Information Sciences 545 (2021) 620–632

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

An automatic clustering technique for query plan recommendation

Elham Azhir [a], Nima Jafari Navimipour [b,c], Mehdi Hosseinzadeh [d,e,*], Arash Sharifi [a], Aso Darwesh [f]

[a] Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran
[b] Future Technology Research Center, National Yunlin University of Science and Technology, 123 University Road, Section 3, Douliou, Yunlin 64002, Taiwan, Republic of China
[c] Department of Computer Engineering, Tabriz Branch, Islamic Azad University, Tabriz, Iran
[d] Institute of Research and Development, Duy Tan University, Da Nang 550000, Vietnam
[e] Mental Health Research Center, Psychosocial Health Research Institue, Iran University of Medical Sciences, Tehran, Iran
[f] Department of Information Technology, University of Human Development, Sulaymaniyah, Iraq

ARTICLE INFO

Article history:
Received 9 October 2019
Received in revised form 21 January 2020
Accepted 17 September 2020

Keywords:
Query processing
Incremental DBSCAN clustering
NSGA-II
Cluster validity indices
Multi-objective optimization (MOO)

ABSTRACT

The query optimizer is responsible for identifying the most efficient Query Execution Plans (QEP's). The distributed database relations may be kept in several places. These results in a dramatic increase in the number of alternative query' plans. The query optimizer cannot exhaustively explore the alternative query plans in a vast search space at reasonable computational costs. Henceforth, reusing the previously generated plans instead of generating new plans for new queries is an efficient technique for query processing. To improve the accuracy of clustering, we've rewritten the queries to standardize their structures. Furthermore, TF representation schema has been used to convert the queries into vectors. In this paper, we've introduced a multi-objective automatic query plan recommendation method, a combination of incremental DBSCAN and NSGA-II. The quality of the results of incremental DBSCAN has been influenced by Minpts (minimum points) and Eps (epsilon). Two cluster validity indices, Dunn index and Davies–Bouldin index, have simultaneously been optimized to calculate the goodness of an answer. Comparative results have been shown against the incremental DBSCAN and K-means regarding an external cluster validity index, namely, the ARI. By comparing different types of query workloads, we've found that the introduced method outperforms the other well-known approaches.

1. Introduction

Cloud computing technology delivers computing resources as services and manages the expenses more efficiently [1]. The clouds shape a large pool of computer system resources that are easily customizable and available, especially data storage and computing power [2]. This paradigm offers computing capabilities in a pay-as-you-go fashion to the users [2,3]. Therefore, Software as a Service (SaaS), Platform as a Service (PaaS), Infrastructure as a Service (IaaS), and Expert as a Service (EaaS) in cloud computing have been delivered to the customers by cloud service providers [4]. A cloud service provider provides

* Corresponding author.
E-mail addresses: JNNima@yuntech.edu.tw (N. Jafari Navimipour), hosseinzadeh.m@iums.ac.ir (M. Hosseinzadeh), a.sharifi@srbiau.ac.ir (A. Sharifi).

https://doi.org/10.1016/j.ins.2020.09.037
0020-0255/

such computing services to users through the Internet [5]. A key cloud computing service is Database as a Service (DBaaS) [6,7], in which data can be stored in virtual pools instead of servers [8]. Furthermore, the data is available through a query for users from anywhere in the cloud. Therefore, reliability and availability are essential for efficient query processing.

One of the well-known NP-hard problems is the optimization of a query in a distributed computing platform [9–11]. It means that there is no polynomial-time algorithm currently available to find the optimal plan at an acceptable time. To have a complicated query processing system, the replication of relations [12], the unpredictable nature of cloud applications and services, and the parallel execution of cloud storage are very important. The search space will be meaningfully larger in the cloud since we have the chance to execute the subqueries everywhere. The execution site has to be determined. Also, the cost of transferring relations between sites must be involved in the cost model, in addition to the Input/Outputs (I/Os). The problem is that for a given query plan, there are a lot of different equivalent execution plans, each with a corresponding execution cost. The plans are equivalent, which means they have the same results, but their costs may vary. Therefore, it is not an easy task for a query optimizer to generate an efficient query plan. Hence, finding an optimal plan with minimum cost is an important issue for DBaaS providers.

In practice, the queries with similar execution plans have been presented to the system to be processed in a cloud environment. The cloud database query engines can use similar execution plans and efficiently reuse a previously-executed query plan for equivalent queries. Therefore, the goal of this paper is to exploit the similarity of query execution plans based on measuring the similarity of the Structured Query Language (SQL) statements. This paper has used an incremental version of the popular density-based clustering algorithm, named Density-Based Spatial Clustering of Applications with Noise (DBSCAN), with a feature representation of SQL statements that is capable of detecting the similarity of queries. Furthermore, Non-dominated Sorting Genetic Algorithm II (NSGA-II) has been used as a parameter-tuning tool for incremental DBSCAN. Therefore, we've introduced a new hybrid method, NSGA-II Optimized Density-based Clustering (NODC), for enhancing the quality of clusters by identifying the ideal parameter settings by exploring the whole parameter space via NSGA-II. Also, two kinds of fitness functions have been developed, according to the clustering validation indices.

This paper contributes to the following:

(i) Using the query semantic standardization techniques to improve query clustering.
(ii) Proposing a hybrid incremental clustering algorithm to improve the performance of access plan recommendation approaches.
(iii) Simultaneously optimizing the cluster validity indices, Dunn index, and Davies–Bouldin index, for automatic determination of the incremental DBSCSAN parameters, and enhancing the quality of clusters.
(iv) An evaluation of the proposed algorithm on multiple query datasets with different structures.

The remainder of the article is as follows: The previous works are reviewed in Section 2. Section 3 reviews the incremental DBSCAN and describes the plan reuse-based optimization approach and the definitions of cluster validity indices. Section 4 introduces the proposed NODC algorithm. In Section 5, the simulation results are provided. Eventually, Section 6 presents the conclusion and suggests some indications for future researches.

2. Related work

Query optimization is an approach for providing efficient and cost-effective solutions for query processing. For complex queries or queries involving multiple data stores in a cloud computing context, the query optimization problem becomes much more challenging [13,14]. The query optimization process is an expensive one for the servers. The query plan reuse-based optimization mechanisms [15–17] benefit from machine learning techniques and similarity identification between queries to recommend a previously generated query plan to the optimizer. In this section, we've first discussed the previous studies about the optimization approaches exploiting the similarity of queries and then introduced the automatic clustering approaches.

Ghosh, Parikh [15] have presented a plan reuse-based optimization tool, named PLASTIC (PLAn Selection Through Incremental Clustering), to divide the queries into different clusters based on their structures and statistics. The presented tool has used the cluster representative query plans to execute all future queries. The PLASTIC has used the Leader Clustering Algorithm proposed by Hartigan [18]. We've evaluated the efficiency of the introduced tool by experimental results. The results have indicated that PLASTIC can predict the correct plan choice in most cases. Therefore, the proposed tool has high accuracy, low optimization time, and low space overhead, according to the experimental results. To extend the usability of PLASTIC, Sarda and Haritsa [19] have augmented the PLASTIC query's feature vector. Furthermore, a decision tree classifier has been incorporated into PLASTIC for efficient cluster assignments.

Zahir, El Qadi [17] have introduced an effective plan reuse-based optimization method for SQL queries. They've aimed at reusing the previous query plans to execute future queries instead of generating new ones. Their approach is based on detecting similarity between old and new queries [20–23]. They've used a clustering technique based on Expectation-Maximization (EM) and K-Means algorithms with SQL queries representations to identify the similarity between the queries. Furthermore, N-GRAM and Vectorized Term Frequency and Inverse Document Frequency (TF-IDF) have been used for queries representation. The outcomes have indicated that the introduced approach reduces the optimization cost.

Also, Zahir and El Qadi [16] have proposed a method for access plan recommendation based on the identification of the similarity of SQL statements. In this paper, they have used some primary classification techniques, including Association Rule (AR), Support Vector Machine (SVM), and Naive Bayes (NB) to discover similarities among queries. The experimental results have shown that the AR can provide more accurate predictions than the SVM and NB. Furthermore, proposing an approach that can use adaptive plan recommendations for enhancing the prediction accuracy is a research direction for the future.

Numerous scholars have integrated the clustering methods with meta-heuristic algorithms to get better outcomes in clustering. For instance, Guan, Yuen [24] have presented a new hybrid method, called Particle swarm Optimized Density-based Clustering and Classification (PODCC), based on combining DBSCAN clustering algorithm and Particle Swarm Optimization (PSO) algorithm. The PODCC addresses the parameter selection problem in DBSCAN clustering. It has been applied to explore the parameter space for identifying the ideal parameters for density-based clustering and classification. Several cluster validation indices have been used as fitness functions of supervised and unsupervised PODCC. They've done several experiments to evaluate the functionality of the proposed PODCC method. The results have been compared to those of DBSCAN, K-means, and SVM. According to the experimental results, the unsupervised PODCC algorithm outperforms the DBSCAN and K-means. Furthermore, the results of supervised PODCC were more accurate than those of the SVM with an appropriate fitness function.

Saini, Saha [25] have developed a multi-objective clustering technique that automatically detects the number of clusters, Self-organizing Map-based Multi-Objective Document clustering technique (SMODoc_clust). It is a combination of the Self-Organizing Map (SOM) and Multi-Objective Differential Evolution (MODE) approach. SMODoc_clust optimizes two cluster validity indexes (Pakhira-Bandyopadhyay-Maulik index, and Silhouette index) simultaneously to improve the cluster quality. They've indicated the efficiency of the introduced SMODoc_clust technique in partitioning several scientific articles and web documents. The results have revealed that SMODoc_clust can achieve the optimal global solution as compared to several clustering methods like three multi-objective clustering ones_ MOCK [26], VAMOSA [27], and NSGA-II-Clust [28,29]_ Single-Objective Genetic Algorithm (SOGA) clustering technique [30], K-means [31], and single-linkage clustering [31].

Bandyopadhyay and Maulik [32] have used the searching capability of Genetic Algorithms (GAs) for automatically detecting the number of clusters in a dataset. However, a single cluster validity measure, Davies–Bouldin index, has been applied to calculate the fitness of chromosomes. According to the experimental results, the introduced algorithm can efficiently detect hyper-spherical-shaped clusters.

Chatterjee and Mukhopadhyay [33] have been developed a Multi-Objective Evolutionary Clustering Ensemble Algorithm (MOECEA) based on NSGA-II. In their proposed MOECEA, two types of objectives (maximizing the Adjusted Rand Index (ARI) and minimizing the Standard Deviation (SD)) have been applied to provide efficient solutions. They've done several experiments with well-known diverse datasets to evaluate the performance of MOECEA as compared to the well-known existing cluster ensemble algorithms, namely Cluster-based Similarity Partitioning Algorithm (CSPA), Hypergraph Partitioning Algorithm (HGPA), Meta-Clustering Algorithm (MCLA), and the single-objective version of the proposed method. The experimental results have revealed that the method outperforms the other current approaches.

We've introduced an effective query optimization method based on reusing previously-executed query plans that consider the accuracy. We have also shown the efficacy of incorporating the NSGA-II multi-objective optimization method in the clustering process of our query plan recommendation approach.

3. Background

This section describes the background of the query plan recommendation approach and presents the key concepts we've used. Initially, we've proposed the query plan recommendation approach. Then, we've reviewed the incremental DBSCAN clustering, and three clustering validity indices used in this paper.

3.1. Plan recommendation

In this section, a similarity-based approach to query optimization has been introduced to amortize the cost of query optimization. Our main goal was to reuse the optimized access plans for new incoming queries based on the similarity between new queries and the old ones used by the query optimizer. We've used the incremental DBSCAN clustering algorithm with a textual representation to group the queries in different clusters. The presented approach has represented the statements of queries as vectors of features and then compared them with those vectors to measure the similarity between them. This paper has tried to improve the prediction accuracy for the similarity detection of queries.

Fig. 1 shows our proposed approach. It has two main phases: query representation and clustering phases. Moreover, these phases have the following steps: query standardization, feature weighting, clustering, and plan recommendation. Breaking the queries' texts into tokens and assigning a weight to each token have been carried out in the query representation phase. In the second stage, the incremental DBSCAN algorithm has been applied to the queries' clustering problem. Finally, we've compared every new query to every cluster centroid to recommend and reuse the previous access plans.

The incremental DBSCAN is a suitable clustering technique for mining in large dynamic databases. It can enhance the functionality of the query plan reuse-based optimization for large-scale cluster computing. As mentioned above, we've used

an incremental DBSCAN clustering algorithm to find the query clusters. Moreover, we've used the Cosine similarity along with incremental DBSCAN to identify similar queries. The Cosine similarity formula is given by Eq. (1):

$$\cos(\theta) = \frac{\sum_{k=1}^{n} x_{1k} x_{2k}}{\sqrt{\sum_{k=1}^{n} x_{1k}^2} \sqrt{\sum_{k=1}^{n} x_{2k}^2}} \tag{1}$$

3.2. Incremental DBSCAN algorithm

Clustering is a method used to group a set of objects using some similarity measurement. Algorithms based on partitioning, hierarchy, grid, density, and graph are different methods of clustering [34]. The performance of these clustering methods has been evaluated by various factors like the number of input parameters, shapes of clusters, data size, noise, and previous knowledge about the number of clusters.

The DBSCAN is a density-based clustering algorithm. It can find clusters of arbitrary shapes. One necessity of static DBSCAN algorithm [35] is that all data collecting should be done before running the algorithm, and then we can re-cluster all the data along with updates. Because of the dynamic nature of most datasets, we cannot collect all data objects before running the clustering algorithm. Henceforth, static DBSCAN is not efficient for dynamic datasets. The incremental DBSCAN [36] is an extension of DBSCAN that groups the newly-arrived data and updates new clusters to the previous clustering results. The incremental DBSCAN is a practical algorithm for reducing wastes in computing resources. Given two global input parameters, Epsilon (Eps) and Minimum points (Minpts), the related concepts are as follows:

- *Eps neighborhood:* For a given distance threshold (Eps), the Eps neighborhood of a point p in a dataset D includes points within a radius of Eps from point p. It is expressed as $N_{Eps}(p) = \{q | dis(p,q) \ll Eps\}$.
- *Directly density-reachable:* A point p is directly density-reachable from a point q if p is in the Eps neighborhood of q (i.e., $p \in N_{Eps}(q)$) and q is a core point (i.e., $|N_{Eps}(q)| \gg Minpts$).
- *Density-reachable:* A point p is density-reachable from a point o if a set of points $r_1, r_2, ..., r_n$, $r_1 = o$, $r_n = p$ exists such that r_{i+1} is directly density-reachable from r_i.
- *Density-connected:* Two points p and q are called density-connected, if they are density-reachable from a point o.

Here, the clusters have been created based on the existing data points and two global input parameters, i.e. Minpts and Eps. When new data has been added to the database, the incremental DBSCAN updates the created clusters. We've clustered the new data by computing the means of that data and every core points of the current clusters. Algorithm 1 indicates the pseudo-code of incremental DBSCAN.

Algorithm 1- Incremental DBSCAN Algorithm

1. Let, C_i (where $i = 1, 2, 3, \cdots$) is the new incoming data object.
2. Let, K be the existing clusters.
3. **For** i = 1 **to** n **do**
4. Find some mean M in some cluster K_p in K such that $dis(C_i, M)$ is the smallest.
5. **If** ($dis(C_i, M)$ is minimum) **&&** ($C_i \leq eps$) **&&** ($size(K_p) \geq Minpts$) **then**
6. K_p = K_p UC_i
7. **Else If** $dis(C_i! = min)$ || ($Ci > eps$) ||($size(K_p) < Minpts$)**then**
8. $C_i \rightarrow$Outlier (O_i)
9. **Else If** Count (O_i)≥ Minpts **then**
10. O_i Form new cluster (M_i)
11. **End for**
12. Repeat step 2 till all the data objects are clustered.

The incremental DBSCAN lacks a method to determine the two global input parameters (i.e., Minpts and Eps). This drawback of incremental DBSCAN can be explained by a simple problem of clustering of 593 queries in 14 groups. As shown in Table 1, the two global input parameters have randomly been set for four different cases. The second column shows the parameter settings, the 3rd column shows the number of clusters, and two last columns show the Dunn Index (DI) [37] and Davies-Bouldin Index (DBI) [38] values, respectively. Table 1 indicates that the ground truth clustering results have not obtained. Also, the higher value of the Dunn index and the lower value of the Davies-Bouldin index have improved the accuracy of the partitioning in comparison to that of the ground truth clustering.

We have conducted several runs of incremental DBSCAN with varying the input parameter Epsilon ($Eps = 0.02, \cdots, 0.18$) in steps of 0.02. The other input parameter Minpts has also been set in a range from $Minpts = 0, \cdots, 19$. We have computed

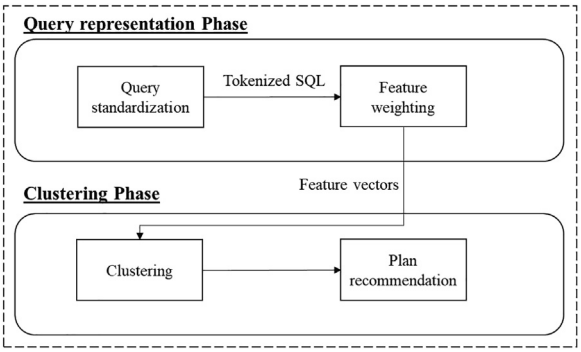

Fig. 1. The query plan recommendation approach.

Table 1
The clustering results of the query log.

Case	(Minpts, Eps)	No. of Clusters	DNI (max)	DBI (min)
1	(1, 0.12)	28	13.822	0.007
2	(7, 0.06)	22	16.708	0.009
3	(9, 0.1)	17	17.980	0.008
4	(13, 0.12)	15	18.617	0.008

the Dunn and Davies-Bouldin indices for each run. These indices have clearly indicated which cluster partitioning is more appropriate. For compact and well-separated clusters, high values of the Dunn index and small values of Davies-Bouldin are expected. Fig. 2 displays the Dunn and Davies-Bouldin indices as functions of the number of clusters for several different values of Epsilon and Minpts parameters.

As depicted in Fig. 2 (a), the Dunn index increases as the number of clusters with a global maximum formed for the division into 14 groups. Also, the lowest values can be observed for partitioning into 14 groups for the Davies-Bouldin diagram (Fig. 2 (b)).

This paper has proposed the NODC system to find the most appropriate parameters for incremental DBSCAN. This paper has also used the Dunn and Davies-Bouldin indices as fitness functions for NODC.

3.3. Cluster validity measures/indices

It is difficult to define when a clustering result is acceptable. Therefore, several clustering validation techniques and indices have been developed. The purpose of the cluster validation is to discover the partition that matches the basic data in the right way. In this paper, we've applied the Dunn index, Davies-Bouldin index, and Adjusted Rand Index (ARI) clustering evaluation metrics to confirm the clustering outcomes.

The Dunn Index is a clustering evaluation metric that measures the minimum distance between clusters and the maximum distance among data points from the identical cluster. This index aims at identifying the compact and well-separated

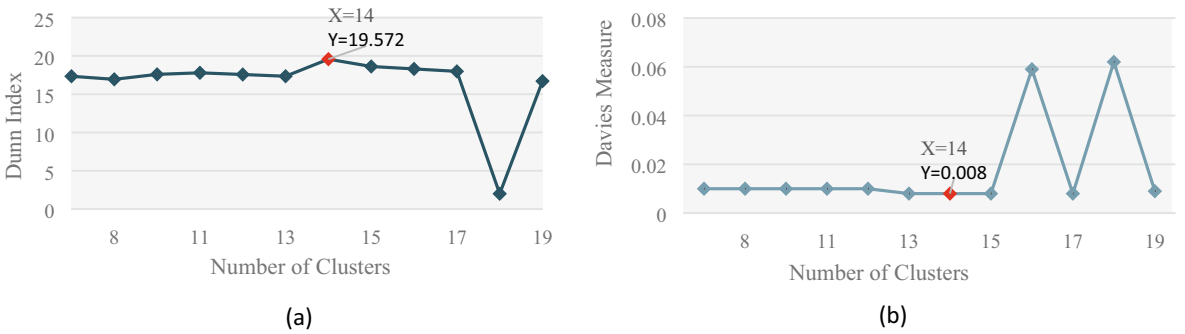

Fig. 2. (a) The Dunn index vs. the number of clusters. (b) Davies measure vs. the number of clusters.

clusters. The higher values of the Dunn index mean that there are compact clusters and well-separated clusters. The Dunn index is determined by equation (2)

$$D_k = min_{i=1,\cdots,k}\left\{ min_{j=i+1,\cdots,k}\left(\frac{d(c_i,c_j)}{max_{r=1,\cdots,k} diam(c_r)}\right)\right\} \tag{2}$$

where, $d(c_i,c_j)$ is the dissimilarity among clusters. $c_i$and c_j are determined by $d(c_i,c_j) = min_{x\in c_i, y\in c_j}(d(x,y))$. $diam(c)$ is the diameter of the cluster; $diam(c) = max_{x,y\in c}(d(x,y))$.

Moreover, the Davies-Bouldin index finds the compressed and well-divided clusters. The goal of the Davies-Bouldin index is to find the clusters with minimum intra-cluster distances. A smaller DBI means longer distance among the clusters and compactness in them. The DBI is defined as follows:

$$DBI = \frac{1}{k}\sum_{i=1}^{k} D_i \tag{3}$$

where k is the number of clusters and D_i the tightness of the cluster C_i, taking the worst-case situation. It is determined by:

$$D_i = max_{j;i\neq j} R_{i,j} \tag{4}$$

In which, i and j are cluster indices, $R_{i,j}$ is the summary assessment of two clusters of a ratio among the sum of tightness of the clusters and looseness among them.

Rand Index (RI) [39,40] computes the similarity between the two clustering solutions. By the perfect agreement between the two partitions, the RI reaches its highest value. A few issues are common in RI like the fact that the anticipated value of the RI for two random partitions is not constant, or that the Rand statistic goes to its upper limit of unity with the increasing count of clusters. The Adjusted Rand Index (ARI) has been proposed by Hubert and Arabie [39] as an enhancement of RI. We've suggested the ARI as the index of choice for assessing the correspondence among the two partitions in clustering analysis with variant counts of clusters. We can compute the ARI as below:

$$\mathrm{ARI} = \frac{\binom{n}{2}(\mathrm{a}+\mathrm{d}) - [(a+b)(a+c)+(c+d)(b+d)]}{\binom{n}{2}^2 - [(a+b)(a+c)+(c+d)(b+d)]} \tag{5}$$

where,

- a: the two data points have been allocated to one cluster in the two partitions,
- b: the two dissimilar data points have been allocated to one cluster,
- c: the two similar data points have been allocated to diverse clusters,
- d: the two dissimilar data points have been allocated to diverse clusters.

Table 2 shows how to interpret the values of each measure to assess the clustering functionality.

4. Proposed methodology

The proposed NSGA-II Optimized Density-based Clustering (NODC) is based on the concepts of using NSGA-II and cluster measurement indexes to improve the input parameter settings for the incremental DBSCAN algorithm. Fig. 3 presents the data flow diagram of the NODC which starts by generating an initial population of individuals. In addition, the source code of the NODC is available on the Mendeley Data repository[1]. The key steps of this algorithm are as follows:

1. *Generate initial population:* The initial population has been randomly produced. While the proposed algorithm attempts to identify the ideal values for Minpts and Eps parameters that can properly divide the dataset, the values of Minpts and Eps parameters encoded in diverse solutions are changed within a range. This group of solutions with changing values of Minpts and Eps creates the initial population.
2. I*ncremental DBSCAN process:* The phases of the incremental DBSCAN clustering method have been done on the total dataset, given the Minpts and Eps values encoded in the solution as the input parameters, to obtain a partitioning corresponding to a solution in the population.
3. *Evaluate objective functions:* Given an initial set of randomly generated candidate solutions, their objective functions have been calculated. The concept of dominance has been applied to classify the solutions. Arbelaitz, Gurrutxaga [41] have classified 30 cluster validity indices into three groups based on their success rate and found that the Dunn Index and the Davies – Bouldin index have fallen into the first group and achieved better results compared to other groups. Henceforth, two internal cluster validity indexes [42], Dunn index and Davies – Bouldin index, have been computed and applied as the objective functions of the existing solution to calculate the goodness of the partitioning encoded in a solution.

[1] . https://doi.org/10.17632/8m34tr5xs9.1

Table 2
The performance indication of the cluster validity measures.

Measure	Optimization type	Range
DNI	Maximum	[0,∞]
DBI	Minimum	[0,1]
ARI	Maximum	[0,1]

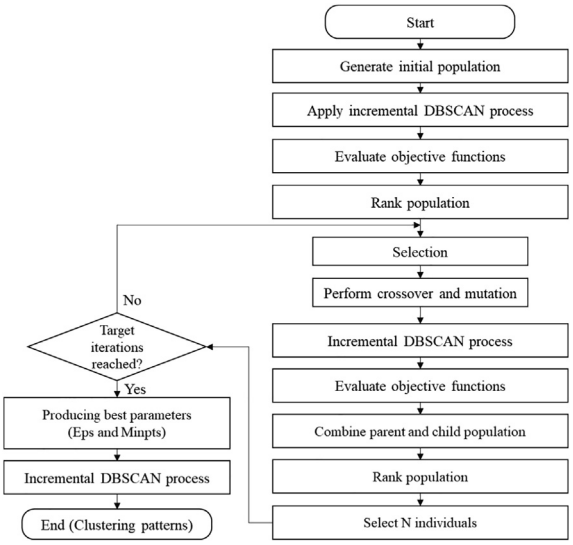

Fig. 3. The Data flow diagram outlining the operation of NODC.

4. *Rank population:* The initial population has been sorted based on the objective values to generate several non-dominated fronts. According to the notion of dominance, the solutions have been separated into K non-dominated fronts, $F = \{F_1, F_2 ... F_k\}$. Each front was comprised of candidate solutions that are non-dominated by any candidate solution of any subsequent front. We've allocated a fitness value (rank) to each member in each front. So, solutions in the F_1 have higher ranks than those of the F_2.

   Furthermore, the Crowding Distance (CD) for each member in each front has been calculated as follows:
   For each front F_k, n is the count of individuals.

- Initialize the distance of every individual to zero, $F_k(d_j) = 0$, where j denotes the j^{th} individual in front F_k.
- For each objective m

   -Sort the individuals in front F_k based on objective m, $I = sort(f_k, m)$.
   -Assign an infinite distance value to boundary solutions, $I(d_1) = I(d_n) = \infty$.
   -For $p = 2$ to $(n-1)$, set $I(d_p) = I(d_p) + (I(p+1).m - I(p-1).m/(f_m^{max} - f_m^{min})$
   Where, $I(p).m$ shows the value of the mth objective function of the p th individual in I, and f_m^{min} and f_m^{max} are the minimum and maximum values for objective m.

5. *Selection:* According to the rank and crowding distance, we've chosen the parent population using binary tournament selection. Thus, we've randomly chosen the first two solutions from the population. After that, if the solutions have different fronts, the one with a lower rank has been selected, or else the one with a higher crowding distance. Fig. 4 presents a summary of the choosing process.
6. *Crossover and Mutation:* Then, we've done the Simulated Binary Crossover (SBX) [43] and polynomial mutation [44] to produce offspring population from the parent. The crossover operator exchanges information among two chromosomes for producing two childeren. The SBX creates two children, $c_1(i)$ and $c_2(i)$, form two parent solutions, $p_1(i)$ and $p_2(i)$, as follows:

$$x < y \ \ if \ \ x_{rank} < y_{rank}$$
$$if \ x_{rank} = y_{rank} \ \text{then} \ x_{CD} > y_{CD}$$

Fig. 4. The binary tournament selection.

$$c_1(i) = 0.5[(1+\beta)p_1(i) + (1-\beta)p_2(i)]$$
$$c_2(i) = 0.5[(1-\beta)p_1(i) + (1+\beta)p_2(i)] \quad (6)$$

with a spread factor β defined by (8),

$$\beta = \begin{cases} (2u)^{\frac{1}{\eta+1}}, if u \ll 0.5 \\ (\frac{1}{(2(1-u))})^{\frac{1}{\eta+1}} Otherwise \end{cases} \quad (7)$$

where u shows a random number in the interval $[0, 1]$and η is a non-negative real number.

7. I*ncremental DBSCAN process, and Evaluate objective functions:* The produced population has been passed to the incremental DBSCAN algorithm to generate clustering results. The fitness values of clustering results have been calculated by the chosen fitness functions.
8. *Combine parent and child population, and Rank population:* Then, the two populations of parent and offspring have been combined and sorted based on non-domination and crowding distance comparison. So, we've chosen the top-ranked solutions and added them to the subsequent generation.
9. *Termination Condition:* The maximum number of iterations has been determined to make sure of the termination of the iterative procedure. The loop will be ended when the pre-defined maximum count of repetitions is touched. Lastly, the best values for Eps and Minpts parameters have been returned and applied to incremental DBSCAN function to generate the ideal clustering result.

5. Experimental setup and results

In this section, we've performed some experiments to assess the performance of the NODC method to solve the plan recommendation problem. These experiments have been described in Section 5.4. In Section 5.1, we've outlined the datasets used. Moreover, standardization rules and representation schema used to convert the queries into vectors have been discussed in Section 5.2. After that, the environment used to conduct the simulations has been described in Section 5.3.

5.1. Dataset

In this paper, we've conducted the experiments to validate the presented approach using the dataset of the APB1 benchmark [45]. The star diagram of this benchmark has one fact table, Actvars (33,323,400 tuples), and four dimension tables, prodlevel (9,900 tuples), Custlevel (990 tuples), Timelevel (24 tuples), and Chanlevel (10 tuples).

We've used the APB1 benchmark query templates to produce 3515 queries in ten sets with different structures, including selection queries, and selection/join queries. Moreover, different values have been assigned to the bind variables. Table 3 presents the nature of the SQL logs.

When an SQL statement is executed in Oracle, a hash value is generated for its execution plan. First, we've classified each query log manually using the queries plan' hash value, giving us the actual numbers of clusters and queries per cluster. As shown in Table 3, dataset S1 contains 95 simple queries having different execution plans, in three classes.

5.2. Preprocessing

As stated in Section 3.1, there are two main steps in the query representation phase. We've overviewed the query representation phase in two steps_ query standardization and feature weighting. The query standardization and feature weighting steps are described in Sections 5.2.1, and 5.2.2, respectively.

5.2.1. Query standardization

Feature space is a collection of features used to describe data. The clustering algorithms operate on a feature space to group data points into different clusters. Our aim, in this step, was to represent the textual queries in a feature space. Henceforth, the queries' texts must be tokenized to split their text into words. The words' frequency has been considered as the queries features for clustering.

Nevertheless, we can meaningfully enhance the clustering quality by rewriting the queries to standardize certain SQL features. First, we need to parse the queries' texts to remove the string constants, number constants, tables and columns aliases,

Table 3
The summary of the datasets.

Features	Dataset Name	No. of Classes	No. of Individuals
Selection	S1	3	95
	S2	7	235
	S3	10	389
	S4	12	481
	S5	14	593
Selection/join, from, group-by and order-by	SJ1	4	60
	SJ2	7	199
	SJ3	10	389
	SJ4	12	481
	SJ5	14	593

syntactic sugaring, database namespaces, and standardizing nested query predicates [22]. As shown in Table 4, we have used many different standardization rules to enhance the clustering quality.

The JSQL parser has parsed the SQL statements and provided the capability of syntactic analysis and manipulation of SQL statements. Furthermore, the parser has tried to tokenize the query into the elementary form. We have used a customized JSQL parser library to manipulate the SQL statements and qualified each token with the SQL group, including select, from, where, group-by, and order-by.

5.2.2. Feature weighting

The TF method has been developed to assess the significance of a word in a given document. The TF method measures the frequency of a certain word in the document. In this paper, we've weighted the generated tokens (features) based on their frequency to balance the most common or unusually-rare tokens. It is shown as $tf(f, q)$, the count of times the feature f appears in query q.

5.3. Simulation environment

The tests have been performed on a PC with the subsequent specifications:
CPU: Intel Core i7, 2 GHz
RAM: 6 Gigabytes
OS: Windows 10 Enterprise
Software: NetBeans 8.2

Table 4
The standardization rules for queries expressions standardization [22,23].

Rule	Raw SQL query	Transformed SQL query
Remove constants (string and number)	**SELECT** id, name **From** user **WHERE** id = '1' and name **like** 'ali%'	**Select** id, name **From** user **WHERE** id = ? and name **like** ?
Remove aliases	**SELECT** name AS id **FROM** user	**SELECT** name **FROM** user
Rewrite equivalent statements	x **BETWEEN** (a,b) x **IN** (a,b, ...) isnull (x,y)	a ≤ x **AND** x ≤ b x = a **OR** x = b **OR** ... **CASE WHEN** x is null **THEN** y END
EXISTS, IN, ANY and ALL operators' normalization	x **IN** (**SELECT** y ...) x < **ANY** (**SELECT** y ...) x < **ALL** (**SELECT** y ...)	**EXISTS** (**SELECT** * ...**WHERE** x = y) **EXISTS** (**SELECT** * ...**WHERE** x < y) **NOT EXISTS** (**SELECT** * ...**WHERE** x ≥ y)
DNF normalization	**SELECT** name **FROM** user **WHERE** name = 'Oliver' **AND** (id = 1 **OR** id = 3)	**SELECT** name **FROM** user **WHERE** name = 'Oliver' ∧ (id = 1 ∨ id = 3)
SQL operator precedence (*, /, -, +, NOT, AND, OR)	**SELECT** name, 12*salary + 100 **FROM** employees	**SELECT** name, (12*salary) + 100 **FROM** employees **FROM**
Flattening FROM clause subqueries	**SELECT** c1 **FROM** t1 **WHERE** c1 **IN** (**SELECT** c1 **FROM** t2)	**SELECT** t1.c1 **FROM** t1, t2 **WHERE** t1.c1 = t2.c1
Nested-query de-correlation (Converts some EXISTS predicates into joins for SELECT DISTINCT or a duplicate-insensitive aggregate parent queries)	**SELECT** ... **FROM** R **WHERE** **EXISTS** (**SELECT** ... **FROM** S **WHERE** q)	**SELECT** ... **FROM** R, (**SELECT** ... **FROM** S) **WHERE** q
OR-UNION transformation	**SELECT** ... **WHERE** q **OR** p **OR** ...	**SELECT** ... **WHERE** q **UNION SELECT** ... **WHERE** p **UNION** ...
Union Pull-Out (Push selection predicates down into the union)	**SELECT** * **FROM** A **UNION** **SELECT** * **FROM** B	**SELECT** * **FROM** (A **UNION** B)

5.4. Results

The experiments have been used in query clustering on several query input sets with different structures. Given a set of queries categorized based on their plan' hash values, we've aimed at understanding how well different algorithms can match the queries with similar plan hash values, and discriminate the queries having dissimilar plan hash values. Each algorithm has been evaluated based on how well it aligns with the actual cluster labels.

We've compared the functionality of the introduced unsupervised NODC algorithm with those of the incremental DBSCAN and K-means for several query datasets. For the experiments, we have used the following NODC parameter settings: The population size, crossover rate, mutation rate, and the maximum count of generations have been set at 50, 0.9, 0.01, and 50, respectively. Moreover, MinPts and Eps decision variables have been set to random values (0–20). The values of K for the K-means have been specified based on the given count of clusters for each dataset.

Fig. 5 shows the Pareto optimal solutions of our introduced method for datasets S1 and S2. Each point in the Pareto optimal front of Fig. 5 indicates a non-dominated solution. It is essential to identify a single answer from the group of non-dominated ones obtained from the final generation to analyze the results. Hence, in the proposed algorithm, we have used the Adjusted Rand index to choose a solo solution from the Pareto optimal front. We've computed the ARI values for all the partitioning solutions of the final Pareto front. Table 5 shows the solutions with the maximum value of ARI. The ideal outcome of NODC matches with ARI = 0.7392 with three obtained clusters for dataset S1.

The experiments on the algorithms have been repeated several times to obtain the desired values. To assess the functionality of unsupervised NODC with those of the incremental DBSCAN and K-means regarding the accuracy, for every one of the datasets, the amounts of the Adjusted Rand index for the ideal clustering results have been applied (Table 5). The maximum values of ARI imply better clustering results. We've determined the number of obtained clusters and the best input parameters by the introduced method for different datasets. Table 5 shows them.

As shown in Table 5, for all the considered datasets, the Adjusted Rand index values show that NODC achieves more accurate results than the existing incremental DBSCAN algorithms. Our experimental results on various query logs have shown that using NSGA-II as a parameter-tuning tool, the clustering performance of incremental DBSCAN can be significantly improved. Indeed, for most of the query logs, NODC produces clustering results competitive to or even slightly better than those of the K-means algorithm. Furthermore, k-means cluster the queries under the assumption that the number of query clusters is known. We've considered the more general and realistic case, in which the number of clusters is unidentified and it's likely to have queries not fitting to any of the clusters. We've found that the k-means algorithm was objectively inferior in all respects.

Overall, the proposed NODC with DBI and DNI can find better results than incremental DBSCAN. In the experiments, the NSGA-II applied in the incremental DBSCAN clustering method performs better than K-means in some cases (S2-S4 and SJ1-SJ2). On the other hand, as shown in Table 5, it is more difficult for the NODC performs a right clustering for complex queries, because the ARI present poor values when the complexity of queries increases. Based on all these results, it can be concluded that the proposed method is far from perfect and further work is needed for selection/join datasets.

5.4.1. Statistical analysis

We have also performed Welch's *t*-test [46] at a 5% level of significance to check the statistical significance of the results. In our *t*-test, we've compared two groups. One includes the list of Adjusted Rand index values produced by our algorithm. The other group includes the list of Adjusted Rand index values generated by other methods. The null hypothesis is that no meaningful difference exists among the median values of the two sets, and the alternative is that a meaningful change exists between median values of the two sets. If the p-value associated with the *t*-test is less than 0.05, then the null hypothesis is rejected. Table 6 shows the calculated p values to compare the functionality of the proposed NODC with that of the incre-

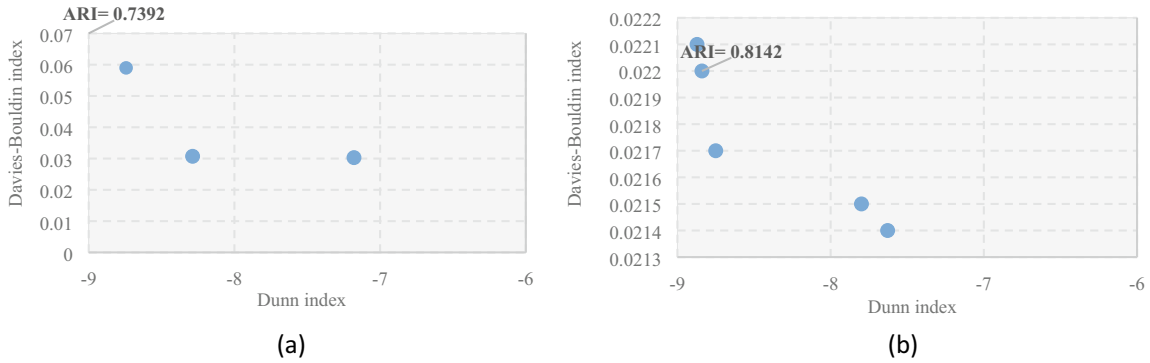

Fig. 5. A set of solutions produced after the application of the proposed NODC algorithm on query logs. (a) Dataset S1. (b) Dataset S2.

Table 5
The comparison of the clustering algorithms using the Adjusted Rand index.

Dataset	NODC			K-means	Incremental DBSCAN	
	No. of clusters	ARI (best)	Best Parameters	ARI (best)	ARI (best)	Best Parameters
S1	3	0.7392	(4,0.05)	**0.9070**	0.6997	(8,0.03)
S2	7	**0.8142**	(4,0.02)	0.6932	0.7504	(16,0.1)
S3	10	**0.8726**	(6,0.02)	0.8233	0.6956	(16,0.08)
S4	12	**0.7814**	(10,0.05)	0.7794	0.7348	(16,0.12)
S5	14	0.6452	(20,0.04)	**0.7239**	0.7011	(16,0.12)
SJ1	4	**0.7882**	(4,0.16)	0.5927	0.4905	(8,0.06)
SJ2	7	**0.7978**	(7,0.11)	0.7872	0.7361	(8,0.06)
SJ3	8	0.6800	(9,0.10)	**0.7523**	0.4859	(8,0.09)
SJ4	14	0.6247	(7,0.04)	**0.7090**	0.6084	(8,0.06)
SJ5	16	0.5200	(7,0.03)	**0.6962**	0.5062	(8,0.09)

Table 6
p values associated with the *t*-test.

Dataset	NODC	K-means	Incremental DBSCAN
S1	0.1117	7.122e−07	0.05341
S2	0.05304	0.002093	0.7745
S3	0.005815	0.4756	0.0001269
S4	0.0589	0.09636	1.72e−09
S5	4.456e−05	0.009728	0.3902
SJ1	0.0001185	0.5147	0.00598
SJ2	0.02499	0.2021	2.442e−06
SJ3	0.5063	0.006379	0.0001204
SJ4	0.06726	7.091e−05	0.1693
SJ5	0.0711	3.169e−06	0.09201

mental DBSCAN and K-means regarding Adjusted Rand index values in Table 5. Table 6 indicates the p values, obviously supporting the outcomes of Table 5.

5.4.2. Threats to validity

Here, we've presented the possible risks for validity:

There are some threats limiting our ability to generalize our findings. There are various factors affecting the quality of clustering methods, including the size of data, shapes of clusters, the noise of data, and the number of input parameters. Our evaluation is based on ten query logs with different structures, limiting the generalization of the conclusions. These query logs have been selected of different structure and size to mitigate this threat. It is important to note that we have not aimed at proposing a general clustering algorithm for all kinds of datasets; instead, we've just claimed that the clusters founded in the target query logs show the feasibility of the proposed algorithm.

The validity of the ground-truth query logs poses another threat to the study. To mitigate this threat, we've used two internal cluster validity indices as objective functions of the proposed method, Dunn index, and Davies–Bouldin index, which measure the quality of a clustering independent of any ground-truth query log.

6. Conclusion

We've introduced a novel automatic multi-objective query plan recommendation approach, called NODC, using the exploration ability of NSGA-II. The proposed method is a combination of NSGA-II and incremental DBSCAN. We've aimed at improving the efficiency of semantic-based query clustering used to access plan recommendations in terms of accuracy. We've investigated two objective functions, both calculating the compression and distinction of clusters, and optimized them simultaneously to enhance the cluster quality. We have standardized the query semantics and used token frequencies for creating the query vectors.

This paper has compared the performance of NODC with that of the existing incremental DBSCAN and K-means when they have been applied on different query datasets. As a result, NODC presents more accurate results for all the considered query logs as compared to the incremental DBSCAN. Also, NODC and K-means algorithms produce comparable results.

In our future work, we will explore the efficiency of various evolutionary and clustering algorithms and other query representations for plan recommendation. Also, more experiments are required to study the performance of the above approach in solving some other real-life problems. We would also like to investigate the effect of using various internal and external clustering validation indices in the proposed clustering framework.

CRediT authorship contribution statement

Elham Azhir: Conceptualization, Methodology, Software, Data curation, Writing - original draft, Writing - review & editing. **Nima Jafari Navimipour:** Supervision, Conceptualization, Writing - original draft, Validation, Writing - review & editing. **Mehdi Hosseinzadeh:** Supervision, Conceptualization, Writing - original draft, Writing - review & editing. **Arash Sharifi:** Conceptualization, Writing - review & editing. **Aso Darwesh:** Writing - review & editing.

Declaration of Competing 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.

References

[1] A. Naseri, N.J. Navimipour, A new agent-based method for QoS-aware cloud service composition using particle swarm optimization algorithm, J. Ambient Intell. Hum. Comput. (2018) 1–14.
[2] Buyya, R., J. Broberg, and A.M. Goscinski, Cloud computing: Principles and paradigms. Vol. 87. 2010: John Wiley & Sons.
[3] A. Vakili, N.J. Navimipour, Comprehensive and systematic review of the service composition mechanisms in the cloud environments, J. Network Comput. Appl. 81 (2017) 24–36.
[4] Á.L. García, E.F. del Castillo, P.O. Fernández, Standards for enabling heterogeneous IaaS cloud federations, Computer Stand. Interfaces 47 (2016) 19–23.
[5] Rahi, S.B., S. Bisui, and S.C. Misra, Identifying critical challenges in the adoption of cloud-based services. International Journal of Communication Systems, 2017. 30(12): p. e3261-n/a.
[6] W. Al Shehri, Cloud database database as a service, Int. J. Database Manage. Syst. 5 (2) (2013) 1.
[7] F. Mehak et al, Security Aspects of Database-as-a-Service (DBaaS) in Cloud Computing, in: Cloud Computing, Springer, 2014, pp. 297–324.
[8] Nachiappan, R., et al., Cloud storage reliability for Big Data applications: A state of the art survey. Journal of Network and Computer Applications, 2017. 97(Supplement C): p. 35-47.
[9] Singh, V. Multi-objective Parametric Query Optimization for Distributed Database Systems. in Proceedings of Fifth International Conference on Soft Computing for Problem Solving. 2016. Springer.
[10] T. Neumann, S. Michel, Algebraic query optimization for distributed top-k queries, Informatik-Forschung und Entwicklung 21 (3–4) (2007) 197–211.
[11] Han, M., J. Youn, and S.-g. Lee. Efficient query processing on distributed stream processing engine. in Proceedings of the 11th International Conference on Ubiquitous Information Management and Communication. 2017. ACM.
[12] B.A. Milani, N.J. Navimipour, A systematic literature review of the data replication techniques in the cloud environments, Big Data Res. (2017).
[13] Azhir, E., et al., Deterministic and non-deterministic query optimization techniques in the cloud computing. Concurrency and Computation: Practice and Experience, 2019: p. e5240.
[14] E. Azhir et al, Query optimization mechanisms in the cloud environments: a systematic study, Int. J. Commun Syst 32 (8) (2019) e3940.
[15] A. Ghosh et al, Plan selection based on query clustering. in VLDB'02: Proceedings of the 28th International Conference on Very Large Databases, Elsevier, 2002.
[16] J. Zahir, A. El Qadi, A recommendation system for execution plans using machine learning, Math. Comput. Appl. 21 (2) (2016) 23.
[17] Zahir, J., A. El Qadi, and S. Mouline. Access plan recommendation: A clustering based approach using queries similarity. in Complex Systems (WCCS), 2014 Second World Conference on. 2014. IEEE.
[18] J.A. Hartigan, Clustering Algorithms, Wiley, New York, 1975.
[19] Sarda, P. and J.R. Haritsa. Green query optimization: Taming query optimization overheads through plan recycling. in Proceedings of the Thirtieth international conference on Very large data bases-Volume 30. 2004. VLDB Endowment.
[20] J. Aligon et al, Similarity measures for OLAP sessions, Knowl. Inf. Syst. 39 (2) (2014) 463–489.
[21] K. Aouiche, P.-E. Jouve, J. Darmont, Clustering-based materialized view selection in data warehouses, East European Conference on Advances in Databases and Information Systems, Springer, 2006.
[22] G. Kul et al, Similarity metrics for sql query clustering, IEEE Trans. Knowl. Data Eng. 12 (2018) 2408–2420.
[23] V.H. Makiyama, J. Raddick, R.D. Santos, Text mining applied to SQL queries: a case study for the SDSS SkyServer, SIMBig (2015).
[24] C. Guan, K.K.F. Yuen, F. Coenen, Particle swarm Optimized density-based clustering and classification: supervised and unsupervised learning approaches, Swarm Evol. Comput. 44 (2019) 876–896.
[25] N. Saini, S. Saha, P. Bhattacharyya, Automatic scientific document clustering using self-organized multi-objective differential evolution, Cognitive Computation 11 (2) (2019) 271–293.
[26] J. Handl, J. Knowles, An evolutionary approach to multiobjective clustering, IEEE Trans. Evol. Comput. 11 (1) (2007) 56–76.
[27] S. Saha, S. Bandyopadhyay, A symmetry based multiobjective clustering technique for automatic evolution of clusters, Pattern Recogn. 43 (3) (2010) 738–751.
[28] S. Bandyopadhyay, U. Maulik, A. Mukhopadhyay, Multiobjective genetic clustering for pixel classification in remote sensing imagery, IEEE Trans. Geosci. Remote Sens. 45 (5) (2007) 1506–1511.
[29] P. Dutta, S. Saha, Fusion of expression values and protein interaction information using multi-objective optimization for improving gene clustering, Comput. Biol. Med. 89 (2017) 31–43.
[30] S. Bandyopadhyay, S. Saha, A new principal axis based line symmetry measurement and its application to clustering, International Conference on Neural Information Processing, Springer, 2008.
[31] Jain, A.K. and R.C. Dubes, Algorithms for clustering data. Englewood Cliffs: Prentice Hall, 1988, 1988.
[32] S. Bandyopadhyay, U. Maulik, Genetic clustering for automatic evolution of clusters and application to image classification, Pattern Recogn. 35 (6) (2002) 1197–1208.
[33] S. Chatterjee, A. Mukhopadhyay, Clustering ensemble: a multiobjective genetic algorithm based approach, Procedia Technol. 10 (2013) 443–449.
[34] S.F. Galán, Comparative evaluation of region query strategies for DBSCAN clustering, Inf. Sci. 502 (2019) 76–90.
[35] Ester, M., et al. A density-based algorithm for discovering clusters in large spatial databases with noise. in Proceeding of 2nd Int. Conf. On Knowledge Discovery and Data Mining. 1996. Portland.
[36] Ester, M., et al. Incremental clustering for mining in a data warehousing environment. in Proceedings of 24th VLDB Conference. 1998. New York, USA: Citeseer.
[37] J.C. Dunn, A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters, J. Cyber. 3 (3) (1973) 32–57.
[38] D.L. Davies, D.W. Bouldin, A cluster separation measure, IEEE Trans. Pattern Anal. Mach. Intell. 1 (2) (1979) 224–227.
[39] L. Hubert, P. Arabie, Comparing partitions, J. Classif. 2 (1) (1985) 193–218.
[40] W.M. Rand, Objective criteria for the evaluation of clustering methods, J. Am. Stat. Assoc. 66 (336) (1971) 846–850.
[41] O. Arbelaitz et al, An extensive comparative study of cluster validity indices, Pattern Recogn. 46 (1) (2013) 243–256.

[42] J. Xie et al, A new internal index based on density core for clustering validation, Inf. Sci. 506 (2020) 346–365.
[43] J.-H. Yi et al, Behavior of crossover operators in NSGA-III for large-scale optimization problems, Inf. Sci. 509 (2020) 470–487.
[44] K. Deb, R.B. Agrawal, Simulated binary crossover for continuous search space, Complex systems 9 (2) (1995) 115–148.
[45] Council, O., Apb-1 olap benchmark, release ii. 1998.
[46] B.L. Welch, The generalization ofstudent's' problem when several different population variances are involved, Biometrika 34 (1/2) (1947) 28–35.