+6 | 3.84E+6 |
|                       | CRO  | 50    | 312212.47      | 307476.32  | 51969.0022                      | 321769.49   | 332062.72   | 2.25E+6 | 3.84E+6 |
|                       | CS   | 50    | 311784.21      | 306961.52  | 52278.0342                      | 321288.57   | 342270.86   | 2.45E+6 | 4.02E+6 |
|                       | ADO  | 50    | 344313.32      | 301705.94  | 58809.7922                      | 332014.76   | 358476.69   | 2.36E+6 | 4.06E+6 |
|                       | GWO  | 50    | 312821.10      | 308404.94  | 52100.3275                      | 323201.59   | 333219.63   | 2.25E+6 | 3.84E+6 |
|                       | ABC  | 50    | 319777.12      | 393392.30  | 22136.1052                      | 315021.67   | 337337.57   | 2.25E+6 | 4.06E+6 |
| Total                 |      | 300   | 319777.12      | 393392.30  | 22136.1052                      | 315021.67   | 337337.57   | 2.25E+6 | 4.06E+6 |

| ANOVA                  | Sum of | df  | Mean Square | F    | Sig.   |
|-----------------------|--------|-----|-------------|------|--------|
| sevenDimFitness       | Squares| 6   | 5.48E+11    | 4.067 | .001   |
| Between Groups        | 2.94E+12|    | 4.11E+13    |      |        |
| Within Groups         | 4.39E+13| 294 | 1.39E+11    |      |        |
| Total                 | 4.39E+13| 299 |             |      |        |
| sevenDimDCSR          | Squares| 5   | 94.84       | 11.273| .000   |
| Between Groups        | 324.243|    | 1691.257    |      |        |
| Within Groups         | 1691.257| 294 | 5.753       |      |        |
| Total                 | 2015.499| 299 |             |      |        |

Figure 14. Post Hoc Tuckey HSD test for fitness cost and DCSR in WWI dataset

| Tukey HSD               | Mean Difference (I-J) | Std. Error | Sig. | 95% Confidence Interval for Mean | Lower Bound | Upper Bound |
|-------------------------|-----------------------|------------|------|---------------------------------|-------------|-------------|
| Dependent Variable      | (J) Algo              | (I) Algo   |      |                                 |             |             |

| sevenDimFitness         | PSO                   | CRO        | -180910.55 | 74784.8482 | 1.18 | -404439.50 | 24618.4023 |
|                         | CRO                   | CS         | -216578.7  | 74784.8482 | 0.034 | -431107.51 | -2049.7050 |
|                         | CRO                   | ADO        | -5108.3607 | 74784.8482 | 1.00 | -209420.58 | 219637.321 |
|                         | CRO                   | GWO        | -4814.4018 | 74784.8482 | 1.00 | -215010.39 | 214047.511 |
|                         | CRO                   | ABC        | -0.01050  | 47969.999 | 1.00 | -1.366 | 1.3655 |
|                         | CS                    | .00181     | 47969.999 | 1.00 | -1.3259 | 1.4282 |
|                         | ADO                   | 2.85504    | 47969.999 | 0.000 | 1.4790 | 4.2311 |
|                         | GWO                   | .00521     | 47969.999 | 1.00 | -1.3708 | 1.3813 |
|                         | ABC                   | .45571     | 47969.999 | .933 | -.9203 | 1.8318 |
to slower results, as seen in Tables 2 and 3. Thus, PSO, CRO, and GWO were the most suitable approaches for PMVS, achieving optimal solutions quickly.

CONCLUSION

This paper revisited the prioritized materialized view selection (PMVS) problem and applied six evolutionary algorithms—PSO, CRO, CS, ACO, GWO, and ABC, to find the best fitness value for a priority and frequency-based cost model. A brief description, pseudo-code, and parameter settings for each algorithm is presented with the transformation approach to building a binary-coded solution required for the problem. CS and ACO did not perform well and failed to reach optimal solutions, while ABC proved to be a highly time-intensive algorithm. The experimental results showed that PSO, CRO, and GWO algorithms performed the best by reaching minimized fitness costs and high DCSR. In addition, they minimized the overall query processing cost and answered a significant percentage of high-priority queries by materializing the right set of views. Thus, the authors recommend using PSO, CRO, and GWO algorithm to solve PMVS.

FUNDING AGENCY

Publisher has waived the Open Access publishing fee.
REFERENCES

Arun, B., & Kumar, T. V. (2017). Materialized view selection using artificial bee colony optimization. *International Journal of Intelligent Information Technologies, 13*(1), 26–49. doi:10.4018/IJIIT.2017010102

Azgomi, H., & Sohrabi, M. K. (2019). A novel coral reefs optimization algorithm for materialized view selection in data warehouse environments. *Applied Intelligence, 49*(11), 3965–3989. doi:10.1007/s10489-019-01481-w

Dorigo, M., Birattari, M., & Stutzle, T. (2006). Ant colony optimization. *IEEE Computational Intelligence Magazine, 1*(4), 28–39. doi:10.1109/MCI.2006.329691

Eldem, H., & Ülker, E. (2017). The application of ant colony optimization in the solution of 3D traveling salesman problem on a sphere. *Engineering Science and Technology, an International Journal, 20*(4), 1242-1248.

Emary, E., Zawbaa, H. M., & Hassanien, A. E. (2016). Binary grey wolf optimization approaches for feature selection. *Neurocomputing, 172, 371–381.* doi:10.1016/j.neucom.2015.06.083

Gherboudj, A., Layeb, A., & Chikhi, S. (2012). Solving 0-1 knapsack problems by a discrete binary version of cuckoo search algorithm. *International Journal of Bio-inspired Computation, 4*(4), 229–236. doi:10.1504/IJBIC.2012.048063

Gosain, A., & Heena, . (2016). Materialized cube selection using particle swarm optimization algorithm. *Procedia Computer Science, 79, 2–7.* doi:10.1016/j.procs.2016.03.002

Gosain, A., & Madaan, H. (2018a). Query prioritization for view selection. In *Progress in Intelligent Computing Techniques: Theory, Practice, and Applications* (pp. 403–410). Springer. doi:10.1007/978-981-10-3373-5_40

Gosain, A., & Madaan, H. (2018b). Efficient approach for view materialisation in a data warehouse by prioritising data cubes. *IET Software, 12*(6), 498–506. doi:10.1049/iet-sen.2017.0310

Gosain, A., & Sachdeva, K. (2020a). Random Walk Grey Wolf Optimizer Algorithm for Materialized View Selection (RWGWOMVS). In *Novel Approaches to Information Systems Design* (pp. 101-122). IGI Global.

Gosain, A., & Sachdeva, K. (2020b). Materialized View Selection for Query Performance Enhancement Using Stochastic Ranking Based Cuckoo Search Algorithm. *International Journal of Reliability Quality and Safety Engineering, 27*(03), 2050008. doi:10.1142/S0218539320500084

Harinarayan, V., Rajaraman, A., & Ullman, J. D. (1996). Implementing data cubes efficiently. *SIGMOD Record, 25*(2), 205–216. doi:10.1145/235968.233333

Karaboga, D., & Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. *Journal of Global Optimization, 39*(3), 459–471. doi:10.1007/s10898-007-9149-x

Kennedy, J., & Eberhart, R. C. (1997, October). A discrete binary version of the particle swarm algorithm. In *1997 IEEE International conference on systems, man, and cybernetics. Computational cybernetics and simulation* (Vol. 5, pp. 4104-4108). IEEE. doi:10.1109/ICSIMC.1997.637339

Kimball, R., & Caserta, J. (2004). *The Data Warehouse ETL Toolkit Practical Techniques for Extracting, Cleaning, Conforming and Delivering Data.* Wiley Publishing, Inc.

Kotidis, Y., & Roussopoulos, N. (1999). Dynamat: A dynamic view management system for data warehouses. *SIGMOD Record, 28*(2), 371–382. doi:10.1145/304181.304215

Kumar, T. V., & Kumar, S. (2012, December). Materialized view selection using simulated annealing. In *International Conference on Big Data Analytics* (pp. 168-179). Springer. doi:10.1007/978-3-642-35542-4_15

Lin, W. Y., & Kuo, I. C. (2004). A genetic selection algorithm for OLAP data cubes. *Knowledge and Information Systems, 6*(1), 83–102. doi:10.1007/s10115-003-0093-x

Loureiro, J., & Belo, O. (2006, May). A Discrete Particle Swarm Algorithm for OLAP Data Cube Selection. *ICEIS, 1,* 46-62.

Microsoft. (2010). *Microsoft Contoso BI Demo Dataset for Retail Industry.* https://www.microsoft.com/en-us/download/details.aspx?id=18279
Heena Madaan received her B.Tech degree in 2012 and M.Tech degree in 2014, with specialization in Computer Science Engineering, from Guru Gobind Singh Indraprastha University, Delhi, India. She is currently pursuing her Ph.D. degree in Computer Science at University School of Information, Communication and Technology, Guru Gobind Singh Indraprastha University. She has more than three years of working experience in Data Analytics and Data Engineering field. Her research interests include database systems, data warehouse and data analysis.

Anjana Gosain is working as a Professor in University School of Information & Communication Technology. She obtained her Ph.D. from GGSI Indraprastha University & M.Tech in Information Systems by receiving gold medal from Netaji Subhas Institute of Technology (NSIT) Delhi. Prior to joining the school, she has worked with Computer Science department of YMCA Institute of Engineering, Faridabad (1994-2002). She has also worked with REC, Kurukshetra. Her technical and research interests include data warehouse, data mining, requirements engineering, databases, software engineering, object orientation and conceptual modeling. She has published more than 80 research papers in International / National journals and conferences.