# An Effective Hybrid Approach for Solving Prioritized Cube Selection Problem Using Particle Swarm Optimization and Tabu Search

Anjana Gosain and Heena Madaan(✉)

University School of Information and Communication Technology, Guru Gobind Singh Indraprastha University, New Delhi, India
heenamadaan100@gmail.com

**Abstract.** Materialized view selection is a major challenge in data warehouse management, and prioritized cube selection is further approach to find an optimal set of prioritized cubes under resource constraints. In this paper, we introduce a hybrid approach combining particle swarm optimization (PSO) algorithm with tabu search (TS) to solve the prioritized cube selection problem. Our proposed hybrid algorithm deals with PSO's premature convergence problem through integration of TS local neighbourhood search, and thus significantly improves the solution quality. We also present a neighbourhood reduction strategy based on cube information obtained during PSO search to intensify the search of TS for better solutions. Finally, we prove the effectiveness of our proposed hybrid algorithm for high-dimensional prioritized cube selection problem by comparing the results with PSO algorithm results.

**Keywords:** Prioritized cube selection · Hybrid algorithm · Particle swarm optimization · Tabu search · Data warehouse · View materialization

## 1 Introduction

Data warehouse is a storage of enterprise-wide huge data that helps decision makers of an organization in making strategic decisions. The decision makers issue complex online analytical processing (OLAP) queries in data warehouse to gain business insights. In order to provide fast processing of OLAP queries, data warehouse materializes a set of cubes needed by some of the OLAP queries. The selection of an appropriate set of cubes under resource constraints, widely known as a materialized view selection problem, has been a major challenge in data warehouse design and management [1]. In the state-of-art for selecting views [2–4], authors have defined a cost model which minimizes the cost of processing queries under resource constraints for selection of optimal views. The cost model includes factors such as query and view frequency, view size, view update frequency and view update costs. These works considered all queries and cubes to be of equal importance, however, in the real environment, queries have different relevance based on the user and system requirements. Thus, in [5, 6], authors introduced a 'priority'

© Springer Nature Singapore Pte Ltd. 2021
C. R. Panigrahi et al. (eds.), *Progress in Advanced Computing and Intelligent Engineering*, Advances in Intelligent Systems and Computing 1198, https://doi.org/10.1007/978-981-15-6584-7_34

parameter for queries and cubes and used ‘cube priority’ as an additional factor in the cost model (PFBCM). They converted cube selection problem into a prioritized cube selection problem and used particle swarm optimization (PSO) algorithm to optimize the PFBCM and selected an optimal set of prioritized cubes under space constraint.

Search space for the cube selection problem can be defined as a lattice framework [2] representing all the possible cubes and dependency relationships among the cubes. The lattice framework containing m cubes results in 2 m possible cube selection ways, and the corresponding search space in the lattice framework grows exponentially with the problem size, thus classifying cube selection as NP-hard problem [3]. NP-hard problems are unsolvable by exact algorithm in satisfactory time period, and thus a variety of heuristic algorithms (categorized as population-based and single-solution based) has been used in the literature for finding an approximate solution utilizing less computation effort [7]. State-of-art for view selection problem has applied either population-based optimization algorithms such as genetic algorithm (GA) [4, 8], particle swarm optimization (PSO) algorithm [5, 6, 9], or single-solution-based local search algorithms such as simulated annealing (SA) [10], greedy algorithm [3], to select an optimal views. The population-based optimization algorithms search the global space efficiently for large scale problems but suffer from convergence problems and may not yield global optimum [11, 12], whereas single-solution-based local search algorithms search the neighbourhood solutions in a limited search area but have a high dependence on the starting position and become impractical for high-dimensional problems [11]. Authors in [6] also applied PSO algorithm to select an optimal set of prioritized cubes under space constraints. However, in case of high-dimensional space, where several local optimal solutions exist, PSO suffers from premature convergence which may be due to poor initialization of particles or particles’ incapability to explore diverse regions when all the particles fail to escape the local optimum region [12]. Hence, PSO is combined with tabu search (TS) widely in large global optimization problems [13], such as scheduling problems [14, 15], vehicle routing problem [16], to speed up computations and find improved optimized solutions. The hybridization of the two algorithms incorporates advantages of both the algorithms: combining TS’s local search ability of intensifying search by exploring neighbours in limited space with the PSO’s global search ability of fast optimization and high efficiency by randomly exploring the space; therefore, increasing the likelihood of finding the global optimum.

In this paper, we design a hybrid algorithm which combines PSO and TS algorithm to solve the prioritized cube selection problem. Our proposed algorithm incorporates the best solution information obtained from PSO into TS to intensify the search leading to a better solution. In the algorithm, PSO searches over the complete search space of lattice framework till a pre-defined number of non-improving iterations occur. Then, the algorithm initiates TS on the best solution achieved by the PSO. TS searches over a reduced neighbourhood of cubes derived from the cube information extracted during PSO search. The cube information is collected from the global best solutions obtained in PSO iterations which is then used intensify the tabu search. Further, we analysed and compared our proposed algorithm results with the PSO algorithm results [6] and our results show a significant improvement in the optimal solution achieved.

The paper organization is as follows—Section 2 provides some of the related work. Section 3 summarizes the prioritized cube selection problem. Section 4 describes the methodology adopted in the PSO-TS hybridization. Section 5 presents the experimental results and analysis. Lastly, Sect. 6 provides the paper conclusion and the future work.

## 2 Related Work

Numerous works have been provided in the field of materialized view selection problem which selects optimal data cubes by applying global optimization algorithm, local search-based optimization algorithm or both to solve the view selection problem. Authors in [8] proposed a solution that uses a mix of evolutionary algorithm and heuristic algorithm for selecting global processing plans and views. Hybrid approach outperformed the single optimization algorithm results. Authors in [4] proposed a genetic algorithm solution to find data cubes for materialization and greedy approach to repair infeasible solutions found during the global search. Authors in [5, 6] applied PSO algorithm to find optimal prioritized cubes for materialization and outperformed GA results. Authors in [10] proposed SA algorithm which improves the set of views selected by greedy algorithm for high-dimensional data. None of the works used a hybrid combination of PSO and TS which has been proved to converge to a good optimal solution for various widely known optimization problems in other application domains. These include job scheduling problem [14, 15], vehicle routing problem [16], manpower scheduling problem [17] and frequency assignment problem [18].

## 3 Prioritized Cube Selection Problem

In a real-world scenario, when a user issues an OLAP query, the query can be of varying importance for the data warehouse system depending on how immediate the user wants it to be answered, what type of query is being issued, what is the level of user's department and what is the position of the user in the organization structure hierarchy. Considering the above factors affecting the query's importance value, authors in [6] assigned a priority to OLAP queries composed of local and global values. The local value is obtained as per how quickly user requires results and the global value is obtained from the organizational structure, wherein the value is assigned from the user level, department level and query type. Since an OLAP query is answered from the data cube which meets up the attributes requirement of the OLAP query, therefore, each cube can be of varying importance depending on how much value the issuing query for the cube holds. Thus, in [6], each of the data cube is prioritized depending on the priority value of the queries answered from the data cube. Finally, a cost model PFBCM is proposed which includes 'cube priority' as an additional factor along with the other existing factors such as cube frequency, cube size, cube update frequency and cube update cost.

According to authors in [6], given a set of $n$-dimensional cube set $c = \{c_1, c_2, \ldots c_n\}$, where each cube $c_i$ having a frequency $f_{c_i}$ from $f = \left\{f_{c_1}, f_{c_2}, \ldots, f_{c_n}\right\}$, priority $p_{c_i}$ from $p = \left\{p_{c_1}, p_{c_2}, \ldots, p_{c_n}\right\}$ and update frequency $g_{c_i}$ from $g = \left\{g_{c_1}, g_{c_2}, \ldots, g_{c_n}\right\}$ sets, prioritized cube selection is defined as a problem of selecting an optimal set $M$ of prioritized cubes for materialization by minimizing the sum of query processing cost and view maintenance cost bound to space constraint $S$.

The PFBCM cost model [6] ("Reproduced by permission of the Institution of Engineering and Technology") is defined [see (1)] where the notation used in the model is summarized as follows

$\alpha$: coefficient of optimism to assign weights to priority and frequency factor;
$q(c_i, M)$: cost of answering query from smallest ancestor cube $c_i$ selected in materialized cube set $M$;
$U(c, M)$: maintenance cost of cube c selected in materialized cube set $M$;
$S(c)$: space required by cubes c being considered for materialization.

$$\begin{aligned}\text{minimize}\,\tau(M) &= \sum_{i=1}^{n}\big((\alpha p_{c_i} + (1-\alpha)f_{c_i}) * q(c_i, M)\big)\\ &\quad + \sum_{c\in M} g_c u(c, M)\\ \text{under space constraint}\, S(M) &= \sum_{c\in M} S(c) \le S \qquad (1)\end{aligned}$$

## 4 Proposed Hybrid Algorithm

In this section, we introduce the hybrid approach combining PSO and TS algorithm to solve the prioritized cube selection problem. In the below subsections, we define the representation of candidate solution and different search spaces, general procedure for the hybridization and detailed description of adopted PSO and TS phases followed by the hybrid algorithm steps.

### 4.1 Candidate Representation and Search Space

Given a lattice framework containing $n$ cubes $c = \{c_1, c_2, \ldots, c_n\}$, a candidate solution for cube selection is encoded as a $n$-dimensional bit vector $(c_1, c_2, \ldots, c_n)$ where $c_i$ is set to 1 if cube $c_i$ is materialized and set to 0 otherwise.

Let $\varphi$ be the set of all possible $n$-dimensional bit vectors representing complete search space containing feasible and infeasible solutions:

$$\varphi = \{c : c \in \{0, 1\}^n\} \qquad (2)$$

A feasible search space $\varphi_F$ containing all the solutions which follow the defined space constraints for cube selection is defined as:

$$\varphi_F = \left\{c \in \{0, 1\}^n : \sum_{i=1}^{n} S(c) \le S\right\} \qquad (3)$$

A limited feasible search space $\varphi_{FK}$ containing all the feasible solutions restricted in a $k$-dimensional hyperplane and $k$ is defined by the index of cubes eligible for search exploration.

$$\varphi_{FK} = \left\{ c \in \{0, 1\}^n : \sum_{i=1}^{n} S(c) \leq S \cup c_k \subset c \right\} \tag{4}$$

## 4.2 Hybridization Procedure

Our hybrid algorithm comprises of two search phases. First phase applies PSO search till a specified number of iterations with non-improving solution is reached. Then, the second phase applies TS on the solution obtained from the first phase to obtain a final solution. The subsections below describe the detailed procedure of the two search phases.

### 4.2.1 First Phase—Particle Swarm Optimization (PSO)

PSO starts with a randomly initialized swarm comprising of particles and each particle is encoded as $n$-dimensional bit vector and a velocity vector. All the particles pass information in the swarm during the search to accomplish a common goal of reaching to an optimized solution. During each iteration, each particle improves its flying experience to reach an optimized solution by updating its position ($p$_best_pos) and velocity vector ($v_i$) which is based on its self-best ($p_{\text{best}}$) solution and the swarm-best ($g_{\text{best}}$) solution [19, 20].

We adopted the PSO approach used by authors in [6, 21] with a modification of maintaining an $n$-dimensional count vector as $c_{\text{vector}} = \{\text{count}_1, \text{count}_2, \ldots, \text{count}_n\}$. The count vector contains information about the data cubes chosen for materialization in the best solution during the search process. In every iteration, count value for the data cube being selected in the best solution's ($g_{\text{best}}$) position $g_{\text{best}_{\text{pos}}} = \left(c_{g_1}, c_{g_2}, \ldots, c_{g_n}\right)$ is updated in the count vector as follows

$$c_{\text{vector}} = \begin{cases} \text{count}_i = \text{count}_i + 1 & \text{if } c_{g_i} = 1; \\ \text{count}_i = \text{count}_i & \text{if } c_{g_i} = 0; \end{cases} \quad \text{for } \forall i \in \{1, \ldots, n\} \tag{5}$$

Thus, count vector maintains a promising set of data cubes that are chosen to be a part of the best solution in the global search of PSO process. PSO search is terminated when the best solution does not improve for a defined count of iterations.

### 4.2.2 Second Phase—Tabu Search (TS)

This phase starts when PSO failed to improve the solution further and TS is initiated on the best solution found by the PSO. TS [22] is a local search strategy that tries to find the best solution in its neighbourhood while preventing cycling back to the recent past solutions. This is achieved by declaring the moves resulting in the recent past solutions as tabu moves, stored as a short-term memory (i.e. tabu list), and such moves are prohibited from search for some fix number of iterations (i.e. tabu tenure). Two essential elements for TS are its neighbourhood structure and its search space [23]. The neighbourhood structure defines the local transformations to be performed on the current solution that produce its neighbourhood solutions and the search space defines the possible search regions to consider during the neighbourhood search.

*Neighbourhood Structure for TS* In our work, we use two neighbourhood structures for TS: restricted one-flip neighbourhood $N_1(x)$ [23] and restricted two-flip neighbourhood $N_2(x)$. These neighbourhood structures are defined from two moves: one-flip move and two-flip move, respectively, that have been used in the literature [23, 24] for solving binary problems.

For a solution $x = (c_1, c_2, \ldots, c_n)$, the one-flip move is defined by changing a cube bit $c_q$ with its complement value $\left(1 - c_q\right)$ and the restricted one-flip neighbourhood $N_1(x)$ [see (6)] is formed from all the feasible neighbouring solutions obtained by performing one-flip moves on cubes.

For a solution $x = (c_1, c_2, .., c_n)$, the two-flip move is defined by changing two cube bits $c_p$ and $c_q$ with the complement value $s\left(1 - c_p\right)$ and $\left(1 - c_q\right)$, respectively, and the restricted two-flip neighbourhood $N_2(x)$ [see (7)] is formed from all the feasible solutions obtained by performing two-flip move on cubes.

$$N_1(x) = \left\{ x \oplus \text{flip}(q) : \sum_{i=1}^{n} S(c) + S\left(1 - c_q\right) \le S, \quad 1 \le q \le n \right\} \tag{6}$$

$$N_2(x) = \left\{ \begin{array}{l} x \oplus \text{flip}(p, q) : \sum_{i=1}^{n} S(c) + S\left(1 - c_p\right) + S\left(1 - c_q\right) \le S, \\ 1 \le p \le n, 1 \le q \le (n - 1) \end{array} \right\} \tag{7}$$

*Search Space for TS Heuristic* For an $n$-dimensional solution $x = (c_1, c_2, \ldots, c_n)$, a complete search space consists of $|n|$ possible one-flip moves and $|n * (n - 1)/2|$ possible two-flip moves. As the search space becomes very large for high-dimensional space, the space and time complexity increase exponentially. Thus, in this study, we try to reduce both the complexity by reducing the neighbourhood search space. The aim is to intensify the local search in the direction of neighbouring cubes that appear to be promising. The information of promising cubes is maintained in the count vector $c_{\text{vector}}$ that records the cubes that have been spotted most of the times in the global best solutions seen in PSO iterations. To select the promising cubes for reduced neighbourhood, we sort values of $c_{\text{vector}}$ in descending order and choose top-$k$ cubes from the list. Thus, we narrowed our search space from $\left(|n| + \left|\frac{n(n-1)}{2}\right| \approx n^2\right)$ to $\left(k + \left(\frac{k(k-1)}{2}\right)\right) \approx k^2$ where $k^2 \ll n^2$. The narrowed space corresponds to the limited feasible search space $\varphi_{FK}$ defined above in Sect. 4.1 and the $k$-dimensional hyperplane comprises of top-$k$ cubes chosen from the count vector.

Our TS chooses a best neighbour solution in $\varphi_{FK}$ search space by applying neighbourhood structures defined in the above subsection and returns the best-found solution.

## 4.3 Hybrid Algorithm

Algorithm 1 provides the proposed hybrid algorithm code and describes the two search phases working as follows—the first phase applies PSO search methodology which randomly initiates the swarm population (Step 1) and explores through complete lattice search space $\phi$ (Step 2). It explores both the feasible and infeasible regions and tries to finds a best solution ($g_{\text{best}}$) in each iteration. Simultaneously, it also maintains a count vector ($c_{\text{vector}}$) to record the data cubes being selected in the best solution's position $\left(g_{\text{best}_{\text{pos}}}\right)$ for each iteration. The count vector indicates how many times each data cube was selected in the best solution and hence stores the most promising data cubes. When PSO does not improve the best solution for a pre-defined number of iterations, the algorithm initiates the second search phase by passing the best solution and the count vector to the second phase.

The second phase employs tabu search which considers the best solution obtained from the first search phase as its starting point (Step 3). It explores the limited feasible search space $\varphi_{FK}$ consisting of promising reduced neighbourhood search regions obtained from the explored zones of PSO which are stored in the count vector (Step 4). TS provides the best solution $s^*$ found in the neighbourhood and $s^*$ is provided as the final solution. Such hybrid methodology aims to intensify the neighbourhood search of TS by gaining information from the PSO search and provides an improved optimized solution.

```
Algorithm 1 Step 1  First phase (PSO) initialization: Randomly initialize PSO's
        swarm particles position and velocities. Initialize $c_{\text{vector}}$ to zero;
Step 2  First phase search: For each iteration until the termination condition is reached,
        run Step 2; else go to Step 3. Termination condition is a pre-defined count of
        iterations with non-improving fitness value.

        (a) Evaluate each particle's fitness value using fitness function given in Eq. (1).
            Update each particle's $p_{\text{best}}$ value and swarm's best $g_{\text{best}}$ value if improved
            from previous values;
        (b) Update count vector $c_{\text{vector}}$ as defined in Eq. (5) based on current iteration's
            $\left(g_{\text{best}_{\text{pos}}}\right)$;
        (c) Update velocity of the particle $i$ using equation

            $v_i(t+1) = \omega v_i(t) + c_1 R_1\left(p_{\text{best}_{\text{pos}_i}} - x_i(t)\right) + c_2 R_2\left(g_{\text{best}_{\text{pos}_i}} - x_i(t)\right)$

        d) Update particle's position as defined below

            $\text{if}\,\text{rand}() < S(v_{\text{id}}(t+1))$
            $\text{then}\,x_{\text{id}}(t+1) = \text{exchange}(x_{\text{id}}(t))$
            $\text{else}\,x_{\text{id}}(t+1) = x_{\text{id}}(t)$
            $\text{where}\,S(v_{\text{id}}) = 2|\text{sigmoid}(v_{\text{id}}) - 0.5|$

Step 3  Second phase (TS) initialization:

        (a) Initialize TS particle to the $g_{\text{best}}$ particle obtained from PSO and tabu list
            to ø;
```

(b) Arrange count vector $c_{\text{vector}}$ in descending order and select top-$k$ cubes to form limited feasible search space $\phi_{FK}$

Step 4 Second phase search: For every iteration, until specified number of generations have been completed, repeat Step 4; otherwise go to Step 5.

(a) Evaluate fitness value using Eq. (1) of all the neighbours lying in the search space $\phi_{FK}$ where neighbours are formed following the neighbourhood structures defined in Sect. 4.2.

b) Get the best move $v$ (either one-flip move or two-flip move) and the new best neighbour solution $x'$, then update the best solution $x$ with $x'$ and update tabu list with move $v$ tabued till its tabu tenure.

Step 5 The solution found by TS is the best solution $s*$ for the hybrid algorithm.

## 5 Experimental Analysis

This section provides complete details of our experimental setup and analysis of the obtained results. Section 5.1 details the datasets on which experiments are performed. Section 5.2 describes the parameter settings done for conducting experiments. Section 5.3 provides the experimental results comparing the performance of our proposed algorithm with PSO algorithm.

### 5.1 Datasets

In our study, we consider the publicly available data warehouse datasets (i.e. Microsoft's Contoso Data Warehouse [25] and Microsoft Azure's WideWorldImporters Data Warehouse [26]), also used by authors in [6], for experimental evaluation and comparisons. We consider the five-dimensional star schema with fact table 'Movement' and seven-dimensional star schema with fact table 'Order' from WideWorldImporters Data Warehouse; and six-dimensional star schema with fact table 'FactOnlineSales' from Contoso Data Warehouse for building a three different dimensional lattice framework required for experimental analysis. We calculated the frequency and priority value of queries and cubes following the same procedure defined by authors in [6].

For our experimental setup, we considered 100, 200 and 300 queries workload for five-dimensional, six-dimensional and seven-dimensional dataset, respectively, under Gaussian distribution. Our proposed algorithm's performance is measured on the average value obtained from 200 runs performed on each dimensional dataset using our proposed hybrid algorithm and the previously used PSO algorithm in MATLAB, and results are compared for both the algorithms in terms of PFBCM fitness value.

### 5.2 Parameter Settings

The two phases in the proposed hybrid algorithm require several parameter values to be set. The parameters for the first search phase (PSO) have been set to the common settings adapted in previous research [20]. The swarm size has been set to 40, $c_1$ and $c_2$ as 2.0, and inertia weight ω increases linearly from 0.2 to 0.6. Termination criteria for PSO have been set to 10 non-improving iterations determined experimentally.

The parameter values for the second search phase (TS) have been set empirically. The termination criteria for TS, i.e. the number of iterations is 5 for five-dimensional and six-dimensional lattice and 10 for seven-dimensional lattice framework as no improvement was observed after the specified number of iterations. Tabu tenure has been set to 3. An important parameter in TS is the *k*-dimensional hyperplane defining the size of TS neighbourhood that directly affects algorithm's computational efficiency. We analysed our algorithm's performance with different neighbourhood size sets in an additional experiment. In the experiment, we choose values of *k* as {5, 10, 15, 20, 25, 30} for 5-D lattice and 6-D lattice; whereas, for 7-D lattice, we considered values of *k* as {10, 20, 30, 40, 50, 60}. The algorithm was run for 100 instances independently for each value of k and different dimensional datasets. The results are averaged for the 100 instances which are plotted in Fig. 1 and recorded in terms of runtime value and fitness value achieved for different k values.

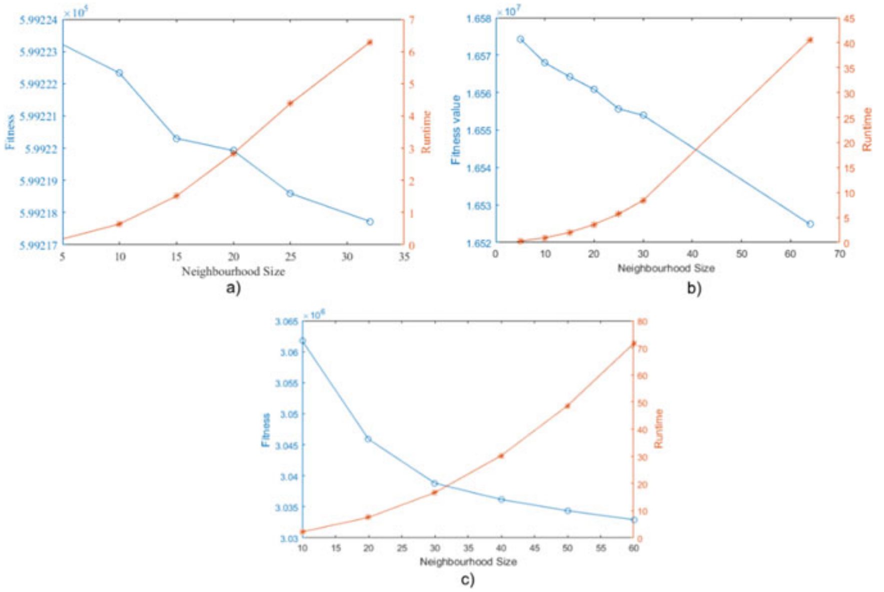

**Fig. 1.** Fitness and runtime trade-off with respect to different neighbourhood size for **a** five-dimensional, **b** six-dimensional and **c** seven-dimensional lattice

Figure 1a shows that in a 5-D lattice framework with the increase in the neighbourhood size, there is no significant difference in fitness value and increase in runtime. However, on the other hand, Fig. 1b, c shows that as the problem size increases, i.e. for 6-D and 7-D lattice framework, respectively, there is a significant improvement in the fitness value, but runtime also increases linearly with the increase in neighbourhood size. Although when considering full neighbourhood, we observed improvement in fitness value but the time consumption of the algorithm is also drastically large due to a large neighbourhood search.

In summary, for selection of $k$-parameter in tabu search, a small value of $k$ reduces runtime but restricts the search region and can miss high-quality solutions, while the large value of $k$ may result in high-quality solutions but it is very time-consuming to search large neighbourhood. Thus, in general, selecting a medium value of k maintains a good balance between the fitness value quality and the computing speed. Therefore, we have selected one-third size of the complete neighbourhood set, i.e. $k = 10$ for five-dimensional 32 cubes set; $k = 20$ for six-dimensional 64 cube set; and $k = 40$ for seven-dimensional 128 cube set; which resulted in good-quality solutions within acceptable running time.

## 5.3 Results and Comparison

Our experimental results for 5-D, 6-D and 7-D datasets are summarized in Table 1. Columns 2 and 3 show the average fitness value obtained by running previously applied PSO algorithm [6] and our proposed hybrid PSO-TS algorithm, respectively, for 200 instances. Column 4 presents the percentage improvement in the fitness value obtained by our proposed algorithm compared to the PSO algorithm. Column 5 shows how many instances improvement in the fitness value was seen after applying our proposed algorithm. Moreover, we also conducted paired $t$-test on the fitness values to prove the statistical significance of our algorithm compared to PSO.

**Table 1.** Average fitness value results reported by PSO algorithm and our proposed hybrid PSO-TS algorithm

| $n$-dimensional problem | PSO algorithm | Our proposed hybrid PSO-TS algorithm | % improvement in fitness value obtained (%) | # instances when fitness improved |
|---|---|---|---|---|
| 5-D | 597,646.3 | 597,587 | 0.015 | 23 (11.5%) |
| 6-D | 16,925,031 | 16,871,207[a] | 0.29 | 171 (88.5%) |
| 7-D | 4,073,202 | 4,016,931[a] | 1.38 | 200 (100%) |

[a]Significant at $p<0.001$

Figure 2a compares the fitness values obtained by our proposed algorithm and PSO algorithm on five-dimensional dataset. The results show that there is no significant improvement in the fitness value and the same is also indicated statistically from paired $t$-test results. Our algorithm could improve fitness value for only 11.5% instances. However, our proposed algorithm significantly outperforms PSO algorithm when the problem size increases.

Figure 2b, c shows the comparison of fitness values obtained from our proposed algorithm and PSO algorithm for six-dimensional and seven-dimensional datasets, respectively. The results show that the PSO algorithm alone could not obtain global optimum

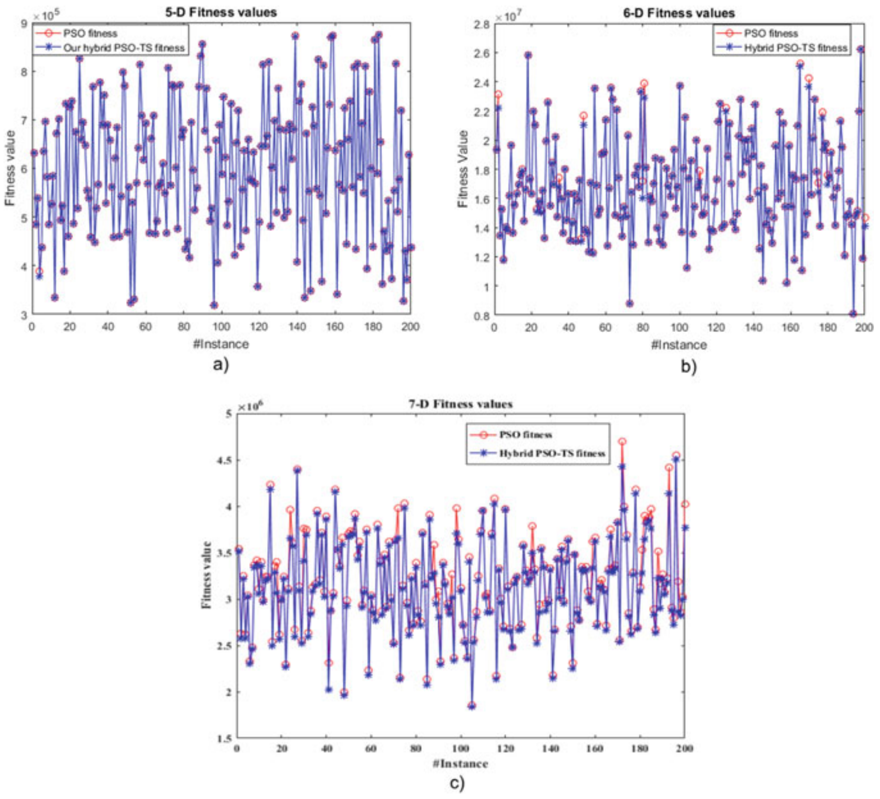

**Fig. 2.** Fitness value comparison for 200 instances obtained by our proposed PSO-TS algorithm vs PSO algorithm for **a** five-dimensional, **b** six-dimensional and **c** seven-dimensional lattice

solution (shown in red circles) as the algorithm got stuck in premature convergence problem. This proves the PSO's disadvantage of facing premature convergence when the problem size increases. However, our proposed algorithm overcomes the PSO's premature convergence problem and improves the solution obtained from PSO algorithm for 88.5% and 100% instances (as shown in blue marks) in 6-D and 7-D lattice framework, respectively. Further, paired *t*-test was conducted to check if the mean fitness value obtained by our proposed algorithm is significantly different from PSO algorithm. The statistical results prove a significant improvement in the fitness value obtained by our proposed algorithm in comparison with PSO algorithm at 99.9% confidence interval level as depicted in Table 1.

In summary, with the increase in the dimensional space, our proposed hybrid algorithm outperforms the PSO algorithm significantly with an increase in the improvement in obtained fitness value and number of cases having improved fitness value.

## 6 Conclusion and Future Work

In this paper, we proposed an algorithm hybridizing PSO and TS to solve prioritized cube selection problem. Earlier, in the literature, heuristic algorithms such as genetic algorithm, PSO had been applied to optimize the cost function for cube selection problem but such algorithms suffer from premature convergence problem. Therefore, we overcame the premature convergence problem of PSO in this paper by combining PSO with TS heuristic. The proposed algorithm started searching in the global search space using PSO algorithm, and when stuck in local optimum, initiated TS algorithm to move search in the promising zone of cubes. A reduced neighbourhood is defined, to reduce the search and time complexity for TS, which is based on the promising cubes information collected during PSO search. Finally, we evaluated the performance of our algorithm in terms of fitness value and the experimental and statistical results showed that our hybrid algorithm outperforms the PSO algorithm results. In our future work, we will try to refine TS neighbourhood structure and search space, and explore more diversified hyperplanes for solving prioritized cube selection problem.

## References

1. Widom J (1995) Research problems in data warehouse. In: 4th international conferences on information knowledge management. Baltimore, Maryland, USA, pp 25–30 (1995)
2. Harinarayan V, Rajaraman A, Ullman JD (1996) Implementing data cubes efficiently. ACM SIGMOD Record 25(2):205–216
3. Gupta H (1997) Selection of views to materialize in a data warehouse. In: ICDT. Springer, Berlin, Heidelberg, pp 98–112
4. Lin WY, Kuo IC (2004) A genetic selection algorithm for OLAP data cubes. Knowl Inf Syst 6(1):83–102
5. Gosain A, Madaan H (2016) Query prioritization for view selection. In: ICACNI'16. Springer, Singapore, pp 403–410 (2016)
6. Gosain A, Madaan H (2018) Efficient approach for view materialisation in a data warehouse by prioritising data cubes. IET Softw 12(6):498–506
7. Kokash N (2005) An introduction to heuristic algorithms. Department of Informatics and Telecommunications (2005)
8. Zhang C, Yao X, Yang J (2001) An evolutionary approach to materialized views selection in a data warehouse environment. IEEE Trans Syst Man Cybern Part C Appl Rev 31(3):282–294
9. Loureiro J, Belo O (2006) A discrete particle swarm algorithm for OLAP data cube selection. ICEIS Paphos Cyprus: 46–62
10. Kumar TV, Kumar S (2012) Materialized view selection using simulated annealing. In: International conference on big data analytics. Springer, Berlin, Heidelberg, pp 168–179 (2012)
11. Simulation and inverse modeling of semiconductor manufacturing processes. http://www.iue.tuwien.ac.at/phd/heitzinger/ (2002)
12. Kalivarapu V, Foo FL, Winer E (2009) Synchronous parallelization of particle swarm optimization with digital pheromones. Adv Eng Softw 40(10):975–985
13. Shinichi N, Ishigame A, Yasuda K (2007) Particle swarm optimization based on the concept of tabu search. In: IEEE congress on evolutionary computation. Singapore, pp 3258–3263
14. Guohui Z, Shao X, Li P, Gao L (2009) An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem. Comput Ind Eng 56(4):1309–1318

15. Gao H, Kwong S, Fan B, Wang R (2014) A hybrid particle-swarm tabu search algorithm for solving job shop scheduling problems. IEEE Trans Ind Inf 10(4):2044–2054
16. Marinakis Y, Marinaki M, Dounias G (2010) A hybrid particle swarm optimization algorithm for the vehicle routing problem. Eng Appl Artif Intell 23(4):463–472
17. Shahnazari-Shahrezaei P, Tavakkoli-Moghaddam R, Kazemipoor H (2013) Solving a multi-objective multi-skilled manpower scheduling model by a fuzzy goal programming approach. Appl Math Model 37(7):5424–5443
18. Hadji HE, Babes M (2016) Integrating Tabu Search in Particle Swarm Optimization for the frequency assignment problem. China Commun 13(3):137–155
19. Kennedy J, Eberhart RC (1997) A discrete binary version of the particle swarm algorithm. Comput Cybern Simul: 4104–4108
20. Nezamabadi-pour H, Rostami-Shahrbabaki M, Maghfoori-Farsangi M (2008) Binary particle swarm optimization: challenges and new solutions. CSI Comput Sci Eng 6(1):21–32
21. Gosain A (2016) Materialized cube selection using particle swarm optimization algorithm. Procedia Comput Sci 79:2–7
22. Glover F (1990) Tabu search: a tutorial. Interfaces 20(4):74–94
23. Lai X, Hao JK, Glover F, Lu Z (2018) A two-phase tabu-evolutionary algorithm for the 0–1 multidimensional knapsack problem. Inf Sci 436:282–301
24. Gendreau M, Potvin JY (2010) Tabu search. Handbook of metaheuristics. Springer, Berlin, pp 41–59
25. Microsoft. Microsoft contoso BI demo dataset for retail industry. https://www.microsoft.com/en-in/download/details.aspx?id=18279 (2010)
26. Microsoft. WideWorldImportersDW database catalog. https://docs.microsoft.com/en-us/sql/samples/wide-world-importers-dw-database-catalog?view=sql-server-2017 (2018)