


RESEARCH

Open Access



On K-means clustering-based approach for DDBSs design

Ali A. Amer* 

*Correspondence:
aliaaa2004@yahoo.com;
aliaaa2004@gmail.com
College of Science, Computer
Science Department, Taiz
University, Taiz 6803, Yemen

Abstract

In Distributed Database Systems (DDBS), communication costs and response time have long been open-ended challenges. Nevertheless, when DDBS is carefully designed, the desired reduction in communication costs will be achieved. Data fragmentation (data clustering) and data allocation are on popularity as the prime strategies in constant use to design DDBS. Based on these strategies, on the other hand, several design techniques have been presented in the literature to improve DDBS performance using either empirical results or data statistics, making most of them imperfect or invalid particularly, at least, at the initial stage of DDBSs design. In this paper, thus, a heuristic k-means approach for vertical fragmentation and allocation is introduced. This approach is primarily focused on DDBS design at the initial stage. Many techniques are being joined in a step to make a promising work. A brief yet effective experimental study, on both artificially-created and real datasets, has been conducted to demonstrate the optimality of the proposed approach, comparing with its counterparts, as the obtained results has been shown encouraging.

Keywords: DDBS, Data allocation, Data replication, Query clustering, K-means algorithm

Introduction

During the last years, a significant progress has been made in DDBS design. Mostly, this progress has been concentrated on fragmentation and allocation techniques due to their critical impact on DDBS productivity, particularly in relational databases. On one extreme, the fragmentation process (horizontal, vertical or mixed) describes how each relation could be split into different data fragments (smaller relations). On the other extreme, data allocation seeks to promote DDBS performance by placing the properly-broken fragments into their relative sites in which they are most needed. Consequently, when data fragmentation and allocation are well performed, DDBS throughput is substantially optimized. This optimization is often met by promoting performance through minimizing the irrelevant access for data (i.e. transmission minimization), which is already stored in different sites, as distributed query under processing. Briefly, paper's contributions are summarized as follows:

1. Developing K-means clustering based vertical fragmentation method in the relational database context. Unlike most of earlier techniques, this work does not need data statistics, empirical results, mid-term predicates, affinity, attributes affinity matrix or even query frequency matrix to perform data fragmentation and allocation, at least, at the initial stage. What is just taken: the considered queries as the most frequently used, and the Query Usage Matrix (QUM) in which each point refers to whether a specific site releases the relevant query or not. In fact, this step marks the novelty and creativity of the proposed work as it is essentially committed to the initial stage of DDBS design.
2. Proposing a novel algorithm for the fragments refinement process. This algorithm produces the non-overlapping schemes out of the overlapping schemes generated from the clustering process.
3. Compiling many techniques into the proposed approach to make an effective work so the data locality maximization and communication costs reduction are met. Among these techniques are: K-means-based process for query clustering, schemes refinement process, fragmentation evaluation technique, site clustering process, and data allocation and replication. Consequently, a competitive DDBS design approach is expected to meet the acquired performance of DDBS in either a static or dynamic environment.
4. Finally, the proposed work of this paper has been evaluated, on both artificially-created and real datasets, against two counterparts in DDBS design literature. Experimental results illustrated a significant performance for the proposed work comparing with its peers.

The rest of this paper is structured as follows. In “[Related work](#)” section, the earlier relevant studies of DDBS design are explored. The proposed methodology including the approach’s heuristics and architecture, fragmentation and allocation cost model and clustering process, is elegantly given in “[Proposed methodology](#)” section). Results and discussion are presented in “[Results](#)” and “[Discussion](#)” section. Finally, “[Conclusions and future work](#)” draws the conclusions along with future work.

Related work

According to the literature, the fragmentation techniques are horizontal, vertical and hybrid. While fragmentation is often done independently of data allocation. The data allocation process, however, is always heavily contingent on the fragmentation process. In other words, it is done on the assumption that fragmentation is antecedent. In its turn, vertical fragmentation (VF) grabs DDBS researchers’ attention. In fact, this is back to the positive effects vertical fragmentation has on DDBS rendering. As first of its kind in the DDBS field [1], came as a fine-grained taxonomy on DDBS design. The basic issues examined in this taxonomy were data fragmentation and allocation. Data replication was significantly scrutinized as well. This taxonomy was comprehensively analyzed that all these issues were considered to classify and analyze a big number of the previous works. The driving aim of this taxonomy was to take the observation of earlier works’ drawbacks to increase the likelihood of producing more productive methods to improve DDBS performance. The decrease of transmission costs (TC),

involving the costs of communication, has been the objective for which most of DDBS design works have been striving to meet. However, to meet this objective, the DDBS design work has to maximize the data locality and minimize the access for remote data significantly. It was observed in [1] that most of the studied works failed to draw a clear “unified or consensually agreed-upon” definition for TC as a metric for DDBS performance which is considered a huge shortcoming.

A simultaneous relational-model-based vertical fragmentation and allocation technique was proposed along with a cost model in [2]. Communication costs minimization was the prime motivation of the work. However, when authors performed fragmentation, there has been no involvement for any cost model to evaluate the resulting fragmentation solution(s) due to the fact that only one single solution is set to be produced regardless of its quality. Moreover, authors did not consider the site clustering or distinguish given for the reading and write queries. Finally, the replication strategy had not been addressed as well. While in [3], the DAP problem was sought to be solved through the hybrid solution using the algorithm of differential evolution (DE) along with the technique of variable neighborhood search (VNS). The key intention revolved around promoting DE rendering through the operators of selection and crossover. The given work was experimentally seen effective as it explored the search space by DE along with the technique of neighborhood search. On the other extreme, hypothesizing the existence of interlocking horizontally-fragmented data, the data replication problem (DRP) was deeply dealt with in [4] as an integer linear problem. Hence, data replication was addressed as the problem of optimization for the sake of keeping copies of fragments and sites at a minimum. On the same line [5], developed a particle swarm optimization-based method (PSO) to reduce TC costs. The aim was to use PSO to find a solution for the data allocation problem (DAP) only.

While most of the earlier work used an attribute affinity as the key element to fragment data, there have been many clustering-based fragmentation techniques. In [6], a heuristic technique for vertical fragmentation and data allocation was elegantly evolved. The work was the first of its kind that sought to incorporate several techniques in one single work with the aim of maximizing DDBS performance. Extensive evaluation on several data allocation scenarios was performed to assert the proposed work's effectiveness. As a follow-up optimization [7], came to add a new data allocation scenario to the work of [6]. This scenario was shown to be non-efficient in some cases in which update queries grow steadily, though. Moreover [8], came to further enhance DDBS performance by proposing a new approach based on an aggregated similarity measure used to cluster queries. The authors proposed a greedy algorithm to solve the data allocation problem. A comprehensive evaluation was promised to be made with [6] to assert the proposed work's superiority. No evaluation was given yet, though. On the same line [9], evolved an enhanced technique to design DDBS. This work was also evaluated against [6] and shown to behave slightly better in most cases. Moreover, an acceptable experimental study was conducted to prove the concept. Following the same of pattern of [6, 9, 10] came to propose an enhanced scheme for vertical fragmentation and allocation. The aim of the work was to improve DDBS performance through finding an influential solution for the round-trip response time minimization. Comparing with the relative works,

author claimed that their proposed schemes decreased the round-trip response time by 23%.

On the other hand [11], worked on finding a vertical fragmentation method. An algorithm based on differential bond energy (DBE) was proposed. Based on the global affinity measure (GAM), a comparison was made for the proposed algorithm with a classical bond energy algorithm (BEA) in terms of performance. The experimental results attested that developed DBE was suitable for the high dimensional problems with having a high GAM value comparing with BEA on several datasets. For improving DDBS performance [12], proposed a non-redundant dynamic fragment allocation approach. This approach amended the read and write data volume to Threshold Time Volume and Distance Constraints Algorithm. Fragments were being allocated based on access patterns made to each fragment. On the same page, an enhanced approach to split data at the initial stage of DDBS design and then assign data at runtime over the cloud environment was proposed in [13]. The data replication scenario was adopted in a way that allowed DDBSMs to work simultaneously to meet the client's orders. In [14], a method to boost the performance and reliability of distributed systems was presented. The method sought to find the optimum placement in network nodes of the information and technological reserve (ITR). This method used information and software redundancy in the form of distributed copies of ITR. It was mathematically shown able to boost the reaction of DDPS. It established that after setting ITR copies and allocating them over the network, these copies served as an information base for DDPS when requests of users are being processed.

Finally, in the same line for solving the DAP problem, [15] came to present a greedy based algorithm called ASGOP to tackle DAP. The data allocation was treated as an optimization problem and the cost model solved using the knapsack algorithm. In each time, each fragment was not allocated to the intended site unless it was guaranteed that this site is the prime container based on its transmission costs. Two data allocation scenarios were addressed, the replication-based and non-replication based. The experiment results shown that ASGOP outperformed its counterparts in terms of data allocation due to its being greedy.

Proposed methodology

Requirements

To perform data fragmentation and allocation, the next information requirement is needed:

- A set of relations of Database (R_1, R_2, \dots, R_r), where (r) represents the number of considered relations.
- For each R_i ($A_1; A_2; \dots; A_n$): is the data schema, R , which consists of (N) attributes.
- A data query set running against R_i , Q ($Q_1; Q_2; \dots; Q_q$), where (q) is the number of running queries.
- Query Access Matrix: each QAM value refers to whether query Q_k is released from site S_j or not, be given by DBA. Where (k and j) are just indices for query and site respectively.

Motivations to prefer K-means over hierarchical clustering (HC)

Hierarchical clustering (and its variations) is efficiently used in applications where points/patterns are at the range of tens and even hundreds and it has long proven effective [6, 7, 9]. Nevertheless, as the size of data sets is increasing, HC is being infeasible due to its non-linear time that grows exponentially with dataset size and the growing demands of space required. As a matter of fact, it is not an easy task to visualize a dendrogram for, let say, “1000” patterns (and not to mention the complexity involved when patterns in thousands). To accurately examine the number of patterns in HC, an exponential time is required to perform the task at hand. To sum up, HC (and its variation) does not scale up perfectly in the context of large-scale applications that would involve thousands and even millions of patterns. On the other hand, naturally, K-means is a hard clustering algorithm that is adequately applied often on large datasets and it has long proven efficient in literature [16]. Moreover, there has been a dominant property features k-means algorithm which is its ability to successively minimize the sum of patterns squared deviations implicitly (called in literature, squared-error criterion) from the center in each cluster. Formally speaking, assuming there has been cluster X_i and \mathcal{M}_i is its center, then the criterion function that sought to be minimized by k-means is drawn in Eq. (1):

$$\sum_{I=1}^{\#clusters} \sum_{j=1}^{\#X} (X(j) - \mathcal{M}(i))^T (X(j) - \mathcal{M}(i)) \quad (1)$$

On the other hand, as a crucial drawback, the k-means algorithm does not secure the globally-optimal fragments due to two basic causes: (1) the poor selection for initial seeds (centers), and (2) The traditional k-means algorithm which leverages the “winner-take-all” technique as pattern given to only and only the winning cluster to eventually generate the hard fragment. So, to tackle this shortcoming and enhance the results, K-mean is being applied according to the next mechanism: initial seeds were chosen heuristically. The widely-known heuristic is to pick up the initial “k” centres. These centers are supposed to be as far away from each other as possible. In literature, this heuristic worked well practically. In practice, picking the pair of patterns which are greatly dissimilar in the set as the initial seeds leads to decrease the dependency on the initialization process. Experimentally speaking, this mechanism serves the interest of the proposed wok substantially in terms of finding a competitive solution for DDBS design efficiency. This efficiency is obviously being reflected by the better results that come in favour of K-means-based work comparing with its counterparts. It is worth indicating that there have been other alternatives for centroid selection in literature like Random generation, Buckshot approach and ranking technique [17, 18]. Some of these strategies are also tested in our work, but no one draws better results and serves the major interest of paper like the “greatly dissimilar” strategy that is already being leveraged.

Finally, the time complexity is $O(NKID)$, where N is the number of considered patterns, K is the number of generated clusters, I is the iterations number and D is the dimensionality. The space requirement is $O(KD)$. This complexity is lower than HC complexity making K-means largely appealing to be used for DDBS design.

Heuristics

Our work is a three-fold approach as drawn in Fig. 1, and detailed as follows

The phase (1): the query set, most-repeatedly-run queries, was identified. For each query in the set, the contained attributes replaced by binary value (0, 1) distinguishing its presence or absence in the query. In doing so for all queries, Attribute Access Matrix (AAM) was constructed so that its rows represented queries and column represented attributes (see Table 4). This matrix was used with the help of the hamming distance metric [19] to find the different values among patterns that were already being recognized. These difference values would be drawn into a matrix called Query Difference Matrix (QDM). Then, using QDM as initial input, the K-means clustering process was being activated as presented in the “Clustering methodology” section.

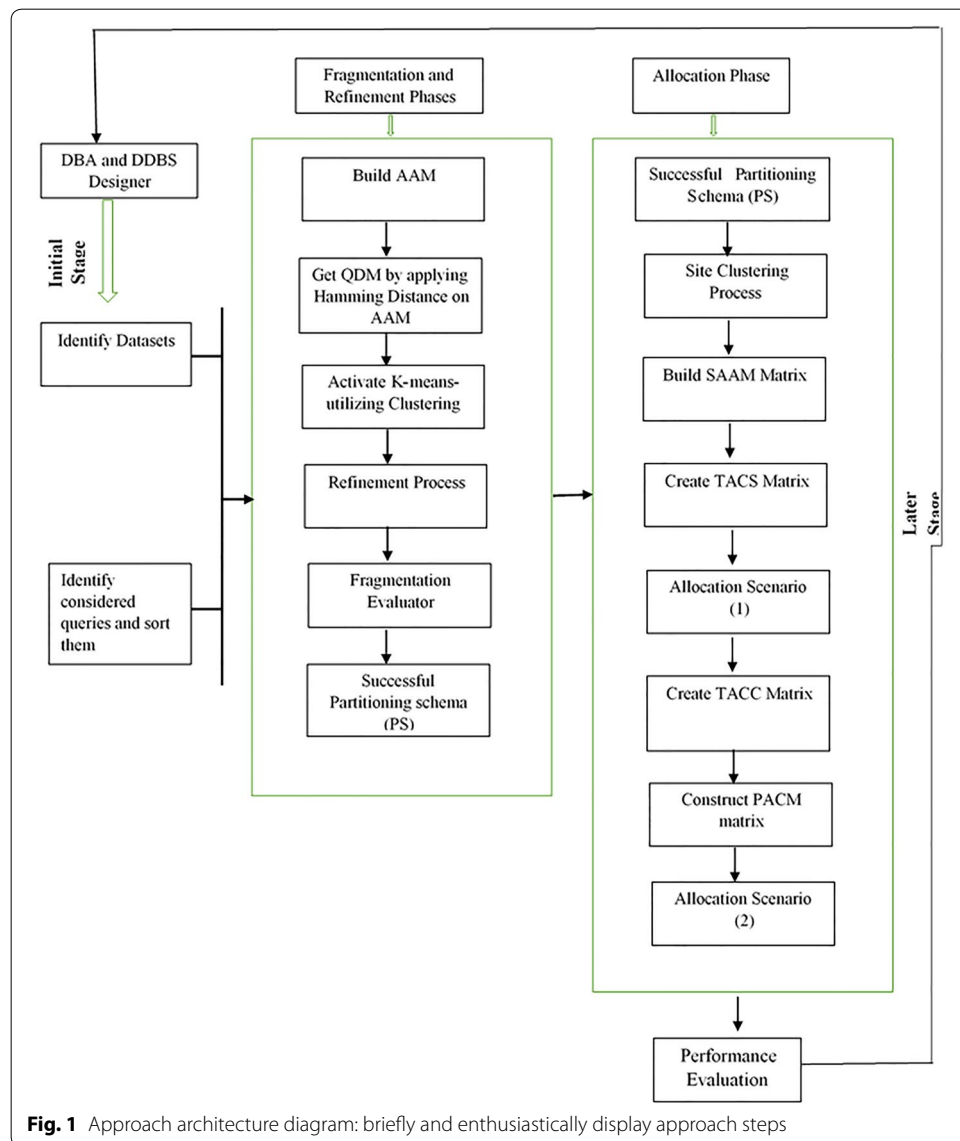


Fig. 1 Approach architecture diagram: briefly and enthusiastically display approach steps

The phase (2): the refinement process was drawn to guarantee securing the non-overlapping fragments. All fragmentation combination has to be created for each cluster. The input parameters for this process were all over-lapping (PSs) out of the fragmentation phase. So, the expected results of this process were bound to be the non-overlapping schemes. However, when query clusters were examined, it happened that attribute(s) went missing in some clusters. Such attribute(s) missed through the clustering process due to the loss of some queries in each cluster chiefly as clusters are aggressively grown. Therefore, since the prime goal of this approach was to keep as high percentage of binding “connection” among attributes regarding their relevant original queries as possible, this attribute(s) would be added according to the proposed function called the affinity function ($aff_func(\text{Partition}, \text{attribute/attributes})$, Eq. (2)). For each cluster, this function would check the strength of the connection of attribute(s) with all partitions of each Partitioning Schema (PS) individually based on Eqs. (3) and (4) at the same time. Then, whenever happened that certain partition(s) had the max connection with attribute(s) at the question, it was the prime candidate container to store them. Nevertheless, if an attribute(s) has been equally required by “N” partition in PS, it was added “N” time(s). In each time, attribute(s) was being added to each partition making a new PS in each addition.

A clear manifestation is drawn in Table 15 so that p3 and p4 were derived from the original PS3. This connection was calculated based on the attribute’s appearance, in each partition of each cluster, concerning their relevant original queries. In the sense that any partition yielded a higher connection with the relevant missing attribute(s), it was the candidate partition to store it. However, if an attribute(s) had a zero connection, it was created as a new partition on its own inside the underlying PS. The proposed Function of affinity was presented in Eq. (2) as follows;

$$func - aff \left(partition, \frac{A}{As} \right) = \begin{cases} add(A, partition), & con = T \\ CNP \left(\frac{A}{As} \right), & con = F \end{cases} \quad (2)$$

where P stands for concerned partition, A is a shortcut for Attribute/attributes, the con is a logical factor to distinguish whether there had been the connection or not, and CNP stands for creating a new partition. T and F stand for true and false flags. By strictly following this procedure, the possibilities of getting PSs with a minimum of the remote access costs and maximum of data locality has been increased. In other words, among all generated combinations, only schemes of a high percentage of connection rate would be taken into fragmentation evaluator. However, to select those schemes, an optimality measure (OM) “parameter” was proposed. Actually, OM was a criterion to reflect the recorded correlation rate of access costs between each PS and all their relevant considered queries, Eqs. (3) and (4).

$$Optimality\ Measure\ (OM) = 1 - Correlation\ Rate \quad (3)$$

$$Correlation\ Rate = \frac{\sum_{i=1}^{ps} \sum_{j=1}^q Cr_{++}}{Q} \quad (4)$$

where Cr is an integer counter. The Correlation rate, in its turn, was used to reflect the maximum remote access for all queries to reach that relevant PS. In the sense that PS which gave higher remote access, it was neglected. Whenever OM was bigger the remote access was in fact minimized and local access was maximized. Consequently, a proportion of technique design objective was met. Finally, the ultimate decision to exclude or include PS into FE was accomplished as per Eq. (5). Figure 2 depicts the steps of the process professionally.

$$Decision\ Making(PS) = \begin{cases} OM \geq 50\%, & \text{Include PS into FE} \\ \text{Otherwise,} & \text{exclude PS from FE} \end{cases} \quad (5)$$

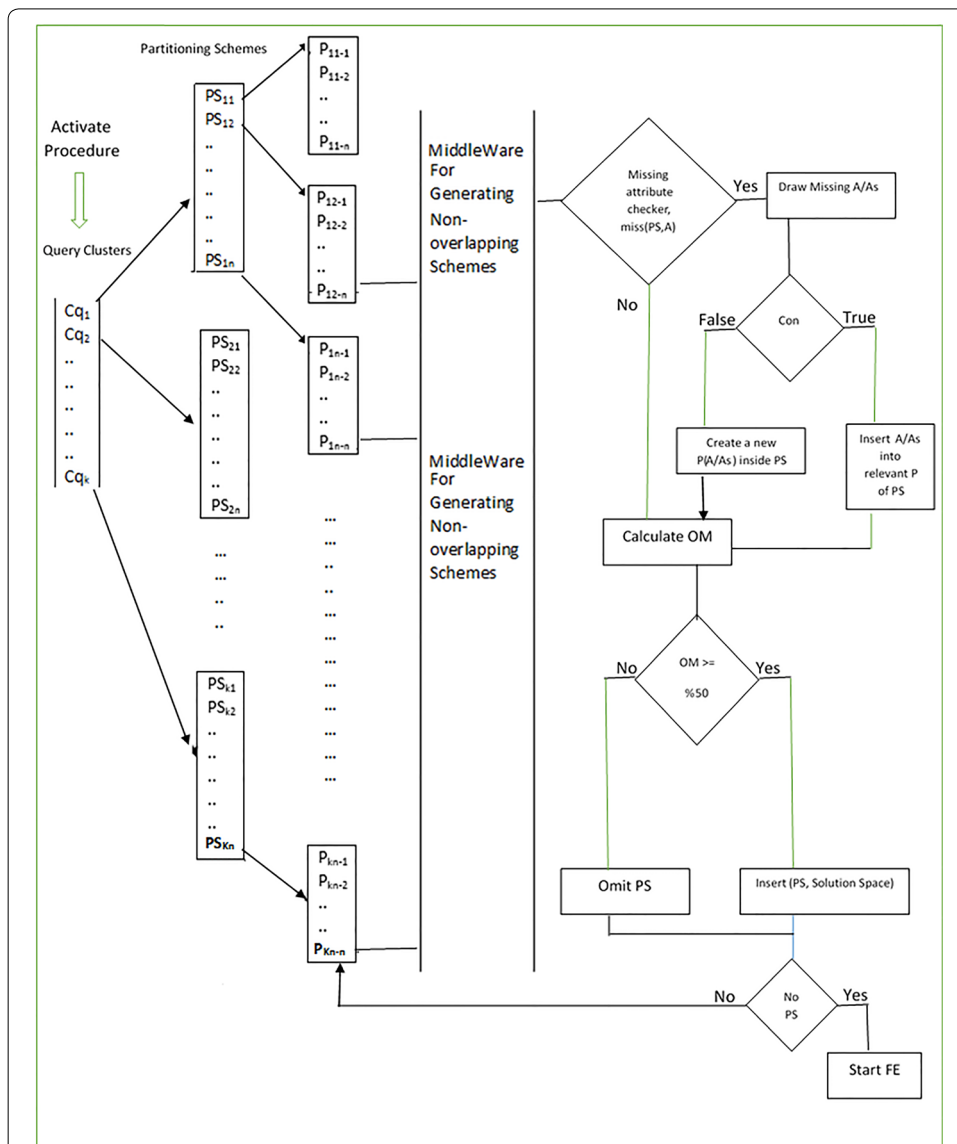


Fig. 2 Refinement procedure diagram: exhibit the steps taken by refinement procedure to produce disjoint schemes

Finally, for this phase, if it happened to have schema duplication, only one copy was kept. This phase keeps only one copy of each schema if it happened to have schema duplication.

The phase (3): the evaluation step was done using the presented fragmentation evaluator (FE) [6]. FE uses two measures to assess schemes which were: the relevant remote access and irrelevant local access. Basically, according to [6], the successful partitioning schema is that of the lowest value of FE. This schema would be considered for the allocation process, as a result.

Fragmentation and allocation cost model

Based on the running queries, the matrix of attribute access (AAM) is being concluded. In this matrix, each aam_{ij} signifies the approaching of A_i by Q_k , with the assumption that the query usage matrix (QUM) was already supported by DBA [6] so that each qum_{ij} indicates the site S_j , from which Q_k was launched. Hence, using these requirements, the process of data fragmentation and allocation is done based on the next functions:

$$\text{Similarity}(Q_{k1}, Q_{k2}) = \sum_{k1=1}^q \sum_{k2=1}^q (1 - \text{difference}(P(Q_{k1}), P(Q_{k2}))), \tag{6}$$

where “Sim”, “dif” and $P(Q)$ stands for similarity, the difference between queries, and the numerical pattern of Q_k respectively. Using Eq. (6), query difference matrix (QDM) is being constructed as seen in Eq. (7). Each qdm_{k1k2} represents the similarity calculated between each query pair.

$$QDM_{k1k2} = \sum_{k1=1}^q \sum_{k2=1}^q \text{Similarity}(Q_{k1}, Q_{k2}) \tag{7}$$

Proposed allocation and replication model

Suppose we have a “K” query set, $Q = \{Q_1, Q_2, \dots, Q_k\}$ reach N attributes $A = \{A_1, A_2, \dots, A_n\}$. These queries were tied into CN several clusters $\{C_{q1}, C_{q2}, \dots, C_{q_{cn}}\}$. Query clusters were placed into a set of M sites $S = \{S_1, S_2, \dots, S_m\}$. Sites, in their turn, were clustered into CM clusters $Cs = \{Cs_1, Cs_2, \dots, Cs_{cm}\}$, and $F = \{F_1, F_2, \dots, F_m\}$ be the disjointed fragments already produced out of the process of query clustering. Then, the proposed model of data allocation pursues to optimally distribute each fragment (F) over cluster set, Cs, and then over sites of each cluster.

Allocation scenarios

First scenario: first Phase (replication adopted): each (F) was replicated over all clusters. It is worth indicating that the replication concept of the proposed work has adopted the replication principles given in [4] to replicate the data when it is needed.

Second scenario: First Phase (non-replication adopted): each (F) was assigned to C of the maximum cost of access.

Both scenarios: Second Phase 2 (no replication inside each cluster): The total cost to each S_j , to reach all A(s) of F_i was the assignment controller. So, the total cost of access of each S_j (NACS_{ij}) has to be precisely computed. NACS matrix was created

using both site attribute access (SAAM), which was computed in Eq. (8), and communication costs (CMS) matrices, see Eq. (9). In NACS, the site of maximum cost for the intended fragment was selected as the candidate site to store the fragment at the question.

Allocation cost functions

$$SAAM = \sum_{k=1}^q \sum_{j=1}^m \sum_{i=1}^n AAM_{ik} * QUM_{ji} \tag{8}$$

$$NACS = \sum_{j1=1}^m \sum_{i=1}^n \sum_{j=1}^m SAAM_{ij1} * CMS_{ji} \tag{9}$$

$$NACC = \sum_{k=1}^{cn} \sum_{i=1}^n \sum_{j=1}^m SAAM_{ij1} + 1 \tag{10}$$

$$PACM = \sum_{k1=1}^{cn} \sum_{i=1}^n \sum_{k=1}^{cn} NACC_{ik1} * CCM_{ki} \tag{11}$$

where AAM, QUM, CMS, and CMS stands for attribute access matrix, query usage matrix, communication costs between sites and communication costs between clusters respectively. Equation (8) built SAAM which used later to build the NACS matrix along with using Eq. (9). Equation (10) accumulated the access costs for each attribute over its relevant clusters with respect to sites contained in each cluster. Lastly, Eq. (11) drew the last step of fragments allocation over each site cluster in the second scenario. Last but not least, the following constraints were maintained throughout the data allocation process. It is worth indicating that this cost model (including equations) has been solved using integer linear programming (ILP) as the objective function of the whole work is to maximally minimize transmission costs.

$$\sum_{k=1}^{cq} Size(F_k) \leq Capacity(S_j), \quad \forall j = 1, \dots, m. \tag{12}$$

$$LA_i \leq \sum_{i=1}^n X_{ij} \leq UA_i, \quad \forall j = 1, \dots, m. \tag{13}$$

$$X_{ij} \in (0, 1), \tag{14}$$

Constraint (12) ensured that the net size of fragments that already assigned to one site must not overpass the site capacity, as shown in Table 1. On the other hand, constraint (13) guaranteed that the number of assigned attributes was between the lower limit of the allowed attribute (LAL) and the upper limit (UAL). Finally, constraint (13) was the

Table 1 Site constraints

Site	S1	S2	S3	S4	S5	S6
Capacity (KB)	10,000	15,000	9000	12,000	9000	10,000
LAL	1	1	1	1	1	1
UAL	12	14	11	7	10	14

decision variable on the binary form. Table 1 describes these constraints as capacity measured in Megabyte, LAL, and UAL.

Fragmentation evaluator (FE)

This work used FE to evaluate schemes. FE has two metrics: relevant remote access and irrelevant local access. The first metric calculates the total costs of remote access for attributes that already exist at the remote site. The second metric associated with attributes that are processed locally. The first metric of FE is computed in Eq. (15), which gave the local access costs:

$$E_{nf}^2 = \sum_{i=1}^{nf} \sum_{q=1}^Q \left[TFQ * |Al_{ik}| \left(1 - \frac{|Al_{ik}|}{|NA_{il}|} \right) \right] \tag{15}$$

where TQF is the total frequency of query (how many times query is being released over network sites) that accesses data, $|Al_{ij}|$ is the number of attributes in F_i that are locally approached by Q_k , NA is the number of attributes of targeted relation. On the other hand, Eq. (16) provided the second term of FE as it computes the ratio of remote attributes being accessed:

$$E_{ad}^2 = \sum_{i=1}^{nf} \sum_{q=1}^Q \left[\sum_{j=1}^m TFQ_{qi}^j * |Ad_{ik}| * \left(\frac{|Ad_{ik}|}{|NA_{iqk}|} \right) \right] \tag{16}$$

where $|AD|$ is the attributes number in F_i which was remotely reached with regard to F_j , by Q_k . Hence, FE was given by its two metrics as follows;

$$FE = E_{nf}^2 + E_{Ad}^2 \tag{17}$$

Clustering methodology

The clustering process is iteratively applied upon query difference matrix (QDM) until satisfying either one of two conditions. While the first condition is that each cluster should reach its stable state. The second condition is placed to keep repeating the process until each pattern takes its turn of being centroid. After that, for the second condition, the most stable cluster would be selected at the final step. To group similar patterns in each loop, the least difference value (LDV) is utilized (see Eqs. (5) and (6), respectively). In this process, each pair of patterns “queries” will be compared in the bottom-up method until all query clusters constituted in a box named a solution space. This space will eventually hold all partitioning schemes combinations.

As a matter of fact, the rationale behind proposing the solution space is to contain all partitioning schemes (PSs) that surpass OM threshold (see Eq. (5)) since each PS represents an optimal solution with a certain percentage. Therefore, containing all possible combinations, in solution space, should contribute to finding optimal schema at a success rate of roughly 100%. This scientific fact is being demonstrated in the performance evaluation section.

K-means clustering process

Technically, K-means clustering process which was presented in [16] and being utilized with slight modifications as follows:

- 1) Determining the number of clusters (CN) is made using Eq. (18).

$$CN = \left\lceil \sqrt{\frac{n}{2}} \right\rceil \quad (18)$$

where n is the number of queries has to be clustered and CN is an even number.

- 2) Choosing members of least dissimilarity values among all numerical patterns of original queries to be a centroids of the already-drawn empty-filled clusters $CQ_1, \dots, CQ_{n/2}$.
- 3) For each cluster, numerical patterns are pulled hierarchically [20] based on the proposed difference value metric. This step shall be repeatedly performed until no member excluded. If pattern (P_i) has the same difference value with more than one centroid, P_i would be added to the cluster with which it is mostly being tied as per the calculated average cost of access for each cluster. However, if P_i still has difficulty joining the relative cluster, P_i could be added to either one.
- 4) After getting all the patterns involved in the first loop, the results are to be kept in the solution space. Then, the clustering process starts over again by releasing all patterns of clusters and randomly choosing new centroids, other than those of the first loop, for all clusters.
- 5) Step 3 and step 4 will be iteratively repeated until each cluster reach its stable state or each pattern successfully takes its turn to be centroid. All clusters in each loop would be kept in solution space. It is worth pointing that the stability state is satisfied when the cluster reaches a “non-change” state. Thus, whenever a cluster is being stable, the process would isolate and maintain this cluster as “a stable cluster”, with no additional processing.

Site clustering algorithm

Initialization: Given a set of sites M as input, communication costs matrix between sites, let the initial clusters initiated using LDV value which was basically proposed in [21].

Loop: for any new site, do the following:

1. Calculate communication costs between the new site and each site cluster using average communication costs. Average costs will be used as a decisive membership for each site with respect to clusters under consideration.
2. The site cluster of the lowest average cost is bound to be the candidate container for the site at hand.
3. If more than one candidate container recorded, the container of the lowest distance with the targeted site is the primary and sole container.
4. Repeat steps (1–3) until all sites are clustered successfully.

Results

This work has been implemented in C++ programming language using a processor of 2.2 GHz Intel (R) Dual-Core (TM) i3CPU with 4 GB of main memory and 80-GB hard drive. It is worth indicating that all requirements like queries and the query frequencies are hypothesized to be collected from the workload of DDBS. Within the relational DB context, the proposed approach was implemented in the virtual fully-connected six-site network. To conduct experiments, the “Car” dataset was artificially proposed based on the description given in Table 2. This dataset has six attributes and initially stacked with 400 data rows. To make an external evaluation with [6, 9], five more problems were addressed on the same dataset while maximizing the number of queries to reach 300 queries and the number of records to reach 4000 records. In the first three problems, the retrieval queries were occupying the larger area of considered queries. However, in the last two problems, the update queries were assumed to have a larger percentage. For computation simplicity, the attributes: Car-no, Car-model, Engine-id, Speed-Limit, Manufacturer, and workshop-id referred to as A_1 , A_2 , A_3 , A_4 , A_5 , and A_6 respectively.

For the first problem, it was hypothesized that only eight queries as the most-occasionally operating against the “Car” database (see Table 2). If the attribute appeared in the query, it means that it is being accessed attribute by the intended query. Using this hypothesis on queries, the matrix of attribute access was established. In its turn, each aam_{kj} , (see Table 3), indicates whether attribute A_j is accessed or not by the corresponding query.

Furthermore, only QUM (Table 4) was needed as it was assumedly given by DB administrator [6]. Each qum_{kj} refers to whether query Q_k was issued from the underlying site S_j or not.

Table 2 Car database description

Attributes	Symbol	Type	Length (bytes)
Car-no	A1	Nominal	4
Car-model	A2	Categorical	30
Engine-id	A3	Categorical	4
Speed-Limit	A4	Numerical	3
Manufacturer	A5	Categorical	5
Workshop-id	A6	Nominal	4

Table 3 Attribute Access Matrix (AAM)

Query/attribute	A ₁	A ₂	A ₃	A ₄	A ₅	A ₆
Q ₁	1	1	0	0	1	1
Q ₂	0	0	1	0	1	0
Q ₃	0	1	0	1	1	0
Q ₄	1	0	1	0	0	1
Q ₅	1	1	0	0	1	0
Q ₆	0	0	1	1	0	1
Q ₇	0	1	0	0	0	1
Q ₈	1	0	1	0	1	1

Table 4 Query Usage Matrix (QUM)

Query/site	S ₁	S ₂	S ₃	S ₄	S ₅	S ₆	SFM
Q ₁	1	0	0	0	1	1	3
Q ₂	1	1	0	0	1	0	3
Q ₃	0	0	1	1	0	0	2
Q ₄	0	0	0	1	0	1	2
Q ₅	0	1	0	0	0	1	2
Q ₆	0	1	1	1	0	0	3
Q ₇	1	0	0	1	0	0	2
Q ₈	0	1	1	0	1	1	4

Table 5 Query Difference Matrix (QDM)

Query	Q ₁	Q ₂	Q ₃	Q ₄	Q ₅	Q ₆	Q ₇	Q ₈
Q ₁	0	4	3	3	1	5	2	2
Q ₂		0	3	3	3	3	4	2
Q ₃			0	6	2	4	3	5
Q ₄				0	4	2	3	1
Q ₅					0	6	3	3
Q ₆						0	3	3
Q ₇							0	4
Q ₈								0

Then, the hamming distance is used in congruence with AAM to build Query Difference Matrix (QDM) in Table 5, (see Eqs. (2) and (3)). On the other hand, each qdm_{ij} gives the value of the difference between the numerical pattern pairs of queries.

K-means-utilizing clustering process

Using the algorithm described in query clustering (see “Clustering methodology” section) along with QDM matrix, the first results were shown in Tables 6, 7, 8, 9, 10.

It is worth indicating that query membership was calculated as per the least difference value measure.

Table 6 First loop

Centeroid/query	Q ₂	Q ₃	Q ₄	Q ₆	Q ₇	Q ₈
Q ₁	4	3	3	5	2	2
Q ₅	3	4	2	6	3	3

Table 7 Member’s affiliation (first loop)

Centeroid/query	Q ₂	Q ₃	Q ₄	Q ₆	Q ₇	Q ₈
Q ₁	0	0	1	1	1	1
Q ₅	1	1	0	0	0	0

Table 8 Second loop

Centeroid/query	Q ₁	Q ₂	Q ₄	Q ₅	Q ₆	Q ₈
Q ₇	2	4	3	3	3	4
Q ₃	3	3	6	2	4	5

Table 9 Member’s affiliation (second loop)

Centeroid/query	Q ₁	Q ₂	Q ₄	Q ₅	Q ₆	Q ₈
Q ₇	1	0	1	0	1	1
Q ₃	0	1	0	1	0	0

Table 10 Solution space

Cluster	Members
CQ ₁	Q ₂ , Q ₃ and Q ₅
CQ ₂	Q ₁ , Q ₄ , Q ₆ , Q ₇ and Q ₈

Second loop (Q₇, Q₂)

Release members of clusters and select new centroids so that the newest were those which have the least difference values with the oldest. Thus, Q₇ chosen as centroid for the first cluster and Q₃ as the centroid of the second.

From the second loop (affiliation Tables 7 and 9) it is obvious that the obtained clusters are in the stable case since they are a complete replica of those of the first loop. Finally, the net results of this process would be drawn in the solution space (Table 10).

Refinement process

The refinement process was elegantly drawn in Table 11 according to the procedure that was earlier given in “Heuristic” section.

From Table 11, we can see the impact of the correlation rate on reducing the number of partitions yielded from each partitioning scheme. For example, P1 is yielded from the cluster (CQ₂₃₅) but A₆ is missing. So, instead of randomly generate several combinations

Table 11 Refinement process

Cluster	Cluster members (overlapping clusters)	P#	Partitioning schemes	Missing partition (added)	Schemes combinations Non-overlapping PS	PS state	Optimality measure (OM)	Decision making	
CQ235	Q2(A3,A5)	P1	(A3,A5) (A2,A4)	A6	(A3,A5) (A2,A4)	Kept	37.5%	Exclude	
	Q3(A2,A4,A5)		(A1)		(A1,A6)				
	Q5(A1,A2,A5)	P2	(A2,A4,A5)	(A1,A3)	(A2,A4,A5)	Kept	62.5%	Include	
			(A1,A3)		(A1,A3,A6)				
	P3	(A1,A2,A5)	(A3,A4)	(A1,A2,A5)	Kept	62.5%	Include		
		(A3,A4)		(A3,A4,A6)					
	P4			(A1,A2,A5,A6)	Kept	75%	Include		
CQ14678	Q1(A1,A2,A3,A6)	P5	(A1,A2,A5,A6)	None	Replica of P4	Deleted	–	–	
	Q4(A1,A3,A6)	P6	(A3,A4)		Replica of P2	Deleted	–	–	
	Q6(A3,A4,A6)		(A1,A3,A6)		Replica of P3	Removed	–	–	
	Q7(A2,A6)	P7	(A2,A4,A5)		(A3,A4,A6)	(A1,A2,A5)	Kept	37.5%	Exclude
	Q8(A1,A3,A5,A6)		(A3,A4,A6)		(A1,A2,A5)				
	P8	(A2,A6)	(A1,A3,A5)	(A2,A6)	Kept	37.5%	Exclude		
	P9	(A1,A3,A5,A6)	(A4)	(A1,A3,A5,A6)	Kept	75%	Include		
			(A2,A4)	(A2,A4)					

Table 12 Fragmentation evaluation

PS number	PS	E_L^2	E_M^2	PE value
1	(A2,A4,A5) (A1,A3,A6)	56	97	153
2	(A1,A2,A5) (A3,A4,A6)	58	87	145
3	(A1,A2,A5,A6) (A3,A4)	58	82	140
4	(A1,A3,A5,A6) (A2,A4)	52	61	113

which are three, four and also four combinations (in total it is “11” combinations) for P_1 , P_2 , and P_3 respectively, as done in [6, 9], the proposed technique measures the affinity of A_6 with each partition in P_1 using Eq. (3). As per calculation, A_6 is found to have a higher rate with the partition (A_1), so it is being joined with this partition. Consequently, the solution space is being reduced significantly and the number of schemes that would be passed into FE will much less compared with techniques used in [6, 9]. This also interprets the statistics drawn in Tables 22, 23, 24 which come in favour of proposed work.

Fragmentation evaluation

The next step was to pass fragments through the fragmentation evaluator to be examined so that the optimal schema was to be selected, Table 12 and Fig. 3.

From Table 12 and Fig. 3, it was clear that the successful schema is the PS_4 since it was of the lowest FE value. This schema was evaluated as per Eqs. (12, 13 and 14), NA means that no access possible to that attribute, as follows:

For local access:

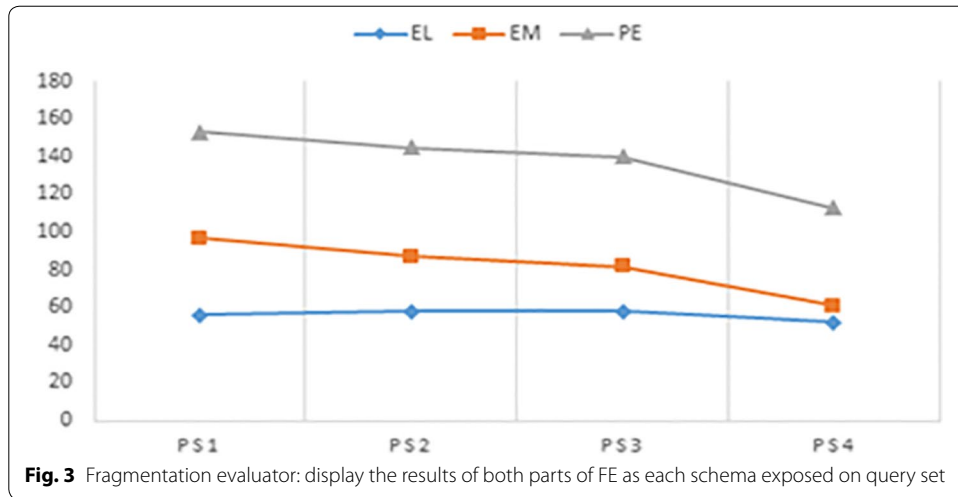


Table 13 Site Access Attribute Matrix (SAAM)

Site	Attribute					
	A ₁	A ₂	A ₃	A ₄	A ₅	A ₆
S ₁	2	4	2	0	4	4
S ₂	3	2	3	1	5	2
S ₃	1	3	2	4	4	2
S ₄	1	4	4	6	3	5
S ₅	3	1	5	0	5	2
S ₆	9	5	4	0	6	7

$$\text{Cost of F1} + \text{cost of F2} = (7 + 9 + 3 + 3 + 4 + 9 + 3 + 0) + (5 + 0 + 0 + 0 + 2 + 5 + 2 + 0) = 38 + 14 = 52.$$

For remote access:

$$\text{Cost of F1} + \text{cost of F2} = (5 + 0 + 8 + 0 + 1 + 3 + 1 + 0) + (20 + \text{NA} + 4 + \text{NA} + 5 + 12 + 2 + \text{NA}) = 18 + 43 = 61$$

So, the total access cost is: $52 + 61 = 113$

As a result, the query clusters obtained from the clustering process were drawn in the following SQL statements:

DataF₁ = CQ₁: Select A₂, A₄ from Table; Size (DataF1) = 9900 Byte

DataF₂ = CQ₂: Select A₁, A₃, A₅, A₆ from Table; Size (DataF2) = 5100 Byte

Allocation process

Firstly, based on QUM and AAM along with Eq. (5), Site Attribute Access Matrix (SAAM) was constructed. In SAAM, every point of each row describes the total cost for each site S_j, through its relevant queries to reach certain Attribute A_i, Table 13.

Then, by multiplying SAAM with communication cost matrix between sites (Eq. (6)), Net Access Cost Matrix (NACS) was to be constructed as shown in Table 14.

Table 14 Net Access Cost Matrix (NACS)

Site	Attribute					
	A ₁	A ₂	A ₃	A ₄	A ₅	A ₆
S ₁	106	86	98	54	144	96
S ₂	81	98	87	46	126	107
S ₃	91	85	79	25	116	100
S ₄	94	59	94	15	120	77
S ₅	81	101	91	79	112	120
S ₆	49	72	79	54	104	77

Table 15 Scenario (1)—Decision Allocation Matrix

Sites' cluster/ fragment	Sites	Attributes							
		A ₂	A ₄	Total cost (F ₁)	A ₁	A ₃	A ₅	A ₆	Total cost (F ₃)
CS ₁	S ₁	86	54	140	106	98	144	96	444
	S ₅	101	79	180	81	91	112	120	404
CS ₂	S ₂	98	46	144	81	87	126	107	401
	S ₃	85	25	110	91	79	116	100	366
	S ₄	59	15	74	94	94	120	77	385
CS ₃	S ₆	72	54	126	49	79	104	77	309

Table 16 Scenario (1)—final allocation

Fragment/sites' cluster	CS ₂		CS ₃		Cs3	
	S ₁	S ₅	S ₂	S ₃	S ₄	S ₆
F ₁		1	1			1
F ₂	1		1		1	

Table 17 Net Access Cost Cluster Matrix (NACC)

Cluster #	A1	A2	A3	A4	A5	A6
C1	6	9	7	5	13	8
C2	4	6	9	6	8	7
C3	9	5	4	0	6	7

The first Allocation scenario (fragments replicated over cluster of sites):

Phase 1: The first phase was more straight-forward and fragment would be directly allocated to all sites' clusters using replication principal as shown in Table 15.

Phase 2: Data fragment F₁ and F₂ were set to be assigned to sites of each cluster. The Decision Allocation Matrix is drawn in Table 16.

For scenario 1, depending upon the Decision Allocation Matrix, the final allocation for fragments was drawn in Table 16.

The second scenario (no replication adopted over clusters of sites)

Site Attribute Access Matrix (SAAM) would be used as input parameters for this process. The results should be the net access cost each sites' cluster (NACC) which was

Table 18 PACM Matrix

Cluster #	A1	A2	A3	A4	A5	A6
C1	38	40	53	30	52	49
C2	66	65	51	25	89	68
C3	28	42	50	34	58	44

Table 19 Scenario (2)—Decision Making Matrix

Cluster #	A2	A4	Total cost of F1	A1	A3	A5	A6	Total cost of F3
C1	40	30	70	38	53	52	49	192
C2	65	25	90	66	51	89	68	274
C3	42	34	76	28	50	58	44	180

Table 20 Scenario (2)—the final allocation of data fragment

Sites' Cluster/Fragment	Sites	Attributes							
		A ₂	A ₄	Total cost (F ₁)	A ₁	A ₃	A ₅	A ₆	Total cost (F ₃)
CS ₂	S ₂	98	46	144	81	87	126	107	401
	S ₃	85	25	110	91	79	116	100	366

Table 21 Scenario (2)—the final allocation of data fragment

Cluster	Sites	Fragment	
		F1	F2
CS ₂	S ₂	1	1
	S ₃		

needed to reach attributes individually (Table 17). The summation of all queries cost for each cluster to access every attribute A_i was calculated as shown in TACC matrix which was produced as applying Eq. (7) on SAAM.

Then, multiplying TACC matrix by communication cost matrix of clusters (Eq. (8)), the pay of attributes access matrix (PACM), across clusters, would be produced as shown in Table 18.

Phase 1: Fragments are allocated to all clusters of sites using non-replication principal so as to each fragment was to be assigned to the cluster of maximum access cost as shown in Table 19.

Phase 2: Data fragment F₁ and F₂ were set to be assigned to sites of each cluster. The Decision Allocation Matrix was drawn in Tables 20 and 21.

The competition, to have these fragments allocated, had been between all sites of each cluster, Table 20. Each fragment was directly assigned to the site of the highest fragment access cost, Table 21.

Discussion

As mentioned earlier, the prime pursuit of this work is directed solely at the performance optimality of DDBS. So, this technique was designed with the aim of optimally, at least to a large extent, maximizing data locality and minimizing remote access. In the quest to achieve the intended objective, besides fragmenting data precisely, these fragment should be assigned to those sites in where it is intensively accessed. In doing so, the costs of communication and response time are bound to be significantly mitigated. To verify whether this goal was achieved, a simple yet effective assessment has been conducted. The evaluation has been made in two parts. For the first part of the evaluation, five problems have been simulated, namely, 8 queries on a datasets of 300 records; 24, 32 queries on a dataset of 500 records; and 64 and 120 queries on a dataset of 1200 records. As given in “Result” section, the first problem was done exclusively for demonstration purposes. It is limited for queries of retrieval type, though. The consecutive two problems addressed queries of both retrieval and update type. The retrieval queries took a bigger space. On the other hand, the fourth and fifth problems addressed both types while update queries taking in the larger space. For site clustering, on the other hand, the procedure given in [18] has been adopted. The second part of evaluation composes of two sections. Section (1) sought to increase the volume of the artificially-created datasets (up to 4000 records) and the size of query set (up to 300 query) to assert the superiority of proposed k-means based work in terms of performance factors under consideration which are execution time (ET), solution space size, fragmentation evaluator (FE) values, and finally the overall reduction observed in transmission costs (TC). Section (2) of second part, sought to meet the same objectives on real datasets that was retrieved from the machine learning repository [22]. The number of records of this dataset is 48842, and the size is 3.8 MB.

Firstly, in regard to performance element represented in the time (calculated in seconds) taken to perform fragments using both hierarchical clustering and the proposed work of this paper which depends on K-means (Fig. 4). From queries of the first experiment to almost 40 queries of experiment 3, it was obvious that time taken to fragment data using K-means went slightly in parallel with the time taken by hierarchical clustering (HC) with a noticeable increase in HC time. However, as shown from experiment 3 to experiment 5, as query set grew, HC time started to grow exponentially while the time taken by K-means grew slightly and gradually. In short, as the first contribution of this work, k-means based clustering process (proposed work of this paper) initially proved its efficiency over HC based works in [6, 9].

As it is drawn in Fig. 4, there has been a significant gap in time between HC based procedure and K-means based procedure particularly when the query set is growing steadily from 80-160 queries. For problem (1), both works moved at a closer rate. However, starting from the problem (2) till Problem (5), the time gaps between both works started to be steadily and significantly widened. It can be even concluded that the size of the problem (when the number of queries growing in each successive problem) has a substantial contribution in widening the time gap between all techniques under consideration. The bigger the size of the query set, the longer time would be needed to process such set in all works with K-means based work being faster as drawn in Fig. 4. In fact, the recorded gaps which come in favour of K-means

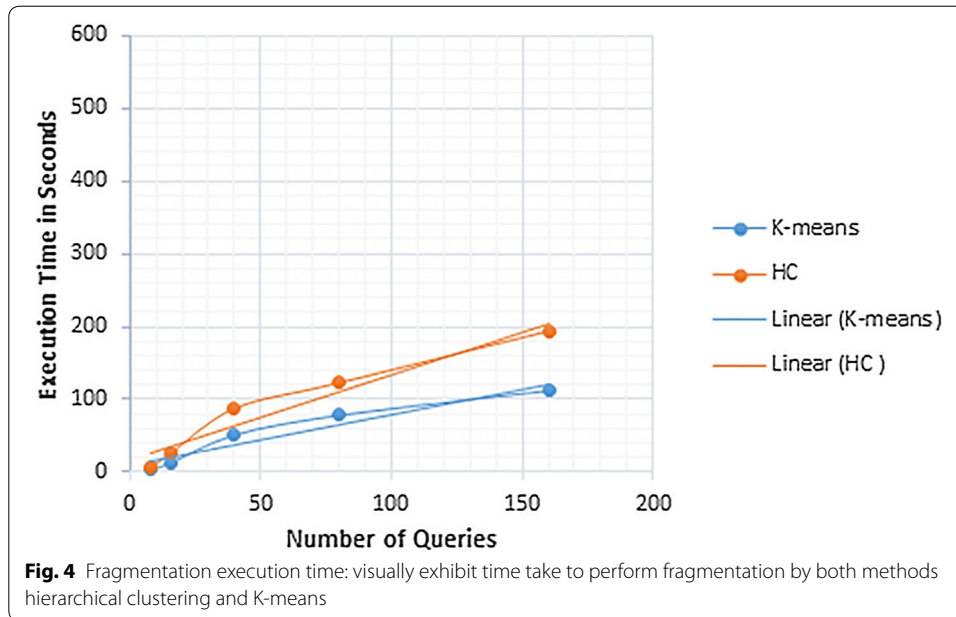


Table 22 Partitioning schemes [6] and [9]

PS number	PS	E_L^2	E_M^2	PE value
1	(A ₁ ,A ₂ ,A ₄ ,A ₅) (A ₃ ,A ₆)	199	292	491
2	(A ₂ ,A ₅) (A ₁ ,A ₃ ,A ₄ ,A ₆)	244	315	559
3	(A ₁ ,A ₂ ,A ₅) (A ₃ ,A ₄ ,A ₆)	234	283	517
4	(A ₂ ,A ₄ ,A ₅) (A ₁ ,A ₃ ,A ₆)	184	266	450
5	(A ₁ ,A ₂ ,A ₄ ,A ₅ ,A ₆) (A ₃)	230	158	388
6	(A ₁ ,A ₂ ,A ₅ ,A ₆) (A ₃ ,A ₄)	222	237	459
7	(A ₂ ,A ₄ ,A ₅ ,A ₆) (A ₁ ,A ₃)	252	307	559

based work come down to some reasons. First, the nature of K-means itself which is faster than hierarchical in finding the partitioning schemes. Second, the proposed filtering procedure in k-means-based approach which sought to reduce the solution space substantially while this process is completed randomly in [6, 9] till all solutions contained. In consequence, as the third reason, FE in the K-means-based approach took much less time to evaluate the non-overlapping schemes comparing with that time taken in HC based approach. That is why K-means based work is faster than HC based works [6, 9].

Secondly, the second contribution was recorded for the performance factor represented in fragmentation evaluator values. These results (values obtained of FE, Tables 22, 23, 24) were accurately compared with results drawn in [6, 9] and clearly shown that the proposed work more effective. The cost reduction of values and cost reduction rate in total, which greatly reached to unexpected rate, were drawn in Table 24 to accentuate proposed work optimality in reducing remote access costs and rising local access costs at the same time.

Table 23 Fragmentation evaluation (proposed work of this paper)

PS number	PS	E_L^2	E_M^2	PE value
1	(A2,A4,A5) (A1,A3,A6)	56	97	153
2	(A1,A2,A5) (A3,A4,A6)	58	87	145
3	(A1,A2,A5,A6) (A3,A4)	58	82	140
4	(A1,A3,A5,A6) (A2,A4)	52	61	113

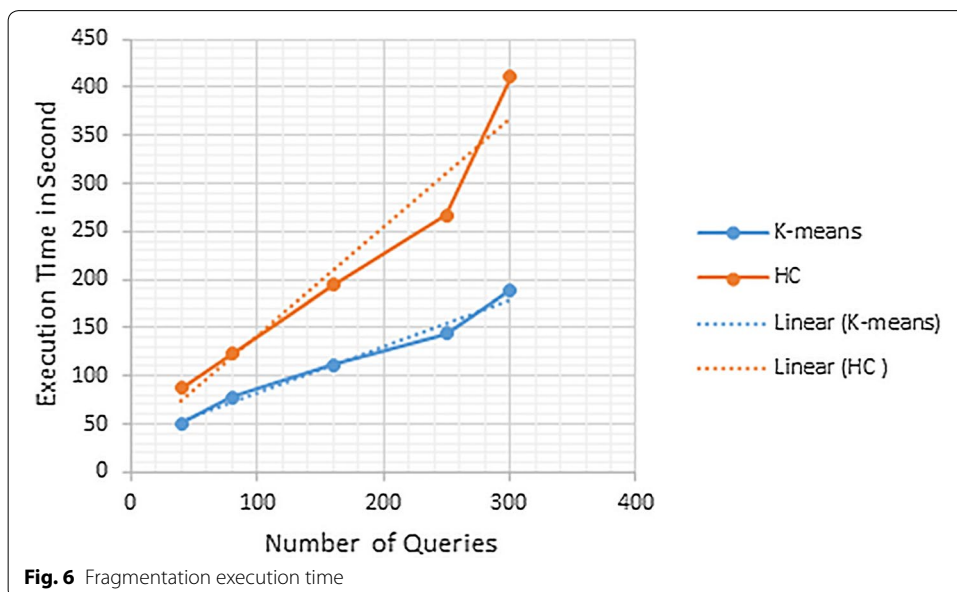
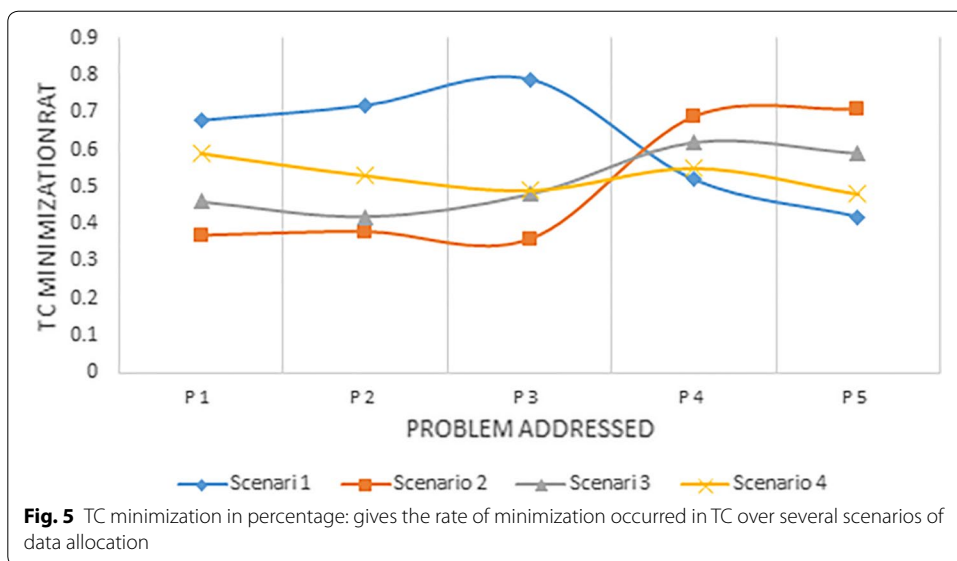
Table 24 FE values of schemes produced in (proposed work, and [6, 9])

Criterion/approach	Experiment#	[6]	[9]	Proposed work
Solution space size	1	7	7	4
Costs reduction (FE)	1	3423/7 = 489	3423/7 = 489	551/4 = 137.75
Costs reduction rate	1	0.22	0.22	0.78
Solution space size	2	9	9	5
Costs reduction (FE)	2	712	712	241
Costs reduction rate	2	0.25	0.25	0.75
Solution space size	3	11	11	6
Costs reduction (FE)	3	1312	1312	489
Costs reduction rate	3	0.27	0.27	0.73
Solution space size	4	14	14	8
Costs reduction (FE)	4	1688	1688	711
Costs reduction rate	4	0.30	0.30	0.70
Solution space size	5	18	18	9
Costs reduction (FE)	5	2132	2132	893
Costs reduction rate	5	0.29	0.29	0.71
Cost reduction rate in Total	1-5	0.27	0.27	0.73

In Table 24, both works [6] and [9] were seen to have the same numbers in terms of solution space size and costs reduction due to the fact that both work used the same HC based procedure to fragment datasets.

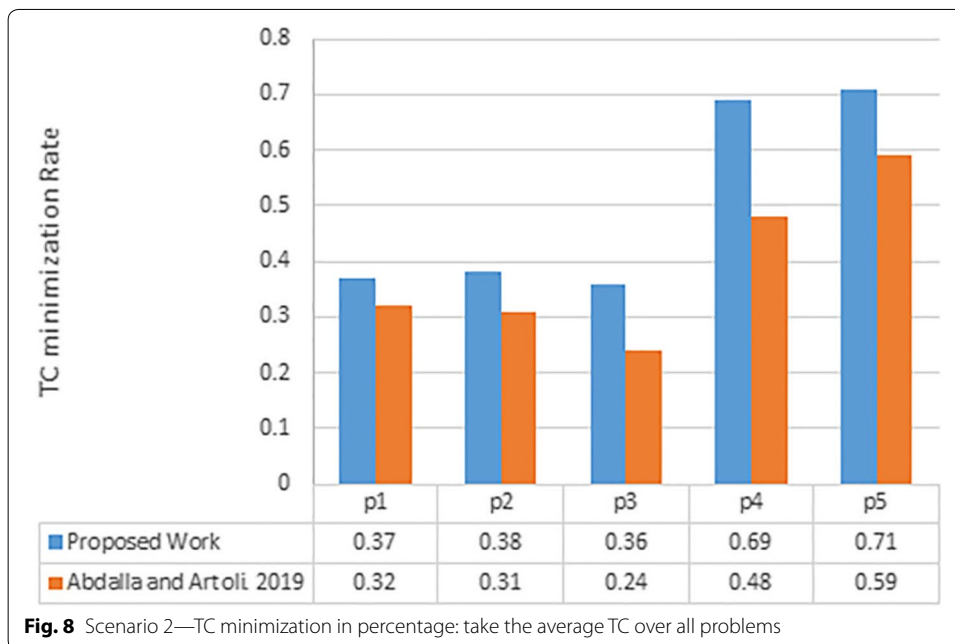
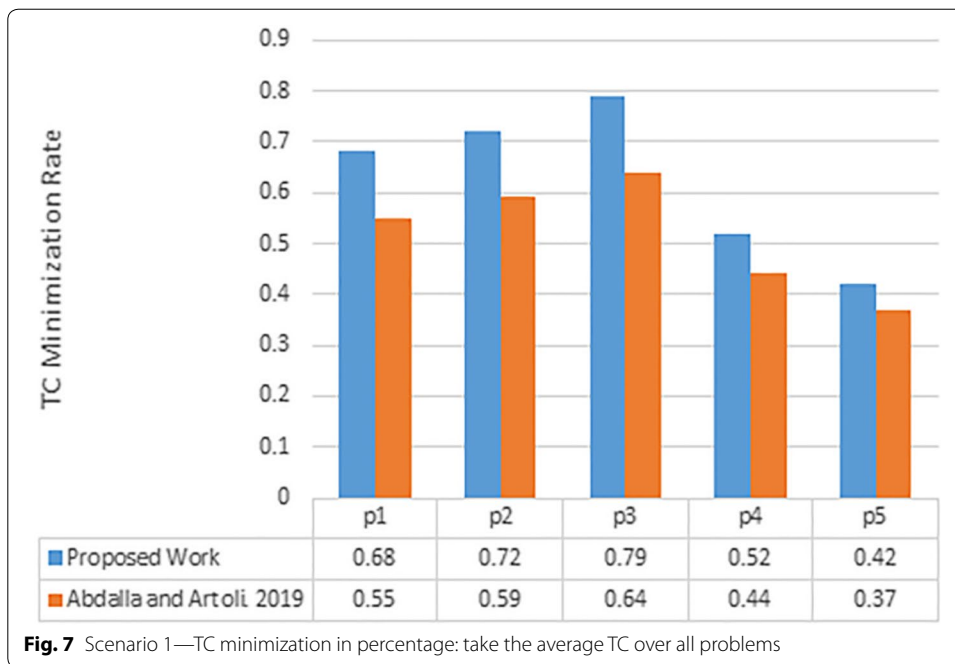
Thirdly, the third contribution has been represented in meeting the desired minimization of TC which was monitored for each problem over all works. Every query, in the considered set, was tested upon “Car” dataset according to four scenarios designed for data allocation: 1) data allocation scenario (1) in which the partial replication adopted; 2) data allocation scenario (2) in which no replication adopted; 3) The non-replicated random allocation; and 4) The random allocation for the whole non-fragmented relation.

Figure 5 clearly induces the great effects the data replication has on TC minimization, particularly when query set of retrieval type were bigger than update. Moreover, four more problems were addressed as follows; 24, 32 queries with dataset of 500 records and 64 and 120 queries along with dataset of 1200 records, respectively. As a result, the results of Fig. 5 showed that scenario (1) was the best option when query of retrieval type occupied bigger space (P1, P2 and P3). On the other hand, the second and third scenarios were seen to be the best option as queries of update type bigger than retrieval type (P4 and P5). Unsurprisingly, the great results obtained with data replication in scenario (1) came with price on TC minimization when update-type



queries were grown. Interestingly, these results came in complete consistent with the theories given in [8] in which replication impact on DDBS performance was thoroughly investigated.

To further assert proposed work optimality in almost all scenarios, the proposed work was experimentally tested against [9] on a dataset of 4000 records and a query set of 300. Five more experiments were conducted to draw the averaged results of all experiments as shown in Figs. 7, 8, 9, 10. Number of queries was varied from 40, 80, 160, 250, and 300. Once again, asserting the results of Fig. 4, Fig. 6 comes to assert that from queries of first experiment to almost 80 queries of experiment 2, it was clear that time taken to fragment data using K-means went in parallel with time taken by



hierarchical clustering (HC) with slight increase in HC time. However, as shown from experiment 3 to experiment 5, as query set grew, HC time started to grow exponentially while time taken by K-means grew slightly and gradually. In short, for the interest of proposed work, k-means based clustering process showed its efficiency over HC based works in [6, 9].

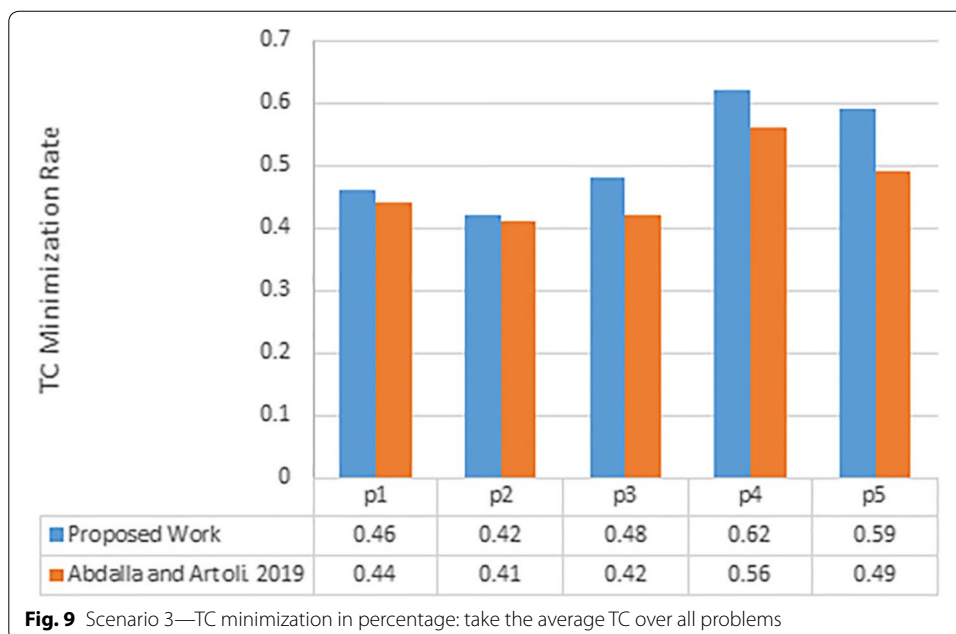


Fig. 9 Scenario 3—TC minimization in percentage: take the average TC over all problems

Like Fig. 4, there has been a significant gap in time between AHC based procedure and K-means based procedure particularly when the query set is growing steadily from 80–300 queries. The rationale behind these gaps is already-given earlier in the interpretation that followed Fig. 4.

In terms of TC, in Fig. 7, the proposed work was clearly proven to outperforms [9] in all problems addressed in scenario (1) in which partial replication was adopted. The same results were evidently affirmed in both scenario (2) and (3) in which non-replication and the whole relation allocation were adopted, (see Figs. 7 and 9). However, in scenario (3) in which random allocation was chosen, both works were seen to have close results particularly in problems (P1–P3) in which retrieval queries were of larger space. The proposed work was seen to behave better, in particular, problems P4 and P5 in which update queries were of larger portion, though. Moreover, even in scenario (4), proposed work still has the lead over [9].

To recap, over all addressed scenarios, our proposed work was shown to outweigh [9] either highly (Figs. 7, 8 and 10) or slightly (Fig. 9). In fact, the rational reason behind this lied in the well-documented behaviour of K-means that lead to having a significant reduction in the solution space that contains schemes, and then much lower FE costs in terms of remote access and local access costs of survival schemes (see Table 24). As a result, the proposed work outperforms its peers in almost all cases.

Last but not least, to confirm the proposed work’s superiority on real datasets, three problems (P1–P3) had been artificially created to conduct the final three experiments on the Adult database [22] which is being retrieved from the machine learning repository. The number of records of this DB is 48842, and the size is 3.8 MB. The adult description is drawn in Table 25.

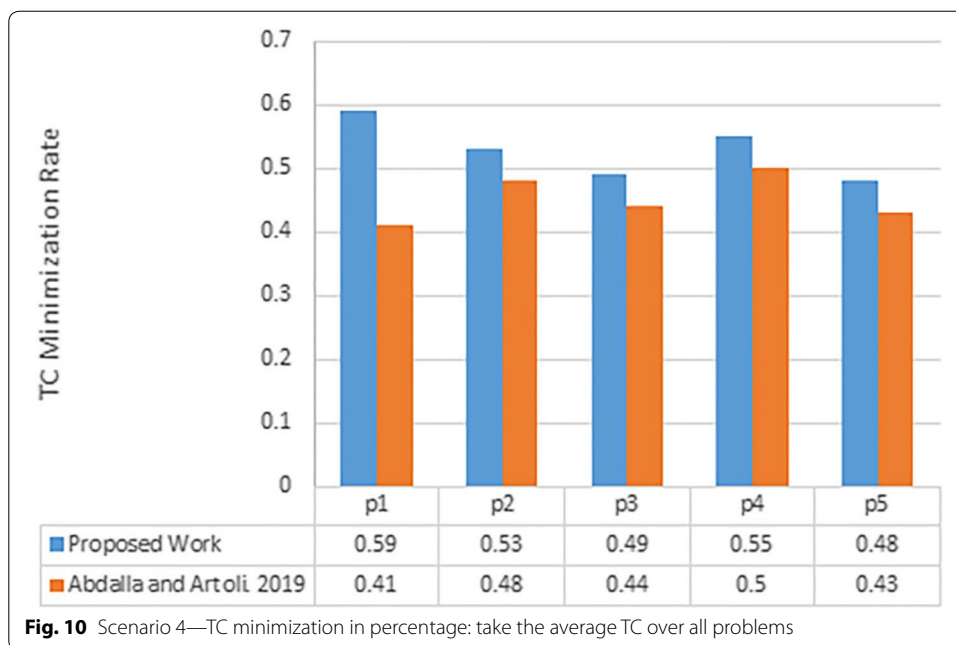


Fig. 10 Scenario 4—TC minimization in percentage: take the average TC over all problems

Table 25 Adult description (UCI, 1999)

Attribute	Type	Size (bytes)
Age	Continuous	1
WorkClass	Nominal	16
Final weight	Continuous	4
Education	Nominal	12
Education	Continuous	1
marital-status	Nominal	21
Occupation	Nominal	17
Relationship	Nominal	14
Race	Nominal	18
Sex	Nominal	6
Capital-gain	Continuous	3
Capital-lose	Continuous	2
Hours-per-week	Continuous	1
Native-country	Nominal	26
Class	Nominal	5

The query set (150 queries) for the Adult dataset is generated in the same way queries of the first experiment, in the result section, were generated with 40, 50 and 60 queries used in all three experiments consecutively. The transaction workload has 80% SELECT (120 queries), and 20% UPDATE (INSERT and DELETE, 30 queries). However, over all these experiments, not all queries used all attributes as 60% queries used 9 attributes, and 40% queries used 6 attributes. Over both scenarios, the results of TC had been accumulated over all three experiments and averaged the same way done when evaluating the given-above Car datasets for both works under consideration. Only two scenarios are tested as follows; in the first scenario, the partial replication was adopted for the fragmented database (which gives five fragments, F1–F5).

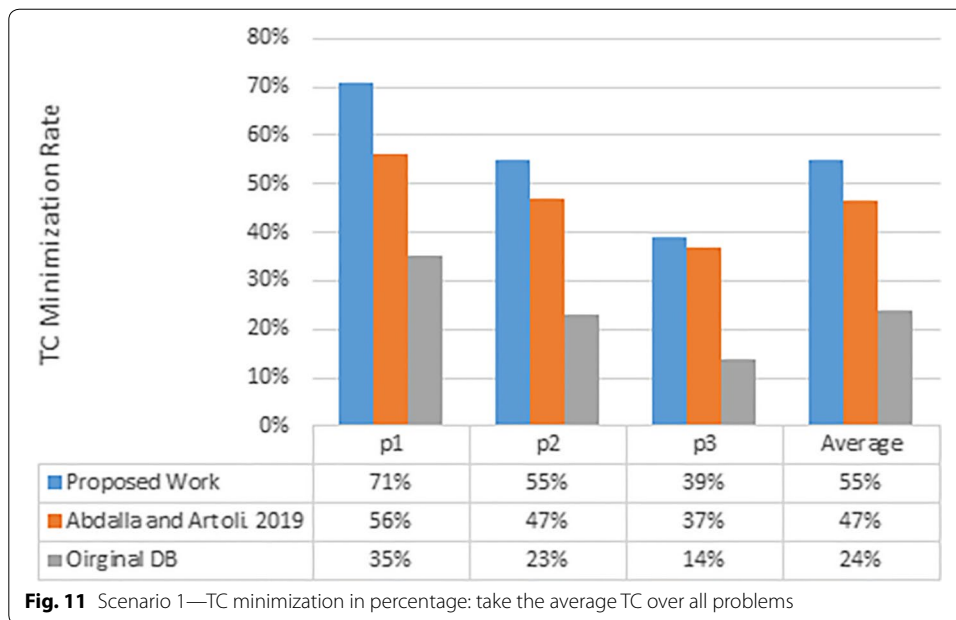


Fig. 11 Scenario 1—TC minimization in percentage: take the average TC over all problems

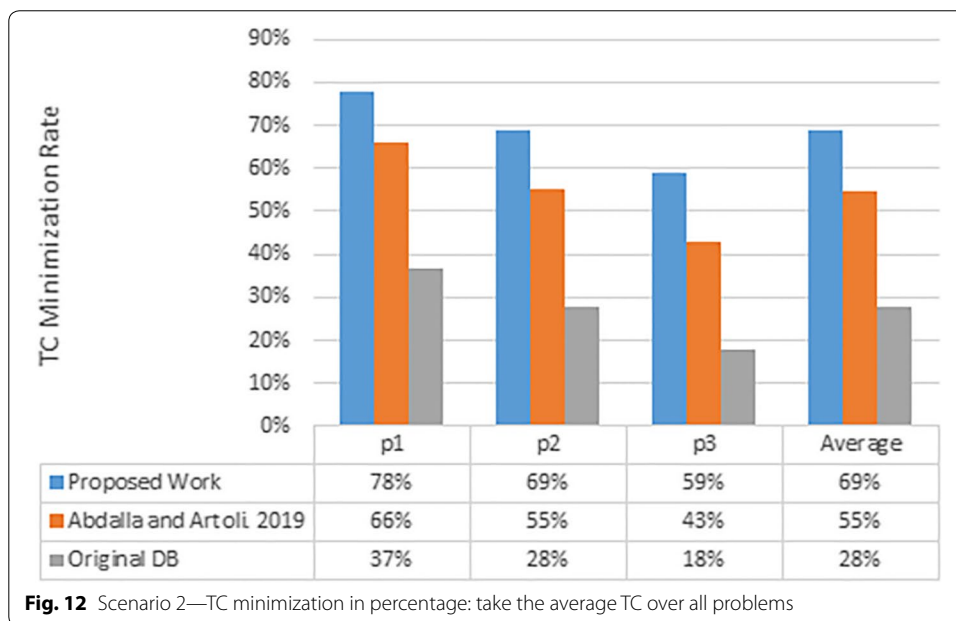


Fig. 12 Scenario 2—TC minimization in percentage: take the average TC over all problems

The DB before fragmentation was tested with its being partially replicated using the same queries. In the second scenario, the non-replication was adopted for both works as well as for the original DB. According to Figs. 11 and 12, the results in both scenarios showed that the proposed K-means based work still has the lead over [9]. On the other hand, the worst performance was seen for the original DB before fragmentation. Indisputably [9], has been significantly superior comparing with the original DB.

In the first scenario (Fig. 11), both works behaved closely with the proposed work being slightly superior as TC minimization average reached 55% while it was 47% in

[9]. That is because of the update queries that affected DDBS performance negatively as each update query needed to be multiplied by the number of sites in which the concerned query was run. In the first two experiments, both works were close to each other though. However, as the query set grew, the proposed work showed a significant TC minimization. In the average results, the proposed work had a noticeable lead comparing with [9] and superior lead comparing with original DB. Both works, nevertheless, had a close performance in P2. On the other hand [9], was seen superior comparing with the original DB. In its turn, original DB had seen to have the worst performance with only 24% in TC minimization.

On the other extreme, in the second scenario (Fig. 12), the proposed work behaved much better comparing with [9] and highly superior comparing with original as TC minimization average reached 69%, 55%, and 28% respectively. Unlike scenario 1, update queries, which negatively affected DDBS performance, had not been multiplied by the number of sites as no replication was adopted. In all three experiments, the proposed work was significantly superior to [9] and highly superior to the original DB. In the averaged results, the proposed work had a significant lead comparing with [9] and superior lead comparing with original DB. Nevertheless [9], was seen much superior comparing with the original DB. In its turn, similarly to scenario 1, the original DB had the worst performance with only 28% in TC minimization.

It can be also deduced that, from both figures, both [9] and original DB had a difference in performance in both scenarios with [9] had a significant improvement in both scenarios. However, the proposed work had a maximally significant improvement with a highly noticeable difference in its behaviour in both scenarios as it was promoted from 55% in the first scenario up to 69% in the second scenario.

Finally, theoretically, it is worth indicating that the proposed K-means-based work of this paper differs significantly from [6, 9] in some important aspects as follows;

- (1) For data fragmentation, this work used K-means to perform clustering of queries while they used hierarchical clustering. As shown from in this paper, K-means based work demonstrated a better performance in terms of finding the survival schemes of the lowest remote access. Moreover, it contributed highly to reducing the size of solution space that would contain all solutions found out of the fragmentation process.
- (2) While [6, 9] used a random way for performing the filtering process that would help eliminate the overlapping schemes, the proposed K-means based work used a heuristic technique as drawn in paper to remove this overlapping. In doing so, the K-means based work contributed vividly in finding the schemes of the lowest remote access and highest local access at the same time comparing with hierarchical based works. This is being reflected in the overall performance of DDBS, though.
- (3) Third, Unlike [6, 9], the proposed K-means based work sought to find the solution for DDBS design without even the need for query frequencies which makes the K-means based work capable of being effectively implemented in the initial stage (in which query frequencies have rarely provided, or they have not been given at

all) of DDBS design. These claims have been emphasized along the paper and demonstrated by the running example and experimental results.

- (4) The proposed K-means based work aimed to reduce the time of computation across applying the clustering algorithm on the Query Difference Matrix which has been directly deduced from applying hamming on QUM matrix. So, only two steps were taken to find this matrix. However, [6, 9] used a lengthy procedure to find their QACM matrix which would also lead to the QDM matrix. Consequently, their hierarchical clustering based works were shown to consume a longer time than the proposed K-means based work.

Conclusions and future work

In this work, for DDBS design, a heuristic K-means clustering algorithm based approach was proposed for data fragmentation and allocation. Several design-related techniques were consolidated. The proposed approach has been shown to shrink the transmission costs of distribution significantly. On the other extreme, the proposed work committed to be operated within the relational database context at the initial and later stages of DDBSs design.

The proposed approach was a three-fold methodology aimed at vertically dividing relations and then assigning the resulted data fragments to network sites. The K-means-based clustering method was introduced to produce disjoint fragments. These fragments, in their turn, were packed into the solution space in each loop of the clustering process. The K-means based solution space was seen to have a great reduction in the number of solutions contained compared to the hierarchical clustering based approaches as drawn in “Discussion” section. The solution space would then hold the overlapping schemes. These schemes are in need to be refined by passing them through the refinement process. In its turn, the refinement process was precisely designed so that it worked mathematically on producing non-overlapping schemes (PSs). This process was also seen effective as it remarkably contributed in reducing the number of schemes (PSs) passed into fragmentation evaluator. These PSs were thoroughly examined by proposed FE so the successful allocation-considered schema is found. The FE technique has two components. These components were used together to assess the quality of partitioning schemes (PSs) that are already held in the solution space.

In the second phase, sites of the network were clustered hierarchically driven by the idea of clustering sites based on their lowest communication costs. In the third phase, an effective allocation algorithm was presented based on the proposed cost model. Moreover, data replication was considered to support data availability and reliability. In doing so, DDBS performance has been improved. Through this work, to prove the concepts, several practical experiments were conducted to verify the proposed approach on different data allocation scenarios. The experiments have been conducted on both artificially-created and real datasets. Four data allocation scenarios were addressed with the major aim of searching the best scenario in which the minimization of transmission costs (tacitly involves communication costs) was being achieved. A brief yet effective external evaluation has been made with two earlier works in the same line on both the artificially-created and real datasets. The proposed K-means based work was seen more

efficient in terms of execution and more effective in terms of TC reduction compared to its HC based peers. Finally, it is worth stressing that the proposed work does not argue the superiority of K-means clustering over the hierarchical clustering algorithm. The proposed work just proved that K-means based work behaved better than hierarchical cluster-based works in terms of DDBS design problem for which the current paper comes essentially to find a solution.

To conclude, our work basically sought to merge many techniques into one efficient work to promote DDBS performance across using K-means-based clustering method to fragment data, proposing a process for data refinement, merging the fragmentation evaluator, utilizing clustering of network sites, and finally allocate and replicate survival data fragments according to four scenarios.

The future work will be dedicated toward conducting in-depth experiments with the state-of-art while varying the query types according to [23] just to confirm the proposed work optimality in the context of comparative study so the proposed work would be implemented in cloud environment effectively.

Abbreviations

DDBS: Distributed Database Systems; TC: Transmission costs; QUM: Query usage matrix; VF: Vertical fragmentation; DRP: Data replication problem; R: Relation; A: Attribute; Q: Query; S: Site; AAM: Attribute Access Matrix; QDM: Query Difference Matrix; PS: Partitioning Schema; OM: Optimality Measure; Cr: An integer counter; FE: Fragmentation evaluator; K: Queries; N: Attributes; CN: Clusters of queries; M: Sites; CM: Clusters of sites; F: Fragment; NACS: Net access cost for each site; CMS: Communication costs between sites; CMS: Communication costs between clusters; SAAM: Site Attribute Access Matrix; LAL: Lower attribute limit; UAL: Upper attribute limit; $|A_{ij}|$: The number of attributes in F_i that are locally approached by Q_k ; NA: Attributes number of targeted relation R; |AD|: The attributes number in F_i which was remotely reached with regard to F_j , by Q_k ; LDV: Least difference value.

Acknowledgements

The authors would like to sincerely express his an eternal thanks to both the Journal of Big Data Team (Editors, in particular) along with those respected anonymous reviewers for their valuable comments and directions that otherwise this research article would not be released.

Authors' contributions

AAA has been the key contributor in conception and design, implementing the approach and analyzing results of all experiments, and the preparation, writing and revising the manuscript. The author read and approved the final manuscript.

Authors' information

Ali A. Amer is an assistant professor in Computer Science Department at Taiz University (YEMEN). He has been publishing many research papers in highly-ranked and top-tier journals as well as refereed International conferences. He has also acted as reviewer for many top-venue platforms. Off these platforms, he has published in, and reviewed for: ACM Computing Surveys, IEEE Access, Computer in Human Behavior, International Journal on Semantic Web and Information Systems, Universal Computer Science Journal, Journal of Supercomputing, Heliyon, and Journal of Evolutionary Intelligence, to name a few. His primary interest of research falls into: Information Systems, Database, Distributed and parallel Database Systems, Data Integration, Data Mining, Network and Information Retrieval.

Funding

No funding received.

Availability of data and materials

Not applicable.

Competing interests

The authors declare that they have no competing interests.

Received: 12 November 2019 Accepted: 19 April 2020

Published online: 11 May 2020

References

1. Nashat D, Amer A. A comprehensive taxonomy of fragmentation and allocation techniques in distributed database design. *ACM Comput Surv.* 2018;51(1):1–25.

2. Hui M, Schewe K, Kirchberg M (2006). A heuristic approach to vertical fragmentation incorporating query information. 7th International Baltic Conference on Databases and Information Systems.
3. Lotfi N. Data allocation in Distributed Database Systems: a novel hybrid method based on differential evolution and variable neighborhood search. *SN Appl Sci*. 2019. <https://doi.org/10.1007/s42452-019-1787-3>.
4. Wiese L, Waage T, Bollwein F. A replication scheme for multiple fragmentations with overlapping fragments. *Comput J*. 2016;60(3):308–28.
5. Mahi M, Baykan O, Kodaz H. A new approach based on particle swarm optimization algorithm for solving data allocation problem. *Appl Soft Comput*. 2018;62:571–8.
6. Sewisy A, Amer A, Abdalla H. A novel query-driven clustering-based technique for vertical fragmentation and allocation in Distributed Database Systems. *Int J Semant Web Inf Syst*. 2017;13(2):27–54.
7. Amer A. Data replication impact on DDBS system performance. In: Lytras SM, Aljohani N, Damiani E, Chui K, editors. *Semantic web science and real-world applications*. 1st ed. Pennsylvania: IGI Global; 2019. p. 134–62.
8. Amer A, Mohamed M, Al-Asri K (2018). On an effective hierarchical clustering based model for data fragmentation and allocation in relational DDBS: review and proposal. In: *Proceedings of ACM ICCES conference, Kuala Lumpur, Malaysia, July 14–16, (ICCES'18)*.
9. Abdalla H, Artoli A. Towards an efficient data fragmentation, allocation, and clustering approach in a distributed environment. *Information*. 2019;10(3):112.
10. Torshiz M, Esfaji A, Amintoosi H. Enhanced schemes for data fragmentation, allocation, and replication in Distributed Database Systems. *Int J Comput Syst Sci Eng*. 2020;35(2).
11. Mehta S, Agarwal P, Shrivastava P, Barlawala J. Differential bond energy algorithm for optimal vertical fragmentation of distributed databases. *J King Saud Univ Comput Inform Sci*. 2018. <https://doi.org/10.1016/j.jksuci.2018.09.020>.
12. Zar Lwin NK, Naing TM (2018). Non-redundant dynamic fragment allocation with horizontal partition in Distributed Database System. In: *International conference on intelligent informatics and biomedical sciences (ICIIBMS), Bangkok*, p. 300–305.
13. Abdel Raouf A, Badr N, Tolba M. Dynamic data reallocation and replication over a cloud environment. *Concurr Comput*. 2018;30(13):e4416.
14. Somov S. Creation of information-technological reserve in Distributed Data Processing Systems. *Autom Remote Control*. 2019;80(4):781–90.
15. Amer A, Mohamed M, Al-Asri K. ASGOP: an aggregated similarity-based greedy-oriented approach for relational DDBSs design. *Heliyon*. 2020;6(1):e03172.
16. Jain A, Dubes R. *Algorithms for clustering data*. Englewood Cliffs: Prentice Hall; 1988.
17. Zahra S, Ghazanfar MA, Khalid A, Azam MA, Naeem U, Prugel-Bennett A. Novel centroid selection approaches for K-means-clustering based recommender systems. *Inf Sci*. 2015;320:156–89.
18. Sandhya N, Raja Sekar M. Analysis of variant approaches for initial centroid selection in K-means clustering algorithm. In: Satapathy S, Bhateja V, Das S, editors. *Smart computing and informatics. Smart Innovation, Systems and Technologies*, vol. 78. Singapore: Springer; 2018.
19. Hamming R. Error detecting and error correcting codes. *Bell Syst Tech J*. 1950;29(2):147–60.
20. Koga H, Ishibashi T, Watanabe T. Fast agglomerative hierarchical clustering algorithm using locality-sensitive hashing. *Knowl Inf Syst*. 2006;12(1):25–53.
21. Amer A, Sewisy A, Elgendy T. An optimized approach for simultaneous horizontal data fragmentation and allocation in Distributed Database Systems (DDBSs). *Heliyon*. 2017;3(12):e00487.
22. UCI. (1999). Machine learning repository content summary. Retrieved March 4, 2020, from <http://www.ics.uci.edu/~mlearn/MLSummary.html>.
23. Amer A, Abdalla H. (2012). A heuristic approach to re-allocate data fragments in DDBSs. *Information Technology and e- Services (ICITeS)*, International Conference on IEEE.

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Submit your manuscript to a SpringerOpen[®] journal and benefit from:

- Convenient online submission
- Rigorous peer review
- Open access: articles freely available online
- High visibility within the field
- Retaining the copyright to your article

Submit your next manuscript at ► springeropen.com
