Query Driven Data Subspace Mapping

Panagiotis Fountas, Maria Papathanasaki, Kostas Kolomvatsos, Christos Anagnostopoulos

Query Driven Data Subspace Mapping

Panagiotis Fountas[1][0000−0002−4555−6606], Maria Papathanasaki[1][0000−0002−7440−8485], Kostas Kolomvatsos[1][0000−0002−9442−3340], and Christos Anagnostopoulos[2][0000−0003−1517−6757]

[1] Department of Informatics and Telecommunications, University of Thessaly, Papasiopoulou 2-4, 35131, Lamia, Greece
[2] School of Computing Science, University of Glasgow, Lilybank Gardens 18, G12 8RZ, Glasgow, UK
{pfountas,mpapathanasaki,kostasks}@uth.gr, christos.anagnostopoulos@glasgow.ac.uk

Abstract. The increased use of multiple types of smart devices in several application domains, opens the pathways for the collection of humongous volumes of data. At the same time, the need for processing of only a subset of these data by applications in order to quickly conclude tasks execution and knowledge extraction, has resulted in the adoption of a very high number of queries set into distributed datasets. As a result, a significant process is the efficient response to these queries both in terms of time and the appropriate data. In this paper, we present a hierarchical query-driven clustering approach, for performing efficient data mapping in remote datasets for the management of future queries. Our work differs from other current methods in the sense that it combines a Query-Based Learning (QBL) model with a hierarchical clustering in the same methodology. The performance of the proposed model is assessed by a set of experimental scenarios while we present the relevant numerical outcomes.

Keywords: Data mapping · Data Management · Query-Based Learning · Hierarchical Clustering · Data Retrieval

1 Introduction

In the era of the intense increase of data collection and production, analysts have to cope with a massive amount of datasets which are spread across multiple locations. As a consequence, several challenges have emerged for data management and information retrieval. Data mapping is the process of combining data from various sources into a particular dataset storing it in a consistent manner. This procedure is essential in various processing activities including data migration, data integration, etc. One of the major challenges, when data management is the case, is the estimation of the appropriate responses to requests for processing performed in the minimum possible time. Such requests are, usually, formulated in the form of queries defining specific constraints and conditions for data retrieval. Queries are reported by streams to processing nodes that may be present

at various locations being directly connected with smart devices. Devices are 'responsible' to collect data and report them feeding the geo-distributed datasets (DDSs).

In the current effort, we present a hierarchical clustering model adopted to detect the proper data for executing queries delivered to a server (SV) in the minimum time. We elaborate on a model which takes into consideration the data requested by a number of historical queries to detect and collect those that a future query may require avoiding to 'scan' the entire dataset. The proposed model performs two different types of clustering: (i) a fuzzy clustering using the Fuzzy C-Means (FCM) algorithm, and (ii) a hard clustering using K-Means (KM) algorithm. Both methods are adopted to group similar queries based on the data they request. When a new query arrives, the proposed model checks to find the most similar clusters. We also involve into our model a mechanism which concludes the overlap of the 'data area' of interest between two queries. We consider a set $DS = \{DS_1, \ldots, DS_w\}$ of DDSs and a SV. We also assume that a group Z_i of devices present in the Internet of Things (IoT) infrastructure is connected with a DDS DS_i such that each Z_i is connected with only one DS_i. IoT devices collect and report data into the respective DS_i in the form of multivariate vectors; $X_j^t = [x_1^t, \ldots, x_d^t]$, where the index j expresses the IoT device reporting X_j^t and the index t shows the time instance that X_j^t is reported. Additionally, d is the number of dimensions of X_j^t. DDSs receive X_j^t and store them in the appropriate format to be the subject of further processing activities. The SV receives queries $Q = \{q_1, q_2, \ldots\}$ from applications and users. We consider that every vector can be represented as a point in the d-dimensional space. q_i requires a number Φ of points as a response and can contain range selection operators for one or more dimensions. This means that q_i can create the boundaries of the area in which the requested data are located. Apparently, the SV has to detect the appropriate data to formulate the final responses to the incoming queries.

The novelty of this paper is that we combine the QBL [13], [14] with a hierarchical clustering scheme into a model that is able to predict which data should be retrieved for similar future queries. The following list reports on the contributions of our work: **(i)** We propose a hierarchical clustering-based model combining two different types of clustering methods for the detection of the appropriate data that correspond to queries requests in the minimum possible time and with the minimum error; **(ii)** We adopt QBL to retrieve the appropriate data for the incoming queries while relying on the retrieved data of previously executed queries; **(iii)** We argue on an data overlap metric between the 'areas' of the incoming queries to detect the existence of common data points between past and future queries' requests.

The rest of the paper is organized as follows. Section 2 reports on the related work while Section 3 elaborates on the preliminary information before we present our model. Section 4 discusses the proposed model and Section 5 presents the adopted experimental evaluation approach. Finally, Section 6 concludes this paper by presenting our future research plans.

2 Related work

Any data management process consists of an important factor for the effectiveness of various tasks like as the generation of analytics upon a specific dataset. The research community has proposed several models and mechanisms for the improvement of various data management processes.

The authors of [1] present a solution for assigning queries and tasks in the appropriate edge computing nodes to reduce the response time. For this purpose, the authors propose a method for the estimation of the computational burden, that an allocation of a query will add to a node. Also, the authors develop an ensemble similarity scheme, responsible to deliver the complexity class for each query or task and a probabilistic decision-making model. In [2], the authors define the concept of a Query Controller (QC) that assigns each query into a processor. Based on this model, they develop a framework for query assignment which involves two learning schemes, i.e., a Reinforcement Learning (RL) and a clustering scheme. In [3], the authors introduce an adaptive, reciprocity-based Machine Learning mechanism, to estimate the answers of a variety of aggregate queries (AQs). The mechanism learns from past analytical-query patterns while the authors develop solutions to correspond in changes in queries' requests. In [4], the authors discuss the Data Canopy framework, adopted to reduce the time needed to compute the statistical information of a dataset. The proposed framework is a smart cache designed for statistical exploratory analysis. It calculates and caches the fundamental primitives of statistical measures, thus, it composes results for future queries without having to return to the base data. The authors of [5] propose WANalytics, a geo-didtributed system that copes with the Wide-Area Big Data problem, a challenge that concerns the supporting rich Directed Acyclic Graphs (DAGs) of computation over globally distributed data. WANalytics is formed by two major parts. The first one is a runtime layer that distributes user DAGs around data centers, and the other part pertains to a workload analyzer that constantly checks and improves the user workload.

The difference of our work compared to other efforts in the domain, is that we do not focus on the query allocation problem to nodes that exhibit a low load and a high processing speed, but on the creation of query clusters that share similar data requests. We do not deal with storing aggregates to synthesize answers for future queries, but we detect what historical queries are similar to the future ones, and return the historical responses as answer to the future queries. In addition, we deviate from approaches that propose solutions for the minimization of the required bandwidth, and we focus on the accuracy of the final results and the time requirements for delivering the outcome and not on the network performance. This is because we consider that queries do not require too many network resources to be reported and responded.

3 Preliminaries

K-Means Algorithm. K-Means algorithm (KM) is one of the most popular unsupervised clustering models. It groups the given data into K clusters trying

to minimize the following objective function: $J = \sum_{i=1}^{\mathrm{K}} \sum_{p \in C_i} \|p - c_i\|^2$. The minimization of the discussed equation is equivalent to the minimization of the distance between points in a cluster C_i with the centroid c_i. We consider a set $D = \{\overrightarrow{p}_1, \ldots, \overrightarrow{p}_n\}$ which consists of n vectors considered as a d-dimensional points. KM has the goal of splitting the n vectors into K clusters, where $\mathrm{K} \leq n$ should be defined in advance. The steps of the KM are as follows [8][9][10]: (i) K points from D are randomly selected to be the centroids of clusters; (ii) In the second step, the algorithm computes for every point in D the distance from the available centroids. Afterwards, every data point is assigned to the closest cluster; (iii) The algorithm recalculates the centroids of each cluster adopting the mean of the cluster members; (iv) The algorithm iterates to the second step till the stopping condition is true. For instance, a specific number of iterations l could be chosen. We have to notice that the cluster centroids could be virtual data points. The time complexity of KM depends on three parameters; the number of data vectors n, the number of clusters K and the number of iterations l that the algorithm needs to cluster the data. Consequently, the time complexity is $O(n \cdot \mathrm{K} \cdot l)$ [6].

Fuzzy C-Means. FCM algorithm is a fuzzy clustering algorithm which assigns each data point to a cluster according to a membership value $u_{gi} \in [0, 1]$ (g is the index of every data vector). u_{gi} indicates the correlation between a data point and the cluster center c_i, thus, the data point is assigned to the cluster with the highest membership value [7]. Given D, the FCM results a $n \times \Gamma$ matrix $\mathbf{U}$ which includes the membership values for each data point, and a set $C = \{C_1, C_2, C_3, \ldots, C_\Gamma\}$ of clusters with centroids $\{c_1, c_2, c_3, \ldots, c_\Gamma\}$. The FCM requires three parameters: (i) the fuzzier $m \in [1, \infty)$ that controls the fuzziness of the clustering; (ii) the number of clusters Γ and; (iii) the stopping criterion value $\beta \in [0, 1]$. The FCM intends to minimize the following objective function: $J_m = \sum_{i=1}^{\Gamma} \sum_{g=1}^{n} (u_{gi})^m \cdot \|p_g - c_i\|^2$. The steps of the algorithm are referred below [8], [9], [11], [12]: (i) Choose of the parameters m,Γ,β; (ii) Initialize the membership matrix $\mathbf{U}$; (iii) Calculate the centers of every cluster in C; (iv) Update the membership matrix $\mathbf{U}$; (v) Repeat steps (iii), (iv) until the divergence is less than β; (vi) Output $\mathbf{U}$,C. The following equations are adopted to deliver u_{gi} and c_i:

$$u_{gi} = \frac{1}{\sum_{j=1}^{\Gamma} \left\{ \frac{\|p_g - c_i\|}{\|p_g - c_j\|} \right\}^{\frac{2}{m-1}}} \tag{1}$$

$$c_i = \frac{\sum_{g=1}^{n} (u_{gi})^m \cdot p_g}{\sum_{g=1}^{n} (u_{gi})^m} \tag{2}$$

4 The Proposed Approach

We elaborate on the proposed model adopted to limit the time for providing responses to the incoming queries by incorporating statistical information retrieved by past queries. We call our model Hierarchical Mixed Clustering Model

(HMCM). HMCM comprises two phases: **(i)** the 'warm up' phase and; **(ii)** the 'performance' phase. The 'warm up' phase consists of two stages; the preparation stage and the hierarchical clustering execution stage. In the preparation stage, when a query is reported to the SV, the HMCM performs a sequential scan of the entire database to determine the suitable data points. When a z number of queries have been sent to SV, the HMCM proceeds to the second stage of the 'warm up' phase. In this second stage, the hierarchical clustering execution stage, a hierarchical clustering is performed to create clusters and subclusters upon historical queries, which will take part of the 'performance' phase. The performance phase targets to map the appropriate data as the response to a future incoming query. Initially, the HMCM adopts the FCM to create the aforementioned set of clusters C upon z queries. Afterwards, the model divides every cluster further into a set of $N_{C_i} = \{S_1, S_2, \ldots, S_M\}$ using the KM algorithm. In the 'performance' phase, the HMCM is activated every time a query is reported to the SV. More specifically, the proposed model uses a similarity metric, to identify the top-r clusters whose members have the same data requests as the incoming query. Then, the HMCM utilizes again a correlation/similarity metric to identify, for each of the top-r clusters, the top-ℓ subclusters. Therefore, we focus only on groups of queries that have the most relevant data requests to the incoming query. In our implementation, we adopt the Euclidean Distance to realize the similarity/correlation metric. Moreover, we propose the adoption of an overlapping metric to detect the 'matching' between the data requests of queries. We consider that every query is represented by the area of points that targets to receive the final response. The Area Overlap Metric (AOM) is provided by the following equation:

$$\text{AOM}(q_{inc}, q_{member}) = \frac{q_{inc} \cap q_{member}}{\textit{incoming query area}} \tag{3}$$

In the above equation, the numerator is the overlap area between two queries. q_{inc} is the incoming query while q_{member} depicts the queries in top-ℓ subclusters of every cluster present on the top-r list. The denominator is the area where the requested data may be present. The AOM indicates the percentage of the q_{inc} area which is covered by the area of q_{member} queries. The HMCM, after the detection of the appropriate clusters and subclusters, examines the members of the detected subclusters, to find those queries q_{member} where the AOM between them and the q_{inc} exceeds a pre-defined threshold θ. Afterwards, the HMCM retrieves only the data points that belong to the q_{member} queries, which satisfy the aforementioned condition. The q_{member} queries selection approach, gives the HMCM the ability to 'avoid' queries that belong into the top-ℓ subclusters of top-r clusters but they do not require data from the common area. This approach also allows the HMCM to ignore queries with which the q_{inc} shares a common area, but the AOM realization does not exceed θ. Algorithm 1 presents the steps adopted by the HMCM model to deliver the final outcomes.

For comparison purposes, we elaborate on two additional methods. We call the first as the Baseline Method (BM). BM is executed every time that a query is reported locally and scans the entire dataset, selecting data that satisfy the

query. This approach identifies all the required data points and is the theoretical optimal threshold in terms of error. However, scanning all data in the DDSs for every query, it is a time consuming process. The second method is called Hard Clustering Based Method (HCBM) and is based on the clustering of 'similar' queries that the SV receives during a period of W time instances. Initially, the HCBM is trained over z incoming queries (the training phase), using the BM to identify the data required for their execution and, then, utilizes KM to cluster them. When a query is sent to the SV, the HCBM is 'triggered' and uses a correlation/similarity metric to find the top-r similar clusters to the incoming query. The HCBM uses the top-r clusters that have been detected to retrieve data points that satisfy the incoming query.

Algorithm 1 Pseudocode of the HMCM algorithm

D: The set of data

```
counter = 0
while counter ≤ z do
    Receive q_inc
    counter++
    reponse = ScanDataset()
    Q.add(q_inc)
end while
C = PerformFCM(Q)
for C_i ∈ C do
    N_{C_i} =ExecuteKMeans(C_i)
end for
while true do
    Receive q_inc
    TopRCIds=FindTopRSimilarClusters(C,q_inc)
    for ω in TopRCIds do
        TopℓSIds= FindTopℓSimilarSubspaces(N_{C_ω}, q_inc)
        for j in TopℓSIds do
            DetectedData=DetectedData ∪ FindDataThatSatisfyQueryUsingAOM(q_inc)
        end for
    end for
    Send(DetectedData)
end while
```

5 Experimental Setup and Evaluation

The experimental evaluation of the proposed model relies on the Query Analytics Workloads Dataset [3]. The dataset contains range/radius query workloads from Gaussian distributions over a real dataset. In our experiments, we focus on range queries, and are based on the file Range Queries Aggregates to create three datasets which are named Warming Dataset (D_W), Dataset of two-dimensional points (D_{2d}) and Test dataset (D_T). Each range query is stored in

[3] http://archive.ics.uci.edu/ml/datasets/Query+Analytics+Workloads+Dataset

the following format $\{X, Y, Xr, Yr, Count, SUM, AVG\}$. However, we take into consideration only the first four attributes which refer to the range query. The first two attributes are the coordinates for the x-axis and y-axis for the center of a 'data rectangle', respectively. The third and fourth attributes represent the ranges of the first two attributes. The D_W consists of $1,000$ queries of the format $q_i = \{X_i, Y_i, Xr_i, Yr_i\}$. We randomly generate for each q_i a number $\delta_i \in [20, 30]$ of two-dimensional data points which are located inside the rectangle of q_i. The total number of two-dimensional points is equal to $24,923$ and constitutes the D_{2d} dataset. The last dataset D_T contains $\psi = 1,000$ incoming Range queries with the same distribution and format with the D_W. Our goal is to confirm that the proposed model has the ability to detect the data that an incoming query requests. We adopt the D_W to 'train' the HCBM and HMCM models while the D_T is used to test the performance of BM, HCBM and HMCM.

We evaluate the described models for both, the error levels and the time that they need to detect the correct data. The evaluation relies on the metrics of Precision (PRE), Recall (REC), Accuracy (ACC), False Positive Rate (FPR) and F1-score (FSC) as they ae realized by the calculation of True Positives (TP), True Negatives (TN), False Positives (FP), False Negatives (FN). We define as TP the number of data that an incoming query q_i demands being correctly detected while TN is the number of data that the q_i does not demand and the model correctly rejects them. On the other hand, FP is the number of data that the q_i does not demand but the model retrieves them, and FN is the number of data that the q_i demands but the model does not detect them. The equations that realize PRE, REC, ACC, FPR and FSC are defined as follows: PRE = TP / (TP+FP); REC = TP / (TP+FN) ; ACC = (TP+TN) / (TP+TN+FP+FN); FPR = FP / (FP+TN); FSC = $(2 \cdot$ TP) / $(2 \cdot$ TP+FP+FN). The required time for each query is represented τ. The aforementioned metrics are used to calculate the performance of models for each q_i. However, the performance of the models has to be estimated over all the incoming queries, thus, we utilize the mean values of the aforementioned metrics. We also calculate the mean time that every model requires to detect the appropriate data which satisfy the incoming queries. The following equation holds true:

$$\mu_{\Omega_i} = \frac{\sum_{q_i \in D_T} \Omega_i}{\psi}, \Omega \in \{PRE, REC, ACC, FPR, FSC, \tau\} \tag{4}$$

The models have the best performance when μ_{ACC}, μ_{PRE}, μ_{REC}, μ_{FSC} reach the unity and μ_{FPR}, μ_{τ} are close to zero. In our experiments, we pay significant attention on μ_{FPR} and μ_{τ} as the former gives the average rate of data that the models retrieve but the incoming queries do not demand them, while the latter gives the mean time that our methods require to respond into the incoming queries. We present the performance of the proposed models for different number of clusters regardless the method adopted to deliver them, i.e., HMCM or HCBM. We have to mention that in our plots and especially for the error metrics, we do not consider the BM model because it detects all the

required data with no error since it scans the entire dataset and consists of the theoretical optimal model.

In Figure 1, we compare the HMCM and the HCBM when they create 24 clusters. As we can see, the HMCM exhibits the best performance for all metrics except the μ_{REC} where the two models have the same performance. The dominance of the HMCM is clearly revealed from the Figure 2 where the mean time that the HMCM needs to respond to the incoming queries is significantly less than the BM method and much less than the HCBM.

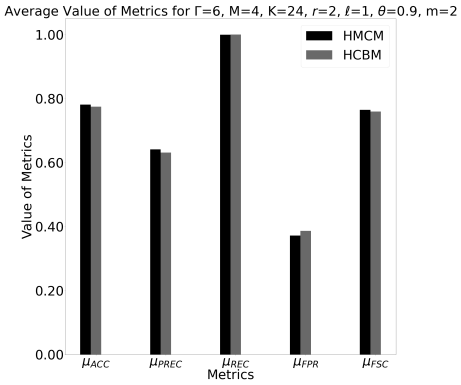

Fig. 1. Comparison between HMCM and HCBM for Γ=6, M=4 and K=24

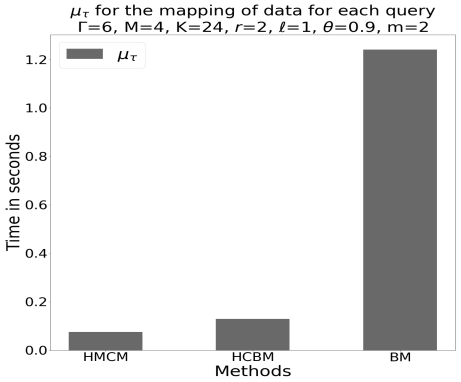

Fig. 2. Average Time comparison between the models for Γ=6, M=4, K=24

Figure 3 shows the comparison of the HMCM and the HCBM for 42 clusters. We observe that the HMCM overcomes the HCBM for all metrics except the μ_{REC}, where the HMCM has slightly lower performance than the HCBM. Nevertheless, both μ_{FSC} and μ_{FPR} confirm that the HMCM has better performance when the estimation error is in our focus. Figure 4 strengthens the conclusion that we deduce from Figure 4 since the HMCM achieves better performance than the HCBM in less conclusion time.

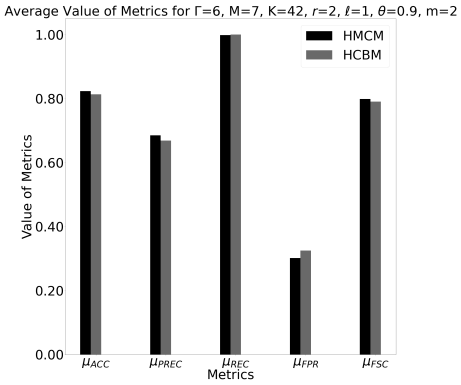

Fig. 3. Comparison between HMCM and HCBM for Γ=6, M=7 and K=42

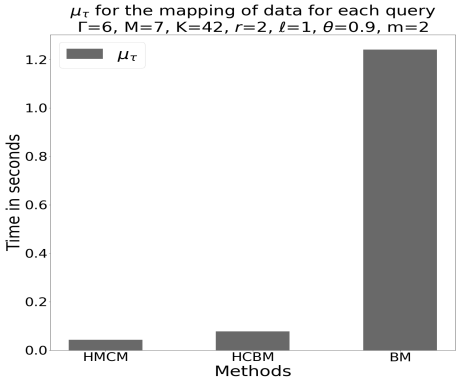

Fig. 4. Average Time comparison between the models for Γ=6, M=7, K=42

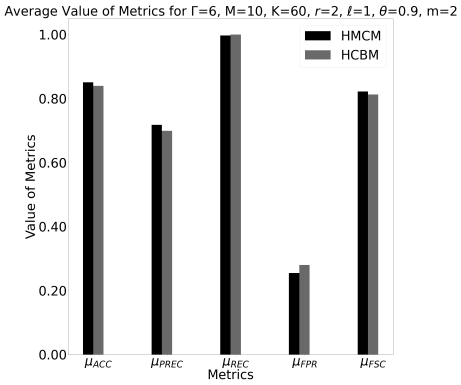

Fig. 5. Comparison between HMCM and HCBM for Γ=6, M=10 and K=60

In Figures 5 & 6, the comparison of error and time metrics is presented between the models for 60 clusters, respectively. In Figure 5, same as in the previous performance outcomes, the HMCM has better performance then the

HCBM in the majority of the error metrics. As far as the time metric concerns, the HMCM maps the data in less time than the other models, as it is presented by Figure 6. Again, in the most important metrics for inferring the conclusion of the designation of the best model, the HMCM clearly outperforms the HCBM.

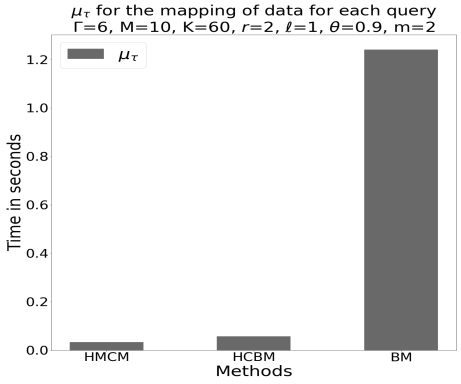

Fig. 6. Average Time comparison between the models for Γ=6, M=10, K=60

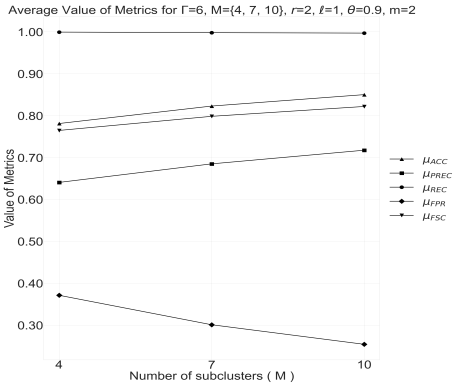

Fig. 7. Performance of error metrics for different number of subclusters in the HMCM

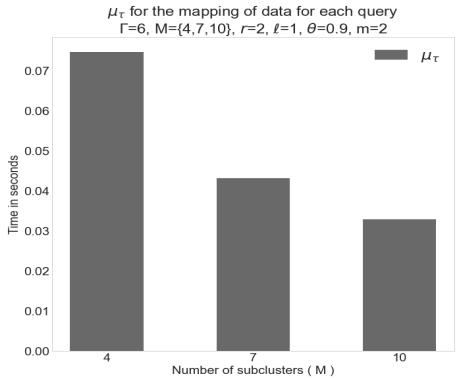

Fig. 8. Comparison of the time metric for different number of subclusters in the HMCM

In Figures 7 & 8, we evaluate the effect of the number of subclusters in the performance of our model both for the error and the required conclusion time. We can easily observe that the increase of the number of subclusters clearly improves the performance of the HMCM and simultaneously decreases the mean required time for the mapping of data.

Figures 9 & 10 present the effect of the increase of clusters in error and time metrics for the HMCM. We notice that the higher the number of clusters, the better the performance becomes, while the mean time is affected in the opposite direction.

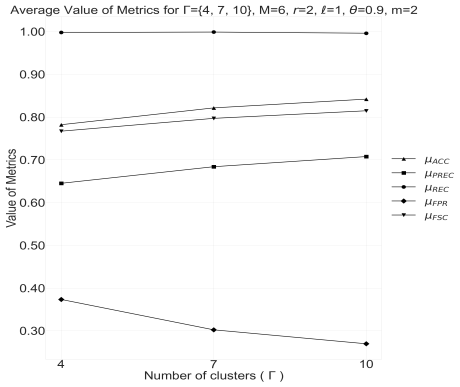

Fig. 9. Performance metrics for different number of clusters in the HMCM

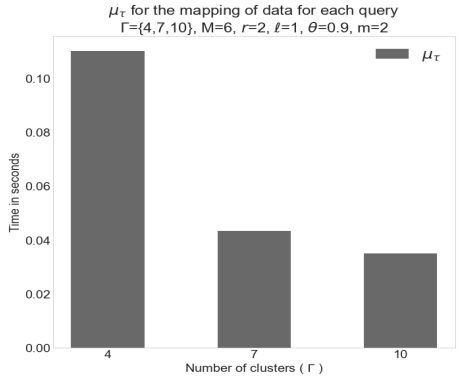

Fig. 10. Comparison of the time metric for different number of clusters in the HMCM

6 Conlusions and Future Work

Data mapping is a significant data management process which plays an important role in many application domains. This process becomes more complex when the data are geo-distributed. Data mapping can be improved if we can identify relations between the data in the DDSs and the defined queries. In this paper, we focus on the efficient data mapping and propose a solution for an effective mapping in the minimum possible time. We are based on the answers of past

queries, and propose a hierarchical clustering scheme, which groups them, relying on the similarity of their data requests. Also, we involve a mechanism for the calculation of the matching data area between two queries. Hence, we create small groups of queries with similar data requests and try to benefit from the exclusion of non-similar groups with an incoming query to reduce the response time. A future extension of this work could be the incorporation of a more complex methodology for the improvement of both error and time metrics. We could also adopt a deep learning model that will be able to be adapted on continuous changes in queries requirements.

References

1. Kolomvatsos, K., Anagnostopoulos, C.: A probabilistic model for assigning queries at the edge. Computing. 102, 865-892, (2020).
2. Kolomvatsos, K., Anagnostopoulos, C.: Reinforcement Learning for Predictive Analytics in Smart Cities. Informatics, (2017).
3. Savva, F., Anagnostopoulos, C., Triantafillou, P.: Adaptive learning of aggregate analytics under dynamic workloads. Future Generation Computer Systems. 109, 317-330, (2020).
4. Wasay, A., Wei, X., Dayan, N., Idreos, S.: Data Canopy. Proceedings of the 2017 ACM International Conference on Management of Data. (2017).
5. Vulimiri, A., Curino, C., Godfrey, P., Jungblut, T., Karanasos, K., Padhye, J., Varghese, G.: WANalytics. Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data. (2015).
6. Na, S., Xumin, L., Yong, G.: Research on k-means Clustering Algorithm: An Improved k-means Clustering Algorithm. 2010 Third International Symposium on Intelligent Information Technology and Security Informatics. (2010).
7. Schwämmle, V., Jensen, O.: A simple and fast method to determine the parameters for fuzzy c–means cluster analysis. Bioinformatics. 26, 2841-2848 (2010).
8. Etehadtavakol, M., Sadri, S., Ng, E.: Application of K- and Fuzzy c-Means for Color Segmentation of Thermal Infrared Breast Images. Journal of Medical Systems. 34, 35-42 (2008).
9. A. M. Jamel, A., Akay, B.: A Survey and Systematic Categorization of Parallel K-Means and Fuzzy-C-Means Algorithms. Computer Systems Science and Engineering. 34, 259-281 (2019).
10. Na, S., Xumin, L., Yong, G.: Research on k-means Clustering Algorithm: An Improved k-means Clustering Algorithm. 2010 Third International Symposium on Intelligent Information Technology and Security Informatics. (2010).
11. Gupta, R., Muttoo, S., Pal, S.: Fuzzy C-Means Clustering and Particle Swarm Optimization based scheme for Common Service Center location allocation. Applied Intelligence. 47, 624-643 (2017).
12. Stetco, A., Zeng, X., Keane, J.: Fuzzy C-means++: Fuzzy C-means with effective seeding initialization. Expert Systems with Applications. 42, 7541-7548 (2015).
13. Chang, R., Hsu, H., Lin, S., Chang, C., Ho, J.: Query-Based Learning for Dynamic Particle Swarm Optimization. IEEE Access. 5, 7648-7658 (2017).
14. Albishre, K., Li, Y., Xu, Y., Huang, W.: Query-based unsupervised learning for improving social media search. World Wide Web. 23, 1791-1809 (2019).