Materialized View Selection Using Backtracking Search Optimization Algorithm

Anjana Gosain and Kavita Sachdeva

Abstract Selecting the materialized views optimally is very important in designing a data warehouse and is NP-hard problem. Various evolutionary algorithms exist in literature for the appropriate selection of materialized views. In this paper, we have examined the application of backtracking search optimization algorithm (BSA), for selecting the materialized views in data warehouse. According to our experiments, the results obtained by our proposed backtracking search optimization-based materialized view selection algorithm (BSMVSA) are superior to those found using particle swarm optimization and genetic algorithm. The solution obtained by BSMVSA greatly reduces the total cost within the storage constraint.

Keywords Aggregation · Dimension · Measure · Heuristic
Evolutionary

1 Introduction

A data warehouse (DW) is a data repository that provides an integrated environment for reporting, analyzing, and supporting queries which requires complex aggregations of huge amounts of historical data [1, 2]. It is a big challenge to operate and manage such an integrated data store in a cost-effective way. Materialized views are the intermediate results stored in a data warehouse [3] that avert

A. Gosain · K. Sachdeva (✉)
University School of Information and Communications Technology,
GGS Indraprastha University, New Delhi 110078, India
e-mail: kavitasachdeva4@gmail.com

A. Gosain
e-mail: anjana_gosain@hotmail.com

V. Bhateja et al. (eds.), *Intelligent Engineering Informatics*, Advances in Intelligent Systems and Computing 695, https://doi.org/10.1007/978-981-10-7566-7_25

accessing the original data sources and thus increase the efficiency of the queries posed to a data warehouse. Optimal selection of materialized views is NP-hard problem [4] and is needed to design a data warehouse effectively. Recently, lot of attention is given to solve this problem. Researchers have given various frameworks and algorithms to deal with this problem. Data cube and lattice [5–13], MVPP [14–17] and AND-OR view graphs [18–21] are various frameworks that exist in literature. Lattice framework captures dependency among aggregate views and is used in building the data cubes with multiple dimensions. MVPP is a global query processing plan for the complete query set and exploits the existence of common sub-expressions. AND-OR view graph is used to express all the possible execution plans for evaluating a query in the query set. Using the above frameworks, various algorithms like heuristic-based algorithms [11, 20], greedy algorithms [5, 8, 18, 22], evolutionary algorithms such as genetic algorithm [9, 10, 14, 15, 21, 23, 24] and simulated annealing [13, 16, 17]have been presented for view selection. However, heuristic-based algorithms cannot compute a perfect solution within the acceptable time due to the complex nature of problem. Greedy algorithms are highly problem-dependent and are susceptible to poor local minima, while evolutionary algorithms (EA) work on randomly selected multiple solutions simultaneously to find out the optimum most solution and can be applied on various problems. Various evolutionary algorithms like particle swarm optimization (PSO) [25, 26], bee colony optimization (BCO) [27], ant colony optimization (ACO) [28], and differential evolution (DE) [29]have been used in context of materialized view selection. Backtracking search optimization algorithm (BSA) comes under the category of evolutionary algorithm and aids in solving various optimization problems [30]. With a simple structure, it is effective, fast, and has strategies for generating trial populations by taking favor of the experiences gained from previous generations and gives BSA very powerful capabilities for exploring and exploiting the population [30]. In this paper, we have implemented BSA on lattice framework to find an optimal set of views within the space constraint, thereby minimizing the total cost of query processing. We have compared our results with genetic algorithm and PSO to prove the effectiveness of BSMVSA.

This paper is organized as follows: Definition and mathematical model of materialized view selection problem is given in Sect. 2. Section 3 gives the brief review of backtracking search optimization algorithm (BSA), while Sect. 4 presents BSA-based view selection in a stepwise manner. Experimental results under different space constraints, along with comparative performance analysis with other algorithms, are discussed in Sect. 5. Section 6 ends up with conclusions.

2 Materialized View Selection

2.1 Problem Definition

Materialized view selection [31] problem can be illustrated as follows: Given some storage space X and query set Z, the problem is to determine a set of materialized views V that reduce the total query processing cost of the selected views, under the constraint that the total space occupied by selected views, V, is less than X. View selection is an important decision in the effective design of data warehouse. We have used the lattice framework [5] in determination of materialized view selection as lattice framework captures and models dependencies among aggregate views. Group-by clause [32] characterizes a data cube (DC). A path exists between the two cubes ci and cj if there is a dependency relation $c_i \leq c_j$, exists between the two cubes c_i and c_j, implying if a query can be resolved from c_i, then it can also be resolved by using cube c_j. 2^N data cubes are possible in the lattice framework, for a fact relation having N dimensions. For example, sales transactions in a data warehouse system (taken from TPC-H star schema benchmark [33]) have three dimensions, part (P), supplier (S), and customer (C), and in fact depicts the sale of parts(P) from suppliers(S) to customers(C). This fact relation will generate $2^3 = 8$ data cubes in the lattice as shown in Fig. 1.

Each level of the lattice has data cubes which are aggregated at same level of aggregation along different dimensions. The number beside each cube indicates its size (i.e., number of rows). The top cube, i.e., psc, is the base cuboid at the lowest level of aggregation. Bottom cuboid has highest level of aggregation. If a query can be directly answered from cube (*, s,*) then it can also be answered from any of its parent cubes (p, s,*), (*, s, c), or (p, s, c) by summarizing the data along some dimensions [26].

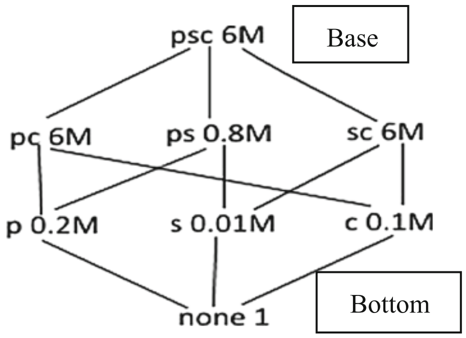


Fig. 1 Lattice framework with 8 possible data cubes [5]

2.2 Mathematical Model of Materialized View Selection Problem

Selection of materialized views is crucial for designing an efficient data warehouse. It aims at reducing the query response time by selecting an optimal set of materialized views within the storage space and cost considerations, to help accelerating the entire data warehouse. We have used the lattice framework, where cubes correspond to the views to be selected, whenever a query is invoked by user. Query invoking frequency corresponds to the cube invoking frequency. We have followed the linear cost model as proposed in [5], for evaluating the cost of answering query. This query answering cost is equivalent to the no. of rows to be accessed in the corresponding cube for the query. So, the materialized view selection (MVS) using lattice framework [9] can be defined as follows: Given a cube-lattice X, having a set of s cubes $B = (b_1, b_2, \ldots, b_s)$, set of y user queries $U = (u_1, u_2, \ldots, u_y)$, a set of query frequency values $W = (w_{u1}, w_{u2}, \ldots, w_{uy})$, set of update frequency values $CF = (g_{b1}, g_{b2}, \ldots, g_{bs})$ of the cubes in B, constrained by storage space T. Our objective function is to select a set of cubes (views) P to minimize the cost function defined in Eq. 1, under the space constraint $\sum_{b \in P} |B| \leq T$ [9].

$$\sum_{i=1}^{k} wui*R(ui\ , P) + \sum_{b \in P} gb*M(b\ , P) \tag{1}$$

where $R(u_i, P)$ and $M(b, P)$ depict the cost to evaluate query u_i and the maintenance cost of cube b, with reference to the set of materialized cubes (views) P.

3 Brief Review of Backtracking Search Optimization Algorithm

Backtracking search optimization algorithm (BSA) is one of the most popular evolutionary algorithms, for finding solution to global optimization problems. It has a simple structure, i.e., fast and efficient in solving various problems. BSA retains a memory in which it caches a population from previous generation and uses the past generated solutions while searching for solutions with superior fitness values. It has strategies for generating trial populations by taking favor of the experiences gained from previous generations and gives BSA very powerful capabilities for exploring and exploiting the population [30]. Random mutation and non-uniform crossover strategy of BSA produces very useful trial populations in each generation and enhances its ability to solve problems. It can be described by partitioning its functions into five major processes: initialization, selection-I, mutation, crossover, and selection-II [30]. Algorithm 1 presents the pseudocode for BSA.

Algorithm1: Pseudocode for BSA [30]

```
Input: ObjFun, N, D, maxcycle, mixrate, $low_{1:D}$, $up_{1:D}$
Output: globalminimum, globalminimizer
// rnd ~ U(0,1), rndn ~ N(0,1), w=rndint(.), rndint(.) ~ U(1,.) | w Ɛ (1,2,3,……..,.)
//INITIALIZATION
1 function bsa(ObjFun,N,D,maxcycle,low,up)
2   globalminimum = inf
3   for i from 1 to N do
4     for j from 1 to D do
5       $P_{i,j}$ =rnd.($up_j$ – $low_j$) + $low_j$ //Initialization of population,P
6       $oldP_{i,j}$ = rnd.($up_j$ – $low_j$) + $low_j$ //Initialization of oldP
7     end
8     $fitnessP_i$ = ObjFun($P_i$) //Initial fitness values of P
9   end
10  for iteration from 1 to maxcycle do
    //Selection – I
11    if ($a < b$ | a,$b$ ~ $U$(0,1)) then oldP := P end
12    $oldP$ := permuting ($oldP$) // permuting arbitrary changes in positions of two
       Individuals in oldP
13    Generation of Trial-Population
      // Mutation
14    mutant= P+3.rndn.(oldP - P)
      //CROSSOVER
15    $map_{1:N,1:D}$= 1 //Initial-map is an N-by-D matrix of ones.
16    if (c < d | c,d ~ $U$(0,1)) then
17      for $i$ from 1 to $N$ do
18        $map_{i,u(1:\lceil mixrate \cdot rnd \cdot D\rceil)}$ = 0 | $u$ = permuting((1,2,3,…,$D$))
19      end
20    else
21      for $i$ from 1 to $N$ do, $map_{i,randi(D)}$ = 0 , end
22    end
23    $T$ := mutant
24    for $i$ from 1 to $N$ do
25      for $j$ from 1 to $D$ do
26        if $map_{i,j}$ = 1 then $T_{i,j}$ := $P_{i,j}$
27      end
28    end
      // Boundary Control Mechanism
29    for $i$ from 1 to $N$ do
30      for $j$ from 1 to $D$ do
31        if ($T_{i,j}$ < $low_j$) or ($T_{i,j}$ > $up_j$) then
32          $T_{i,j}$ = rnd.($up_j$ – $low_j$) + $low_j$
33        end
34      end
35    end
36  end
    // SELECTION-II
37  fitnessT = ObjFnc (T)
38  for i from 1 to N do
39    if $fitnessT_i$ < $fitnessP_i$ then
40      $fitnessP_i$ := $fitnessT_i$
        $P_i$ := $T_i$
41    end
42  end
43  $fitnessP_{best}$ = min(fitnessP) | best Ɛ {1,2,3,…,N}
44  if $fitnessP_{best}$ < globalminimum then
45    globalminimum := $fitnessP_{best}$
      globalminimizer := $P_{best}$
      // Export globalminimum and globalminimizer
46  end
47 end
```

4 BSA-Based Materialized View Selection (BSMVSA)

The steps of the BSA-based algorithm for materialized view selection are elaborated below.

Step1: Population initialization: In this step, a set of views V is created constrained by the rule that the total space occupied by V is less than S as shown in (2) and the objective function, i.e., cost function of (1) is defined as

$$V_{ab} \sim U\,(x, y, z) \tag{2}$$

where b = 1, 2, … z, a = 1, 2, …, y, x represents the number of cubes (views), z corresponds to the problem dimension, and y represents the number of user queries. Each V_a is a target individual in the view set V. U is the uniform distribution. Fitness function, of the initial population (set of views V), fitness V_a is initialized as the objective function.

Step2: Selection-I: Selection-I of BSMVSA gives the historical population oldV according to (3). It is used for directing the search toward the set of

$$\text{old}V_{ab} \sim U(x, y, z) \tag{3}$$

The historical population oldV can be redefined at the start of each iteration by the 'if-then' rule shown in (4):

$$\text{if } m < n \text{ then } \text{old}V := V|m, n \sim U(0,\ 1), \tag{4}$$

After determining oldV, (5) is used to alter the order of views in oldV randomly.

$$\text{old}V := \text{permuting}(\text{old}V) \tag{5}$$

Step3: Mutation: BSMVSA's mutation process gives the basic form of trial set of views,M using (6)

$$M = V + R.\,(\text{old}V - V) \tag{6}$$

In this, R manages the amplitude of (oldV − V). The value of R = 3.rndn.

Step4: Crossover: During the crossover, BSMVSA generates the final form of the trial set of views S with the help of two steps as shown in Algorithm1 (Lines 15–21). The boundary control mechanism in (Algorithm1, lines 29–34) is used to reconstruct the individuals.

Step5: Selection-II: In this stage, the trial set of views S_i's having better fitness values than the analogous set of views V_i's are used to update the V_i's.

5 Experimental Results

We have implemented BSMVSA using MATLAB and conducted experiments by running it over standard test data sets of TPC-H star schema benchmark [33]. To test and validate the effectiveness of BSMVSA, we have compared it with PSO and genetic algorithm. The main parameters of BSMVSA are as follows: population size k = 50, problem dimension d = 3, and mix rate = .5. We have calculated the cost function on varying the dimensions of lattice, query invoking frequency, and cube invoking frequency to show the effectiveness of BSA for materialized view selection.

5.1 Considering Different Space Constraints

If unlimited space is provided to store materialized views, then all the cubes can be materialized to attain minimum query processing cost. But providing unlimited space is infeasible, so we considered different cases of space constraints as 5, 10, 15, 20, 25, 30, 35, and 40% to examine the performance of the proposed algorithm in selecting views under different space constraints. It is from Fig. 2 that aimlessly increasing the storage space cannot reduce the cost. According to the test data set, the optimal and effective storage space in terms of cost is about 20% of the total views.

5.2 On Varying the Dimensions of Lattice

We have considered three cases, i.e., by using (a) three dimensions (b) four dimensions, and (c) five dimensions, respectively. We have used the part of TPC-H benchmark [33]. For three dimensions, we chose Product p, Customer c, and

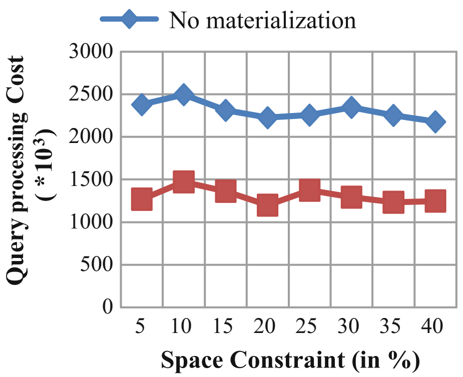
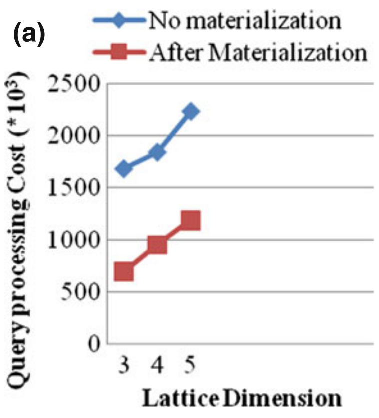


Fig. 2 Results of BSMVSA under different space constraints



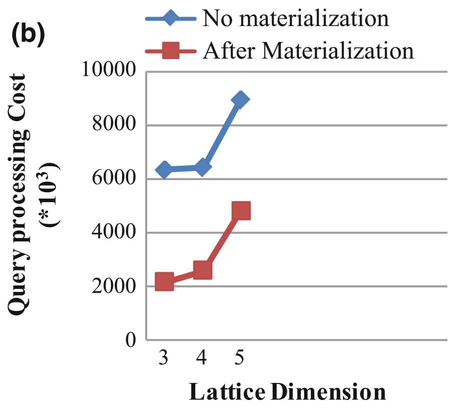


Fig. 3 **a** Results of BSMVSA on varying lattice dimensions keeping the query frequency and update frequency uniform **b** Results of BSMVSA on varying lattice dimensions with random query frequency and update frequency

Supplier s from the test set. For the fourth dimension, we included Time t dimension from the benchmark. For considering five dimensions, we added an additional dimension, Location l. The results of the experiments on changing the number of dimensions of lattice are depicted in Fig. 3a, b with uniform frequency sets and random frequency sets, respectively. It has been observed that on an exponential rise in the number of views, the processing cost of query increases linearly.

5.3 *On Increasing the Number of User Queries*

To examine the performance distribution of query processing cost achieved, we conducted an experiment on increasing the user queries. Keeping the query invoking frequency and cube invoking frequency uniform or random, the difference between query processing cost without materialization and query processing cost with materialization decreases on increasing the number of user queries as shown in Fig. 4a, b. This apparently depicts that our proposed algorithm, BSMVSA is scalable.

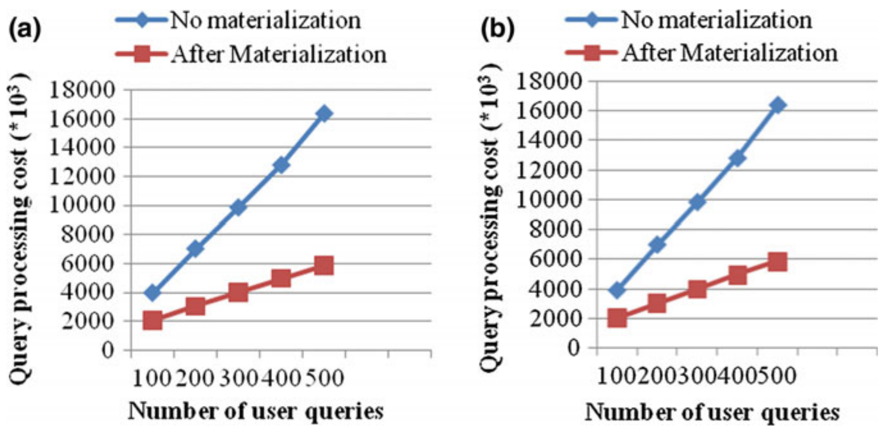


Fig. 4 **a** Number of user queries versus query processing cost (keeping query frequency and update frequency uniform) **b** Number of user queries versus query processing cost (with random query frequency and update frequency)

5.4 Comparison with PSO and Genetic Algorithm

BSMVSA yields much better results than PSO and genetic algorithm, inspite of storage and invoking frequency of data set used. Thus, BSMVSA is a better option than PSO and GA in selecting materialized views with lower query processing cost (Fig. 5).

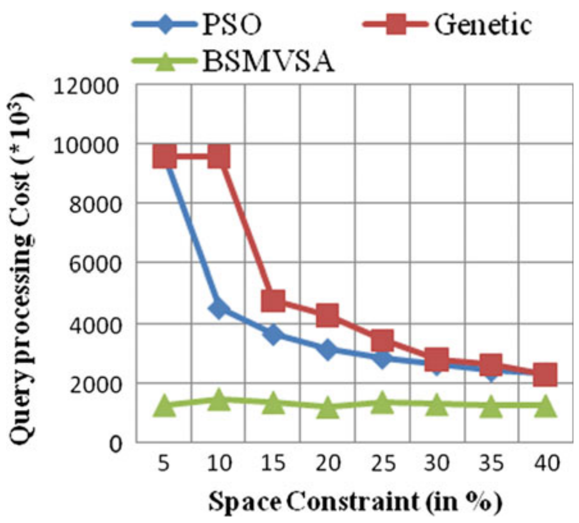


Fig. 5 Comparative results using genetic, PSO, and BSMVSA

6 Conclusion

In this study, we have implemented BSA for selection of materialized views using lattice framework. Experiments were conducted using TPC-H benchmark data set on different lattice dimensions, on different frequency sets, and considering different cases of storage space. According to the experimental results, BSMVSA always generates a superior solution. The total cost of processing and maintaining the queries approaches a minimum when space is approximately 20% of size of all views. This clearly shows that any further increase beyond this amount does not notably reduce the cost. To prove its effectiveness, over other view selection methods, it is compared with PSO and genetic algorithm.

References

1. Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Morgan Kaufman, San Francisco, CA, USA (2001)
2. Morse, S., Isaac, D.: Parallel Systems in the Data Warehouse. Prentice Hall, Upper saddle River, NJ, USA (1998)
3. Jain, H., Gosain, A.: A comprehensive study of view maintenance approaches in data warehousing evolution. ACM SIGSOFT Soft. Eng. Notes **37**(5) (2012)
4. Gupta, H., Mumick, I.S.: Selection of views to materialize under a maintenance cost constraint. In: Proceedings of the 7th International Conference on Database Theory, pp. 453–470. Springer (1999)
5. Harinarayan, V., Rajaraman, A., Ullman, J.D.: Implementing data cubes efficiently. In: Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Que, Canada, pp. 205–216 (1996)
6. Yang, D.L., Huang, M.L., Hung, M.C.: Efficient utilization of materialized views in a data warehouse. In: Advances in Knowledge Discovery and Data Mining, pp. 393–404. Springer, Berlin, Heidelberg (2002)
7. Yu, X., J., Yao, X., Choi, C.-H., Gou, G.: Materialized view selection as constrained evolutionary optimization. IEEE Trans. Syst. Man Cybern. Part C: Appl. Rev. **33**(4) (2003)
8. Vijay, K.T.V., Haider, M.: Materialized views selection for answering queries. In: Data Engineering and Management, pp. 44–51. Springer, Berlin Heidelberg (2012)
9. Lin, W.Y., Kuo, I.C.: A genetic selection algorithm for OLAP data cubes. Knowl. Inf. Syst. **6** (1), 83–102 (2004)
10. Lawrence, M.: Multiobjective genetic algorithms for materialized view selection in OLAP data warehouses. In: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation. ACM (2006)
11. Gou, G., Xu Yu, J., Lu, H.: A*search: an efficient and flexible approach to materialized view selection. IEEE Trans. Syst. Man Cybern. Part C: Appl. Rev. **36**(3), 411–425 (2006)
12. Talebian, S.H., Sameem, A.K.: Using genetic algorithm to select materialized views subject to dual constraints. In: International Conference on Signal Processing Systems. IEEE (2009)
13. Vijay, K.T.V., Kumar, S.: Materialized view selection using simulated annealing. In: Big Data Analytics, pp. 168–179. Springer, Berlin, Heidelberg (2012)
14. Zhang, C., Yao, X., Yang, J.: An evolutionary approach to materialized views selection in a data warehouse environment. IEEE Trans. Syst. Man Cyberne. Part C: Appl. Rev. **31**(3), 282–294 (2001)

15. Horng, J.-T., Chang, Y.-J., Liu, B.-J.: Applying evolutionary algorithms to materialized view selection in a data warehouse. Soft. Comput. **7**(8), 574–581 (2003)
16. Derakhshan, R., Dehne, F., Korn, O., Stantic, B.: Simulated annealing for materialized view selection in data warehousing environment. In: Databases and Applications (2006)
17. Derakhshan, R., Dehne, F., Korn, O., Stantic, B.: Parallel simulated annealing for materialized view selection in data warehousing environments. In: Algorithms and Architectures for Parallel Processing, pp. 121–132. Springer, Berlin, Heidelberg (2008)
18. Gupta, H., Mumick, S.: Selection of views to materialize in a data warehouse. IEEE Trans. Knowl. Data Eng., 24–43 (2005)
19. Mami, I., Coletta, R., Bellahsene, Z.: Modeling view selection as a constraint satisfaction problem. In: Database and Expert Systems Applications. Springer, Berlin, Heidelberg (2011)
20. Tamiozzo, A.S., Ale, J.M.: A solution to the materialized view selection problem in data warehousing. In: XX Congreso Argentino de Ciencias de la Computación (Buenos Aires, 2014)
21. Horng, J.-T., Chang, Y.-J., Lin, B.-J., Kao, C.-Y.: Materialized view selection using genetic algorithms in a data warehouse system, evolutionary computation, 1999. In: CEC 99 Proceedings of the 1999 Congress on. vol. 3. IEEE (1999)
22. Vijay, K.T.V., Ghoshal, A.: A reduced lattice greedy algorithm for selecting materialized views. In: Information Systems, Technology and Management. pp. 6–18. Springer, Berlin, Heidelberg (2009)
23. Wang, Z., Zhang, D.: Optimal genetic view selection algorithm under space constraint. Int. J. Inf. Technol. **11**(5), 44–51 (2005)
24. Talebian, S.H., Sameem, A.K.: Using genetic algorithm to select materialized views subject to dual constraints. In: International Conference on Signal Processing Systems. IEEE (2009)
25. Sun, X., Ziqiang, W.: An efficient materialized views selection algorithm based on PSO. In: Proceeding of the International Workshop on Intelligent Systems and Applications, ISA 2009, Wuhan, China (2009)
26. Gosain, A., Heena: Materialized cube selection using particle swarm optimization algorithm. In: 7th International Conference on Communication, Computing and Virtualization, Elsevier (2016)
27. Vijay Kumar, T.V., Arun, B.: Materialized view selection using improvement based bee colony optimization. Int. J. Softw. Sci. Comput. Intell. 7(4) (2015)
28. Song, X., Gao, L.: An ant colony based algorithm for optimal selection of materialized view. In: International Conference on Intelligent Computing and Integrated Systems (ICISS) (2010)
29. Vijay Kumar, T.V., Kumar, S.: Materialized view selection using differential evolution. Int. J. Innov. Comput. Appl. **6**(2) (2014)
30. Civicioglu, P.: Backtracking search optimization algorithm for numerical optimization problems. Appl. Math. Comput. **219**, 8121–8144 (2013)
31. Gupta, H.: Selection of views to materialize in a data warehouse. In: Proceedings of the 6th International Conference on Database Theory, pp. 98–112. Springer (1997)
32. Gray, J., Layman, A., Bosworth, A., Pirahesh, H.: Data cube: a relational aggregation operator generalizing group-by, cross-tabs and subtotals. Data Min. Knowl. Discovery **1**(1), 29–53 (1997)
33. O'Neil, P.E., O'Neil, E.J., Chen, X.: The star schema benchmark (SSB). Pat (2007)