The Similarity Measures and their Impact on OODB Fragmentation Using Hierarchical Clustering Algorithms

ADRIAN SERGIU DARABANT, HOREA TODORAN, OCTAVIAN CRET, GEORGE CHIS
Dept. of Computer Science
Babes Bolyai University, Technical University
1, Kogalniceanu and 15 C-tin Daicoviciu, Cluj Napoca,
ROMANIA
dadi@cs.ubbcluj.ro, htodoran@euro.ubbcluj.ro, cret@cs.utcluj.ro, gchis@econ.ubbcluj.ro,
http://www.cs.ubbcluj.ro

Abstract: - Class fragmentation is an essential phase in the design of Distributed Object Oriented Databases (DOODB). Due to their semantic similarity with the purpose of database fragmentation (obtaining sets of similar objects with respect to the user applications running in the system), clustering algorithms have recently begun to be investigated in the process of database fragmentation. This work proposes a study on the impact of different similarity measures applied in hierarchical agglomerative clustering algorithms for horizontal fragmentation of classes with complex attributes. This study would eventually help finding formal, automatic, approaches in choosing a particular similarity measure in accordance with: the applied clustering algorithm, the structure of the database inheritance/aggregation hierarchies, the semantics of data, etc.

Key-Words: - Distributed database design, horizontal fragmentation, data mining methods, performance evaluation.

1 Introduction*

As opposed to centralized databases where the design phase handles only logical and physical data modeling, the design process in Distributed Object Oriented Databases involves both data partitioning and allocation to the nodes of the system. This process is usually called *database fragmentation* and is an important aspect of distributed database design. Horizontal fragmentation, in Object Oriented Database Systems, distributes class instances into fragments. Each object in every fragment has the same structure and a different state or content. Thus, a horizontal fragment of a class contains a subset of the whole class extension.

Most of the existing Object Oriented (OO) fragmentation approaches are usually inspired from the relational fragmentation techniques. The accumulated experience in the relational field has already helped developing the first techniques in object oriented data models. While this proves to be a good starting point for approaching the fragmentation problem, there is definitely a limit in applying these techniques to data models featuring all the complex characteristics of a real OO model.

As a result, new approaches for OO database fragmentation are emerging. While some of them are based on graph theory, others use clustering techniques for splitting the classes and their extensions into fragments.

Some of the proposed techniques use data mining clustering algorithms for data fragmentation. Clustering algorithms are not only used in OO database fragmentation but also for optimizing object storage and retrieval. Clustering similar objects closely on disk pages helps faster data retrieval and easier navigation in the aggregation and inheritance hierarchies when accessing related objects.

In [11, 12, 13, 14] we have proposed some new database fragmentation techniques based on data mining methods, using clustering algorithms. Basically two major techniques are presented in these works: a *hierarchical clustering* algorithm and a *k-means clustering* algorithm. Both algorithms are applied in the context of horizontal fragmentation and partition data according to the similarity between objects. Two objects are similar when they are accessed in the same way by queries - they behave in the same way when running queries against the database. The resulting clusters finally form the database fragments.

Essentially, the algorithms group objects together by their similarity with respect to a set of user queries with conditions imposed on data. Similarity (dissimilarity) between objects is defined in a vector space model and is computed using different metrics. As a result, objects that are highly used together by queries are placed in the same fragment.

This paper presents an important aspect of the *fragmentation by clustering* methods – the significance of the similarity measures and its impact on the performance of the resulting database schema. The study is performed on complex class hierarchies (complex attributes and methods) and compares the performance influence of similarity measures.

The paper is organized as follows. The next section briefly presents some related work handling horizontal data fragmentation in object oriented databases followed by the motivation to this work. Section 2 presents the data model. Section 3 presents our numerical database model and the way it captures objects and their relations. Section 4 briefly presents the hierarchical clustering algorithm studied in this work and section 5 presents the comparative results.

1.1 Related Work

Fragmentation methods for OODB environments, or flat data models have been generally considered in Karlapalem [2], Ezeife [3], Karlapalem [4][5]. Ravat [6] uses the Bond Energy Algorithm (BEA) for vertical and horizontal fragmentation. Ezeife [7] presents a set of algorithms for horizontally fragmenting models with simple attributes/methods and complex attributes/methods. Bellatreche et al. [8] propose a method that emphasizes the role of queries in the horizontal fragmentation.[11] presents a first fragmentation approach based on hierarchical agglomerative clustering while in [13] the original problem modeling is improved so that complex class hierarchies could be taken in account.

1.2 Contributions

Clustering fragmentation methods in complex class hierarchies proposed in earlier papers are generally based on similarity measures used to determine the similarity between different instances of the same class. Based on this similarity, objects are grouped into clusters (fragments). This work proposes a comparative study on the influence of different choices of similarity measures on the performance and quality of the obtained fragmentation. The behavior of clustering algorithms is not generally stable. Special patterns of data combined with different similarity measures give totally different performance and quality results. Our aim is to find a formal or automatic way of choosing the best similarity measure for each data fragmentation context for a given algorithm. The comparative study reveals the major properties for each similarity measure and will provide deeper knowledge on the best similarity measure choice.

2 Data Model

The used object-oriented model is one with the basic features as described in the literature [11]. Objects with common attributes and methods are grouped into classes. A class is an ordered tuple $C=(K,A,M,I)$, where A is the set of object attributes, M is the set of methods, K is the class identifier and I is the set of instances of class C. Every object in the database is uniquely identified by an OID. Classes are organized in an inheritance hierarchy, in which a subclass is a specialization of its superclass. An OODB is a set of classes from an inheritance hierarchy, with all their instances. There is a special class Root that is the ancestor of all classes in the database.

An entry point into a database is a meta-class instance bound to a known variable in the system. An entry point allows navigation from it to all classes and class instances of its sub-tree (including itself). There are usually more entry points in an OODB.

Given a complex hierarchy H, a *path expression* P is defined as $C_1.A_1. ...A_n$, n≥1 where: C_1 is an entry point in H, A_1 is an attribute of class C_1, Ai is an attribute of class C_i in H such that C_i is the domain of attribute A_{i-1} of class $C_{i-1}(1\leq i \leq n)$. In the general case, A_i can be a method call. If i<n, then A_i must return a single complex type value (an object).

3 Vector Space Modeling

Let $Q=\{q_1 ,..., q_t\}$ be the set of all queries in respect to which we want to perform the fragmentation. Let $Pred=\{p_1, ..., p_q\}$ be the set of all atomic predicates Q is defined on. Let $Pred(C)=\{p\in Pred|\ p$ imposes a condition to an attribute of class C or to an attribute of its parent$\}$. Given the predicate $p \equiv C_1.A_1. ...A_n\ \theta\ value$, then $p\in Pred(C_n)$ if class C_i is the complex domain of A_{i-1}, i=1..n, and A_n has a complex type or simple type.

Given two classes C and C', where C' is subclass of C, $Pred(C')\supseteq Pred(C)$. The reason behind this fact is explained in [11].

We construct the object-condition matrix for class C, $OCM(C) = \{a_{ij} ,1\leq i \leq |Inst(C)|, 1\leq j \leq |Pred(C)|\}$, where $Inst(C) = \{O_1, \dots O_m\}$ is the set of all instances of class C, $Pred(C) = \{p_1, \dots, p_n\}$:

$$a_{ij} = \begin{cases} 0, if\ \ p_j(O_i) = false \\ 1, if\ \ p_j(O_i) = true \end{cases} \quad (1)$$

$$w_{ij} = \frac{\sum_{l=1, a_{lj}=a_{ij}}^{m} \left[(a_{lj} \mid a_{lj} > 0) + ((1 - a_{lj}) \mid a_{lj} = 0) \right]}{m}, \quad (1')$$

$$i = \overline{1,m}, j = \overline{1,n}$$

Each line *i* in *OCM(C)* is the object-condition vector of O_i, where $O_i \in Inst(C)$. *OCM(C)* is used then to obtain the characteristic vectors for all instances of *C*. The characteristic vector for object O_i is $w_i = (w_{i1}, w_{i2}, ..., w_{in})$, where each w_{ij} is the ratio between the number of objects in *C* respecting the predicate $p_j \in Pred(C)$ in the same way as O_i and the number of objects in *C*. We denote the characteristic vector matrix as *CVM(C)* [11].

3.1 Derived Fragmentation Modeling

All characteristics of simple attributes and methods have been captured so far. The next part focuses on the class relationships in our vector space model. We first model the aggregation and association relations.

Given two classes C_O (owner) and C_M (member), where C_M is the domain of an attribute of C_O, a path expression traversing this link navigates from instances of C_O to one or more instances of C_M. When fragmenting C_O we should take in account the fragmentation of C_M.

Let $\{F_1, \ldots F_k\}$ be the fragments of C_M. Let $Agg(O_i, F_j) = \{O^m \mid O^m \in F_j, O_i \text{ aggregates } O^m\}$.

Given the set of fragments for C_M, the *attribute-link induced object-condition vectors for derived fragmentation* are defined as $ad_i = (ad_{i1}, ad_{i2}, \ldots, ad_{ik})$, where each vector component is expressed by the following formula:

$$ad_{ij} = \text{sgn}\left(\left| Agg(O_i, F_j) \right| \right) \quad (2)$$

For an object $O_i \in Inst(C_O)$ and a fragment F_j of C_M, ad_{ij} is 1 if O_i is linked to at least one object of F_j and is 0 otherwise.

Given the set of fragments for C_M, the *attribute-link induced characteristic vectors for derived fragmentation* are defined as $wd_i = (wd_{i1}, wd_{i2}, \ldots, wd_{ik})$, where each vector component is expressed by one of the following formulas (two alternatives):

$$wd^1_{ij} = \frac{\left| Agg(O_i, F_j) \right|}{\left| \bigcup_{O_l \in Inst(C_1)} Agg(O_l, F_j) \right|} \quad (3)$$

$$wd^2_{ij} = \frac{\left| \left\{ O_l \in Inst(C_1) \mid \text{sgn}\left(\left| Agg(O_l, F_j) \right| \right) = \text{sgn}\left(\left| Agg(O_i, F_j) \right| \right) \right\} \right|}{\left| Inst(C_1) \right|} \quad (4)$$

wd^1_{ij} gives the ratio between the number of objects in fragment F_j of class C_M linked to O_i and the number of all objects in fragment F_j linked to instances of C_O. Each wd^2_{ij} component gives the percentage of objects in C_O that aggregate in the same way as O_i objects from F_j. Two objects O_i and O_l are said to aggregate in the same way F_j if they are both either linked or not linked with objects from F_j. According to the second criteria, two objects are candidate to be placed in the same fragment of C_O in respect to F_j if they are both related in the same way to F_j.

Usually, the fragmentation of a class C_O is performed in two steps: primary fragmentation, according to query conditions, and derived fragmentation, according to the fragments of the member or owner classes. In our case the phases are merged into one single step capturing the semantic of both primary and derived fragmentations. For this we unify the characteristic vector and the attribute-link induced characteristic vectors for each object O_i of the class C_O and obtain the *extended characteristic vector*.

If the class C_O is linked with classes $C_{M1}, C_{M2}, \ldots C_{Mp}$, the *extended characteristic vector* we_i for object $O_i \in Inst(C_O)$ is obtained by appending the attribute-link induced characteristic vectors of $C_{M1}, C_{M2}, \ldots, C_{Mp}$ to the characteristic vector of O_i.

The *extended object-condition vector* ae_i for an object O_i is obtained in the same way by appending its attribute-link induced object-condition vectors to its object-condition vector.

Let *EOCM*(*C*) and *ECVM*(*C*) be the extended object-condition and characteristic matrices for class *C*.

3.2 Similarity between objects

The aim of our method is to group into a cluster those objects that are similar to one another. Similarity between objects is computed using the following *pseudo-metrics*:

$$\cos(we_i, we_j) = \frac{\sum_{k=1}^{n} we_{ik} \times we_{jk}}{\sqrt{\sum_{k=1}^{n} (we_{ik})^2} \times \sqrt{\sum_{k=1}^{n} (we_{jk})^2}} \quad (5)$$

$$d_M(we_i, we_j) = \sum_{k=1}^{n} \left| we_{ik} - we_{jk} \right| \quad (6)$$

$$d_E(we_i, we_j) = \sqrt{\sum_{k=1}^{n} (we_{ik} - we_{jk})^2} \quad (7)$$

Given two objects O_i and O_j, we define the following similarity measures between them in (8):

$$sim_{\cos}(O_i, O_j) = \cos(we_i, we_j)$$

$$sim_M(O_i, O_j) = 1 - \frac{d_M(we_i, we_j)}{|Inst(C)|} \quad (8)$$

$$sim_E(O_i, O_j) = 1 - \frac{d_E(we_i, we_j)}{|Inst(C)|}$$

Fig 1 presents the geometrical interpretation of the three similarity measures. Sim_E and sim_M are based on distance measures (Euclidian and Manhattan), while the third one (sim_{cos}) is the cosine of the angle between the two associated vectors. We expect the measures based on the 2 distances to generally capture the similarity from a distance point of view: as objects get closer they are more similar. The cosine similarity takes in account only the angle between the support vectors. Two objects are similar as their support vectors tend to have the same angle. It doesn't take in account the spread of the objects on the same support axis.

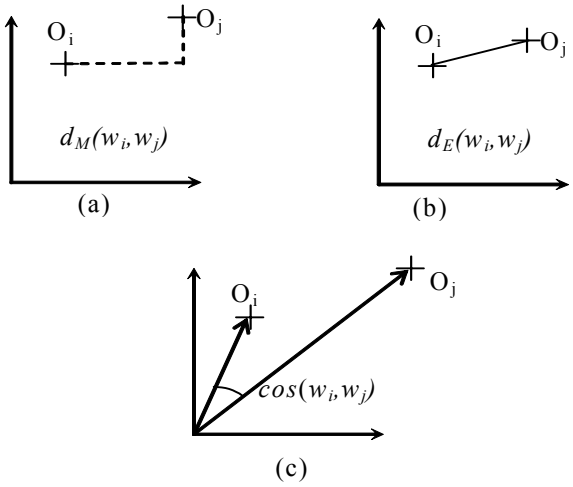

Fig. 1 – Geometrical interpretation of similarity functions for 2-dimensions.objects.

The cosine similarity is not defined for any two object-condition vectors. For extended vectors that have all components zero the cosine similarity measure is not defined. On the other side having all components zero means that the corresponding object is not referred by any application, so its resemblance with other objects is not significant in the fragmentation process in this case. It should be noted that all characteristic vectors have positive coordinates by definition.

4 The Hierarchical Agglomerative Fragmentation

The algorithm presented here is similar to the one in [11] and performs horizontal fragmentation on complex class hierarchies using the numerical database model presented above.

```
Algorithm HierachicalAggFrag is
Input: Class C, Inst(C) to be fragmented, the similarity function sim:Inst(C)xInst(C)-[0,1],
m=|Inst(C)|, 1<k≤ m desired number of fragments, EOCM(C), ECVM(C).
Output: The set of hierarchical clusters F={F1,…,Fk}
Begin
  For i=1 To Inst(C) do Fi={Oi};
  F={F1,…,Fm};
  While |F|>k do
    (Fu*,Fv*):=argmax(Fu,Fv)[sim(Fu,Fv)];
    Fnew=Fu*∪Fv*;
    F=F-{Fu*,Fv*}∪{Fnew};
    End While;
End.
```

At each iteration the algorithm chooses the two most similar clusters and merges them into a single cluster (*argmax(F_u,F_v)[sim(F_u,F_v)]*). As similarity between two clusters F_u and F_v, the average similarity of all pairs of objects is considered:

$$sim(F_u, F_v) = \frac{\sum_{a_i \in F_u} \sum_{b_j \in F_v} sim(a_i, b_j)}{|F_u| \times |F_v|} \quad (9)$$

The algorithm always ends up with k clusters representing the class fragments.

5 Results and Evaluation

This section illustrates the experimental results obtained by applying our fragmentation schemes on real and test object databases. Given a set of queries, we first obtain the horizontal fragments for the classes in the database; afterwards we evaluate the quality and performance of the fragmentation results. It should be noted that the order in which classes are fragmented is significant as it captures the semantic of query path expressions into the fragmentation process [12].

The sample object database in Fig. 2 represents a reduced university database. This is just an example reduced database for practically presenting the

average results obtained by running the algorithms on real and test databases. The inheritance hierarchy is shown in Fig. 2. It represents the average results obtained during our tests.

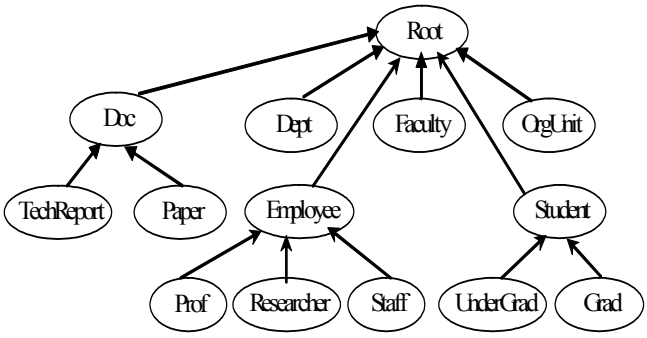

Fig. 2 – An example of a test database inheritance

Each measurement considers a set of applications running on the database. They are given in [11, 12, 13, 14]

For measuring the fragmentation quality we use the partition evaluator function presented in [13]. The cost formulas are:

$$PE(C) = EM^2 + ER^2 \quad (10)$$

$$EM^2(C) = \sum_{i=1}^{M}\sum_{t=1}^{T} freq_{ts}^2 * |Acc_{it}| * \left(1 - \frac{|Acc_{it}|}{|F_i|}\right) \quad (11)$$

$$ER^2(C) = \sum_{t=1}^{T} \min\left\{\sum_{s=1}^{S}\sum_{i=1}^{M} freq_{ts}^2 * |Acc_{it}| * \frac{|Acc_{it}|}{|F_i|}\right\} \quad (12)$$

As explained in [11], the EM term calculates the local access cost, while ER calculates the remote relevant access cost for all fragments of a class.

The fragmentation is better when the local (irrelevant) costs and the remote relevant access costs are smaller. Globally, PE measures how well fragments fit the object sets requested by queries.

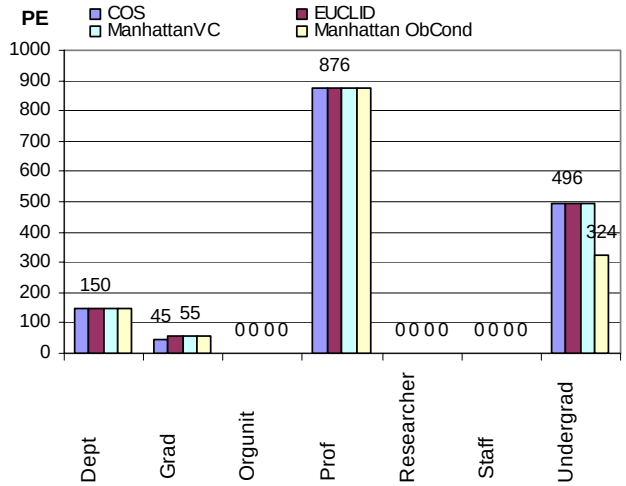

Fig. 3 Comparative PE costs for variant M1 on all classes.

Using the given query access frequency and other input data, the fragments above are allocated to N distributed sites. The presented method uses a simple allocation scheme that assigns fragments to sites where they are most needed.

In Fig. 3. and Fig. 4, M1 conforms to eqn (3) while M2 conforms to eqn (4) for expressing derived fragmentation. Classes are represented in each figure in the order they have been fragmented (from left to right). In each figure we compare the PE values on each fragmented class. It can be seen that all measures perform in about the same way for the first classes, even though the composition of resulting clusters is sometimes different. Classes have been fragmented in the same order for both M1 and M2. As we approach the right side of each figure we can see that the different composition of clusters of the already fragmented classes influences the resulting fragments (class Undergrad for example). This leads to a more clear separation in the induced PE costs for the Undergrad class for each similarity measure. It can be seen that the Euclidian similarity has an overall best place, as it obtains the smaller costs. The next measure in terms of performance is Manhattan applied on object-condition vectors (Manhattan ObCond). It can also be seen that generally the M2 method behaves better than M1 in terms of costs. The cosine (COS) similarity has generally the worst results. There are, however, particular situations where it outperforms the other similarity measures. Manhattan similarity applied on characteristic vectors (ManhattanVC) has in almost all cases an average behaviour.

The first conclusion that can be drawn from the above is that besides similarity measures, the length of dependency cycles in the aggregation hierarchy greatly influences the fragmentation results.

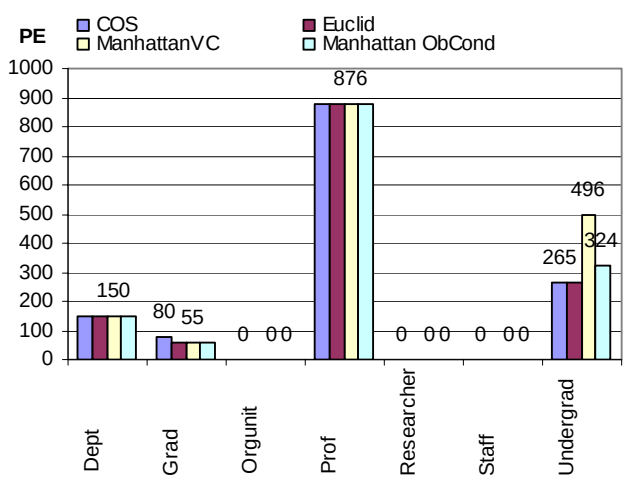

Fig. 4 - Comparative PE costs for variant M2 on all classes.

As the dependency chain is longer – i.e. the number of classes to be fragmented is higher – the small placement errors of objects in clusters tend to have a negative influence on the fragmentation of the classes at the end of the dependency chain. As long as the dependency chains are very short the choice of the similarity measure is insignificant.

In Fig.5 and Fig. 6 we show the overall results of *primary-only* and *complex* (*primary+derived*) class fragmentation for all classes and for both ways of constructing the extended characteristic matrixes. The left side of both figures contains the results of fragmentation in complex class hierarchies, while the right side displays the results of *primary-only* fragmentation for each similarity measure (P-COS, P-Euclid, P-ManhattanVC). Both figures show that for the *primary-only* fragmentation case the choice of the similarity measure doesn't affect much the resulting fragments. All similarity measures obtain similar costs, leading to the idea that they have an equal clustering power.

As seen in Fig.5 and Fig. 6 and as it has already been noted in [13], the results of *primary+derived* fragmentation are always better than *primary-only* fragmentation. This means that the disseminative power of the three similarity measures is not particularly influenced by the type of fragmentation: *primary-only* or *primary+derived*. In other words, the Euclidian measure would not behave better just because it is used in the context of primary-only fragmentation or primary+derived fragmentation, compared to the other similarity measures. Both figures show that the best result is obtained by the Euclidian measure (fig 6).

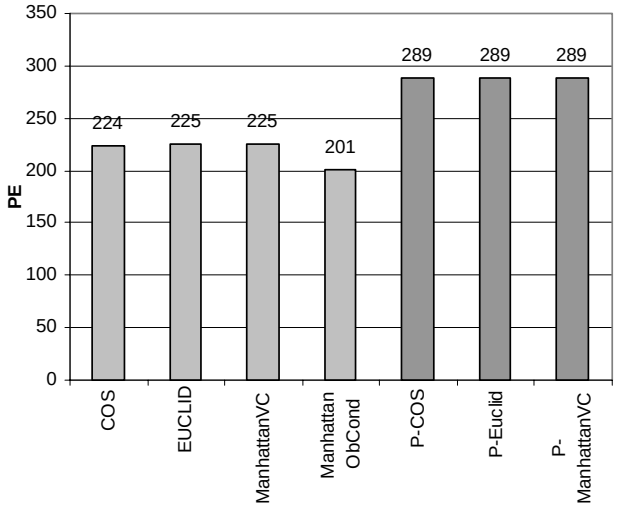

Fig. 5 - PE values for M1 on complex class fragmentation and primary fragmentation.

Overall best result is always achieved either with Euclidian similarity or Manhattan applied on object condition vectors. The best individual results, when combined with both vector construction methods (M1 and M2) are obtained generally by the Euclidian measure.

On the second place comes the Manhattan measure applied on object condition vectors. In this case the method for vectors construction has no influence whatsoever on the final results. It should be noted that Manhattan on object condition vectors always obtains good results – or even close to the optimal ones. The process of constructing the vector space is less elaborate than in the other cases because only qualitative information is managed. There is no explicit quantitative information about the way objects are inter-related. The quantitative information is automatically inferred by the algorithm through the similarity measure. The Manhattan similarity applied on object condition vectors is a good candidate for all cases where there is no prior knowledge on the generated vector data distribution.

The cosine measure shows an important dependence on the construction of the extended characteristic matrixes for derived fragmentation. It performs poorly for method M1 and quite well for method M2. It should be noted though that the cosine similarity cannot always be successfully applied in practice as it depends on the values of the individual components of the vectors. As shown in [14] there are cases where sets of clearly delimited objects cannot be correctly clustered using this measure.

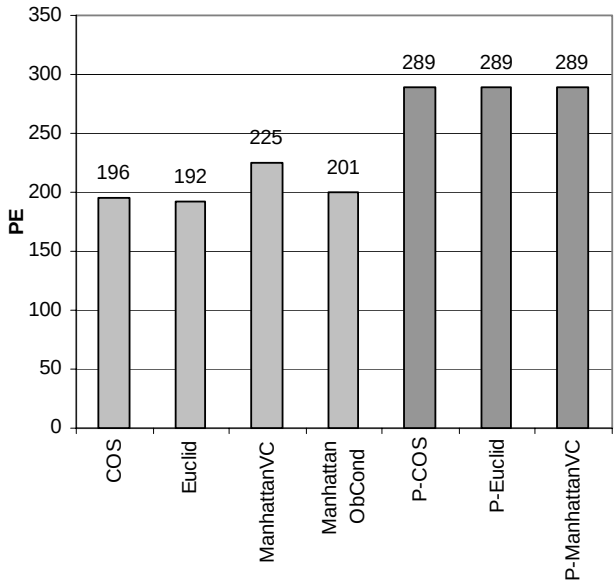

Fig. 6 PE values for M2 on complex class fragmentation and primary fragmentation.

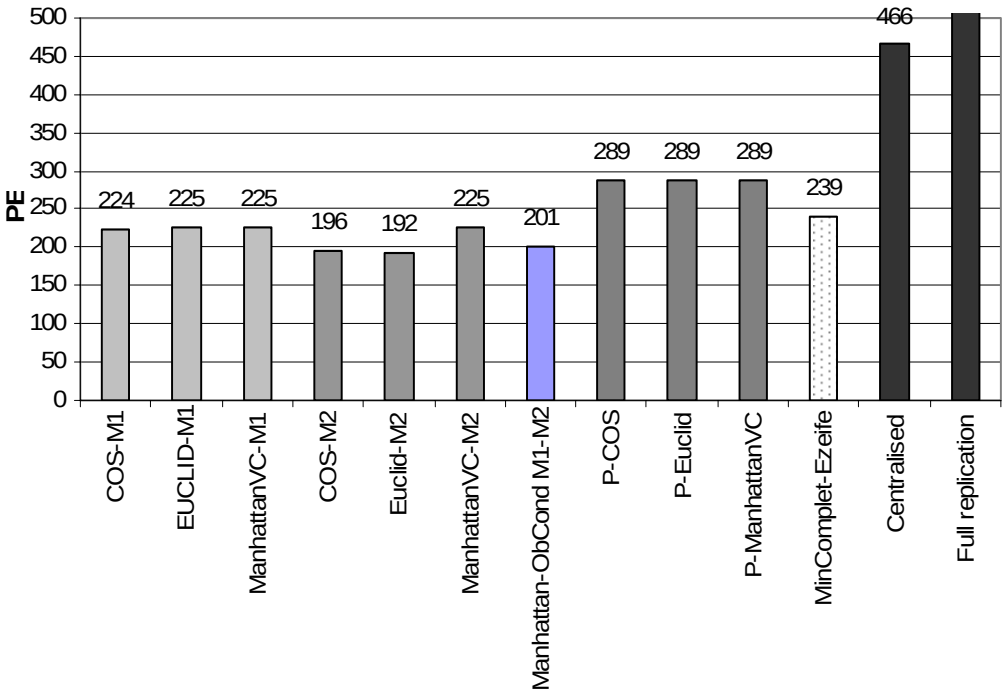

Fig. 7 Comparative PE values for complex class fragmentation and primary fragmentation.

Solutions have been proposed to correct this issue in [14]. On the other side, the Euclidian and Manhattan measures do not have this kind of degenerated behaviour.

Fig 7 presents the comparative study for both methods of constructing vectors (M1 and M2) using the hierarchical clustering algorithm. On the right side of the figure there are also the results for the primary only fragmentation, centralized version of the database and the fully replicated architecture. Also we compare the results of the hierarchical clustering algorithm with the results obtained by applying the fragmentation scheme developed in [2](MinComplet_Ezeife). It can be seen that overall, the centralized and fully replicated databases obtain the worst results. This is due to low concurrency factor in the centralized case and to the high cost of managing fully replicated copies of all objects on all nodes of the system (in the case of full replication). , As already said, primary only fragmentation is not strongly influenced by the choice of the similarity measure and always obtains weaker results than complex class fragmentation. The best overall score and the best individual score (fixed choice of method + algorithm) show a promising improvement over the average scored obtained by the fragmentation method developed by Ezeife in [2] (MinComplet_Ezeife). This algorithm uses a technique adapted from relational database fragmentation and applies it to object schemas.

Finally, it should be mentioned that the hierarchical agglomerative clustering method is characterized by the fact that once a step is done it can never be undone. This could help increase the gap between the results obtained by different similarity measures, while a more behaved clustering method that would allow correcting misplaced objects in the next few iterations is more likely to better reflect the real differences between similarities.

6 Conclusions and Future Work

This paper presents a comparative study of the influence of three similarity measures in the fragmentation process of an Object Oriented Database with complex class relationships, using a hierarchical agglomerative clustering algorithm. We show that the choice of different similarity measures influences the resulting fragmentation in a real database where the number of classes to fragment and the number of inter-class dependencies is likely to be important. Results on multiple database schemas show that the Euclidian and Manhattan (in some cases) similarity measure generally outperforms the other similarity measures. Our conclusions are based on initial suppositions that we support and confirm with experimental results. Given the empirical nature of *proofs only by experiments,* we aim to find formal, mathematical ways for proving the characteristics of different similarity measures applied in the fragmentation

process using clustering techniques. This would help improve the positive *impression* gained by applying clustering methods in the horizontal fragmentation process of distributed databases.

Last but not least, we show that using clustering algorithms for horizontal object database fragmentation proves to be a good choice when compared to other existing fragmentation methods. In this case all the similarity measures behave, when using hierarchical clustering, at least as well as other existing fragmentation algorithms. Generally the obtained results are better than with other existing algorithms. The choice for comparing with the algorithm developed in [2] was made because accurate fragmentation quality measurement was easier in this case. There is work in progress to assess the fragmentation quality for other fragmentation algorithms and compare it with those obtained by clustering.

As a final conclusion, it should be noted that the choice of similarity measures when fragmenting database schemas *by clustering* is important. The fragmentation quality varies with the similarity measure. There are, however similarities that behave almost constantly on a wide range of input data. The Euclidian and Manhattan fall in this category and their results can be trusted as good in the majority of cases.

References:

[1] Karlapalem, K., Navathe, S.B., Morsi, M.M.A. – "Issues in distribution design of object-oriented databases". In M. Tamer Ozsu, U. Dayal, P. Valduriez, editors, *Distributed Object Management*, pp 148-164, Morgan Kaufmann Publishers, 1994.

[2] Ezeife, C.I., Barker, K. – "A Comprehensive Approach to Horizontal Class Fragmentation in a Distributed Object Based System", *International Journal of Distributed and Parallel Databases*, 3(3), pp 247-272, 1995.

[3] Han, J., Kamber, M., *Data Mining: Concepts and Techniques*, The Morgan Kaufmann Series in Data Management Systems, 2000.

[4] Karlapalem, K., Li, Q. – "Partitioning Schemes for Object-Oriented Databases", *In Proceedings of the Fifth International Workshop on Research Issues in Data Engineering-Distributed Object Management*, pp 42–49, Taiwan, 1995.

[5] Karlapalem, K., Li, Q., Vieweg, S. – "Method Induced Partitioning Schemes in Object-Oriented Databases", *In Proceedings of the 16th Int. Conf. on Distributed Computing System* (ICDCS'96), pp 377–384, Hong Kong, 1996.

[6] Ravat, S. – "La fragmentation d'un schema conceptuel oriente objet", *In Ingenierie des systemes d'information* (ISI), 4(2), pp 161–193, 1996.

[7] Ezeife, C.I., Barker, K. – "Horizontal Class Fragmentation for Advanced-Object Modes in a Distributed Object-Based System", *In the Proceedings of the 9th International Symposium on Computer and Information Sciences*, Antalya, Turkey, pp 25-32, 1994.

[8] Bellatreche,L., Karlapalem, K., Simonet, A. – "Horizontal Class Partitioning in Object-Oriented Databases", *In Lecture Notes in Computer Science*, volume 1308, pp 58–67, Toulouse, France, 1997.

[9] Savonnet, M. et. al. – "Using Structural Schema Information as Heuristics for Horizontal Fragmentation of Object Classes in Distributed OODB", *In Proc IX Int. Conf. on Parallel and Distributed Computing Systems*, France, pp 732-737, 1996.

[10] Baiao, F., Mattoso, M. – "A Mixed Fragmentation Algorithm for Distributed Object Oriented Databases", *In Proc. Of the 9th Int. Conf. on Computing Information*, Canada, pp 141-148, 1998.

[11] Darabant, A.S., Campan, A.– "Hierarchical AI Clustering for Horizontal Object Fragmentation", *In Proc of Int. Conf. of Computers and Communications*, Oradea, pp 117-122, May, 2004 .

[12] Darabant, A.S, Campan, A. – "Optimal Class Fragmentation Ordering in Object Oriented Databases", *In Studia Universitatis Babes Bolyai Informatica*, pp. 45-54, Volume XLIX, Number 1 (2004), 2004.

[13] Darabant, A. S.**,** Campan, A., Cret, O, – "Hierarchical Clustering in Object Oriented Data Models with Complex Class Relationships", in *Proc. of the 8th IEEE International Conference on Intelligent Engineering Systems INES2004*, pag: 307-312, Cluj Napoca, 2004.

[14] Darabant, A. S., Campan, A., - "AI Clustering Techniques: a New Approach to Object Oriented Database Fragmentation", in *Proc. Of the 8th IEEE International Conference on Intelligent Engineering Systems, INES2004,* pag: 73-78, Cluj Napoca, 2004.

[*] This work has been partially funded from the CNCSIS research grant A_C No. 1/8 -2005, "Collaborative Information Systems In The Global Economy"