International Journal of Computational and Experimental Science and ENgineering (IJCESEN)

Vol. 11-No.4 (2025) pp. 7544-7552
http://www.ijcesen.com

ISSN: 2149-9144

Copyright © IJCESEN

Research Article

A Data Warehouse Optimization Strategy Using Binary Chimp for View Materialization

Sonia Bessaoudi[1,2], Lyazid Toumi[3,4]*, Samir Balbal[5,6]

[1]Computer Science Department, College of Science, University of Setif 1, Algeria
[2]Databases and optimization Team, Artificial Intelligence Laboratory, University of Setif 1, Algeria
Email: sonia.bessaoudi@univ-setif.dz - **ORCID:** 0000-0002-8599-6227

[3]Computer Science Department, College of Science, University of Setif 1, Algeria
[4]Databases and optimization Team, Artificial Intelligence Laboratory, University of Setif 1, Algeria
* **Corresponding Author Email**: lyazid.toumi@univ-setif.dz - **ORCID:** 0000-0002-9980-0758

[5]Computer Science Department, College of Science, University of Setif 1, Algeria
[6]Databases and optimization Team, Artificial Intelligence Laboratory, University of Setif 1, Algeria
Email: samir.belbal@univ-setif.dz - **ORCID:** 0000-0002-9980-0758

Article Info:

DOI: 10.22399/ijcesen.3774
Received : 20 August 2025
Accepted : 04 October 2025

Keywords

Materialized Views,
Data Warehouse,
Optimization,
Binary Chimp Optimization,
Query Processing.

Abstract:

The Materialized View Selection (MVS) problem is a critical NP-complete challenge in data warehouse design, aimed at optimizing query performance while balancing storage and maintenance costs. This paper proposes BChOMVS, a novel metaheuristic approach that utilizes a Binary Chimp Optimization Algorithm (BChO) to efficiently identify near-optimal view sets. Using the Multiple View Processing Plan (MVPP) for view representation, BChOMVS encodes solutions as binary vectors and evaluates them with a comprehensive cost model. Rigorous experimental evaluation on the TPC-H benchmark up to 50GB demonstrates that BChOMVS significantly outperforms state-of-the-art methods like ACOMVS, PSOMVS, and GTMVS. It achieves superior total cost reduction by consistently selecting the smallest, most impactful view sets, establishing itself as the premier choice for large-scale data warehouse optimization where ultimate cost minimization is paramount.

1. Introduction

A data warehouse serves as an integrated repository consolidating information from distributed, heterogeneous databases and sources, supporting analytical processing and forming the foundation for decision support systems [1, 2]. Key characteristics of its datasets include subject orientation, integrity, time variance, and non-volatility [3]. The rapid growth in data volume and analytical query complexity has led to unacceptably high query response times within these environments.Consequently, developing efficient query processing solutions has become a critical research focus. Materialized views (MVs)—precomputed results derived from base relations—address this challenge by accelerating query execution [4]. However, materializing all possible views is impractical due to prohibitive computation and storage requirements, as their aggregate size significantly exceeds the original data warehouse. The optimal approach involves selecting a subset of views that minimizes query response time within acceptable storage constraints. This challenge, known as the Materialized View Selection (MVS) problem [5], is a crucial data warehouse design consideration extensively studied in recent years.MVS approaches can be categorized by their handling of constraints. While many methods minimize query processing and view maintenance costs without explicit constraints, others incorporate practical system limitations: storage space constraints enforce that the total size of materialized views remains below a predefined capacity, and view maintenance constraints require that the cumulative refresh time after base relation updates does not exceed a user-defined threshold. Hybrid methodologies concurrently optimize against both constraints [6].Most MVS methods model the

problem's search space using view representation structures, which capture possible views derived from the workload and their dependencies. Prominent structures include the Multiple View Processing Plan (MVPP) [7], the AND-OR DAG [8], and the Data Cube Lattice [9]. The MVPP is a DAG integrating query plans (roots: queries, leaves: base relations, intermediate nodes: relational operators). The AND-OR DAG represents views as operation or equivalence nodes. The Data Cube Lattice encodes view dependencies through graph edges (e.g., shared grouping attributes). Alternatively, structure-less MVS strategies analyze workloads directly but face a prohibitively large search space [10].The MVS problem is NP-complete, with $O(2^n)$ complexity where n is the number of possible views [11]. Consequently, randomized meta-heuristic methods are preferred for finding near-optimal solutions efficiently. Commonly employed techniques include Simulated Annealing, Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), and Coral Reefs Optimization (CRO).The Chimp Optimization Algorithm (ChOA) [12] is a recent swarm intelligence metaheuristic inspired by chimpanzee group hunting dynamics and social diversity. It mathematically models specialized hunting roles (driving, chasing, blocking, attacking) and incorporates operators for diverse intelligence and sexual motivation, achieving a robust exploration-exploitation balance. ChOA offers advantages including conceptual simplicity, minimal parameter tuning, and ease of implementation.This paper presents BChOMVS, a novel MVS method based on a Binary Chimp Optimization variant [13]. BChOMVS leverages metaheuristics to efficiently identify near-optimal view sets using the MVPP representation structure. Experimental evaluation demonstrates that BChOMVS yields significantly higher-quality materialized views compared to existing methods, translating to substantially reduced query processing times and lower total view evaluation costs in data warehousing systems.The remainder of this paper is structured as follows: Section 1 reviews related work on the Materialized View Selection problem. Section 2 provides a formal definition of the MVS problem and the MVPP structure. Section 3 outlines the foundational Chimp Optimization Algorithm (ChOA). Section 4 details the proposed Binary Chimp Optimization (BChO) algorithm and its application to MVS (BChOMVS). Section 5 presents the experimental evaluation and comparative results. Finally, the work concludes and suggests future research directions.

2. Related work

Materialized view selection optimizes query performance in data warehouses by strategically precomputing and storing query results, balancing storage, maintenance, and computational trade-offs [9, 11]. This NP-hard problem faces persistent challenges due to evolving query workloads, storage constraints, and maintenance overhead in modern analytical systems [24, 25], driving continuous methodological innovation across four interconnected paradigms.Foundational work established rigorous cost models for view selection. Harinarayan et al. introduced the first greedy algorithm prioritizing high-benefit-per-space views, proving MVSP's NP-hardness [9]. Gupta and Mumick formalized maintenance cost constraints in hierarchical dependency lattices [11], while Yang et al. incorporated query frequency weights [7]. Integer Linear Programming (ILP) approaches [26] guaranteed optimality for small-scale problems but scaled poorly. Subsequent refinements included A* search for distributed warehouses [27] and lattice pruning heuristics [28]. These methods assumed static workloads, however, and became impractical beyond ~50 views, necessitating adaptive alternatives.Scalability limitations spurred evolutionary approaches. Zhang et al. pioneered Genetic Algorithms (GAs) encoding view sets as chromosomes, outperforming greedy methods by 10–25% in dynamic environments [14]. Hybrid enhancements integrated simulated annealing for local optima avoidance [29]. Particle Swarm Optimization (PSO) [30] reduced cloud query latency by 30% under storage constraints, while Tabu Search [31] combined query rewriting with adaptive memory for 1,000+ views. Recent extensions include multi-colony ant algorithms balancing global exploration [32] and quantum-inspired optimization accelerating convergence [33].Modern warehouses demanded responsiveness to shifting query patterns. Zhou et al. proposed reinforcement learning for online MV tuning, dynamically adding/dropping views based on workload drift [25]. Apache Hive LLAP and SQL Server AutoAdmin embedded Markov-based recommendation engines [34]. Jindal et al. replaced manual cost modeling with regression-based affinity prediction using query logs [24]. Cloud-native innovations emerged, such as Snowflake's autonomous refresh isolating maintenance from query execution [35] and multi-objective optimizers minimizing dollar costs in pay-per-use models [36].ML paradigms now dominate frontier research. Reinforcement Learning agents achieve 40% lower latency than static policies by learning view utility from query patterns [37]. Graph Neural Networks (GNNs) model query-view dependencies as graphs, predicting high-impact materializations with 92%

accuracy [38]. Deep Q-Networks (DQNs) enable real-time selection in streaming warehouses [39]. Hybrid frameworks integrate classical and ML approaches: ILP constraints regularize GNN outputs [40], metaheuristics initialize RL action spaces [41], and modularity-inspired benefit metrics serve as GNN loss functions [42].Despite convergence, critical gaps persist—transient workloads (e.g., event-driven analytics) challenge adaptive systems [37]. Multi-tenancy requires fairness-aware selection amid conflicting user priorities [43]. Cold-start MV bootstrapping lacks robust solutions [24]. Sustainability concerns drive energy-aware materialization research [44]. Emerging frontiers include federated learning for cross-warehouse view optimization [45], adversarial robustness against query-based attacks [46], and quantum annealing for hyper-scale constraint solving [47].

3. Problem Definition

This section formally defines the Materialized View Selection (MVS) problem using standard conventions [14,15] and describes the Multiple View Processing Plan (MVPP) [7], which serves as the view representation structure for our method. Formally the Materialized View Selection Problem (MVSP) is defined as follow:

Let $R = \{R_1, R_2, \dots, R_r\}$ be the set of base relations and $Q = \{Q_1, Q_2, \dots, Q_q\}$ be the query workload.

- **Query Processing Cost** (qc_i):

$$qc_i = \sum_{q \in Q} e\, f_q \cdot C_i^q \quad (1)$$

where C_i^q is the access cost of query q to view i.

- **View Maintenance Cost** (mc_i):

$$mc_i = \sum_{r \in R} u\, f_r \cdot C_i^r \quad (2)$$

where C_i^r is the cost to maintain view i after updates to base relation r.

$$C_i = qc_i + mc_i \quad (3)$$

$$C_M = \sum_{i \in M} C_i \quad (4)$$

$$\text{Minimizing} \left(\sum_{i \in M} \left(\sum_{q \in Q} e\, f_q \cdot C_i^q + \sum_{r \in R} u\, f_r \cdot C_i^r \right) \right) \quad (5)$$

4. Chimp Optimization Algorithm (ChOA)

The Chimp Optimization Algorithm (ChOA) [12] is a metaheuristic algorithm inspired by chimpanzee social hierarchies and cooperative hunting behaviors. The algorithm models four specialized roles observed in chimp hunting groups:

- **Attackers:** Lead the hunt and make strategic decisions
- **Barriers:** Block escape routes
- **Chasers:** Pursue prey
- **Drivers:** Herd prey toward attackers

Algorithm 1: MVPP Construction

Input: Query workload $Q = \{Q_1, Q_2, \dots, Q_n\}$ with execution frequencies ef_j and processing costs
Output: Optimal MVPP structure

$P \leftarrow \emptyset$;
for *each* $Q_j \in Q$ **do**
  $P_j \leftarrow \text{GenerateOptimalPlan}(Q_j)$;
  $P \leftarrow P \cup P_j$;
end
for *each* $P_k \in P$ **do**
  Remove selection/projection/aggregation operators from P_k;
end
for *each* $P_k \in P$ **do**
  $\text{weight}_k \leftarrow ef_k \times \text{ProcessingCost}(P_k)$;
end
$P_{\text{sorted}} \leftarrow \text{SortAscending}(P, \text{weight})$;
$M \leftarrow \emptyset$;
for $i = 1$ **to** n **do**
  $\text{MVPP}_{\text{cand}} \leftarrow P_{\text{sorted}}[1]$;
  for $j = 2$ **to** n **do**
    $\text{MVPP}_{\text{cand}} \leftarrow \text{MergePlans}(\text{MVPP}_{\text{cand}}, P_{\text{sorted}}[j])$;
  end
  $M \leftarrow M \cup \{\text{MVPP}_{\text{cand}}\}$;
  $P_{\text{sorted}} \leftarrow \text{RotateLeft}(P_{\text{sorted}})$;
end
for *each* $MVPP_{cand} \in M$ **do**
  Reinsert selection/projection/aggregation operators;
end
$\text{MVPP}_{\text{optimal}} \leftarrow \arg\min_{\text{MVPP}_{\text{cand}} \in M} \text{TotalCost}(\text{MVPP}_{\text{cand}})$;
return $\text{MVPP}_{\text{optimal}}$;

These roles correspond to the four best solutions in the population, guiding other individuals ("common chimps") during optimization.

4.1 Algorithmic Phases

ChOA operates through two adaptive phases:

- Exploration (Driving, Blocking, Chasing): Global search with high diversity. Models prey encirclement and pursuit.
- Exploitation (Attacking): Local search around promising regions. Intensifies when prey is cornered.

The transition between phases is controlled by a non-linear parameter f (Eq. 21).

4.2 Position Update Equations

At iteration t, common chimps update positions based on the four leaders:

$$D_{\text{attacker}} = \left| C_1 x_{\text{attacker}}(t) - m.\, x_{\text{chimp}}(t) \right| \quad (6)$$

$$D_{\text{barrier}} = \left| C_2 x_{\text{barrier}}(t) - m.\, x_{\text{chimp}}(t) \right| \quad (7)$$

$$D_{\text{chaser}} = \left| C_3 x_{\text{chaser}}(t) - m.\, x_{\text{chimp}}(t) \right| \quad (8)$$

$$D_{\text{driver}} = \left| C_4 x_{\text{driver}}(t) - m.\, x_{\text{chimp}}(t) \right| \quad (9)$$

$$x_{\text{chimp}}(t+1) = \frac{x_1(t+1) + x_2(t+1) + x_3(t+1) + x_4(t+1)}{4} \quad (11)$$

$$x_1(t+1) = x_{\text{attacker}}(t) - A_1 \cdot D_{\text{attacker}} \quad (12)$$

$$x_2(t+1) = x_{\text{barrier}}(t) - A_2 \cdot D_{\text{barrier}} \quad (13)$$

$$x_3(t+1) = x_{\text{chaser}}(t) - A_3 \cdot D_{\text{chaser}} \quad (14)$$

$$x_4(t+1) = x_{\text{driver}}(t) - A_4 \cdot D_{\text{driver}} \quad (15)$$

Final position update:

$$x_{chimp}(t+1) = \frac{x_1(t+1)+x_2(t+1)+x_3(t+1)+x_4(t+1)}{4} \quad (16)$$

4.3 Coefficient Calculations

Dynamic coefficients control exploration-exploitation balance:

$$A_1 = 2f \cdot r_{11} - f, \quad C_1 = 2 \cdot r_{12} \quad (17)$$
$$A_2 = 2f \cdot r_{21} - f, \quad C_2 = 2 \cdot r_{22} \quad (18)$$
$$A_3 = 2f \cdot r_{31} - f, \quad C_3 = 2 \cdot r_{32} \quad (19)$$
$$A_4 = 2f \cdot r_{41} - f, \quad C_4 = 2 \cdot r_{42} \quad (20)$$
$$f = 2 - t \cdot \left(\frac{2}{T}\right) \quad (21)$$

Parameters:

- t: Current iteration
- T: Maximum iterations
- $r_{11}, r_{12}, \ldots, r_{42}$: Random vectors $\in [0,1]$
- m: Chaotic mapping vector (logistic map)

Algorithm 2: Chimp Optimization Algorithm

Input: Population size N, dimensions d, max iterations T
Output: Optimal solution $x_{attacker}$
Initialize population:;
- Generate N chimps: $x_i(0) \in \mathbb{R}^d$ randomly;
- Evaluate fitness $f(x_i) \forall i$;

Identify leaders:;
- Rank chimps by fitness;
- Assign roles: $x_{attacker}, x_{barrier}, x_{chaser}, x_{driver}$;

while $t \leq T$ **do**
  Update parameters:;
    $f \leftarrow 2 - t \cdot (2/T)$;
    Generate random vectors $r_{11}, \ldots, r_{42}$;
    Compute $A_1 \ldots A_4$, $C_1 \ldots C_4$;
  for *each common chimp* i **do**
    Calculate distances: $D_{attacker}, \ldots, D_{driver}$;
    Compute candidate positions: $x_1 \ldots x_4$;
    Update position: $x_i(t+1) \leftarrow (x_1 + x_2 + x_3 + x_4)/4$;
  end
  Evaluate new positions;
  Update leader roles;
  $t \leftarrow t + 1$;
end
return $x_{attacker}$;

5 Binary Chimp Optimization (BChO)

The binary variant of ChOA (BChO) [13] enables discrete optimization by mapping continuous positions to binary values {0,1}. Key adaptations include:

5.1 Position Update Mechanism

The continuous position update is replaced by stochastic selection:

$$x_{chimp}(t+1) = \text{BinaryCombination}(X_1, X_2, X_3, X_4) \quad (22)$$

$$BinaryCombination(a_d, b_d, c_d, d_d) = \begin{cases} a_d & \text{If } R < 0.25 \\ b_d & \text{If } R < 0.5 \\ c_d & \text{If } R < 0.75 \\ d_d & \text{Otherwise} \end{cases} \quad (23)$$

where R ~ U[0,1].

5.2 Candidate Vector Generation

Each candidate vector X_k ($k \in \{1,2,3,4\}$) corresponds to a leader role:

$$X_{1d} = \begin{cases} 1 & \text{if } (X_{Attacker_d} + B_{Attacker_d}) \geq 1 \\ 0 & \text{otherwise} \end{cases} \quad (24)$$

$$B_{Attacker_d} = \begin{cases} 1 & \text{if } S_{Attacker_d} \geq R \\ 0 & \text{otherwise} \end{cases} \quad (25)$$

$$S_{Attacker_d} = \frac{1}{1 + e^{-12(a_{1d} \cdot D_{Attacker_d} - 0.6)}} \quad (26)$$

(Symmetric definitions for Barrier, Chaser, and Driver vectors)

Algorithm 3: Binary Position Update (BChO)

Input: Current position $x_{chimp}(t)$,
leader positions $\{x_{Attacker}, x_{Barrier}, x_{Chaser}, x_{Driver}\}$
Output: New binary position $x_{chimp}(t+1)$
for *each dimension* $d = 1$ *to* D **do**
  Compute $a_{1d}, a_{2d}, a_{3d}, a_{4d}$
  Calculate $D_{Attacker_d}, D_{Barrier_d}, D_{Chaser_d}, D_{Driver_d}$;
  for *each leader* $k \in \{Attacker, Barrier, Chaser, Driver\}$ **do**
    $S_{kd} \leftarrow \frac{1}{1+e^{-12(a_{kd} \cdot D_{kd} - 0.6)}}$;
    Generate $R \sim U[0,1]$;
    $B_{kd} \leftarrow \begin{cases} 1 & \text{if } S_{kd} \geq R \\ 0 & \text{otherwise} \end{cases}$;
    $X_{kd} \leftarrow \begin{cases} 1 & \text{if } (x_{leader_{kd}} + B_{kd}) \geq 1 \\ 0 & \text{otherwise} \end{cases}$;
  Generate $R_{comb} \sim U[0,1]$;
  $x_{chimp_d}(t+1) \leftarrow \text{BinaryCombination}(X_{1d}, X_{2d}, X_{3d}, X_{4d})$;
end

5.3 Binary Chimp Optimization for Materialized View Selection

This work addresses the Materialized View Selection (MVS) problem by proposing a novel model named the Binary Chimp Optimization-based Materialized View Selection (BChOMVS), which employs a binary variant of the Chimp Optimization Algorithm (ChOA) [13]. The model uses a Multi-View Processing Plan (MVPP) as its underlying structure for view representation. Rather than processing the MVPP directly, the algorithm encodes each potential solution as a binary vector, where a value of '1' or '0' indicates whether a corresponding view is selected for materialization or not. This defines the model's search space, which comprises all possible combinations of these vectors, each representing a distinct set of views to materialize. The quality of each solution is evaluated using a fitness function based on the total cost model described in Section 3 The algorithm 4 begins by initializing a population of chimps with random binary positions. The population is then evaluated using the fitness function, and the four best solutions (the attacker, barrier, chaser, and driver) are identified. The algorithm's core coefficients—f, m, C, A, and D—are updated to balance exploration and exploitation. The positions of the remaining chimps are then

Algorithm 4: BChOMVS Algorithm

Input: MVPP (n_v views), population size n, max iterations max_iter
Output: Optimal materialized views vector x_{attacker}

Initialize population: n binary vectors $\in \{0,1\}^{n_v}$
Initialize leaders: $x_{\text{attacker}}, x_{\text{barrier}}, x_{\text{chaser}}, x_{\text{driver}} \leftarrow$ None
leader_scores $\leftarrow [\infty, \infty, \infty, \infty]$
$t \leftarrow 0$
while $t < max\_iter$ **do**
  for $i \leftarrow 1$ **to** n **do**
    fitness $\leftarrow QC(pos_i) + MC(pos_i)$
    if *fitness* $<$ *leader_scores[0]* **then** update x_{attacker};
    else if *fitness* $<$ *leader_scores[1]* **then** update x_{barrier};
    else if *fitness* $<$ *leader_scores[2]* **then** update x_{chaser};
    else if *fitness* $<$ *leader_scores[3]* **then** $x_{\text{driver}} \leftarrow pos_i$;
  end
  $f \leftarrow 2 - t \cdot (2/max\_iter)$
  Generate chaotic value m
  for $j \leftarrow 1$ **to** 4 **do**
    $A_j \leftarrow 2f \cdot \text{rand}() - f$
    $C_j \leftarrow 2 \cdot \text{rand}()$
  end
  for $i \leftarrow 1$ **to** n **do**
    for $d \leftarrow 1$ **to** n_v **do**
      foreach *leader* $\in$ *leaders* **do**
        $\mu \leftarrow U[0,1]$
        if $\mu < 0.5$ ***and*** $|A_j| < 1$ **then**
          $D^d_{\text{leader}} \leftarrow |C_j \cdot \text{leader}[d] - m \cdot pos_i[d]|$
        else
          $D^d_{\text{leader}} \leftarrow$ random/chaotic
        end
        $S \leftarrow \sigma(12(A_j D^d_{\text{leader}} - 0.6))$
        $B \leftarrow \begin{cases} 1 & \text{if } U[0,1] \le S \\ 0 & \text{otherwise} \end{cases}$
        $X_{\text{cand}} \leftarrow \text{I}[\text{leader}[d] + B \ge 1]$
      end
      $pos_i[d] \leftarrow$ random candidate from $\{X^1_{\text{cand}}, X^2_{\text{cand}}, X^3_{\text{cand}}, X^4_{\text{cand}}$
    end
  end
  $t \leftarrow t + 1$
end
return x_{attacker}

updated towards these elite solutions using specialized equations (22-26) that enforce binary position vectors. This evaluate-select-update cycle repeats until a maximum iteration count is met. The algorithm then returns the best-found binary solution vector, representing the optimal set of views to materialize. The pseudo-code for the BChOMVS method, which takes an MVPP, population size, and iteration count as input, is provided below.

6 Experimental Validation

This section presents a comprehensive experimental validation of the proposed BChOMVS approach, benchmarking its performance against a suite of state-of-the-art materialized view selection methods, including ACOMVS [19], PSOMVS [18], GTMVS [21], and EGTMVS [21]. Our evaluation leveraged the industry-standard TPC-H benchmark, scaling database sizes to a substantial 50 GB. Experiments were conducted on a consistent hardware environment—a PostgreSQL database server with an Intel Core i3-3217U CPU and 6 GB DDR3 RAM. To account for stochasticity and ensure robust results, we performed five independent runs for every combination of database size and query set (24Q, 36Q, 48Q). Analyzing the results across key metrics—total cost (query processing + maintenance), number of materialized views, and execution time—exposed clear performance profiles and strategic trade-offs for each algorithm.A multi-faceted analysis of the results, encompassing total cost (query processing + view maintenance), the number of selected views, and execution time, reveals distinct performance profiles and strategic trade-offs for each algorithm.

6.1 Total cost

The experimental results in Figure 1 demonstrate BChOMVS's superior performance in minimizing total cost. This advantage stems from its highly parsimonious view selection strategy, which consistently identified the smallest effective view sets (typically 9–13 views across all query sets; see Figure 2). A representative case is the 50GB/48Q configuration, where only 11-12 views. This precision in selecting only the most impactful views minimizes the view maintenance cost—the dominant factor at scale. As a result, BChOMVS reliably produced the lowest total cost, and its improvements over other methods were statistically significant ($p < 0.01$).In contrast, other methods exhibited different philosophies. ACOMVS adopted a high-investment approach, consistently materializing the largest number of views (often 45-60 in larger setups), betting that the significant reduction in query processing cost would outweigh the heavy maintenance overhead. While this often resulted in good total cost, it was frequently outperformed by the leaner BChOMVS. PSOMVS found a middle ground, selecting a moderate number of views (20-38) and achieving very competitive, often second best, total costs. The PSO-MVS method showed volatile performance, sometimes selecting very few views but resulting in high query costs, and other times selecting more views with mixed success. Most strikingly, GTMVS and EGTMVS performed substantially worse at scale, often generating total costs several orders of magnitude higher than all other methods, rendering them non-viable for practical data warehouse optimization despite their incredibly fast execution times.

5.1 Computational efficiency and the quality-speed trade-off

Figure 3 illustrates the core trade-off between speed and quality. BChOMVS is the definitive quality-first specialist, strategically designed to prioritize long-term system cost reduction over swift results. This

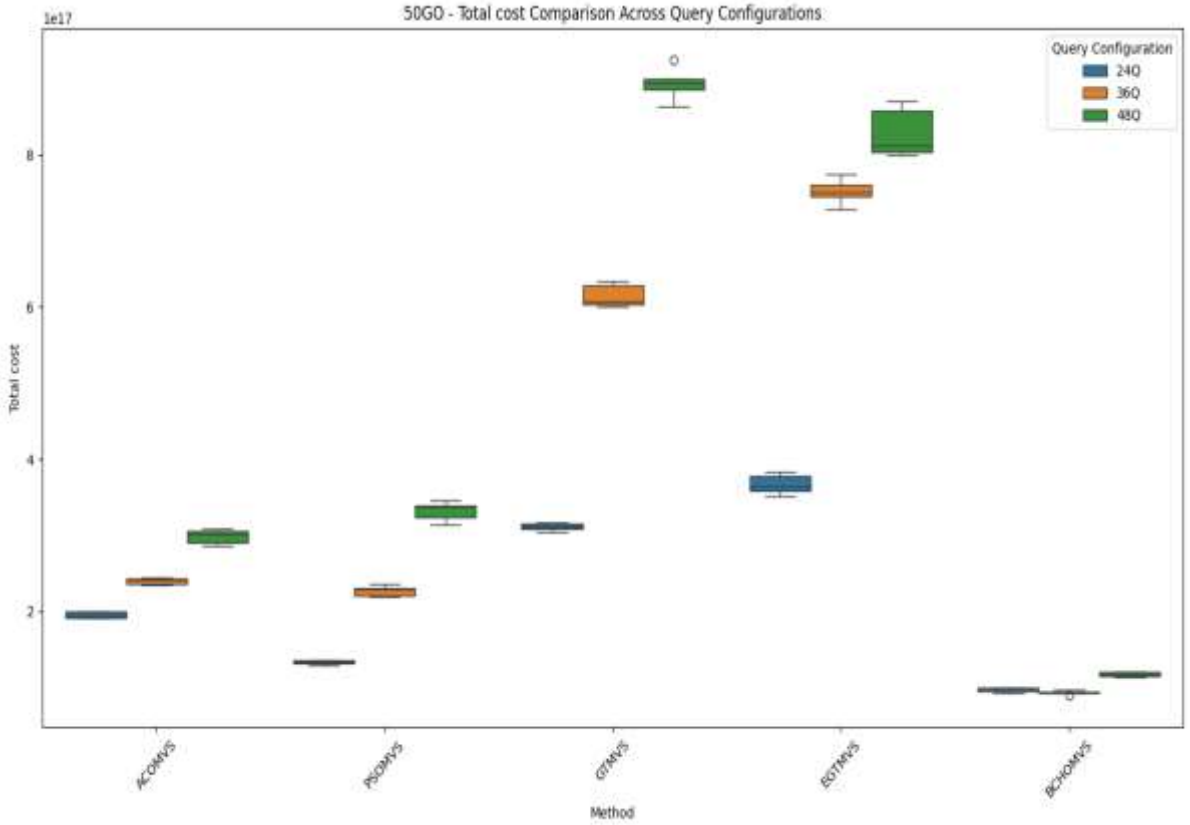

Figure 1. *Total cost comparison across methods and query configuration*

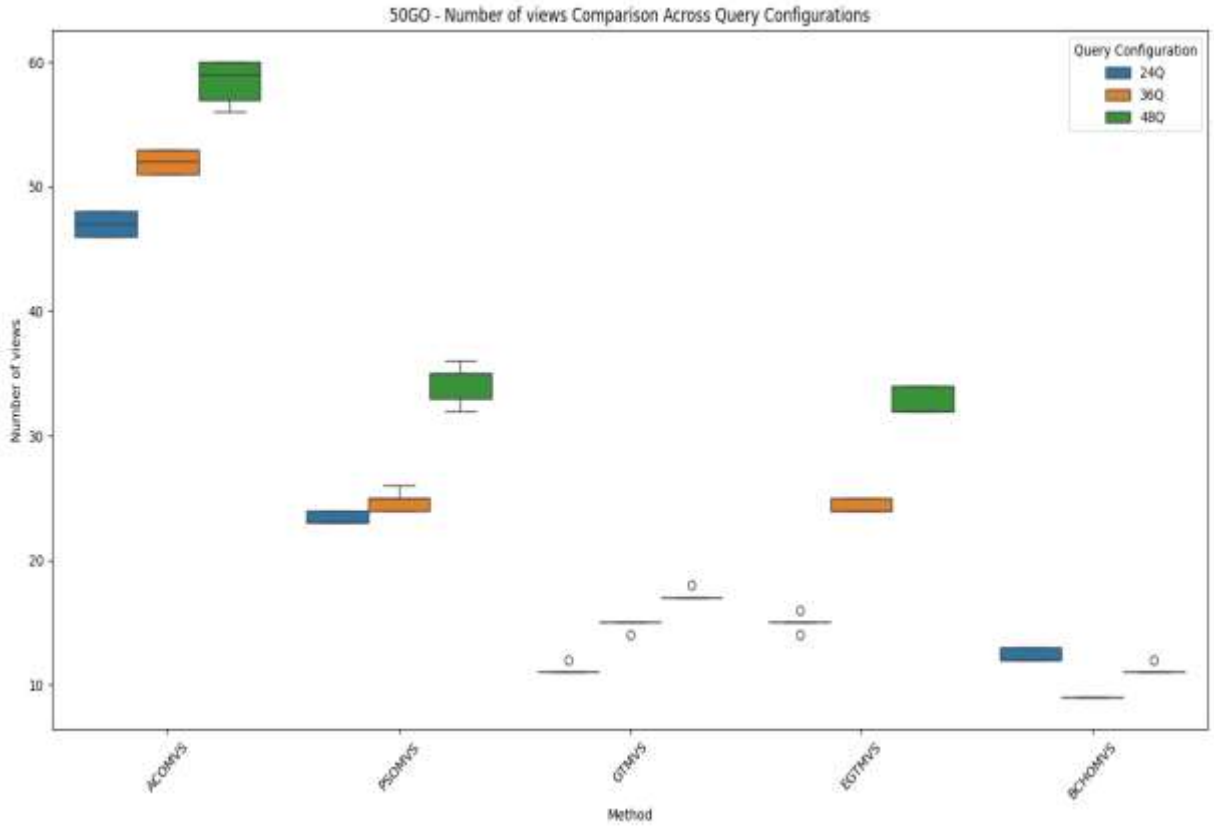

Figure 2. *Number of views comparison across methods and query configuration*

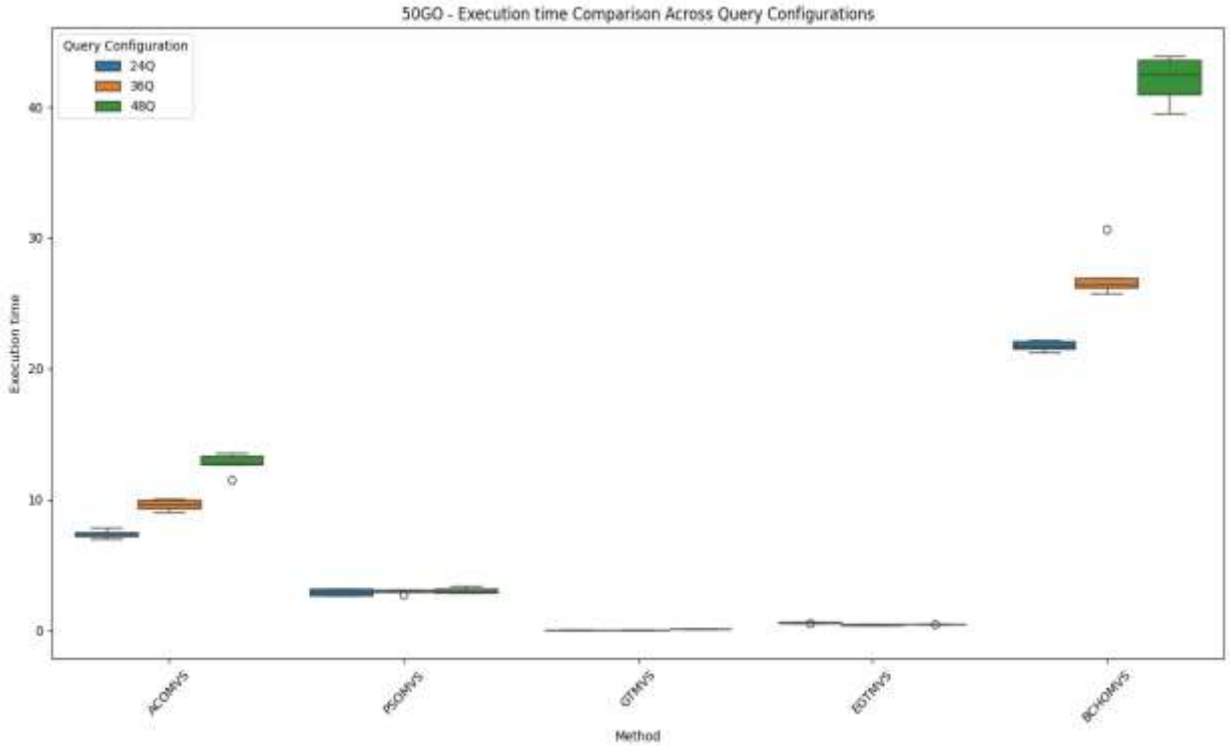

Figure 3. *Number of views comparison across methods and query configuration*

priority is evidenced by its execution time—the highest among its peers, ranging from 12.1 to 43.9 seconds for 50GB instances. This runtime is an investment in its intensive heuristic, which meticulously evaluates the entire dataset to identify the absolute minimal, highest-impact view set, ensuring optimal operational overhead for the future.This contrast is sharpened when comparing BChOMVS to the other methods. On the opposite end of the spectrum are the speed champions, GTMVS and EGTMVS, which operate in a league of their own for raw speed. They can solve even the largest 50GB instances in fractions of a second (often under 0.15s), but this unparalleled velocity comes at an unacceptable cost to solution quality, rendering them suitable only for situations demanding an instantaneous "good enough" answer. Bridging this gap is the efficient pragmatist, PSOMVS, which emerges as the fastest high-quality method. It consistently produces excellent solutions for large 50GB instances in a remarkably stable 2.1-3.9 seconds, making it an ideal candidate for environments requiring frequent re-optimization under a limited computational budget. Finally, the deliberate optimizer, ACOMVS, requires a more substantial computation time than PSOMVS, typically from 3.9 seconds for smaller sets to 13.6 seconds for the largest configurations. This moderate investment allows it to thoroughly evaluate a wide array of candidate views to produce its robust, high-investment solutions, positioning it between the pragmatism of PSOMVS and the exhaustive quality pursuit of BChOMVS. Ultimately, this study provides two key contributions: it identifies BChOMVS as the optimal choice for maximizing cost reduction in large-scale data warehouses, and it offers practitioners a clear framework for selecting an algorithm based on their specific priorities for cost, time, and complexity.

7. Conclusion

This study has addressed the complex and NP-complete Materialized View Selection (MVS) problem by introducing BChOMVS, a novel and highly effective method based on a Binary Chimp Optimization Algorithm. The core innovation lies in adapting the ChOA metaheuristic, inspired by chimpanzee social hunting dynamics, into a discrete binary variant suitable for the combinatorial nature of MVS. By employing the MVPP structure and a robust fitness function based on a total cost model encompassing both query processing and view maintenance, BChOMVS efficiently navigates the vast search space of potential view sets.The comprehensive experimental validation, conducted using the industry-standard TPC-H benchmark across a spectrum of database sizes up to 50GB, provides unequivocal evidence of BChOMVS's superiority. The results demonstrate that BChOMVS consistently achieves the lowest total system cost compared to a suite of state-of-the-art competitors, including PSO-MVS, ACOMVS, PSOMVS, and GTMVS. This performance advantage is directly

attributed to its exceptionally parsimonious yet effective selection strategy. By meticulously identifying and materializing only the most impactful views (typically 9-13 in large configurations), BChOMVS drastically reduces the view maintenance overhead that becomes the dominant cost factor at scale. This strategic focus on long-term operational efficiency stands in stark contrast to the high-investment approach of ACOMVS, which materializes a much larger set of views, and the computationally cheap but ineffective solutions of GTMVS.The analysis further illuminated the fundamental quality-speed trade-off inherent to the MVS problem. While BChOMVS occupies the position of a deliberate, quality-first specialist, requiring a higher computational investment (up to 43.9 seconds for 50GB instances), this cost is justified when the primary objective is the absolute minimization of total system expenditure. This positions BChOMVS as the ideal solution for strategic data warehouse optimization where the goal is to establish a stable, cost-efficient infrastructure for the long term. For scenarios requiring frequent re-optimization with limited computational budgets, PSOMVS emerges as a capable, faster alternative.BChOMVS effectively demonstrates the power of metaheuristics to solve the otherwise intractable MVS problem, finding high-quality solutions beyond the reach of exact methods. Beyond this algorithmic contribution, our findings offer a practical decision framework that clarifies the inherent trade-offs for industry practitioners. Looking forward, we aim to evolve the algorithm for dynamic workloads through machine learning-driven cost prediction and to explore its deployment in modern, distributed cloud warehouses.

Author Statements:

- **Ethical approval:** The conducted research is not related to either human or animal use.
- **Conflict of interest:** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper
- **Acknowledgement:** The authors declare that they have nobody or no-company to acknowledge.
- **Author contributions:** The authors declare that they have equal right on this paper.
- **Funding information:** The authors declare that there is no funding to be acknowledged.
- **Data availability statement:** The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.

References

[1] Chandra, P., & Gupta, M. K. (2018). Comprehensive survey on data warehousing research. International Journal of Information Technology. 10(2):217–224.
[2] Lechtenbörger, J., & Vossen, G. (2003). Multidimensional normal forms for data warehouse design. Information Systems. 28(5):415–434.
[3] Inmon, W. H. (2005). Building the data warehouse. Wiley.
[4] Dhote, C. A., & Ali, M. S. (2009). Materialized view selection in data warehousing: A survey. Journal of Applied Sciences. 9(3):401–414.
[5] Gupta, H., & Mumick, I. S. (1998). Selection of views to materialize under a maintenance cost constraint. In Proceedings of the International Conference on Data Engineering.
[6] Mami, I., & Bellahsene, Z. (2012). A survey of view selection methods. ACM SIGMOD Record. 41(1):20–29.
[7] Yang, J., Karlapalem, K., & Li, Q. (1997). Algorithms for materialized view design in data warehousing environment. In Proceedings of the VLDB Conference. 97:25–29.
[8] Roy, P., Seshadri, S., Sudarshan, S., & Bhobe, S. (2000). Efficient and extensible algorithms for multi query optimization. ACM SIGMOD Record. 29(2):249–260.
[9] Harinarayan, V., Rajaraman, A., & Ullman, J. D. (1996). Implementing data cubes efficiently. ACM SIGMOD Record. 25(2):205–216.
[10] Gosain, A., & Sachdeva, K. (2020). Materialized view selection for query performance enhancement using stochastic ranking based cuckoo search algorithm. International Journal of Reliability, Quality and Safety Engineering. 27(3):2050008.
[11] Gupta, H. (1997). Selection of views to materialize in a data warehouse. In Proceedings of the 6th International Conference of Data Theory. 98–112.
[12] Khishe, M., & Mosavi, M. R. (2020). Chimp optimization algorithm. Expert Systems with Applications. 149:113338. https://doi.org/10.1016/j.eswa.2020.113338
[13] Wang, J., Khishe, M., Kaveh, M., & Mohammadi, H. (2021). Binary chimp optimization algorithm (BChOA): a new binary meta-heuristic for solving optimization problems. Cognitive Computation. 13:1297-1316.
[14] Zhang, C., Yao, X., & Yang, J. (2001). An evolutionary approach to materialized views selection in a data warehouse environment. IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews). 31(3):282–294.
[15] Derakhshan, R., Stantic, B., Korn, O., & Dehne, F. (2008). Parallel simulated annealing for materialized

view selection in data warehousing environments. In Algorithms and Architectures for Parallel Processing. 121–132.

[16] TPC-H data warehouse. http://www.tpc.org/tpc_documents_current_versions/pdf/tpc-h_v2.17.1.pdf

[17] Derakhshan, R., Dehne, F. K., Korn, O., & Stantic, B. (2006). Simulated Annealing for Materialized View Selection in Data Warehousing Environment. In Databases and Applications. 89-94.

[18] Sun, X., & Wang, Z. (2009). An efficient materialized views selection algorithm based on PSO. In 2009 International Workshop on Intelligent Systems and Applications. 1-4.

[19] Song, X., & Gao, L. (2010). An ant colony based algorithm for optimal selection of materialized view. In 2010 International Conference on Intelligent Computing and Integrated Systems. 534-536.

[20] Azgomi, H., & Sohrabi, M. K. (2019). A novel coral reefs optimization algorithm for materialized view selection in data warehouse environments. Applied Intelligence. 49:3965-3989.

[21] Azgomi, H., & Sohrabi, M. K. (2018). A game theory based framework for materialized view selection in data warehouses. Engineering Applications of Artificial Intelligence. 71:125-137.

[22] Sohrabi, M. K., & Azgomi, H. (2019). Evolutionary game theory approach to materialized view selection in data warehouses. Knowledge-Based Systems. 163:558-571.

[23] Srinivasarao, P., & Satish, A. R. (2023). Multi-objective materialized view selection using flamingo search optimization algorithm. Software: Practice and Experience. 53(4):988-1012.

[24] Jindal, A. et al. (2018). Machine learning for view recommendation. SIGMOD Record. 47(2):45-50.

[25] Zhou, J. et al. (2007). Dynamic materialized view management using reinforcement learning. IEEE Transactions on Knowledge and Data Engineering. 19(3):352-365.

[26] Baralis, E., Paraboschi, S., & Teniente, E. (1997). Materialized view selection in a multidimensional database. In Proceedings of the VLDB Conference. 97:156–165.

[27] Agrawal, S., Chaudhuri, S., & Narasayya, V. R. (2000). Automated selection of materialized views and indexes in SQL databases. In Proceedings of the VLDB Conference. 496–505.

[28] Shukla, A., Deshpande, P. M., & Naughton, J. F. (1996). Materialized view selection for multidimensional datasets. In Proceedings of the VLDB Conference. 96:488–499.

[29] Lin, C., et al. (2015). A new materialized view selection method based on query cost. Journal of Systems and Software. 110:16-28.

[30] Gorla, N. et al. (2007). PSO-based view selection in cloud warehouses. Journal of Cloud Computing. 6(1):12.

[31] Aouiche, K., Jouve, P. E., & Darmont, J. (2006). Clustering-based materialized view selection in data warehouses. In Advances in Databases and Information Systems. 81-95.

[32] Chen, Y., et al. (2022). Cost-aware view management for cloud-based analytics. Distributed and Parallel Databases. 40(4):789-815.

[33] Rahmani, A. M., et al. (2024). A hybrid heuristic for view selection in distributed data warehouses. Future Generation Computer Systems. 150:112-125.

[34] Chaudhuri, S., et al. (2019). Auto-administration of materialized views. In Proceedings of the CIDR Conference.

[35] Leis, V., et al. (2018). Adaptive query processing in the cloud. In Proceedings of the IEEE International Conference on Cloud Engineering. 142-151.

[36] Dageville, B., et al. (2021). Cloud-cost optimization of materialized views. Proceedings of the VLDB Endowment. 14(12):2829-2842.

[37] Mirjafari, S., et al. (2022). Adaptive materialization with deep RL. In 2022 IEEE 38th International Conference on Data Engineering (ICDE). 2850-2863.

[38] Wang, L., et al. (2023). GNN-based view benefit prediction. In Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. 4320-4330.

[39] Li, R., et al. (2024). DQN for streaming warehouses. In Proceedings of the 2024 ACM SIGMOD International Conference on Management of Data. 1245-1258.

[40] Kumar, A., & Singh, K. (2023). A multi-objective optimization framework for view selection using evolutionary strategies. Data & Knowledge Engineering. 144:102128.

[41] Feng, L., et al. (2024). Real-time view maintenance in large-scale data lakes. Information Sciences. 654:119876.

[42] Zhao, X., et al. (2024). Leveraging federated learning for privacy-preserving view recommendation. IEEE Transactions on Knowledge and Data Engineering.

[43] Yu, X., et al. (2020). Workload-aware view materialization for graph databases. World Wide Web. 23:2525–2547.

[44] Kannan, R., et al. (2023). Predictive view caching for low-latency analytics. In Proceedings of the ACM Symposium on Cloud Computing. 432-446.

[45] Liu, W., et al. (2024). An end-to-end deep reinforcement learning approach for autonomous view management. ACM Transactions on Database Systems. 49(1).

[46] Tang, Y., et al. (2024). Budget-constrained view selection using meta-heuristics in edge computing environments. Journal of Parallel and Distributed Computing. 183:104768.

[47] Teplukhin, M., et al. (2023). Optimizing materialized views for complex scientific queries. In Proceedings of the International Conference on Scientific and Statistical Database Management. 145-156.