Anul L 2005

S T U D I A
UNIVERSITATIS BABEŞ-BOLYAI

INFORMATICA

2

Redacţia: 3400 Cluj-Napoca, str. M. Kogălniceanu nr. 1 Telefon 405300

SUMAR – CONTENTS – SOMMAIRE

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume L, Number 2, 2005

AN ONTOLOGICAL APPROACH TO WEB APPLICATION DESIGN USING W2000 METHODOLOGY

ANNA LISA GUIDO, ROBERTO PAIANO, AND ANDREA PANDURINO

Abstract. Applications users claim ever more quality in the software product; that does not mean only an improved performance but also a better usability or a more efficient interaction paradigm; in other words, a better "user experience". The designer focus is not on the system but on the user and his applications. This point of view creates new requirements that force the designer to use structured approaches for improving the quality of the design and of the final web application as well. If we consider the web and the applications on this channel (where the "user experience" aspects have an important role), it's possible to evaluate the weight of the new features in the software engineering process.
The new application features don't exactly match design methodologies, which must evolve in order to follow the new requirements. Many methodologies are introducing new primitives and structures that did not exist before. The well-known UML describing language (and its MOF metamodel) is evolving; the Jim Connallen's Web Application Extensions for UML are an example of the need to update the language to these application trends. Supporting the UML goodness, in this paper we describe our experiences in the use of Web Application (WA) design methodology called W2000 (which uses UML) and in the use of ontology language to represent the same methodology.

1. Introduction & Background

During the standard application development, the designer can use well-known traditional software engineering techniques that focus on the system aspects, such as data and transactions; on the contrary, the new user requests put the stress on the hypermedia aspects, such as information, navigation and presentation aspects; the combination of these factors improves the user experience and produces a better quality product. However linear the modeling technique and however legible the language used may be, the need to describe in sufficient details the multiple static and dynamic aspects of the application is a highly complex task. The

Received by the editors: September 19, 2005.

1991 *Mathematics Subject Classification.* [**Information Systems**: hypertext navigation, interfaces, decision support etc.](68U, 35).

1998 *CR Categories and Descriptors.* D.2.0 [**Software**]: Subtopic – *Software Engineering-General* .

designer has to manage the individual aspects and predict the inevitable interactions among them; in many cases, a navigation choice that is valid for one device or context turns out to be completely wrong or unworkable in another; thus, the complexity increases and the methodological instruments to manage it seem to be insufficient.

The complexity to design a modern application is very high; in particular, we need to combine the multi-user and multi-device features with the typical transactional aspects. To obtain high quality applications, a structured methodological approach to model all the user experience aspects is needed.

The design methodologies used to model traditional application are not sufficient to meet the new application requirements and to improve the quality of the final product; therefore, new design methodologies, in particular in the web application domain, are born such as OOHDM [1] ,WSDM [9] , WebML [13] , HDM [3] . The traditional design tool is also updated to include new primitives and construct.

UML (de facto standard in model representations and independent of methodology) must evolve to include the new features: UML [12] defines a formal mechanism to extend its semantics. The "Extension Mechanism" allows to customize and extend UML with the use of stereotypes, constraints, tag definition and tagged values. (The same mechanism, defined in UML 1.x, is also valid in UML 2.0 [11]).

In this paper, we describe our experience in the use of web application design methodology (called W2000 [6] [5]) with UML and highlight the benefits and the problems connected with the large use of stereotypes. Considering that stereotype is the only possible choice to describe the primitives not directly implemented into the UML paradigm, our studies could be applied to all the methodologies that make a large use of stereotypes.

Many problems arise from MOF (core of UML approach). In this paper, we highlight these problems and try to use the ontological language as an alternative.

2. Our experience with MOF approach: W2000 methodology

W2000 methodology uses UML and, thus, has a MOF representation (proposed by OMG [16]) made up of three packages: Information, Navigation and Publishing Model. Each package has sub-packages where a class diagram represents W2000 methodology primitives and relationships among them. Packages, sub-packages and OCL constraints [10] (OCL is an external language used to express all the methodology constraints not directly supported by UML) represent, in MOF terminology, the design methodology *metamodel*, that is a collection of concepts and their relationships; in our case it is a collection of W2000 methodology primitives. The specific WA *model* is produced according to this metamodel. Although MOF W2000 methodology approach [7] is formally correct, there are some practical problems (related to UML1.x version and also present in UML2.0), due to dependence of UML on MOF.

The main problems that we highlight are:

Metamodel and semantics: In MOF approach there are a lot of primitives not directly supported by UML and thus all primitives are represented by stereotypes so metamodel semantic coincides with stereotypes semantic. Furthermore, the lack of semantics creates confusion to the unskilled designer during the practical applications of modeling concepts. The explicit presence of semantics helps the designer to understand how the modeling concepts should be used.
Another problem strictly connected to *semantics concerns semantic relationships among classes*: we can observe that MOF allows to use only two relationships aggregation and association while in W2000 metamodel it is necessary to define specific methodology relationships with its relative semantics.
Relationships among classes: another problem is that relationships among classes are lost in the transition from metamodel to model. Supposing that in the metamodel we have a relationship among classes: when we define the model, relationships must be redefined because they are not inherited by the model. This problem could be solved creating intermediate classes to represent the relationships; the disadvantage of this solution is that it will make the model unreadable for the large number of intermediate classes.
Finally, in MOF approach, if an attribute is the same for two different concepts it is defined once for each class (each attribute is strictly connected to each class). This MOF limit creates confusion letting designers think that each attribute has its semantic but it is not true.
Model Flexibility: another problem is the flexibility, that is the possibility to enrich the model with new primitives or to add new characteristics to the primitives already defined. The solution proposed by UML (both 1.x and 2.0) is to enrich the UML metamodel with Extension Mechanism but this mechanism require a good knowledge of UML and then may require a lot of time compromising the evolution of the WA design methodologies. Another problem related to the language evolution concerns the *unique name assumption* principle: in UML approach different words must refer to different objects. In order to meet WA evolution, it is often necessary to define new version of concepts (defined before) and to use the same name. The unique name assumption makes it impossible. The UML and MOF do not support the *dynamic classification of classes*. It is possible that, when metamodel is extended to include the methodology evolution, two classes must be replaced by their intersection: the instance of the new class contains both previous classes. This is not possible in UML, since every instance can be only the instance of a class and not the instance of two classes at the same time.
Standard description of the model: it is important to have a machine readable description of the model. In MOF approach we use XMI (OMG standard) as a model representation language (but we are free to use any XML description). There are different formats according to the graphic editors that produce XMI but a model description must be understandable in an easy and univocal way by software agent and preferably should be a W3C standard.

An external language to represent constraints: in MOF approach there is an external language, OCL, to describe the methodology constraints. OCL is hard to understand by designer who are unskilled both in W2000 methodology and in the OCL language. If the designer is also an OCL language expert, the model obtained will be formally correct but difficult to understand by developers.

3. Ontological approach to web application design methodology representation

The main problem related to MOF approach is the *lack of semantics* that brings to represent all the W2000 primitives through the same UML primitives: stereotypes.
Another problem is that WA methodologies must follow the evolution of WA requirements and the methodology representation language must meet this evolution too: in MOF approach it is often hard to represent the new WA requirements and the solution is to add new stereotypes that will be increasing the semantic problem. Finally, the language used to represent the methodology must be easy to learn and read for designers and guidelines must be provided during the whole design process; instead, the necessity to learn an external language such as OCL may be a problem.
Considering these comments, the use of UML as a representation language, in our experience, forces methodology to adapt itself to the paradigm imposed by UML, and thus semantics turns flat and it is hard to read and understand the model. *We need to adapt the representation language to methodology and not methodology to representation language as in UML approach.* To solve this problem, we need a *new language* easy to use, more flexible and expressive than UML, allowing to represent directly methodology primitives and helping the designer in his/her task through a better semantics. We explored several alternatives but the use of semantic language able to describe, in general, a domain of knowledge seems more flexible and compliant with our goal which consists in representing WA methodology in a more meaningful way. The language used is OWL [14], the *ontological language*: a text language without graphical notation. Our use of ontology is quite different from the semantic web which is the traditional one. We use OWL to represent both WA design methodology and the WA model obtained through this methodology. We choose to use ontology as a language to represent a methodology starting from its definition: "*ontology is a formal, explicit specification of a shared conceptualization*" (Gruber in 1993 [2]), that is ontology is an abstraction of some concepts made through the definition of its peculiar characteristics. Defining a *metamodel* as a set of concepts and rules necessary to specify a model in the domain of interest, we can state that: "*A valid metamodel is an ontology but not all the ontologies are expressly modeled as metamodel*" [15]. From these considerations we state that *it would be possible to use the ontological languages to express a metamodel and to obtain from it a model.*

3.1. **OWL language in a nutshell.** Before explaining the use of OWL in our approach, it's necessary to understand its main concepts. OWL primitives are:

- *Classes*: allow the abstraction of some concepts. Each class has a set of *properties* (each one for specific concept characteristics). A class would be composed by subclasses.
- *Properties*: There are two types of properties: *DataType* specific to each class and *ObjectProperty* used to create a link between classes. Object-Property has both domains: class (to which the property is connected) and range (the possible values of the property). In each class we can indicate "restrictions" that define constraints.
- *Individuals* are objects that have the characteristics defined by classes and properties. Both classes and properties may have individuals.

3.2. **The architecture of our approach.** MOF approach, proposed by OMG is based on a 4-level architecture. It allows to define a language for the methodology representation and to use this language for model definition. The use of a 4-level architecture is a good choice because it allows to separate different levels of abstraction so we use it but with few changes. Fig. 1 shows the 4-level architecture compared with our ontological approach.

In MOF approach, M3 level is the level where the MOF language, that is the

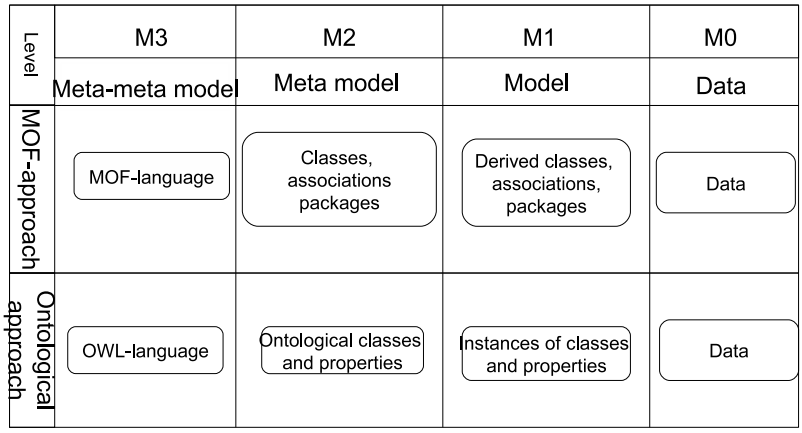


FIGURE 1. MOF and Ontological approaches compared

abstract language used to describe MOF metamodel, is defined. MOF is Object Oriented and strictly connected to UML: UML notation is used to express MOF metamodel. In the ontological approach we use in M3 level OWL language instead of MOF language.

In the M2 level we define, both in MOF and ontological approaches, a metamodel, that is the abstract language that allows the definition of the model in the M1 level. In MOF approach the metamodel definition is made up of classes, association, packages and OCL constraints aimed to representing the particular methodology. In the ontological approach we define the metamodel through ontological classes, that allow to define the methodology primitives, and ontological properties (DataType and Object Property) that allow to give other details about each methodology primitive. Object Property, using domain and range definition, represents the semantic network of the methodology. Restriction on properties allows to define methodology rules without using any external language.
M1 level is the level where the designer, using metamodel (guidelines for methodology), designs the specific application. Finally M0 level represents data of a specific model.
From a technological point of view, in our ontological approach, we use both in the M2 and M1 level, Protégé[4], an open source ontology editor developed by Stanford University. Protégé manages separately the metamodel (classes and properties) and the model (the instances); thus, it's possible to define the metamodel (M2 level) and model (M1 level) layer. To create the ontological metamodel we followed the guidelines defined by the Stanford University researchers [8].

4. Our experience with ontological approach: W2000 methodology

To understand the benefits of our ontological approach we have to explain our experience with WA design methodology W2000: we explain how problems presented in section 3 can be solved.
First of all, it's possible to give a *semantic meaning* to each W2000 primitive directly through OWL without the UML stereotypes in order to improve the designer comprehension of the model. In figure 2 there is the OWL code of the relationship between Entity and Component(Entity and Component are W2000 methodology primitives): the ObjectProperty "madeOfComponent". It has a restriction that forces each entity to have at least one component. There is also the relationship between "Component" and "Entity"("belongsToEntity") defined as inverse of "madeOfComponent": this allows to read in a bi-directional way the concepts linked to the Object Property.

```
<owl:ObjectProperty rdf:about="#madeOfComponent">
    <owl:inverseOf>
        <owl:ObjectProperty
        rdf:ID="belongsToEntity"/>
    </owl:inverseOf>
```

Figure 2. OWL code: "inverse of"

The ontological approach allows to define an attribute once and to use it in different primitives: the domain of this attribute will be the union of all classes

that represent the primitives in which the attribute will be used.
Concerning the *model flexibility*, the architecture proposed allows to add new primitives to metamodel inserting new classes and its properties into the metamodel or new properties to classes. The changes to metamodel are very fast and do not require technical competences about any language: it will be available immediately in the model and also the existing model will be updated automatically with new elements. The OWL language allows to define intersection or union between two classes (dynamic classification of classes) and the equivalence between classes (the unique name assumption is not valid in OWL).
The *relationships among classes* in the transition from the metamodel to the model is naturally taken. The OWL metamodel is immediately ready to create a WA model in the OWL language: to create a model starting from metamodel is sufficient to add instances of classes and properties defined in the metamodel. In our approach the property instances allow to tie together classes without adding, as in MOF approach, classes that represent the relationship between classes defined in the metamodel.
The problem of *standard description of the model* is solved with the OWL use (recommended by W3C); also with the same language it's possible to represent both the metamodel and the model.
The ontological approach avoids the use of *external language to represent constraints* that are described directly in the OWL language using its restriction. For example, to express that an Entity is made up of at least one Component, a restriction on the property "madeOfComponent" is defined (Fig. 3).

```
<owl:Restriction>
   <owl:minCardinality rdf:
datatype="http://www.w3.org/2001/XMLSchema#int>1
   </owl:minCardinality>
<owl:onProperty>
<owl:ObjectProperty rdf:about="#madeOfComponent"/>
```

FIGURE 3. OWL code Property restriction

This approach helps the designer to understand the methodology and then to make a design following all the rules presented in the methodology *without learning an external language*. The model obtained is also understandable by developers who do not have knowledge of the OCL language.

5. Conclusion and future work

Considering that WA design methodologies have peculiar characteristics, different from standard applications, and that the traditional design approaches are not sufficient to take into account all these new features, we highlight the limits of MOF. Then we introduce a new ontology approach that uses the OWL language.

Our approach applied to W2000 methodology is more flexible and allows methodology to adapt itself to WA evolution. The model obtained from the metamodel is more clear, effective and complete than the one obtained with MOF approach and makes the designer work easier. In fact, a better semantics provides clear guidelines for the designer.
At present our work is focused on the development of an editor able to design WA through W2000 methodology and to obtain an OWL description of the model. We are also planning to extend metamodel to include a good description of the operation in an ontological way.

References

[1] D. Schwabe, G. Rossi, S. D. J. Barbosa: *Systematic Hypermedia Application Design with OOHDM* Proc. ACM Conf. Hypertext '96 - Mar 1996 - ACM Press.
[2] Dieter Fensel Ontologies: *A silver Bullet for Knowledge Management and Electronic Commerce* Second Edition, Revised and Extended. Foreword by Michael L.Brodie. Springer-Verlag Berlin Heidelberg 2004 ISBN 3-540-00302-9.
[3] F. Garzotto, P. Paolini, D. Schwabe: *HDM - A Model for the Design of Hypertext Applications*, in Proceedings ACM Hypertext '91, S. Antonio (TX, USA), ACM Press, Dec. 1991.
[4] http://protege.stanford.edu/.
[5] L. Baresi, F. Garzotto, and P. Paolini, *Extending UML for Modeling Web Applications*, Proceedings of 34th Annual Hawaii International Conference on System Sciences (HICSS-34). IEEE Computer Society, 2001.
[6] L. Baresi, F. Garzotto, Paolo Paolini, *From Web Sites to Web Applications: New Issues for Conceptual Modeling*, Proceedings WWW Conceptual Modeling Conference, Salt Lake City, October, 2000.
[7] L.Baresi, F. Garzotto, M.Maritati: *W2000 as a MOF Metamodel* In Proceedings of The 6th World Multiconference on Systemics, Cybernetics and Informatics - Web Engineering track. Orlando (USA), July 2002.
[8] Natalya F. Noy and Deborah L. McGuinnes Stanford University, Stanford, CA, 94305 *Ontology Development 101: A Guide to Creating Your First Ontology*
[9] O.M.F. De Troyer, C.J. Leune, *WSDM: a user centred design method for Web sites*, Proceeding of WWW7 Conference, April 14-18, Brisbane, Australia, 1998.
[10] OMG, ad/97-08-08, *Object Constraint Language Specification*, version 1.1, 1 September 1997.
[11] OMG. *UML 2.0 Superstructure Specification. Version 2.0*, September, 8 2003.
[12] OMG. *Unified Modeling Language Specification.* Version 1.4, September 2001, http://www.omg.org/uml.
[13] S. Ceri, P. Fraternali, A. Bongio: *Web Modeling Language (WebML): a modeling language for designing Web sites*, Proc. Int. Conf. WWW9, Amsterdam, May 5 2000.
[14] *OWL Web Ontology language Reference* W3C Recommendation 10 February 2004.
[15] www.metamodel.com : *What are differences between a vocabulary, a taxonomy, a thesaurus, an ontology and a metamodel.*
[16] www.omg.org.

Dipartimento Ingegneria dell'Innovazione Universitá di Lecce Via per Arnesano, 73100 Lecce, Italy Tel: +39 0832 297229 Fax: : +39 0832 297279
E-mail address: annalisa.guido@unile.it, roberto.paiano@unile.it, andrea.pandurino@unile.it

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume L, Number 2, 2005

UNSUPERVISED SINGLE-LINK HIERARCHICAL CLUSTERING

DANA AVRAM LUPŞA

Abstract. There are many clustering techniques presented in the literature. The particularity of single-link clustering is that it rather discovers the clusters as chains. We aim to identify a method to apply the single link clustering technique so that: it discovers the first level clusters and the user doesn't have to provide any sort of a parameter. We focuses on clusters that are well separated, and so, which have to maximize the intra-cluster similarity and minimize the inter-cluster similarity. We evaluate the method on a two dimensional space, that is planar points.

1. Introduction

In the literature, a vast collection of clustering algorithm [6], [2] is available. There is no clustering technique that is universally applicable in uncovering the variety of structures present in multidimensional data sets. Studies of clustering were made from a long time ([4], 1987), but they also constitute recent preocupations of reserchers ([7],2002). A list of materials about detecting clusters and the number of clusters can be found in [10].

All clustering algorithms will, when presented with data, produce clusters – regardless of whether the data contain clusters or not. If the data does contain clusters, some clustering algorithms may obtain better results than others. We focus on data sets that do contain clusters. More than that, we will also request the data to be relatively uniform distributed inside the clusters.

In the literature, the classification is a method that assigns objects to predefined groups; it is a sort of supervised learning technique [5]. Clustering infers groups based on inter-object similarity; it tends to be an unsupervised learning technique. But clustering techniques needs a semi-supervised parameter - that is the number of clusters, and/or the error or/and a maximum number of steps to be executed. In this paper we suggest a method applicable to hierarchical clustering that do not need any parameter. By using single-link hierarchical clustering, the method

Received by the editors: June 27, 2005.

2000 *Mathematics Subject Classification.* 65-05, 65S05.

1998 *CR Categories and Descriptors.* I.5.2. [**Computing Methodologies**]: PATTERN RECOGNITION – *Design Methodology – Classifier design and evaluation*; I.5.3. [**Computing Methodologies**]: PATTERN RECOGNITION – *Clustering – Algorithms* .

we suggest discover the clusters in which the data is grouped, if there are such clusters and they are well identified.

2. Hierarchical Clustering

The hierarchical clustering algorithm was first defined by S.C. Johnson in *Hierarchical Clustering Schemes* , Psychometrika 1967. Given a set of N items to be clustered, and a $N * N$ distance (or similarity) matrix, the basic process of hierarchical clustering is this:

Step 1: Start by assigning each item to a cluster, so that if you have N items, you now have N clusters, each containing just one item. Let the distances (similarities) between the clusters be the same as the distances (similarities) between the items they contain.

Step 2: Find the closest (most similar) pair of clusters and merge them into a single cluster, so that now you have one cluster less.

Step 3: Compute distances (similarities) between the new cluster and each of the old clusters.

Step 4: Repeat steps 2 and 3 until all items are clustered into a single cluster of size N.

Step 3 can be done in several ways, which is what distinguishes *single-linkage* from *complete-linkage* and *average-linkage* clustering.

In *single-linkage* clustering (also called the *connectedness* or *minimum* method), we consider the distance between one cluster and another cluster to be equal to the *shortest* distance from any member of one cluster to any member of the other cluster. If the data consist of similarities, we consider the similarity between one cluster and another cluster to be equal to the *greatest* similarity from any member of one cluster to any member of the other cluster.

In *complete-linkage* clustering (also called the *diameter* or *maximum* method), we consider the distance between one cluster and another cluster to be equal to the *greatest* distance from any member of one cluster to any member of the other cluster. In *average-linkage* clustering, we consider the distance between one cluster and another cluster to be equal to the *average* distance from any member of one cluster to any member of the other cluster.

The complete-link algorithm produces tightly bound or compact clusters . The single link algorithm, by contrast, suffers from a chaining effect [8]. The single-link algorithm is more versatile than the complete link algorithm.

In most of the applications, the goal of clustering is to identify some clusters in the given data. The hierarchical clustering algorithm won't stop unless we provide a *stop condition*. This can be the number of iterations of step 2 and 3, the number of clusters that we want to obtain or a certain error indicated by an evaluation of obtained clusters. Those stop conditions are chosen accordingly with

some extra information about the data of the problem we want to solve. To choose the appropriate stop condition is not always an easy task.

The problem we address in this paper is to find the clusters by using single-link hierarchical clustering (which, on the other hand is one of the most simple and intuitive methods) without having to bother about providing a stop condition. The idea is to limit the similarity values between elements that can be used to compute the similarity between 2 clusters (step 3) to the best similarities. We will refer to the chosen number of best similarities as *NBS* (Number of Best Similarities).

3. The Basic Idea

The question to which we are going to answer now is how many *similarities are used* for building k clusters for N elements, where $1 \leq k \leq N$.

In single linkage, in each formed cluster, we can build a tree formed by similarity links used to build that cluster. If n_i are the number of the elements in the cluster, than there are $n_i - 1$ similarities used. With the notation:

$$UsedSimi(n_i) = number\ of\ similarities$$

the next relation holds:

$$UsedSimi(n_i) = n_i - 1$$

Suppose there are k clusters build and the number of elements in each cluster are $n_1, n_2 \ldots, n_k$. The next relations hold:

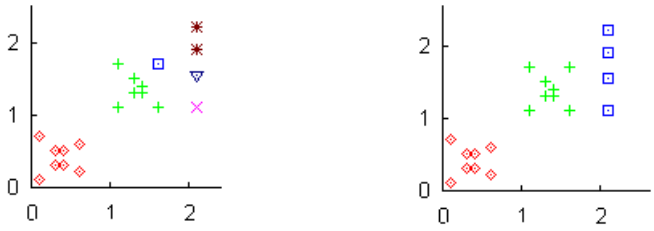
(1) $n_1 + n_2 + \ldots + n_k = N$
(2) the total number of used similarities is the sum of the numbers of used similarities for each cluster; that is:
$AllUsedSimi = \sum_{i=1}^{k} UsedSimi(n_i) = \sum_{i=1}^{k}(n_i - 1) = N - k$
(3) $1 \leq k \leq N$

The idea is to consider only the $N - k$ best similarities to build the clusters. The clustering process will end when there is no similarity left from the best NBS that can be used by the clustering process.

During the clustering process, we are not dealing only with the best similarities among elements to be the similarities that are used for building the cluster. Some of them are lost for other intra-cluster similarity values. This is one important property of the method and we are going to find a way to workaround the error introduced by this property and also to profit by this.

See, for example, the clusters that are produced by using this method and first N (= 20) best similarities, that are larger than any $N - k$, in Fig. 1(a) [1].

[1]The graphics is made by using gnuplot



(a) NBS=N (b) Adjusted clusters

FIGURE 1. Clusters without and with elimination of singular elements

Some of the best similarities are extra-links [2] inside the clusters that are built. The number $N - k$ is too small, the chosen NBS must consider also that the extra-links in the clusters are similarities values that can have large value but will not be used by the algorithm. On the other hand, our formula depends on k and this is an undesired thing. Consequently, if we consider as NBS the maximum value that can be obtained by the above formula, we accomplish two requirements: get a larger value as NBS in order to ignore some extra-links inside the cluster and also have the advantages of having a NBS value that does not depend on k:

$$NBS1: \ max_k(N-k) = N$$

On the other hand, the decision to take only the best N similarity values remains sometimes a little too strong, as we can see in Fig. 1(a). The reason is that maximization of $(N-k)$ does not cover enough extra-links inside the clusters. But we can consider that, there is still a good chance that NBS correctly identifies clusters nuclei.

If we consider that the clusters nuclei are correctly identified, we can improve the result by continuing to group clusters with one element (in single-link hierarchical manner) until the moment when the clusters that must be unified are nuclei determined by NBS.

The question that arises is what can be considered as a cluster nucleus and when an element is *singular*[3], that means that is not part of a nucleus. If the elements are grouped in clusters, we expect that there are elements enough close so that they would be put together in a cluster - for each natural cluster which the data contains. Consequently, we will consider as singular elements the ones that are singular in clusters, and as nuclei - clusters with more than one element.

[2] In this case, we have notated as extra-link the links among elements in a cluster that are not used by the single-link clustering algorithm to form the cluster

[3] We say that an element is singular if it forms a cluster by himself

By applying this method on the same data sets as in Fig. 1(a), the cluster set build in this case is indicated on Fig. 1(b).

4. The Choice of NBS

4.1. **Implementation Issues.** We evaluate our method by hand, by using sets of points in a two dimensional space and evaluating the computed clusters. We use as similarity a measure derived from the *Euclidian metric distance*. If d is the Euclidian distance between two points p and q:

$$d(p,q) = \sqrt{\sum_{i=1}^{2}(p_i - q_i)^2}$$

then the similarity between them can be computed by using the formula:

$$similarity(p,q) = \frac{1}{d(p,q)} \quad (A)$$

Of course, this metrics holds if there are not 2 elements with the same coordinates. If this condition is not satisfied, we can use:

$$similarity(p,q) = \frac{1}{d(p,q)+1} \quad (B)$$

In our experiments, we are in case when there are not 2 elements with the same coordinates, and we have taken formula (A) for computing the similarities.

Computation with real values introduce small errors. On the other hand, we are not interested to put in different clusters the elements that are closest. That is why we are interested in ignoring small variations of similarities. When we identify the best similarity values, we consider as acceptable a variation that is not very small and that depends of the similarity values. In calculus, the variation is interpreted as an acceptable error. We used an *error of 10%* from medium difference between two similarities values, computed with the next formula:

$$\begin{aligned} error &= \frac{(\max - \min)}{(\text{number of similarities})} \\ &= \frac{1}{10} \times \frac{\max - \min}{n*(n-1)/2)} \\ &= \frac{1}{5} \times \frac{\max - \min}{n*(n-1)} \end{aligned}$$

where:

max**:** maximum similarity value
min**:** the minimum similarity value, greater than 0

On the other hand, we are working with a $N * N$ similarity matrix, where the elements on the main diagonal have a special value and are not used. Each other similarity value from the matrix is repeated twice. This means that we use a number of $2 * NBS1$ values from the similarity matrix.

4.2. **Fine tuning the parameter.** The way in which we build the value $NBS = N$ would say that the choice is close to the best one, but it is not necessary the best choice. In order to test this, we will establish a measure of cluster accuracy and we will study the effects of small variation of NBS value.

As is universally accepted there is no universal measure for evaluating cluster sets. We choose for evaluation the next measure:

$$measure = min_{C_i,C_j} dist(C_i, C_j) - min_{p \in Ci}(max_{p,q \in C_i} dist(p, q))$$

and we will earn from the fact that this measure performs well if there are no singular points that must be part of a cluster and they are not. Because it is a measure of optimum (max or min), not of average, and one wrong cluster will modify the result of the evaluation. As we *continue clustering* starting from a determined set of nuclei and as long as the set of nuclei remains unchanged, we get a good chance of eliminating singular points, and so, eliminating the wrong clusters.

The method we propose is to take as the result the set built for NBS greater than N and smaller than $3/2N$ that get a score value at least double compared to the score for the set built for N, or the set built for $NBS = N$ otherwise. We ask for the score value to be double because we considered that, if the improvements are not big, than the better score could appear by cause of the natural tendency of the evaluation function to grow with the decrease of the number of the determined clusters set.

We experimented the result by taking as NBS the values that approximate *the interval*: $N-N/2 \ \ldots \ N+N/2$. That is, we considered as the first best similarities those with the values ranked between $(1; N) \ \ldots \ (1; 3N)$ from the $N*N$ similarities of the similarity matrix. The distinct sets of obtained clusters are shown in Fig. 2. Each set of clusters are accompanied by a short explanation as follows:

- on the first line: the score of the cluster set
- on the second line: the smallest value of $2*NBS$ (value ranks are between N and $3 * N$) that obtain one set of clusters

One very important thing the experiment enlightens is that the result is *not strongly dependent of* the chosen NBS value, in the sense that small variations of NBS keep the result (clusters set) unchanged. Note that there are only 9 distinct cluster sets for a NBS value that vary between $NBS = N = 46$ and $NBS = 3N = 138$, that is for 92 distinct values for NBS.

The experiment confirms that the best result is very *close to* $NBS = N$. Another thing the experiments points is that when NBS grows bigger, the clusters grow bigger too and are less well identified, while the evaluation function value grows either. That is why we cannot use only the evaluation function to identify the best set of clusters identified during the hierarchical clustering process.

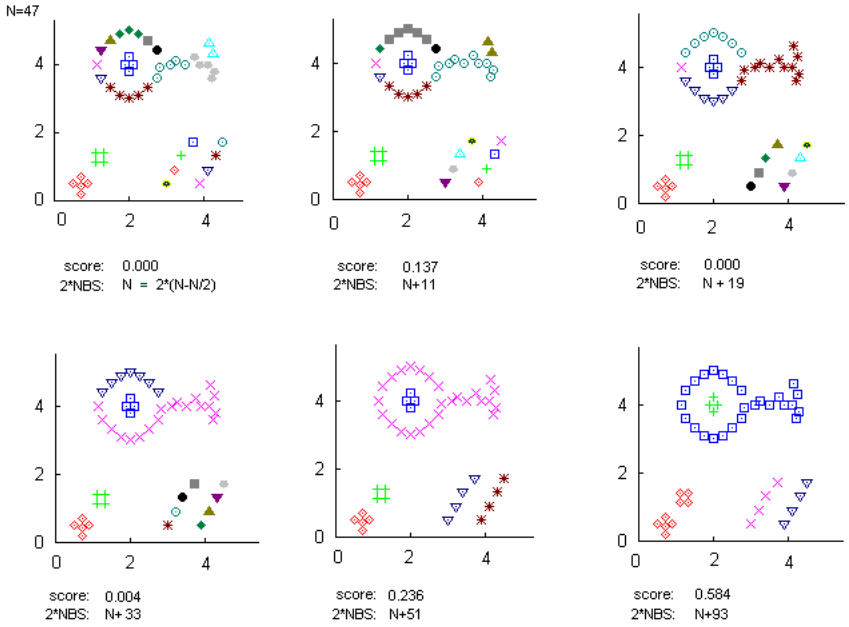


FIGURE 2. All different clusters levels when considering best similarity values, with ranks between (1 and $N/2$) and (1 and $3N/2$

We also studied the behavior of other 10 data sets. Based on the results of these experiments, we are going to improve the method by taking as result the first set of clusters built for an NBS value that satisfies $NBS \geq N$ and $NBS \leq N + N/2$, if the set is evaluated as being better than the set build for $NBS = N$.

4.3. **Best choice. Experimental results.** We have taken 5 different types of input sets, with 2 examples for each type. That would be a collection of 10 data sets. We choose data with characteristics presented in the table 1.

The results of clustering processes are presented in Fig. 3.

Evaluation. We take as precision the elements that are considered by a human subject that are well grouped. We are looking for the most general clusters a human judge would identify. As we evaluated, there are 8 correct cluster results. We indicate the sets 4A and 5A as not being correct. That would indicate a percentage of 80% correct clusters.

4.4. **Discussion of special cases.** One case of bad distributed elements is the case when there should be clusters with few elements in a cluster. The figure 4 presents cluster changes with the variation of the number of points.

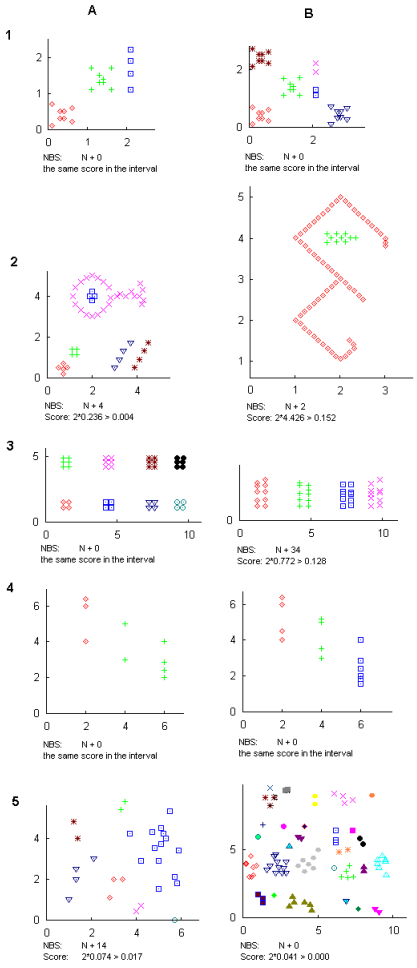


FIGURE 3. Clusters for different values for NBS

If the elements are not relatively uniform distributed inside the clusters, the method won't always obtain good results. In figures 4 and 5 we have the results from data more or less well distributed.

Figure 5 illustrates that, if the clusters are well identified, the result suffers very little from small variation of elements' coordinates.

One could say that the cluster set 3 in the fig. 5 is inaccurate. But what would be the result if the distances between the elements in clusters 3, 4, 5, 6 are modified

Data characteristic	**Identification in fig**
well grouped in clusters	1, 2, 3
well grouped in clusters and known as with problem for hierarchical clustering (there are differences between single-link and complete-link hierarchical clustering)	2
bad cluster identification many points but sparse data	5
small number of elements in a cluster	4
many clusters with a small number of elements in a cluster	3A
small number of clusters with many elements in a cluster	2B

TABLE 1. Characterisitc of the data in the figure 3

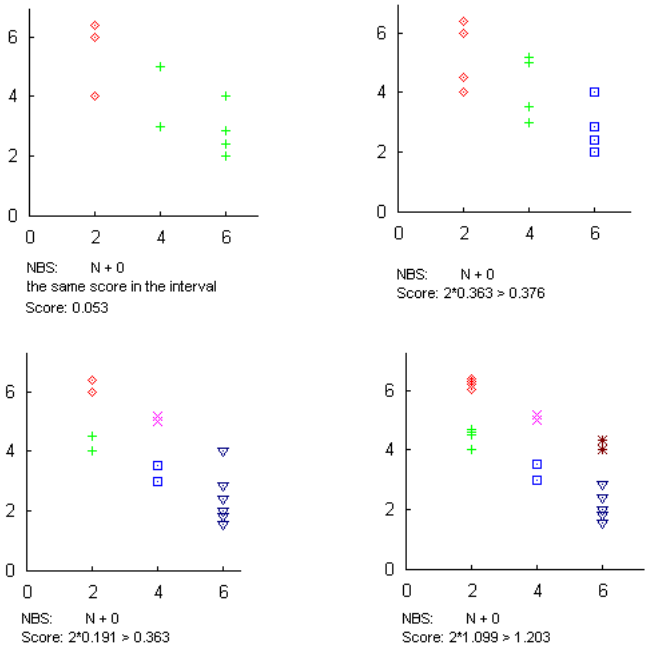


FIGURE 4. Clustering over reduced number of elements

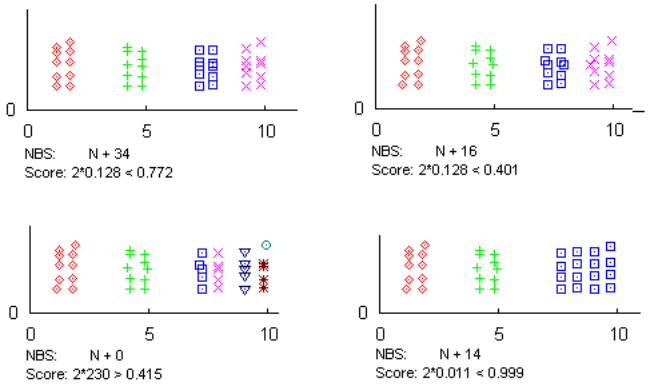


FIGURE 5. Clusters built for small variation of elements coordinates

in order to have close value? The result is represented in the 4^{th} data set in the figure.

As we can see in figure 5, the results for elements that are not uniform distributed in a cluster are disputable also for human judges. Consider the clusters from the figure 6. Which of them do we have to consider best? On the other hand, the elements coordinnates in the two images in the figure are the same. But they are represented to a different scale. In one the separation among clusters is observable, and in the other is not. A human judge won't observe that. For this data set our algorithm identifies clusters indicated in figure.

Sparse data is also an example of not relatively uniform distributed elements in a cluster. In this case, the identified clusters are not very close with those identified by a human judge. If the result are good or not is disputable, as we can see in the figure 3, sets 5A and 5B. The explanation is that the result for sparse data is better when the clusters are more compact. This corresponds to smaller value for NBS.

5. Conclusions and Future Directions

The algorithms we build have the advantage to build the clusters without the need of some stop condition, so it is a really unsupervised method. We consider the result as good, as long as the tests indicate an accuracy of about 90% for data 'well' grouped in clusters. The accuracy of the results depends on dispersion of the elements inside the 'ideal' cluster and the number of the elements inside a cluster (the bigger, the better). Usually, the algorithm does not work so well if the clusters in the data do not have many elements, because the number of all elements is small or the data is sparse.

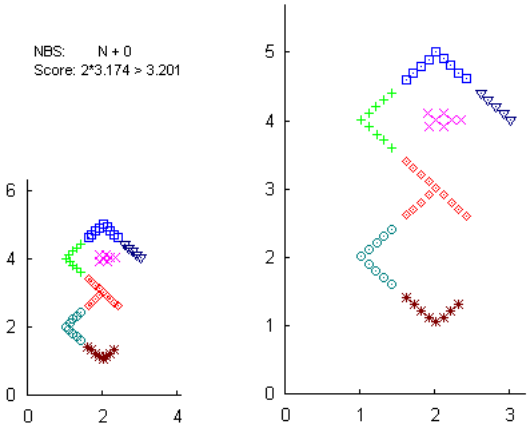


FIGURE 6

One of advantage of the method is that the result do not depend directly from the NBS, there is an interval of values for which the results are the same.

The method was built for cases when clusters are 'well identified' and the elements are relatively uniform distributed inside the clusters. But, for any sparse data, there is no guarantee that there is a human judge that identifies clusters. This is the drawback of the suggested mechanism, if we want to compare it with the absolute case of a human judge. As is known, in case of sparse data, the complete-link method is more appropriate than the single-link.

One of the future directions we are working on is to develop a similar method also for the complete-link hierarchical clustering and compare the results of the two methods.

We also intend to apply this method to pattern recognition in image processing, because we think that this is a domain where the method should apply with best result.

REFERENCES

[1] Baeza-Yates, R.A. *Introduction to data structures and algorithms related to information retrieval*, 1992;
[2] Berkin, P., *Survey of Clustering Data Mining Techniques*, 2002;
[3] Cao, F. et. al., *An a contrario approach to hierarchical clustering validity assessment*, 2004;
[4] Dubes, R.C., *How many clusters are best? - An experiment*, 1987;
[5] Hearst,M., *Applied Natural Language Processing*, Lecture Notes, 2004;
[6] Jain,A.K., M.N. Murty, *Data Clustering: A Review*, ACM Computing Surveys, Vol. 31, No.3, September 1999;
[7] Massey,L., *Determination of Clustering Tendency With ART Neural Networks*, 2002;

[8] Nagy, G., *State of the Art in Pattern Recognition*, Proc. IEEE 56, 836-862, 1968;
[9] Matteucci, M., *A Tutorial on Clustering Algorithms*,
http://www.elet.polimi.it/upload/matteucc/Clustering/tutorial_html/index.html
[10] Annotated Computer Vision Bibliography
http://iris.usc.edu/Vision-Notes/bibliography/pattern617.html

Babes-Bolyai University, Faculty of Mathematics and Computer Science, Department of Computer Science, Cluj-Napoca, Romania
E-mail address: davram@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume L, Number 2, 2005

COMPONENTS EXECUTION ORDER USING GENETIC ALGORITHMS

ANDREEA FANEA AND LAURA DIOŞAN

Abstract. The current trend in software engineering is towards component-based development. A challenge in component-based software development is how to assemble components effectively and efficiently. In this paper we present a new approach for computing components execution order. Using Genetic Algorithms GA we compute the final system from specified components with conditions order. Some numerical experiments are performed.

1. Introduction

Component-based software engineering (CBSE) is the emerging discipline of the development of software components and the development of systems incorporating such components. In a component-based development process we distinguish the development of components from development of systems; components can be developed independently of systems. However, the processes may have many interaction points.

The main advantage of component-based system development is the reuse of components when building applications. Instead of developing a new system from scratch, already existing components are assembled to give the required result.

Unlike previous work [5] which use backtracking algorithm (BA) to determine the execution order for system components, this work looks for designing an execution order using evolutionary methods.

Our main purpose is to evolve the execution order for system components. This basically means that we want to find which component should be executed and which is the order in which these elements are "computed". In this respect we propose a new technique which is used for evolving the execution structure of a component-based system. We evolve arrays of integers which provide a meaning for executing the elements within a system.

Received by the editors: November 10, 2005.

2000 *Mathematics Subject Classification.* 68T20,68N30.

1998 *CR Categories and Descriptors.* I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search – *Heuristic methods*; I.6.5 [**Model Development**]: Modeling methodologies .

The number of evolved orders found with GA is compared to configurations number obtained with a backtracking algorithm from [5]. Numerical experiments show that the GA performs similarly and sometimes even better than standard backtracking approaches for several human-defined systems.

The paper is structured as follows: Section 2 discusses work related to components integration and composition. Section 3 describes, in detail, the proposed model. Several numerical experiments are performed in Section 4. Conclusions and further work are given in Section 5.

2. Related work

There are two issues which need to be addressed where a software system is to be constructed from a collection of components. First there has to be a way to connect the components together. Then we have to get them to do what we want. We need to ensure that the assembled system does what it is required [7], [9].

To successfully incorporate a component [4] in a system, certain steps must be followed: selection, composition and integration, and finally, test and verification must be followed. In the following we discuss integration and composition of components.

Component integration and composition are not synonymous. Component integration is the mechanical process of "wiring" components together, whereas composition takes one step further to ensure that assemblies can be used as components in larger assemblies. Component composition focuses on emergent assembly-level behavior, assuring that the assembly will perform as desired and that it could be used as a building block in a larger system. The constituent components must not only plug together, they must work well together.

The problem of making pieces of software fit together has been the subject of considerable effort. Systems and schemes exist which address these issues (COM, EJB, RMI, MSMQ) [10], [12], [1], [2], [8]. These arrangements work by managing and controlling the interfaces between components. By forcing components to conform to rules about how they interact with the outside world, systems ensure that components do not damage each others when they are connected. To a large extent, this problem may be addressed by managing component interfaces and ensuring that components are only connected through compatible interfaces.

In [5] a formal model for component composition is described. We consider that a compound component is formed from two or more simple components. There are two basic ways [11] in which these simple components can depend on each other, parallel and serial operation:

- **parallel composition**, $A||B$, in which the operations performed on data are independent and there is no dependency between outports(A) and outports(B);
- **serial composition**, $A + B$, in which the B component expects some results from component A.

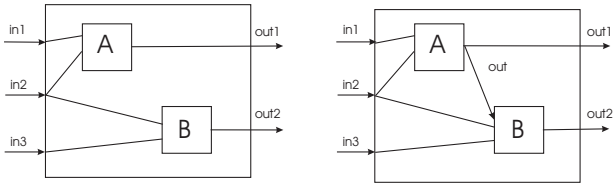


FIGURE 1. Parallel and serial compositions

3. Proposed model

This section presents the proposed model for finding the components execution order. We will use a GA [3, 6] for evolving the execution order of components. Each GA individual is a fixed-length string of genes. Each gene is an integer number, from $[0...NumberOfComponents]$. These values represent indexes order of the components. They will indicate the time moments for execution.

Some components could be executed earlier and some of them are executed later. Therefore, a GA chromosome must be transformed to contain only the values from 0 to Max, where Max represents the number of different time moments (at one moment it is possible that one or more components will be executed).

Example 1. Suppose that we want to evolve the structure of a system that contains 9 components. This means that the algorithm will have chromosomes with 9 genes whose values are in the [1...9] range. The dependencies between components are:

TABLE 1. Conditions for the first example

CondNr	Condition	CondNr	Condition
1	$C1 + C3$	6	$C6 + C7$
2	$C2 + C4$,	7	$C7 + C8$,
3	$C3 + C5$,	8	$C5 + C9$,
4	$C4 + C5$,	9	$C8 + C9$,

A GA chromosome with 9 genes can be:

Genes	1	2	3	4	5	6	7	8	9
Values	2	1	4	3	8	5	6	9	7

For computing the fitness of this chromosome, during one generation, we will execute the components following this order: first, the second component (because the gene with minimum value is the second gene), then the first component and

so on (it is needed to order the ascending the genes values): execute(C2), execute (C1), execute (C4), execute (C3), execute (C6), execute (C7), execute (C9), execute (C5), execute (C8).

In this example at each moment, we execute only one component.

Example 2. Let us consider another example which consists of a chromosome with 9 genes containing only 5 different values.

Genes	1	2	3	4	5	6	7	8	9
Values	6	2	1	4	7	1	6	2	6

In this case components 3 and 6 are executed in same time (also components 2 and 8, but after the previous execution, also the components 1, 7 and 9). Because of this synchronism we need to scale the genes of the GA chromosome to the interval [0 ... 5] (because we need 5 time moments for execution). The obtained chromosome is:

Genes	1	2	3	4	5	6	7	8	9
Values	4	2	1	3	5	1	4	2	4

To establish the correct order of execution for this chromosome the genes values must be ascending sorted. Then we obtain:

Genes	3	6	2	8	4	1	7	9	5
Values	1	1	2	2	3	4	4	4	5

3.1. **Fitness assignment.** The model proposed in this paper is a GA that evolves the order of execution for the components of a system.

The array of integers encoded into a GA chromosome represents the order of execution for system components. The fitness of a chromosome represents the sum of two values: the number of breached conditions and the number of components that are not executed.

Some components depend on the results provided by the other components (one component must wait until other component (s) finished its execution). For instance, we have two number a and b and we want to verify if the greatest common divisor of these two numbers is a prim number, that's equivalent to have two components: $C1$ that computes the greatest common divisor of a and b and $C2$ that verifies if a number is prime. In our problem $C1$ must be executed before $C2$. A system with more components will have more conditions. If one of these conditions is not respected (by the order given by the GA individual), than we increase the fitness.

Even if the number of genes from a GA individual is equal to the components number, it is possible that not all components can be executed. For instance, if the order provided by the GA chromosome supposes to execute the component $C2$ before $C1$, than $C2$ will not be executed and we increase the fitness value.

In the second example, the genes with same execution moments proceed in parallel and the genes with different values are executed serial. But for a parallel execution of two or more components these elements must be independent (are not the subject of one of the system conditions).

The system executes the components $C3$ and $C6$ if and only if there isn't any constrict for these components. Components $C3$ and $C6$ are independent, so the system can execute them in parallel (but also it is possible to execute them in serie). But really, the system executes only $C6$ because $C3$ can't b executed ($C1$ isn't yet executed). Then, component $C2$ is executed in series with $C6$ because the associated genes have different values. Then component $C8$ has the same gene value with $C2$ and there isn't any condition regarding these components, therefore they can be executed in parallel (but $C8$ will not be executed because, $C7$, must be executed first and so on. Finally, this chromosome will have a fitness equal to 8 (four components are not executed and four elements breached the dependencies).

3.2. **Algorithm.** The algorithm used for evolving the integration order of components system is described in this section. The algorithm is a standard GA [3, 6]. We use a generational model as underlying mechanism for our GA implementation.

Population Initialization $P(0)$.
Evaluate $P(0)$.
For $t = 1$ **to** $NumberOfGeneration$ **do**
  Add the best individual from $P(t-1)$ to $P(t)$
  While $P(t)$ is not complete (full, whole) **do**
    Select two parents P_1 and P_2 from $P(t-1)$
    Recombine the parents, obtaining two offspring O_1 and O_2.
    Mutate each offspring.
    Add the two offspring O_1 and O_2 to $P(t)$.
    Evaluate $P(t)$
  End While
End For

The GA starts by creating a random population of individuals. Each individual is a fixed-length (equal to the number of system components) array of integer numbers. The following steps are repeated until a given number of generations is reached: we put in the next generation the best chromosome from current generation. Two parents are selected using a standard selection procedure (binary tournament selection) until we obtain a new complete generation. The parents are recombined (using one-cutting point crossover) in order to obtain two offspring. The offspring are considered for mutation which is performed by replacing some

genes with randomly generated values. Finally, we put the offspring in next generation.

4. Numerical experiments

In order to test our approach we performed three didactical experiments with different number of components involved in the system and with different types of dependences between components. We compared the result obtained with the algorithm from [5] with the presented GA.

Having specified the simple components that we need to compose, a model for component composition [5] is used to obtain all the possibilities of using parallel and serial composition. In the end we will obtain the system that we want. First we have to check if the data can pass between the simple components involved in the composition. That is if inports of all the simple components are inports of the black_box or if they occur as outports of the other components. If the condition above is satisfied then we can obtain the black_box component (final system), respecting dependencies between components.

The dependencies between components must be first determined as follows: we must check if an inport of a component appears as an outport of other component; if an inport in_i of a component B appears as an outport out_j of a component A, ($in_i = out_j$) then we must use component A before component B, and we can use B only with the + operator (for serial composition). These conditions are incorporated in the backtracking algorithm: if the component that we check for integration depends on the results provided by other components than those components must finish their executions before using this component. The dependent component must have a higher position number in the solution array then the components it depends. Also, in the solution array before a component with dependencies we always use serial composition +.

Experiment 1. In this experiment we deal with ten involved components having nine dependencies between them. The system that we want to obtain is presented in Figure 2 with the following computation semantics: *gcd* - greatest common divisor, *sd* - sum of digits, *pd* - prime divisors, *np* - number of prime numbers, *scd* - smallest common divisor, *inv* - inverse, *oed* - product of sum between odd and even digits, *sed* - sum of even digits, *npn* - next prime number and *aa* - arithmetic average.

We denote by "+" the serial composition and by "||" the parallel composition. The dependencies between the involved components are presented in Table 2.

Taking into account the above conditions we compute all the possibilities to obtain the desired system. Using BA from [5] one solution is:

$$(((((((((C2||C6)+C4)||C1)+C7)+C3)+C8)||C9)+C5)+C10).$$

This formula means that the result for computing in parallel $C2$ with $C6$ is then computed serial with $C4$, because $C4$ expects the result from $C2$. Then, the

TABLE 2. Conditions for the first experiment

CondNr	Condition	CondNr	Condition
1	$C1 + C3$,	6	$C7 + C8$,
2	$C2 + C4$,	7	$C5 + C10$,
3	$C6 + C7$,	8	$C8 + C10$,
4	$C3 + C5$,	9	$C9 + C10$.
5	$C4 + C5$,		

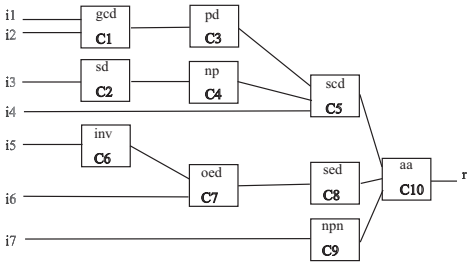


FIGURE 2. Component-based system for experiment 1

current result subsystem is composed in parallel with $C1$. Because component $C6$ was computed we can then serially execute $C7$. The result for computation of $C1$ is already obtained so can serially execute $C3$. Next, we can execute either serial $C5$, serial $C8$ or in parallel $C9$. We cannot compute $C10$ because we have to previous execute $C5$, $C8$ and $C9$. In previously experiment serial $C8$ is integrated. The following possibilities are parallel $C9$ and serial $C5$. Parallel $C9$ is executed and then serial $C5$. Having all the three components executed, we can now serial execute component $C10$.

We compare this approach with an evolutionary one. We use a standard GA for evolving the execution order. For GA we use a population with 200 individuals, each of them with 1 dimension. The gene number for a dimension will be equal to the components number (for this example it is 10). During 10000 generations we apply binary tournament selection, one cut point crossover with probability 0.8 and mutation (we mutate a random chosen gene) with probability 0.2.

But, since the GA uses pseudo-random numbers, it is very likely that successive runs of the same algorithm will generate completely different solutions. This problem can be handled in a standard manner: the genetic algorithm is run multiple times (100 runs in fact) and the number of the "good" chromosomes (with fitness 0) will be the average over all runs.

Applying GA one chromosome is:

We sort this chromosome after genes values and obtain:

Genes	1	2	3	4	5	6	7	8	9	10
Values	2	1	4	2	6	1	3	5	5	6

Genes	2	6	1	4	7	3	8	9	5	10
Values	1	1	2	2	3	4	5	5	6	6

Decoding it we obtain the following execution order: $C2$ in same time with $C6$, then $C1$ and $C4$, and then $C7$ and $C3$ (serial execution). At moment 5 we execute in parallel $C8$ and $C9$. Components $C5$ and $C10$ have the same execution moment, but they aren't executed in parallel because there exists a dependence between these two components. Therefore we execute first $C5$ and then, serial execution, $C10$. The experiment results are presented in Table 3 .

TABLE 3. Experiment 1: ten involved components with nine dependences

Algorithm	Solutions	Time
BA	3024	$2''$
GA	3883	$2''$

Experiment 2. In this experiment we deal with eleven involved components having ten dependencies between them. The computation semantics for each component involved in system integration are similar with those from the first experiment. We only present in Table 4 the dependencies between the involved components.

TABLE 4. Conditions for the second experiment

CondNr	Condition	CondNr	Condition
1	$C1 + C3$	6	$C6 + C7$
2	$C3 + C5$,	7	$C7 + C9$,
3	$C4 + C5$,	8	$C8 + C10$,
4	$C2 + C8$,	9	$C9 + C10$,
5	$C5 + C8$,	10	$C10 + C11$.

Applying algorithm from [5] one solution is:

$$((((((((((C3||C6) + C7)||C1) + C2)||C4) + C9) + C5) + C8) + C10) + C11).$$

This solution is represented in Figure 3. Each component has on the upper-left corner a number representing the order of execution.

Applying GA we obtain a chromosome of the following form:

We sort this chromosome on genes values and obtain:

The experimental results are shown in Table 5.

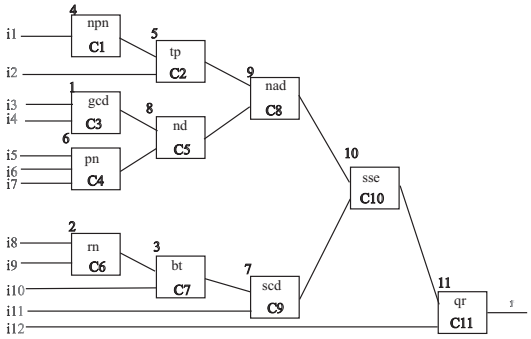


FIGURE 3. Component-based system for experiment 2

Genes	1	2	3	4	5	6	7	8	9	10	11
Values	2	3	1	3	5	1	2	6	4	6	7

Genes	3	6	1	7	2	4	9	5	8	10	11
Values	1	1	2	2	3	3	4	5	6	6	7

TABLE 5. Experiment 2: eleven involved components with ten dependences

Algorithm	Solutions	Time
BA	1680	$1''$
GA	2957	$1''$

5. Conclusions and Further work

In this paper a new method for component integration is proposed. The method is based on GA computing the order possibilities of components execution from a component-based system. A comparison between a previous developed backtracking-based algorithm and a GA algorithm is presented. The experiments on different data sets prove that we obtain much more solutions applying GA than applying other algorithms.

However, taking into account the No Free Lunch theorems for Search [13] and Optimization [14] we cannot make any assumption about the generalization ability of the evolved execution order. Further numerical experiments are required in order to assess the power of the evolved order.

Further works can be done in the following directions:

- how can we use GA to compute all the possibilities of obtaining a correct syntactical component-based system using only the information on the component interface;
- how can we use IA methods to analyze the behavior of a component-based system or to predict the behavior.

References

[1] R. Allen and D. Garlan, *A formal basis for architectural connection*, ACM Trans. on Software Eng. and Methodology, 6(3):213–249, July 1997.
[2] D. Box, *Essential COM*, Addison Wesley, 1998.
[3] H. J. Bremermann, *Optimization through evolution and recombination*, M.C. Yovits, G.T. Jacobi, and G.D. Goldstein, editors, Self-Organizing Systems 1962, Proceedings of the Conference on Self-Organizing Systems, Chicago, Illinois, 22.- 24.5.1962, pp. 93-106, 1962.
[4] I. Crnkovic, M. Larsson, *Building reliable component-based software systems*, Artech House, 2002
[5] A. Fanea, S. Motogna, *A Formal Model for Component Composition*, Proceedings of the Symposium "Zilele Academice Clujene", 2004, pp. 160-167
[6] D. Goldberg, *Genetic algorithms in search, optimization and machine learning*, Addison-Wesley, Boston, USA, 1989.
[7] A. M. Gravell, and P. Henderson, *Executing formal specifications need not be harmful*, Software Engineering Journal, 11(2):104-110, IEE/BCS, March 1996
[8] D. N. Gray, J. Hotchkiss, S. LaForge, A. Shalit and T. Weinberg, *Modern Languages and Microsoft's Component Object Model*, Communications of the ACM 41(5): 55-65 1998.
[9] C. A. R. Hoare, *The role of formal techniques: past, current and future or how did software get so reliable without proof?*, 18th International Conference on Software Engineering (ICSE-18), Berlin, IEEE Computer Society Press, 1996, pp. 233-234.
[10] Object Management Group, http://www.omg.org/
[11] B. Parv, S. Motogna, *A formal model for components*, Bul. Stiint., Univ. Baia Mare, Ser. B, Matematica-Informatica, XVIII(2002), No.2, pag. 269-274
[12] Sun Microsystems, http://www.sun.com/
[13] D. H. Wolpert and W. G. McReady, *No Free Lunch Theorems for Search*, Technical Report SFI-TR-05-010, Santa Fe Institute, USA, 1995.
[14] D. H. Wolpert and W. G. McReady, *No Free Lunch Theorems for Optimization*, *IEEE Transaction on Evolutionary Computation*, Nr. 1, pp. 67-82, IEEE Press, NY, USA, 1997

Department of Computer Science, Faculty of Mathematics and Computer Science, Babeş-Bolyai University, Cluj-Napoca, Romania
E-mail address: afanea, lauras@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume L, Number 2, 2005

ADAPTIVE CLUSTERING USING A CORE-BASED APPROACH

GABRIELA ŞERBAN AND ALINA CÂMPAN

Abstract. This paper studies an adaptive clustering problem. We focus on re-clustering an object set, previously clustered, when the feature set characterizing the objects increases. We propose an adaptive, *k-means* based clustering method, *Core Based Adaptive k-means (CBAk)*, that adjusts the partitioning into clusters that was established by applying *k-means* or *CBAk* before the feature set changed. We aim to reach the result more efficiently than running *k-means* starting from the current clustering. Experiments testing the method's efficiency are also reported.

Keywords: Data Mining, clustering, k-means.

1. Introduction

A large collection of clustering algorithms is available in the literature. The papers [5], [6] and [7] contain comprehensive overviews of the existing clustering techniques.

A well-known class of clustering methods is the one of the partitioning by relocation methods, with representatives such as the *k-means* algorithm or the *k-medoids* algorithm. Essentially, given a set of n objects and a number $k, k \leq n$, such a method divides the object set into k distinct clusters. The partitioning process is iterative and stops when a "good" partitioning is achieved. Finding a "good" partitioning coincides with optimizing a criterion function. The criterion function used in *k-means* is the squared error criterion, which tends to work well with isolated and compact clusters [7].

Generally, these methods apply on a set of objects measured against a known set of features (attributes). But there are applications where the attribute set characterizing the objects evolves. For obtaining in these conditions a partitioning of the object set, the clustering algorithm can be, obviously, applied over and over again, beginning from scratch or from the current partitioning, each time

Received by the editors: October 15, 2005.

2000 *Mathematics Subject Classification.* 62H30, 68U35.

1998 *CR Categories and Descriptors.* 62H30 [**Statistics**]: Multivariate analysis – *Classification and discrimination; cluster analysis*; 68U35 [**Computer science**]: Computing methodologies and applications – *Information systems (hypertext navigation, interfaces, decision support, etc.)*;

when the attributes change. But this can be inefficient. What we want is to propose an adaptive, *k-means* like clustering method, named *Core Based Adaptive k-means (CBAk)*, that is capable to efficiently re-partition the object set, when the attribute set *increases*. The method starts from the partitioning into clusters that was established by applying *k-means* or *CBAk* before the attribute set changed. We aim to reach the result more efficiently than running *k-means* starting from the current clustering.

Related Work

There are few approaches reported in the literature that address the problem of adapting the result of a clustering when the object feature set is extended. Early works treat the sequential use of features in the clustering process, one by one. An example of such a monothetic approach is mentioned in [7]. A more recent paper [10] analyzes the same problem of adapting a clustering produced by a *DBSCAN* like algorithm, using some additional structures and distance approximations in an Euclidian space. However, adapting a clustering resulted from a partitioning algorithm, using partitioning-based methods hasn't been reported by none of these works.

2. Theoretical Model

Let $X = \{O_1, O_2, \ldots, O_n\}$ be the set of objects to be classified. Each object is measured with respect to a set of m initial attributes and is therefore described by an m-dimensional vector $O_i = (O_{i1}, \ldots, O_{im}), O_{ik} \in \Re^+, 1 \leq i \leq n, 1 \leq k \leq m$. Usually, the attributes associated to objects are standardized, in order to ensure an equal weight to all of them [5].

Let $\{K_1, K_2, \ldots, K_p\}$ be the set of clusters discovered in data by applying the *k-means* algorithm. Each cluster is a set of objects, $K_j = \{O_1^j, O_2^j, \ldots, O_{n_j}^j\}, 1 \leq j \leq p$. The centroid (cluster mean) of the cluster K_j is denoted by f_j, where

$$f_j = \left(\frac{\sum_{k=1}^{n_j} O_{k1}^j}{n_j}, \ldots, \frac{\sum_{k=1}^{n_j} O_{km}^j}{n_j} \right).$$

The measure used for discriminating objects can be any *metric* or *semi-metric* function d. We used the *Euclidian distance*:

$$d(O_i, O_j) = d_E(O_i, O_j) = \sqrt{\sum_{l=1}^{m} (O_{il} - O_{jl})^2}.$$

The measured set of attributes is afterwards extended with s ($s \geq 1$) new attributes, numbered as $(m+1), (m+2), \ldots, (m+s)$. After extension, the objects' vectors become $O_i' = (O_{i1}, \ldots, O_{im}, O_{i,m+1}, \ldots, O_{i,m+s}), 1 \leq i \leq n$. We denote by $extO_i' = (O_{i,m+1}, \ldots, O_{i,m+s})$ the s-attribute extension of the vector associated to O_i.

We want to analyze the problem of recalculating the objects' grouping into clusters, after object extension and starting from the current partitioning. We start from the fact that, at the end of the initial *k-means* clustering process, all objects are closer to the centroid of their cluster than to any other centroid. So, for any cluster j and any object $O_i^j \in K_j$, inequality (1) below holds.

$$d_E(O_i^j, f_j) \leq d_E(O_i^j, f_r), \forall j, r,\ 1 \leq j, r \leq p,\ r \neq j. \tag{1}$$

We denote by $K'_j, 1 \leq j \leq p$, the set containing the same objects as K_j, after the extension. By $f'_j, 1 \leq j \leq p$, we denote the mean (center) of the set of K'_j. We denote by $extf'_j = \left(\frac{\sum_{k=1}^{n_j} O^j_{k,m+1}}{n_j}, \ldots, \frac{\sum_{k=1}^{n_j} O^j_{k,m+s}}{n_j} \right)$ the s-attribute extension of the K'_j center (mean). These sets $K'_j, 1 \leq j \leq p$, will not necessarily represent clusters after the attribute set extension. The newly arrived attributes can change the objects' arrangement into clusters. But there is a considerable chance, when adding one or few attributes to objects, that the old arrangement in clusters to be close to the actual one. The actual clusters can be obtained by applying the *k-means* algorithm on the set of extended objects starting from the current clustering. But we try to avoid this process and replace it with one less expensive but not less accurate. With these being said, we agree, however, to continue to refer the sets K'_j as clusters.

We therefore take as starting point the previous partitioning into clusters and study in which conditions an extended object $O_i^{j\prime}$ is still "correctly" placed into its cluster K'_j. For that, we express the distance of $O_i^{j\prime}$ to the center of its cluster, f'_j, compared to the distance to the center f'_r of any other cluster K'_r.

Lemma 1. *When inequality (2) holds for an extended object $O_i^{j\prime} \in K'_j$*

$$d^2(extO_i^{j\prime}, extf'_j) \leq d^2(extO_i^{j\prime}, extf'_r) \tag{2}$$

for all $r = \overline{1,p},\ r \neq j$ then the object $O_i^{j\prime}$ is closer to the center f'_j than to any other center $f'_r, 1 \leq j, r \leq p,\ r \neq j$.

Proof

We prove this statement. For $O_i^{j\prime}$ and $1 \leq r \leq p$

$d^2(O_i^{j\prime}, f'_j) - d^2(O_i^{j\prime}, f'_r) = d^2(O_i^j, f_j) + d^2(extO_i^{j\prime}, extf'_j) - d^2(O_i^j, f_r) - d^2(extO_i^{j\prime}, extf'_r)$.

Using the inequality (1), we have:

$d^2(O_i^{j\prime}, f'_j) - d^2(O_i^{j\prime}, f'_r) \leq d^2(extO_i^{j\prime}, extf'_j) - d^2(extO_i^{j\prime}, extf'_r)$.

If the inequality (2) holds, then the inequality above becomes:

$d^2(O_i^{j\prime}, f'_j) - d^2(O_i^{j\prime}, f'_r) \leq 0$.

Because all distances are non-negative numbers, it follows that:

$d(O_i^{j\prime}, f'_j) \leq (O_i^{j\prime}, f'_r)$, $\forall r,\ 1 \leq r \leq p,\ r \neq j$.

Remark The global complexity of the *CBAk* algorithm is not increased by the cluster cores calculation.

3. The *Core Based Adaptive k-means* Algorithm

We will use the property enounced in the previous paragraph in order to identify inside each cluster $K'_j, 1 \leq j \leq p$, the objects that have a considerable chance to remain stable in their cluster, and not to move into another cluster as a result of the attribute set extension. These objects form the *core* of their cluster.

Definition 1.

a) We denote by $StrongCore_j = \{O_i^{j\prime} | O_i^{j\prime} \in K'_j, O_i^{j\prime}$ satisfies the inequality (2)$\}$, $\forall r,\ 1 \leq r \leq p,\ r \neq j$.

b) Let $sat(O_i^{j\prime})$ be the set of all clusters $K'_r, \forall r,\ 1 \leq r \leq p,\ r \neq j$ not containing $O_i^{j\prime}$ and for which object $O_i^{j\prime}$ satisfies inequality (2).
We denote by $WeakCore_j = \{O_i^{j\prime} | O_i^{j\prime} \in K'_j, |sat(O_i^{j\prime})| \geq \frac{\sum_{k=1}^{n_j} |sat(O_k^{j\prime})|}{n_j}\}$ the set of all objects in K'_j satisfying inequality (2) for at least so many clusters that all objects in K'_j are satisfying (2), in the average.

c) $Core_j = StrongCore_j$ iif $StrongCore_j \neq \emptyset$; otherwise,
$Core_j = WeakCore_j$. $OCore_j = K'_j \setminus Core_j$ is the set of out-of-core objects in cluster K'_j.

d) We denote by $CORE$ the set $\{Core_j, 1 \leq j \leq p\}$ of all cluster cores and by $OCORE$ the set $\{OCore_j, 1 \leq j \leq p\}$.

We have chosen the above cluster cores definition because of the following reasons. It is not sure that there is in cluster K'_j any object that satisfies inequality (2) for all clusters $K'_r, 1 \leq r \leq p,\ r \neq j$. If there are such objects ($StrongCore_j \neq \emptyset$), we know that, according to Lemma 1, they are closer to the cluster center f'_j than to any other cluster center f'_r, $1 \leq r \leq p$, $r \neq j$. Then, $Core_j$ will be taken to be equal to $StrongCore_j$ and will be the seed for cluster j in the adaptive algorithm. But if $StrongCore_j = \emptyset$, for the core not to be empty, we will choose as seed for cluster j other objects, the most stable ones between all objects in K'_j.

The cluster cores, chosen as we described, will serve as seed in the adaptive clustering process. All objects in $Core_j$ will surely remain together in the same group if clusters do not change. This will not be the case for all core objects, but for most of them, as we will see in the results section.

We give next the *Core Based Adaptive k-means* algorithm.

We mention that the algorithm stops when the clusters from two consecutive iterations remain unchanged or the number of steps performed exceeds the maximum allowed number of iterations.

```
Algorithm Core Based Adaptive k-means is
Input: - the set X = {O_1, ..., O_n} of m-dimensional previously clustered
         objects,
       - the set X' = {O'_1, ..., O'_n} of (m+s)-dimensional extended objects
         to be clustered; O'_i has the same first m components as O_i,
       - the metric d_E between objects in a multi-dimensional space,
       - p, the number of desired clusters,
       - K = {K_1, ..., K_p} the previous partition of objects in X,
       - noMaxIter the maximum number of iterations allowed.
Output: - the new partition K' = {K'_1, ..., K'_p} for the objects in X'.
Begin
    For all clusters K_j ∈ K
       Calculate Core_j = (StrongCore_j ≠ ∅)?StrongCore_j : WeakCore_j
       K'_j = Core_j
       Calculate f'_j as the mean of objects in K'_j
    EndFor
    While (K' changes between two consecutive steps) and
          (there were not performed noMaxIter iterations) do
       For all clusters K'_j do
          K'_j = {O'_i | d(O'_i, f'_j) ≤ d(O'_i, f'_r), ∀r, 1 ≤ r ≤ p, 1 ≤ i ≤ n}
       EndFor
       For all clusters K'_j do
          f'_j = the mean of objects in K'_j
       EndFor
    EndWhile
End.
```

The algorithm starts by calculating the old clusters' cores. The cores will be the new initial clusters from which the iterative processing begins. Next, the algorithm proceeds in the same manner as the classical *k-means* method does.

4. Experimental Evaluation

In this section we present some experimental results obtained by applying the *CBAk* algorithm described in section 3.

As case studies, for experimenting our theoretical study described in section 2 and for evaluating the performance of the *CBAk* algorithm, we considered the data

sets described in [1]. The data were taken from the website "http://www.cormactech.com/neunet" and have also been used in [2, 4, 9].

4.1. **Quality Measures.** As a quality measure for our algorithm we take the movement degree of the core objects and of the extra-core objects. In other words, we measure how the objects in either $Core_j \in CORE$, or $OCore_j \in OCORE$, remain together in clusters after the algorithm ends.

As expected, more stable the core objects are and more they remain together in respect to the initial sets $Core_j$, better was the decision to choose them as seed for the adaptive clustering process.

We denote by $S = \{S_1, S_2, \ldots, S_p\}, S_i \subseteq K_i$, a set of clusters' subsets (as $CORE$ and $OCORE$ are). We express the *stability factor* of S as:

$$SF(S) = \frac{\sum_{j=1}^{p} \frac{|S_j|}{\text{no of clusters where the objects in } S_j \text{ ended}}}{\sum_{j=1}^{p} |S_j|} \tag{3}$$

The worst case is when each object in S_j ends in a different final cluster, and this happens for every set in S. The best case is when every S_j remains compact and it is found in a single final cluster. So, the limits between which $SF(CORE)$ varies are given below, where the higher the value of $SF(CORE)$ is, the better was the cores choice:

$$\frac{p}{\sum_{j=1}^{p} |Core_j|} \leq SF(CORE) \leq 1 \tag{4}$$

For comparing the quality of the partitions produced by our algorithm and by *k-means*, we consider the *squared sum error (SSE)* of a clustering K, defined as:

$$SSE(K) = \sum_{K_j \in K} \sum_{O_i \in K_j} d^2(O_i, f_j) \tag{5}$$

When comparing two partitions K_1 and K_2 for the same data set, we will say that K_1 is better than K_2 iff $SSE(K_1) < SSE(K_2)$.

For measuring the clustering tendency of a data set, we use the Hopkins statistics, H [11], an approach that uses statistical tests for spatial randomness. H takes values between 0 and 1, and a value near 1 indicates that data is highly clustered. Usually, for a data set with clustering tendency, we expect for H values greater than 0.5.

4.2. **Results.** In this section we comparatively present the results obtained by applying the *CBAk* algorithm and *k-means*, for the experimental data. We mention that the results are calculated in average, for several executions.

TABLE 1. The comparative results

Experiment	Cancer	Dermatology	Wine
No of objects	457	366	178
No of attributes (m+s)	9	34	13
No of new attributes (s)	4	3	4
No of clusters	2	6	3
No of k-means iterations for m attributes	5.66	11.2	9.28
No of k-means iterations for +s attributes	4	1.33	3.85
No of CBAk iterations for +s attributes	4	5.66	2.42
k-means SSE for +s attributes	13808.784	12683.82	49.016
CBAk SSE for +s attributes	13808.784	12522.95	49.019
SF(CORE)	1.0	0.8119	0.97
SF(OCORE)	0.5	0.646	0.475
H for s attributes	0.666	0.68122	0.7018
H for m+s attributes	0.7148	0.6865	0.7094

From Table 1 we observe that using the *CBAk* algorithm the number of iterations for finding the solution is not always smaller that in case of using *k-means*; but the cores' stability factor, $SF(CORE)$, is high. We mention that for every run of each experiment, SSE(CBAk) has been roughly equal to SSE(k-means). Also, every time, the stability of the objects chosen to be part or cores was greater than the stability of out-of-core objects.

5. Conclusions and Future Work

In this paper we proposed a new method for adapting the result of a clustering when the attribute set describing the objects increases. The experiments on different data sets prove that, in most cases, the result is reached more efficiently using the proposed method than running *k-means* starting from the current partition, on the feature-extended object set. But there are some situations when it is better to resort to a *k-means* clustering of the feature-extended object set, starting from the existing clustering, than using the *CBAk* algorithm. For example, such situations can be: the addition of a large number of features or the addition of new features with large information gain and contradictory information with respect to the old feature set.

Further work may be done in the following directions:

- to isolate conditions to decide when it is more effective to adapt (using *CBAk*) the result of a clustering of the feature-extended object set than to resume its clustering using *k-means*;
- to study how the information brought into the system by the newly added attributes, their correlation with the initial ones, influences the number of iterations performed by the *CBAk* algorithm for finding the solution;
- to apply the adaptive algorithm on precise problems, from where the need of such an adaptive algorithm originated;
- to study how the theoretical results described for non-hierarchical clustering could be applied/generalized for other clustering techniques.

References

[1] Şerban, G., Câmpan, A.: "Core Based Incremental Clustering", Studia Universitatis "Babeş-Bolyai", Informatica, L(1), 2005, pp 89–96
[2] Aeberhard, S., Coomans, D., de Vel, O.: "THE CLASSIFICATION PERFORMANCE OF RDA", Tech. Rep. 92–01, Dept. of Computer Science and Dept. of Mathematics and Statistics, James Cook University of North Queensland, 1992
[3] CorMac Technologies Inc, Canada: "Discover the Patterns in Your Data", http://www.cormactech.com/neunet
[4] Demiroz, G., Govenir, H. A., Ilter, N.: "Learning Differential Diagnosis of Eryhemato-Squamous Diseases using Voting Feature Intervals", Artificial Intelligence in Medicine
[5] Han, J., Kamber, M.: "Data Mining: Concepts and Techniques", Morgan Kaufmann Publishers, 2001
[6] Jain, A., Dubes, R.: "Algorithms for Clustering Data", Prentice Hall, Englewood Cliffs, New Jersey, 1998
[7] Jain, A., Murty, M. N., Flynn, P.: "Data clustering: A review", ACM Computing Surveys, 31(3), 1999, pp 264-323
[8] Quinlan, J. R.: C4.5: "Programs for Machine Learning", Morgan Kaufmann. San Mateo, California, 1993
[9] Wolberg, W., Mangasarian, O. L.: "Multisurface method of pattern separation for medical diagnosis applied to breast cytology" Proceedings of the National Academy of Sciences, U.S.A., Volume 87, December 1990, pp 9193–9196
[10] Wu, F., Gardarin, G.: "Gradual Clustering Algorithms", Proceedings of the 7th International Conference on Database Systems for Advanced Applications (DASFAA'01), 2001, pp 48–57
[11] Tan, P.-N., Steinbach, M., Kumar, V.: "Introduction to Data Mining", Addison Wesley, 2005, chapters 8,9

Babeş-Bolyai University, Faculty of Mathematics and Computer Science, Cluj-Napoca, Romania
E-mail address: gabis@cs.ubbcluj.ro

Babeş-Bolyai University, Faculty of Mathematics and Computer Science, Cluj-Napoca, Romania
E-mail address: alina@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume L, Number 2, 2005

INTEGRATING CONVERSIONS INTO A COMPUTATIONAL ALGEBRAIC SYSTEM

VIRGINIA NICULESCU AND GRIGORETA SOFIA MOLDOVAN

Abstract. Conversions play an important role in any computational algebraic system. This article analyzes two approaches for integrating conversions. The first is based on template method design pattern and the other is based on aspect-oriented programming. The advantages and disadvantages of these approaches are emphasized.

1. Introduction

Object oriented programming and design patterns introduce a high level of abstraction that allows us to implement and work with mathematical abstractions. Classic algebraic libraries and systems, based on imperative programming, contain subalgorithms for working with polynomials, matrices, vectors, etc. Their main inconvenience is the dependency on types.

In [4] we have analyzed the design of the kernel for an object oriented computational algebra system based on design patterns. This approach allows us to work not only with concrete algebraic structures, but also with abstract algebraic structures. The advantages are mainly given by the creational design patterns, by reflection and dynamic loading, and by representation independence. These introduce significant flexibility and abstraction.

Conversions play an important role in a computational algebraic system, and we present here two solutions for integrating them.

2. The Basic Design of the Algebraic System

The main requirement for an alegraic system is the possibility of working with abstract algebraic structures like groups, rings, fields, etc. The user has to be allowed to define concrete algebraic structures by using these abstractions. We restrict the discussion to basic algebraic structures, and to polynomials and vector spaces.

Received by the editors: November 21, 2005.

2000 *Mathematics Subject Classification.* 68R01, 68U99.

1998 *CR Categories and Descriptors.* J.2 [**Computer Applications**]: Physical Sciences and engineering – *Mathematics and statistics* ;

Abstract classes are defined for elements of each abstract algebraic structure. Their hierarchy is shown in Figure 1.

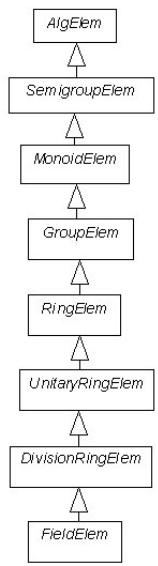


FIGURE 1. The class diagram for basic abstract algebraic structures.

New abstract algebraic structures may be built over the existing algebraic structures; for example polynomials and vector spaces. Polynomials are built over a unitary commutative ring, and they also form a unitary commutative ring. In order to define a vector space we need a group, a field, and an external operation.

The *Composite* design pattern [3] may be used to implement this kind of structures. Using the *Composite* pattern we may define polynomials over other polynomials. Similar examples may be given for matrices – we can define matrices over polynomials, etc.

More details about the system design can be found in [4].

3. CONVERSIONS

Operations between different types of numbers are an important issue for an algebraic system, and the design of the algebraic structures must take into consideration the design of the conversions.

If we add a real number to a complex number, we know that the result is a complex, because a real number is also a complex number, which has the imaginary part equal to zero.

If we generalize this, we arrive to a situation where between two algebraic structures an inclusion relation may be defined. We may have subgroups, subrings, etc. Let us consider that we have a group $(G, +)$, and a subgroup (or submonoid) $(SG, +)$, $SG \subset G$. Corresponding to these, we will have the classes `GElem` and `SGElem`. If we have an element g of type G and an element sg of type SG, we may consider that sg is also an element of type G, and we may use it in operations of class `GElem`. For this we have to allow automatic conversions from SG type to G type.

For example, if we consider the group $(\mathbb{Z}, +)$ and the monoid $(\mathbb{N}, +)$, we have to allow conversions from natural to integer numbers.

One solution to allow this is to use inheritance for defining `SGElem` class (to be derived from `GElem`). Then the operation `g.plus(sg)` would be possible. But the operation `plus` is a commutative one, so we would like to also allow the operation `sg.plus(g)`, but using this solution this is not possible.

But the main disadvantage of the solution based on inheritance can be understood from the following example. We define the class `IntElem` derived from `GroupElem`, corresponding to the group $(\mathbb{Z}, +)$, and the class `NaturalElem` derived from `MonoidElem`, corresponding to the monoid $(\mathbb{N}, +)$. If we choose the solution based on inheritance for conversions, we have to derive `NaturalElem` from class `IntElem`, as well. So, we arrive to a situation when `NaturalElem` is a group, too — which is completely wrong (the corresponding UML diagram is presented in Figure 2).

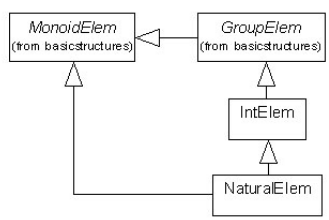


FIGURE 2. A wrong solution for implementing conversions.

A compatibility between two classes is defined when an instance of one class can be converted into an instance of the other.

As a general rule we allow defining conversion if an inclusion type relation can be defined between structures.

Still, we should allow some special cases:

- To add a simple real to a polynomial over reals.
- Consider the group of continuous functions $(\mathbb{R}^A, +)$, where $\mathbb{R}^A = \{f | f : A \to \mathbb{R}, f \text{ continuous}, A \subset \mathbb{R}\}$. If we have $g \in R^A$, we can add it to an $r \in \mathbb{R}$, and the result will be of type $\mathbb{R}^A$. We can add g to r, because r can be seen as $h(x) = r$, for any $x \in A$, hence we actually apply the add operation to g and h.
- New structures may be defined based on the existing ones. For example, the group $(M_n, +)$, where $M_n = \{a + b\sqrt{n} | a, b \in \mathbb{Z}\}$ and $n \in \mathbb{N}$, n is a prime number. The structure $(M_n, +)$ is a group, and $\mathbb{Z} \subset M_n$ (because if $b = 0$, $a + b\sqrt{n} \in \mathbb{Z}$, $\forall a \in \mathbb{Z}$).

So, we may also admit conversions when we have a structure defined over another structure, and when a simple element of the basic structure may form an element of the complex structure.

Another special case when conversions have to be used is related to special representations and precision. If we want to represent structures that are defined over infinite sets, we cannot represent all the elements. So, we may consider only the elements of some subsets. These subsets may be included one into another. Corresponding to these we will have different classes. Integers may be represented using primitive types like `int` or `long`, but also using another representation that could be based on a bigger base. Because they are different representations of the same algebraic structures, they have to be compatible. (The situation is similar to that of the usual conversions that appear in any programming language.) When it is the case, we may base our conversions on the conversions in the implementation language.

3.1. **The Basic Design.** The solution that we suggest is based on *reflection* and *dynamic loading*. These are used for defining new compatibilities between the existing and the new created structures, and for dynamic loading of these compatibilities. The compatibilities are implemented as distinct classes, and their names are stored in a specific file, which can be updated. We will be able to choose whether conversions are accepted or not, or to choose a subset of the set of all defined conversions.

We define a `Conversions` class, which will store all the conversions available in the system. This class will be a singleton [3], because we do not need more than one instance of it. We also define a `Conversion` interface, which will handle the actual conversion, and which has three methods (Figure 3). The methods `getFirstClass()` and `getSecondClass()` are used to determine what type of conversions the concrete class deals with. The `convert(...)` method converts one of the two parameters to the class of the other parameter. Because we do not need to know which parameter has been converted, the result will be an array with two elements.

When the constructor of the `Conversions` class is called, it will dynamically load from the file all the classes that implement the `Conversion` interface, and store

an instance of each class in a list. The method `existConversion(...)` verifies whether there exists a conversion between the two classes given as parameters. If there is one, the `convert(...)` method of the `Conversions` class will be used to make the actual conversion, using the corresponding `Conversion` class.

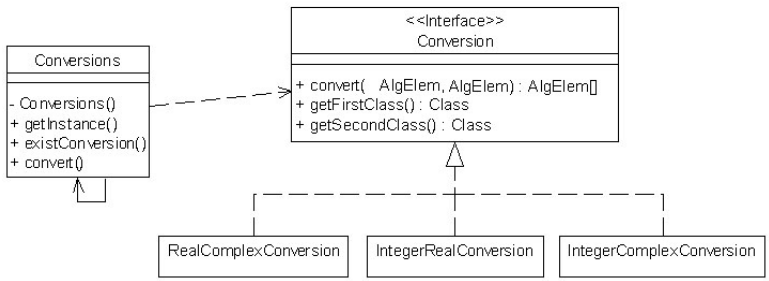


FIGURE 3. Integer - Real - Complex conversions.

The inconvenience of this solution is that a different conversion class must be defined for each possible compatibility. We must define a class for converting an integer to a real, one for converting a real to a complex, and also one for converting an integer to a complex. This inconvenience can be solved using a *graph of types* and then finding the smallest path from one type to another. The vertices of this graph represent the types, and an edge between two vertices represents a compatibility between the two corresponding types.

In the following, we present two approaches to integrate this design into the algebraic system.

3.2. **The *Template Method* Approach.** For each operation `equals()`, `plus()`, `minus()`, etc. the same steps are followed every time, so one approach to integrate the conversions could be based on *Template Method* design pattern [3].

As we have said before, two algebraic elements are considered compatible if they have the same class or there is a conversion between them (Section 3). Any algebraic element can be compared for equality with another algebraic element if they are compatible. So, `AlgElem` has an `equals()` method that compares two `AlgElems` for equality.

It is desirable to build an extendible system, to which the user can add new types. For each new type, the developer will have to override the `equals()` method, where he/she has to verify first if those elements are compatible. The `equals()` method is implemented as a template method. It verifies whether the elements have the same type, and if they do, it calls the method `equalsS()` which compares two elements of the same type. If the elements have different types, but they are compatible, it first calls the `convert()` method and then the

`equalsS()` method to compare them. If the types are incompatible, an exception is thrown. Therefore, the `AlgElem` defines not only the method `equals()` (the template method), but also an abstract method `equalsS()`.

```
public abstract class AlgElem {
 public final boolean equals(AlgElem e)
            throws IncompatibleClassesException{
  if (this.getClass()!=e.getClass()){
   if (Conversions.getInstance().existConversion(
       this.getClass(),e.getClass())){
      AlgElem[] conver=Conversions.getInstance().convert(this, e);
      return conver[0].equalsS(conver[1]);
   }else
      throw new IncompatibleClassesException("There is not defined
       any conversion between "+this.getClass()+" "+e.getClass());
  }else
   return this.equalsS(e);
 }
 protected abstract boolean equalsS(AlgElem o);
}
```

FIGURE 4. Java implementation of the class `AlgElem`.

In the hierarchy of algebraic structures (Figure 1) there are other operations that need to use conversions: `plus(), minus(), times(), etc`. These operations take parameters(operands) which should have the same type. But there are cases when these operations could be called with parameters of different types, because the types are compatible.

The first time such an operation appears in the hierarchy, we define a template method for it. This template method calls another method that does the actual work on the parameters of the same type. The implementation is similar to that for `equals`.

This solution is quite good but it spans over many classes (`AlgElem`, `SemigroupElem`, `GroupElem`, `RingElem` and `DivisionRingElem`) and any modifications/changes to the conversions module will also have to be done in all these classes. This solution also adds a number of new methods to the algebraic structures interfaces, which increases the complexity.

If we decide not to allow conversions, or to only allow a subset of the defined conversions, we have to replace the file that contains them. No recompilation of the program is needed.

3.3. **The *Aspect Oriented* Approach.** The second approach uses Aspect Oriented Programming(AOP) [1].

AOP is a new methodology that provides separation of crosscutting concerns (as logging, authorization, etc.) by introducing a new unit of modularization, the

aspect that crosscuts other modules. With AOP it is possible to implement crosscutting concerns in aspects instead of fusing them in the core modules. An aspect weaver, which is a compiler-like entity, composes the final system by combining the core and crosscutting modules through a process called weaving. The result is that AOP modularizes the crosscutting concerns in a clear-cut fashion, yielding a system architecture that is easier to design, implement, and maintain. Aspect-oriented programming is a way of modularizing crosscutting concerns much like object-oriented programming is a way of modularizing core concerns.

Conversions are concerns that are separated from operations such as `equals(), plus(), etc`. In order to integrate conversions in the system using this approach, all we have to do is define an aspect: `ConversionsAspect` that contains the calls to the conversions. For each method that might use conversions we define a pointcut and an advice. The pointcut gathers the context and other necessary information, and the advice tells what it must be done when the pointcut is reached. The code below presents the pointcut for `equals()` when AspectJ [2] is used:

```
equals(AlgElem e1,AlgElem e2): target(e1) && args(e2) &&
 execution(* AlgElem+.equals(AlgElem) throws IncompatibleClassesException);
```

We consider here as join point the execution of the method `equals()` from `AlgElem` or any of its subclasses. After that, we define an advice for it. Because this method might throw an exception when there is no conversions between the two algebraic elements, and the body of `equals()` is no longer executed, we need to use the `around()` advice:

```
Object around(AlgElem e1,AlgElem e2)
           throws IncompatibleClassesException : equals(e1,e2){
 if (e1.getClass()!=e2.getClass()){
     if (Conversions.getInstance().existConversion(
           e1.getClass(),e2.getClass())){
AlgElem[] conver=Conversions.getInstance().convert(e1,e2);
return new Boolean(conver[0].equals(conver[1]));
    }else
     throw new IncompatibleClassesException("There is not defined
         any conversion between "+e1.getClass()+" "+e2.getClass());
 }else
    return proceed(e1,e2);
}
```

The pointcuts and advices are very similar for all operations, but after a call to conversions the context might change, so we must take into consideration this situation, as well. For example, if we try to compare a real to a complex, the `equals()` method called belongs to the first parameter (which belongs to the `Real` class), but after the conversion we need to call the `equals()` method from the `Complex class`, and this is not possible using AspectJ `proceed()` statement, which remembers the states of each parameters from the advice.

If later we decide not to use conversions, we have to recompile the system without the `ConversionsAspect`. We could replace this aspect with another one to check the compatibility of the parameters type, but it is not mandatory.

Using AOP there is no need to add more methods in the algebraic structure classes, the conversion crosscutting concern is kept in one place, and if in the future more operations that need conversions are added, the only place that must be changed/modified is the aspect. If the Conversion module is changed, modifications must also be done in only one place, the `ConversionsAspect` aspect.

4. Conclusion

We have analyzed how conversions could be added to an algebraic system. The design is based on reflection and dynamic loading, and can be integrated using two approaches. The first one uses *Template Method* design pattern. This approach is easy to understand and allows adding new types and operations that might use convertions without recompilation. But it also has some disadvantages. For each operation a new template method has to be built and this increases the complexity of maintenance and extendibility. If the conversions are removed, the execution is still slown down because of the verifications done by the template methods. The second approach uses Aspect Oriented Programming. Using this approach there is no need to add new methods in the algebraic structures classes, the conversion crosscutting concern is kept in one place, and if in the future more operations that need conversions are added, the only place that must be changed is the aspect. But any addition of new types needs recompilation of the whole system. If the conversions are removed, we need to recompile the whole system, but the execution is not slown down by verifications.

References

[1] G. Kiczales, J. Lamping, A. Menhdhekar, C. Maeda, C. Lopes, J.-M. Loingtier, J. Irwin. *Aspect-Oriented Programming*, Proceedings European Conference on Object-Oriented Programming (ECOOP), Springer-Verlag, 1997, pages 220–242.

[2] The AspectJ web site: http://eclipse.org/aspectj.

[3] E. Gamma, R. Helm, R. Johnson, J. Vlissides, *Design Patterns: Elements of Reusable Object Oriented Software*, Addison-Wesley, 1995.

[4] Niculescu V, Moldovan G.S.,*Building an Object Oriented Computational Algebra System Based on Design Patterns*, Proceedings of 7th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC'05), IEEE Computer Press, 2005.

Babeş-Bolyai University, Faculty of Mathematics and Computer Science, Cluj-Napoca, Romania

E-mail address: vniculescu@cs.ubbcluj.ro

Babeş-Bolyai University, Faculty of Mathematics and Computer Science, Cluj-Napoca, Romania

E-mail address: grigo@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume L, Number 2, 2005

COMPUTING DEFAULT EXTENSIONS. A HEURISTIC APPROACH

MIHAIELA LUPEA

Abstract. Default logics represent a simple but a powerful class of nonmonotonic formalisms. The main computational problem specific to these logical systems is a search problem: finding all the extensions (sets of nonmonotonic theorems - beliefs) of a default theory. GADEL is an automated system based on a heuristic approach of the classical default extension computing problem and applies the principles of genetic algorithms to solve the problem. The purpose of this paper is to extend this heuristic approach for computing all type of default extensions: classical, justified, constrained and rational.

1. Introduction

The *nonmonotonic reasoning* is an important part of human reasoning and represents the process of inferring conclusions (only plausible, not necessary true) from incomplete information. Adding new facts may later invalidate these conclusions, called *beliefs*.

The family of default logics is based on first-order logic and introduces a new type of inference rules, called *defaults*. These special inference rules model laws that are true with a few exceptions, formalizing a particular type of nonmonotonic reasoning, called *default reasoning*. The differences among different variants of default logic are caused by the semantics of the defaults. These logical systems are sintactically very simple, but very powerful in their inferential process.

A *default theory* $\Delta = (D, W)$ consists of a set W of consistent formulas of first order logic (the *facts*) and a set D of *default rules*. A *default* has the form $d = \frac{\alpha:\beta_1,\ldots,\beta_m}{\gamma}$, where: α is called *prerequisite*, $\beta_1, \ldots, \beta_m$ are called *justifications* and γ is called *consequent*.

A default $d = \frac{\alpha:\beta_1,\ldots,\beta_m}{\gamma}$ can be applied and thus derive γ if α is believed and it is consistent to assumed $\beta_1, \ldots, \beta_m$(meaning that $\neg\beta_1, \ldots, \neg\beta_m$ are not believed).

2000 *Mathematics Subject Classification.* 03B79, 68T15, 68T27.

1998 *CR Categories and Descriptors.* I2 [**Artificial Intelligence**]: Logic in artificial intelligence – *default logics, nonmonotonic reasoning, theorem proving* .

Using the classical inference rules and the defaults, the set of facts, W, can be extended with new formulas, called *nonmonotonic theorems* (*beliefs*) obtaining *extensions*.

The set of defaults used in the construction of an extension is called the *generating default set* for the considered extension.

In this paper we will use the following notations ($d = \frac{\alpha:\beta}{\gamma}$):
$Prereq(d) = \alpha$, $Justif(d) = \beta$, $Conseq(d) = \gamma$, $Prereq(D) = \bigcup_{d\in D} Prereq(d)$,
$Justif(D) = \bigcup_{d\in D} Justif(d)$, $Conseq(D) = \bigcup_{d\in D} Conseq(d)$,
$Th(X) = \{A | X \vdash A\}$ the classical deductive closure of the set X of formulas.

The versions (classical, justified, constrained, rational) of default logic try to provide an appropriate definition of consistency condition for the justifications of the defaults, and thus to obtain many interesting and useful properties for these logical systems:

- *Classical default logic* was proposed by Reiter [9]. Due to the individual consistency checking of justifications and thus the loss of implicit assumptions when are constructed the classical extensions, this logical system does not satisfy some desirable formal properties: semi-monotonicity, regularity, existence of extensions, commitment to assumptions.
- *Justified default logic* was introduced by Lukaszewicz [3]. The applicability condition of default rules is strengthen and thus individual inconsistencies between consequents and justifications are detected, but inconsistencies among justifications are neglected. In this logical system the existence of extensions and the semi-monotonicity property is guaranteed.
- *Constrained default logic* was developed by Schaub [10]. The consistency condition is a global one and it is based on the observation that in commonsense reasoning we assume things, we keep track of our assumptions and we verify that they do not contradict each other. This logic is strongly regular, semi-monotonic, strongly commits to assumptions and guarantees the existence of extensions.
- *Rational default logic* was introduced in [7] as a version of classical default logic, for solving the problem of handling disjunctive information. The defaults with mutually inconsistent justifications are never used together in constructing a rational default extension. This logic is strongly regular but does not guarantee the existence of extensions, is not semi-monotonic and does not commit to assumptions.

Automated theorem proving for default logics has began with solving the extension computing problem for particular default theories: normal, ordered semi-normal, and then was extended to general theories. The classical theorem proving

methods: resolution, semantic tableaux method, connection method, were incorporated and adapted in automated systems to solve specific tasks:

- **DeRes** [2] computes classical extensions for stratified default theories, using a semantic tableaux propositional prover;
- **Exten** [1] is based on an operational approach for computing classical, justified and constrained extensions;
- **Xray** [12] represents an approach of the query-answering problem in constrained and cumulative default logics;
- **DARR** [5] is a theorem prover for constrained and rational default logics based on a modified version of propositional semantic tableaux method.

Due to its very high level of theoretical complexity ($\sum_2^p - complete$), caused by the great power of the inferrential process, the problem of finding the extensions of a default theory, can be solved in an efficient manner only for particular classes of default theories.

In the paper [8] a heuristic approach of the classical extension computation problem is presented. An efficient automated system, called **GADEL** [8], which computes the classical extensions for propositional default logic using the principles of genetic algorithms was also developed.

The purpose of this paper is to extend this heuristic approach for computing all type of default extensions: classical, justified, constrained and rational.

2. Default logics

The results from [6] show that default theories can be represented by unitary theories (all the defaults have only one justification, $d = \frac{\alpha:\beta}{\gamma}$) in such a way that extensions (classical, justified, constrained, rational) are preserved. In this paper we will use only unitary default theories.

Definition 1. [13] A set X of defaults is *grounded* in the set of facts W if there is an enumeration $\langle d_i \rangle_{i \in I}$ of the defaults from X such that:

$$\forall i \in I \text{ we have } W \cup Prereq(\{d_0, d_1, ..., d_{i-1}\}) \vdash Prereq(d_i).$$

The following theorems provide global characterizations for classical, justified, constrained and rational extensions of a default theory using the generating default sets.

Theorem 1. [11] *Let (D,W) be a default theory, and let E be a set of formulas. E is a* **classical extension** *of (D,W) if and only if* $E = Th(W \cup Conseq(D'))$ *for a maximal set* $D' \subseteq D$ *such that D' is grounded in W and the following conditions are satisfied:*

- *For any* $\frac{\alpha:\beta}{\gamma} \in D'$: $W \cup Conseq(D') \cup \{\beta\}$ *is a consistent set;*
- *For any* $\frac{\alpha:\beta}{\gamma} \notin D'$: $W \cup Conseq(D') \cup \{\beta\}$ *is inconsistent* ***or*** $W \cup Conseq(D') \cup \{\neg\alpha\}$ *is consistent.*

This theorm shows that the defaults are nonmonotonic inference rules, meaning that conclusions derived using defaults can be later invalidated by adding new facts. The consistency condition for justifications is an individual one.

Theorem 2. [4] *Let (D,W) be a default theory, and let E, J be sets of formulas. (E,J) is a* **justified extension** *of (D,W) if and only if* $E = Th(W \cup Conseq(D'))$ *and* $J = Justif(D')$ *for a maximal set* $D' \subseteq D$ *such that D' is grounded in W and the conditions:*

- $\forall d \in D'$*: the set* $W \cup Conseq(D') \cup Justif(d)$ *is consistent.*

are satisfied.

The justifications of the generating default set satisfy an individual consistency condition (stronger than in the classical default logic) and are memorized in a support set J. Unfortunatly this set may be inconsistent, and thus two formulas of the actual extension E may be derived using contradictory assumptions.

Theorem 3. [10] *Let (D,W) be a default theory, and let E, C be sets of formulas. (E,C) is a* **constrained extension** *of (D,W) if and only if* $E = Th(W \cup Conseq(D'))$ *and* $C = Th(W \cup Conseq(D') \cup Justif(D'))$ *for a maximal set* $D' \subseteq D$ *such that D' is grounded in W and the following condition is satisfied:*

- *the set* $W \cup Conseq(D') \cup Justif(D')$ *is consistent.*

Each constrained extension is generated by a set of defaults whose justifications and consequents are together consistent, and at the same time consistent with the set of facts. The actual extension E is embedded in a consistent context C where all the assumptions (justifications) used in the reasoning process are retained.

Theorem 4. [4] *Let (D,W) be a default theory, and let E, C be sets of formulas. (E,C) is a* **rational extension** *of (D,W) if and only if* $E = Th(W \cup Conseq(D'))$ *and* $C = Th(W \cup Conseq(D') \cup Justif(D'))$ *for a maximal set* $D' \subseteq D$ *such that D' is grounded in W and the conditions:*

- *the set* $W \cup Conseq(D') \cup Justif(D')$ *is consistent;*
- $\forall d \in D \backslash D'$ *we have:*
  - $W \cup Conseq(D') \cup \neg Prereq(d)$ *is consistent* ***or***
  - $W \cup Conseq(D') \cup Justif(D' \cup d)$ *is inconsistent,*

are satisfied.

The theorem above states that the reasoning context C must be consistent and the set of generating defaults must be maximal-active [7] with respect to W and the actual extension E, for a rational extension.

From theorems 1,2,3 and 4 we can conclude that all four types of extensions are deductive closures of the set W (explicit content) and the consequents of the **generating default set *D'***(implicit content).

According to the initial fixed-point definitions of all variants of default logic we have the following definitions for the generating default sets:

Definition 2. Let E_1 be a classical extension, (E_2, J) be a justified extension, (E_3, C_3) be a constrained extension and (E_4, C_4) be a rational extension of the default theory *(D,W)*, then we have:

- $GD^{E_1}_{(D,W)} = \left\{ \frac{\alpha:\beta}{\gamma} \in D | \alpha \in E_1 \text{ and } E_1 \cup \{\beta\} \text{ is consistent} \right\}$
  is the *generating default set* for the classical extension E_1;
- $GD^{(E_2,J)}_{(D,W)} = \left\{ \frac{\alpha:\beta}{\gamma} \in D | \alpha \in E_2 \text{ and } \forall \eta \in J \cup E2, E2 \cup \{\gamma, \eta\} \text{ consistent} \right\}$
  is the *generating default set* for the justified extension (E_2, J);
- $GD^{(E_3,C_3)}_{(D,W)} = \left\{ \frac{\alpha:\beta}{\gamma} \in D | \alpha \in E_3 \text{ and } C_3 \cup \{\beta, \gamma\} \text{ is consistent} \right\}$
  is the *generating default set* for constrained extension (E_3, C_3);
- $GD^{(E_4,C_4)}_{(D,W)} = \left\{ \frac{\alpha:\beta}{\gamma} \in D | \alpha \in E_4 \text{ and } C_4 \cup \{\beta\} \text{ is consistent} \right\}$
  is the *generating default set* for the rational extension (E_4, C_4).

From [11] and [13] we have the following result regarding the generating default sets of different types of extensions.

Theorem 5. *The generating default sets for every type of extension are grounded in the set of facts of the default theory.*

Example: In this example we illustrate many types of contradictory information in consequents and justifications of the defaults and show how the versions of default logic solve them. The default theory *(D,W)* with $W = \{F \vee C\}$ and $D = \left\{d1 = \frac{:A}{B}, d2 = \frac{:\neg A}{C}, d3 = \frac{:\neg B \wedge \neg F}{G}, d4 = \frac{:\neg B \wedge \neg C}{E}\right\}$ has:

- One classical extension: $E1 = Th(\{F \vee C, B, C\})$ with $D1 = \{d1, d2\}$ as a generating default set.
- Three justified extensions:
  - $(E1, J1)$=$(Th(\{F \vee C, B, C\}), \{A, \neg A\})$ with $D1$ as a generating default set;
  - $(E2, J2)$=$(Th(\{F \vee C, G, E\})$,$\{\neg B \wedge \neg C, \neg B \wedge \neg F\})$ with $D2$=$\{d3, d4\}$ as a generating default set;
  - $(E3, J3)$=$(Th(\{F \vee C, C, G\}), \{\neg A, \neg B \wedge \neg F\})$ with $D3 = \{d2, d3\}$ as a generating default set.
- Three constrained extensions:
  - $(E4, C4)$ =$(Th(\{F \vee C, B\}), Th(\{F \vee C, B, A\}))$ with $D4 = \{d1\}$ as a generating default set;
  - $(E5, C5)$ =$(Th(\{F \vee C, C, G\})$,$Th(\{F \vee C, G, \neg A, \neg B \wedge \neg F\}))$ with $D5 = \{d2, d3\}$ as a generating default set;
  - $(E6, C6)$ =$(Th(\{F \vee C, E\}), Th(\{F \vee C, E, \neg B \wedge \neg C\}))$with $D6 = \{d4\}$ as a generating default set.
- Two rational extensions: $(E4, C4)$ and $(E5, C5)$.

3. A heuristic approach of the extension computation problem

In this section we extend the heuristic approach of the classical extension problem from [8] to all types of default extensions: justified, constrained, rational. The theorems from the previous section show that the problem of finding extensions can be reduced to the problem of finding the generating default sets for those extensions.

In this heuristic approach we need to define a search space for generating default sets and an evaluation function to compute the fitness of each element of this space according to the definitions of different types of default extensions.

For a default theory *(D,W)* we define the search space as the set $CGD = 2^D$, representing all possible configurations, called *candidate generating default sets.*

Definition 3. For a default theory *(D,W)* and $X \in CGD$ we define:
- *candidate extension* associated to X: $CE(X) = Th(W \cup Conseq(X))$;
- *candidate context* associated to X: $CC(X) = Th(W \cup Conseq(X) \cup Justif(X))$;
- *candidate support set* associated to X: $CJ(X) = Justif(X)$.

For defining the evaluation function we need four intermediate functions: $f_0^{type}, f_1^{type}, f_2^{type}, f_3^{type}$, where *type=clas* for *clas*sical extensions, *type=just* for *just*ified extensions, *type=cons* for *cons*trained extensions and *type=rat* for *rat*ional extensions.

f_0^{type} rates if the candidate extension / candidate context is consistent or not as follows:

$$f_0^{clas}(X), f_0^{just}(X) = \begin{cases} 0 & \text{if } CE(X) \text{ is consistent} \\ 1 & \text{otherwise} \end{cases}$$

$$f_0^{cons}(X), f_0^{rat}(X) = \begin{cases} 0 & \text{if } CC(X) \text{ is consistent} \\ 1 & \text{otherwise} \end{cases}$$

f_1^{type} rates the correctness of the candidate generating default set according to the definitions of different types of default extensions:

$f_1^{type}(X) = \sum_{i=1}^{n} \pi(d_i)$, where $D = \{d_1, d_2, ..., d_n\}$

The next table defines $\pi(d_i)$ - a penalty for each default of D, where k>0.

case	$d_i \in X$	$CE(X) \vdash \alpha_i$	*Cond-justif*^{type}	$\pi(d_i)$	$d_i = \frac{\alpha_i : \beta_i}{\gamma_i}$
1	true	true	true	0	d_i correctly applied
2	true	true	false	k	d_i wrongly applied
3	true	false	true	k	d_i wrongly applied
4	true	false	false	k	d_i wrongly applied
5	false	true	true	k	d_i wrongly not applied
6	false	true	false	0	d_i correctly not applied
7	false	false	true	0	d_i correctly not applied
8	false	false	false	0	d_i correctly not applied

The condition $Cond - justif^{type}$ represents the consistency condition for the justifications of defaults according to the Definition 2 of the generating default sets for different types of extensions.

- $Cond - justif^{clas} : CE(X) \cup \{\beta_i\}$ is consistent;
- $Cond - justif^{just} : \forall \eta \in CJ(X) \cup \beta_i : CE(X) \cup \{\eta, \gamma_i\}$ is consistent;
- $Cond - justif^{cons} : CC(X) \cup \{\beta_i, \gamma_i\}$ is consistent;
- $Cond - justif^{rat} : CC(X) \cup \{\beta_i\}$ is consistent.

f_2^{type} rates the level of groundness of the candidate generating default set as follows: $f_2^{type}(X) = card(Y)$, where Y is the biggest grounded set $Y \subseteq CGD$.

f_3^{type} checks the groundness property of X:

$$f_3^{type}(X) = \begin{cases} 0 & \text{if } X \text{ is grounded} \\ 1 & \text{otherwise} \end{cases}$$

Definition 4. For a default theory *(D,W)* the evaluation function for a candidate generating default set $X \in CGD$ of an extension of $type \in \{clas, just, cons, rat\}$ is defined by:

$eval^{type} : CGD \longmapsto Z \cup \{\bot, \top\}$
if $f_0^{type}(X) = 1$
  then $eval^{type}(X) = \top$
  else if $f_1^{type}(X) = 0$ and $f_3^{type}(X) = 0$
    then $eval^{type}(X) = \bot$
    else $eval^{type}(X) = f_1^{type}(X) - f_2^{type}(X)$
  endif
endif

The following theorem provides a necessary and sufficient condition for a set of defaults to be a generating set for an extension of $type \in \{clas, just, cons, rat\}$ using the evaluation function $eval^{type}$.

Theorem 6. *Let (D,W) be a default theory. A candidate generating default set $X \in CGD$ generates an extension of $type \in \{clas, just, cons, rat\}$ if and only if $eval^{type}(X) = \bot$.*

Proof of "⇒":

Let (W,D) be a default theory, D' a generating default set for an extension of $type \in \{clas, just, cons, rat\}$ and suppose that $eval^{type}(D') \neq \bot$. We have two cases a) $eval^{type}(D') = \top$ **or** b) $eval^{type}(D') \in Z$.

a) If $eval^{type}(D') = \top$ then according to Definition 4, $f_0^{type}(D') = 1$, which means that:

- $CE(D') = Th(W \cup Conseq(D'))$ is inconsistent for classical and justified default logics.
  But from Theorem 1 and Theorem 2, with D' as a generating default

set, $W \cup Conseq(D')$ is consistent as a subset of consistent sets.
Thus we have that the deductive closure of a consistent set is inconsistent, which is a contradiction.
Therefore for a generating default set of a classical or justified extension we can not have $eval^{type}(D') = \top$.

- $CC(D') = Th(W \cup Conseq(D') \cup Justif(D'))$ is inconsistent for constrained and rational default logics.
But from Theorem 3 and Theorem 4, with D' as a generating default set, $W \cup Conseq(D') \cup Justif(D')$ is consistent.
Thus we have that the deductive closure of a consistent set is inconsistent, which is a contradiction.
Therefore for a generating default set of a constrained or rational extension we can not have $eval^{type}(D') = \top$.

b) If $eval^{type}(D') \in Z$, then according to Definition 4, we have $f_1^{type}(D') \neq 0$ or $f_3^{type}(D') \neq 0$

- $f_3^{type}(D') \neq 0$ means that D' is not grounded in W, which contradicts the fact that a generating default set for every type of extension is grounded in W.
- $f_1^{type}(D') \neq 0$ implies that $\exists d \in D$ such that $\pi(d) \neq 0$.
  - According to the definition of $\pi(d)$ there is only the case 5: $\exists d \in D - D'$ such that $\pi(d) \neq 0$, meaning that the default d is wrongly not applied. This contradicts the maximality of the generating default set D' for all types of extensions, from the theorems 1,2,3,4.
  - There are 3 cases for which $\exists d \in D'$ such that $\pi(d) \neq 0$, with the meaning that the default d is wrongly applied.
    i) case 3 and case 4 from definition of $\pi(d)$ imply that the condition for the prerequisite: $CE(D') \vdash \alpha$ is false, which contradicts the property of groundness for a generating default set.
    ii) case 2 from the same definition of penalty implies that the consistency condition for the justification of the default d is false, which contradicts Definition 1 of the generating default sets for all types of extensions.

Therefore for a generating default set of any $type \in \{clas, just, cons, rat\}$ of extension we can not have $eval^{type}(D') \in Z$.

From a) and b) we can conclude that $eval^{type}(D') = \bot$, where D' is a generating default set of any $type \in \{clas, just, cons, rat\}$ of extension.

Proof of "$\Leftarrow$": Suppose that $X \in CGD$ is a candidate generating default set of $type \in \{clas, just, cons, rat\}$ of extension and $eval^{type}(X) = \bot$. From Definition 4 and $eval^{type}(X) = \bot$ we have that $f_0^{type}(X) = 0$, $f_1^{type}(X) = 0$, $f_3^{type}(X) = 0$.

- $f_0^{type}(X) = 0$ means that $CE(X) = Th(W \cup Conseq(X))$ is consistent for classical and justified extensions and $CC(X) = Th(W \cup Conseq(X) \cup$

$Justif(X)$) is consistent for constrained and rational extension. From here we have that $W \cup Conseq(X)$ is consistent, respectivelly $W \cup Conseq(X) \cup Jusif(X)$ is consistent.

- $f_3^{type}(X) = 0$ means that X is grounded in the set of facts W.
- $f_1^{type}(X) = 0$ in cases 1,6,7,8 implies that the conditions for prerequisites and justifications for the defaults from X are satisfied according to different types of extensions, meaning that the defaults from X are generating defaults. $f_1^{type}(X) = k$ in cases 2,3,4,5 implies that all the defaults from D-X can not be generating defaults.

Now we can easy prove that the conditions from the Theorems 1,2,3 and 4 are satisfied, therefore X is a generating default set for different types of extensions.

Example - continued: we will calculate the evaluation function for different candidate generating default sets, according to different types of extensions.

- $eval^{clas}(D1) = \bot$, and thus D1 is a generating default set for E1 because:
  - $CE(D1) = Th(\{F \vee C, B, C\})$ consistent implies $f_0^{clas}(D1) = 0$;
  - the defaults from *D1* have no prerequisites, and thus $f_3^{clas}(D1) = 0$;
  - $f_1^{clas}(D1) = \pi(d1) + \pi(d2) + \pi(d3) + \pi(d4) = 0 + 0 + 0 + 0 = 0$ according to the definition of the penalty. d1, d2 are correctly applied ($Cond - justif^{clas}$ for d1 and d2 is satisfied) and d3, d4 are correctly not applied ($Cond - justif^{clas}$ for d3 and d4 is not satisfied).

- $eval^{cons}(D1) = \top$, therefore D1 is not a generating default set for a constrained extension because $f_0^{cons}(D1) = 1$ ($CC(D1) = Th(\{F \vee C, B, C, A, \neg A\})$ is inconsistent).

- $eval^{cons}(D5) = eval^{rat}(D5) = \bot$, therefore $D5 = \{d2, d3\}$ is a generating default set for the constrained and rational extension *(E5,C5)* because:
  - $CC(D5) = Th(\{F \vee C, G, \neg A, \neg B \wedge \neg F\})$ is consistent, and thus $f_0^{cons}(D5) = f_0^{rat}(D5) = 0$;
  - no prerequisites for *d2* and *d3* implies $f_3^{cons}(D5) = f_3^{rat}(D5) = 0$;
  - $f_1^{cons}(D5) = f_1^{rat}(D5) = 0$, there is no penalty for the defaults of *D5*: d2, d3 are correcty applied and d1, d4 are correctly not applied.

- $eval^{cons}(D6) = \bot$ but $eval^{rat}(D6) \in Z$, therefore $D6 = \{d4\}$ is a generating default set for the constrained extension *(E5,C5)* but can not be a generating default set for a rational extension as follows:
  - $CC(D6) = Th(\{F \vee C, E, \neg B \wedge \neg C\})$ is consistent, and thus $f_0^{cons}(D6) = f_0^{rat}(D6) = 0$;
  - no prerequisite for *d4* implies $f_3^{cons}(D6) = f_3^{rat}(D6) = 0$;
  - $f_1^{cons}(D6) = 0$, there is no penalty for the defaults of *D6*: d4 is correcty applied and d1,d2,d3 are correctly not applied.
  - $f_1^{cons}(D6) \neq 0$, there is a penalty for the defaults d1 and d2 because they are wrongly not applied (case 5: the conditions for prerequisites and justifications are satisfied, but d1, d2 does not belong to D6).

If we try to add one or both of these defaults to D6 to obtain a new candidate default set, the new candidate context will not be consistent, which means that even if d1, d2 can be applied, their application will give inconsistency.

4. Conclusions

Based on the results from [8], in this paper we proposed a heuristic approach of the extension computing problem for all variants of default logic: classical, justified, constrained and rational. The evaluation function is used to compute the fitness of the elements of the search space according to the definitions of the generating default sets for different types of extensions. Future works will consist in developping an automated system, based on this approach and applying the principles of genetic algorithms.

References

[1] G. Antoniou, A.P. Courtney, J. Ernst, MA. Williams, *A System for Computing Constrained Default logic Extensions*, Logics in Artificial Intelligence, JELIA'96, Lecture Notes in Artificial Intelligence, 1126, 1996, pg. 237-250.

[2] P. Cholewinski, W. Marek si M. Truszczynski, *Default reasoning system DeReS*, Proceedings of KR-96, Morgan Kaufmann, 1996, pg. 518-528.

[3] W. Lukasiewicz, *Considerations on default logic - an alternative approach*, Computational Intelligence, 4, 1988, pg. 1-16.

[4] M. Lupea, *Nonmonotonic reasoning using default logics*, Ph.D. Thesis, Babes-Bolyai University, Cluj-Napoca, 2002.

[5] M. Lupea, *DARR - A theorem prover for constrained and rational default logics*, Studia Universitas Babes-Bolyai, Informatica, XLVII, No.1, 2002, pg. 45-52.

[6] W. Marek, M. Truszczynski, *Normal form results for default logics*, Non-monotonic and Inductive logic, Springer Verlag, LNAI 659, 1993, pg. 153-174.

[7] A. Mikitiuk, M. Truszczynski, *Rational default logic and disjunctive logic programming*, A. Nerode, L.Pereira, Logic programming and non-monotonic reasoning, MIT Press, 1993, pg. 283-299.

[8] P. Nicolas, F. Saubion, I. Stephan, *Genetic algorithm for extension search in default logic*, 8-th International Workshop on Non-Monotonic Reasoning, 2000.

[9] R. Reiter, *A Logic for Default reasoning*, Artificial Intelligence 13, 1980, pg. 81-132.

[10] T.H. Schaub, *Considerations on default logics*, Ph.D. Thesis, Technischen Hochschule Darmstadt, Germany, 1992.

[11] T.H. Schaub, *The automation of reasoning with incomplete information*, Springer-Verlag, Berlin, 1997.

[12] T.H. Schaub, *XRay system: An implementation platform for local query-answering in default logics*, Applications of Uncertainty Formalisms, Lecture Notes in Computer Science, vol 1455, Springer Verlag, 1998, pg. 254-378.

[13] C. Schwind, *A tableaux-based theorem prover for a decidable subset of default logic*, Proceedings CADE, Springer Verlag, 1990.

Babeş Bolyai University, Cluj Napoca,Romania
E-mail address: lupea@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume L, Number 2, 2005

SOME REMARKS ABOUT FEATURE SELECTION IN WORD SENSE DISCRIMINATION FOR ROMANIAN LANGUAGE

DANA AVRAM LUPSA, DOINA TATAR

Abstract. The problem of feature selection in Word Sense Discrimination (a subtask of Word Sense Disambiguation) is crucial for the accuracy of results. The paper proposes as a new feature the length of words [1]. Some combination between this feature and other features usually used are studied and presented.

1. Introduction

The task of Word Sense Discrimination is to divide the occurrences of a word into a number of classes. It differs from the more general Word Sense Disambiguation (WSD) [20, 4] in the fact that we need only to determine which occurrences have the same meanings and not what the meaning actually is. The result is that a reference to an external knowledge source for sense definition is not required for the task of Word Sense Discrimination. Since WSD is a necessary step in a large range of applications, for many problems in information access it is sufficient to solve the discriminating problem only.

In this paper we present some remarks about feature selection in context-group discrimination method. The method was first introduced by Shutze ([15]) and consist in an unsupervised grouping of a set of contextually similar occurrences of an ambiguous word into a same cluster. The approach of this problem is based on the strong contextual hypothesis of Miller and Charles ([6]) which states that "two words are semantically related to the extent that their contextual representations are similar".

In [9] the authors systematically compare unsupervised word sense discrimination using different features of representing contexts and different clustering methods. In this paper we present some experiments made with SenseCluster using a corpus in Romanian language.

The hypothesis we introduce in this paper is that longer words carry more semantic significance on their own.

Received by the editors: November 1, 2005.

2000 *Mathematics Subject Classification.* 68T50, 68Q32.

1998 *CR Categories and Descriptors.* I.2.7 [**Computing Methodologies**]: ARTIFICIAL INTELLIGENCE – *Natural Language Processing* .

The paper is structured as follows. Section 1 and 2 contain an introduction in Word Sense Discrimination problem and a presentation of SenseClusters tool. Section 3 presents the problem of feature selection in learning algorithms as well as a new introduced feature characterisation. Section 4 contains the experiment and evaluations of the results. In Section 5 some conclusions and future directions are presented.

2. SenseClusters

SenseClusters is a freely available word sense discrimination system ([17]) developed at the University of Minnesota, Duluth. It provides support for [9]: feature selection from an input corpus selected by user, for several different context representation methods, for various clustering algorithms and for evaluation of the discovered clusters. SenseClusters creates clusters made up of some contexts (instances) in which a given target ambiguous word occurs. A context is a group of 2 or 3 sentences, one of which contains the target word. Processing starts by selecting a corpus (in format of Senseval contests) and then selecting of features. SenseClusters supports the use of most frequent words (unigrams), the most frequent groups of two words with or without words intervening between them (bigrams) and co-occurence features (bigrams that include the target ambiguous word). The method used by the system is to represent each context as a vector. This vector could be binary, if it shows that a feature occurs or not in the context, or the frequency vector, if it shows how often the feature occurs in the context. This association of features with contexts is called "first order context vector", as different of "second order context vectors", introduced in [15]. There, the context vector is the average of the first order vectors associated with the words that occur in the context.

SenseClusters interface provides support for a number of clustering techniques provided by CLUTO, a Clustering Toolkit ([12]). It also offers the options for a number of similarity measures as simple matching, the cosine, the Jaccard and the Dice measures.

SenseClusters produces clusters of contexts where each cluster refers to a particular sense. The evaluation of these clusters is made with the help of an existing external knowledge of correct senses (the gold standard senses). The system produces a confusion matrix which shows the distribution of correct senses in each of the discovered clusters. The problem of assigning the maximally accurate discrimination becomes one of re-ordering the columns of the confusion matrix to maximize the diagonal sum. This method corresponds to several known methods in operation research.

3. Feature selection in learning algorithms

Notational conventions for WSD used in the following are as in [4], [20]:

- w– the ambiguous word (*target word*);
- $v_1, \cdots, v_J$— words used as contextual features for disambiguation of w.

Regarding $v_1, \cdots, v_J$, there are many possibilities. In [5] the author enumerated some sets of good indicators of word senses which could be selected as features:

- 0-param features, which can be used or not, without any parameter to set (as example the part of speech (POS) of a surrounding word). In addition in [5] are mentioned 0-param features as: verb before, verb after, noun before, noun after, named entity before, named entity after, preposition before, preposition after, pronoun before, pronoun after;
- 1-param features, which have one variable parameter that can be set to a specific value; (for example the length of a window of surrounding words or the position of a collocated word with the ambiguous word w);
- 2-param features, which have two parameters associated. As an example consider "a number" of words (the first parameter) which occur at least "a number" of times (the second parameter). As a second example consider "a number" of bigrams (first parameter) occurring at least "a number" of times (the second parameter).

A system used at contest Senseval 2 during the *English all words* task and *English lexical sample task*, based on these features selection, was ranked as the best performing one in the ranking made before the deadline.

In [7] the features are considered in the following three categories:

- morphological features (number for nouns, tense for verbs);
- POS features of two words immediately preceding and following the ambiguous word;
- collocation features which indicate if a particular word occurs in a window with the ambiguous word.

3.1. **New Word Feature Characterization.** In this paper we propose to take into consideration also the length of the words and we examinate this on the Romanian language word sense discrimination case.

Our ideea is based on two facts:

(1) The first is that longer words carry more semantic information of their own. For example, most prepositions and conjunctions are shorter words in a given language.
(2) The other is that longer words are less predilect to accumulate new meanings .

In Romanian, a special case of written word polysemy exists in the case of different words with different pronunciation and the same spelling. In what follows we will refer to it as *f-homonimy*. Because f-homonims are different words with the same written form, in what follows we need to make the distinction between the word and the vector of letters that constitutes the written form of the word.

Whenever we need to make this distinction, we will use the term "word form" to refere the letters of the written form of the word.

In order to test our intuition (item (2) in the list above), for each possible word length we compared the number of word forms in Romanian language with the number of word forms which have f-homonims. In this paper, we study the case of f-homonimy instead of polisemy for Romanian language, because a free version of DEX ([2]) is available and because distinct words with the same form (f-homonims) are easy to identify because they have distinct entries in DEX.

By using the Romanian DEX ([2]) we have extracted all distinct word forms that are entries in DEX and also all word forms that have more than one entry in DEX. Their absolute frequencies for each possible word length is depicted on the same graphic (figure 1).

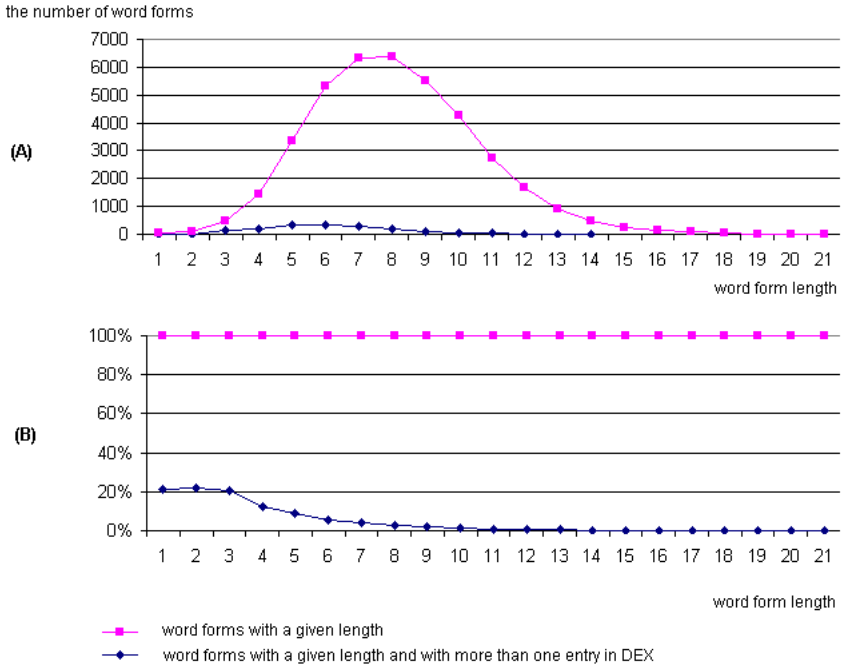


FIGURE 1. The number of F-Homonymic word lenghts compared with lengths of all words for Romanian language

Sub-Figure (1.A) presents the histogram of the absolute word form frequency for a given length and absolute frequency of f-homonimic word forms. Sub-Figure (1.B) represents the ratio between frequency of f-homonimic word forms and frequency of all word forms for each possible word length. The representation from

(1.B) indicates more visibly that the ratio between f-homonimic word forms frequency and word forms frequency is decreasing while word length is increasing. This is the remark (related to 2) we intend to take advantage of when we introduce the feature selection parametrized by the length of the word feature.

4. The Experiment

4.1. **How we evaluate our hypothesis.** The hypothesis we introduce in this paper is that longer words carry more semantic significance on their own and are less polysemantic. A consequence of this is that selecting longer words as attributes of a context should characterize better the context.

One way of doing word sense discrimination is to cluster the contexts of ambiguous word. A cluster of contexts corresponds to contexts of the same meaning of the given word. Choosing better attributes for the contexts should bring better results for the word sense discrimination process.

We evaluate the importance and the influence on clustering process by selecting different word lengths as context attributes and we use that for clustering contexts of some polisemous Romanian words. For clustering contexts of word occurences, we use the SenseClusters program. A presentation of the application was made by its authors in [3], [13], [12]. The use of clustering similar contexts in word sense discrimination and the influence of different parameters is studied in [9], [11],[10].

We want to represent the meaning of a context as an average meaning of the words that appear in the context. Following this idea, we choose to use the agglomerative clustering method with average link criteria function. There are argues in literature [11], [10] that the average link criteria function fares well.

We use the unigram and bigram type of feature. There is not a consentaneous opinion about which of them is better. The work presented in [8] emphasizes the importance of bigram in word sense dezambiguation, while in [11] co-occurences and unigrams achieved the overall best results. For our data, unigrams performed better than bigrams.

From SenseClusters point of view, bigrams features are pairs of words that occur in a given order within some distances from each other. We choose a window size of three, meaning that there could be at most one intervening word between the first and the second word that make a bigram.

Unigrams are single words that occur in the same context as the target word and they are made up of all the words found in context.

The new parameter we introduce is the length of feature words. We used as word length parameter the values 2, 3, ..., 10. In general, the length parameter indicates that the length of the word feature is greater than the parameter value. For unigrams, it means that selected feature words are longer than the indicated value. In the case of bigram attributes, this parameter refers to the two feature words which are selected according to the bigram model. In this case we enforce that both feature words to be longer than the length parameter.

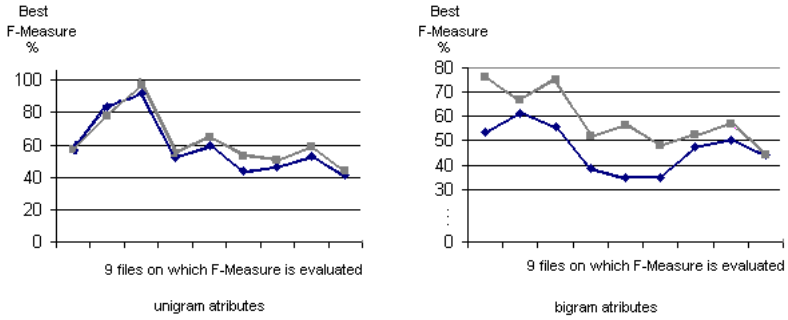


FIGURE 2. Best values of FMeasure (in percent) with and without word length filter (on the gray line - with word length filter; on the black line - without word length filter)

For each dataset and the two type of attribute we have worked with, we compared the best result obtained with using the word length parameter and without selecting attributes based on their word length. The results are presented in table 4.1. The dark color represents the best results obtained without any restriction of word length and the light color represents the results obtained by using the word length restriction.

That type of general results obtained by the application with and without using word length parameter justifies the fact the word length parameter is worthy to be studied. In the next section we are going to present in detail the results for all the combinations of parameters we have studied.

4.2. **Data.** We use as test data a Romanian corpus from SenseEval [1] that is not annotated with POS information. It contains contexts for 39 words, among them there are 25 nouns. There are about 1 million words and 7674 contexts. The file contains 248 number of senses to be disambiguated/discriminated.

We choose the words *actiune* (*action*), *eruptie* (*eruption*), *problema* (*problem*). For each of them we have selected three set of contexts, as follows:

- one is formed by all contexts of the word and its senses in the original selected SenseEval file;
- the other two sets of contexts are built by dividing the corpus into two parts, with almost the same size.

The characteristics of the chosen contexts are presented in table 1.

4.3. **Evaluation Method.** In this study we use all the 9 datasets presented in subsection 4.2 for each word length parameter value in $\{2, 3, \ldots, 10\}$. We evaluated each result by using F-measure values computed by SenseClusters.

[1] We used the file RomanianLS.unlabeled

word	number of		
	senses	contexts	words in contexts
actiune	8	299	about 50000
actiune	7	138	
actiune	8	161	
eruptie	2	54	about 8500
eruptie	2	18	
eruptie	2	36	
problema	6	288	about 45000
problema	5	142	
problema	5	146	

TABLE 1. Characteristics of the sets of word contexts

For overall evaluation we used two methods. One is by computing the average of the F-measure values of all datasets for each word length parameter (figures 3(a) and 4(a)). The other is based on ranking the F-measure values and we present it in what follows.

Borrowing ideas from the notion of Pareto dominance ([18], [19]) we define the dominance number (definition 2) and use it for a second type of evaluation of the importance of word length parameter.

Definition 1 (Better solution)**.** *Let $\mathcal{S}$ be solution space, $x, y \in \mathcal{S}$ and $eval : \mathcal{S} \to \Re$ a solution evaluation function. We define:*

$$better\_solution(x; y) = \begin{cases} 1 & if\ eval(x) \geq eval(y) \\ 0 & if\ eval(x) \leq eval(y) \end{cases}$$

In other words, the definition (1) says that $better\_solution(x; y) = 1$ iff x is a better solution than y; otherwise $better\_solution(x; y) = 0$.

Definition 2 (Dominance number)**.** *Let $\mathcal{S}$ be solution space , $y, x_1, x_2, \ldots x_n \in \mathcal{S}$ and $eval : \mathcal{S} \to \Re$ a solution evaluation function. The dominance number of y over $x_1, x_2, \ldots x_n$ is:*

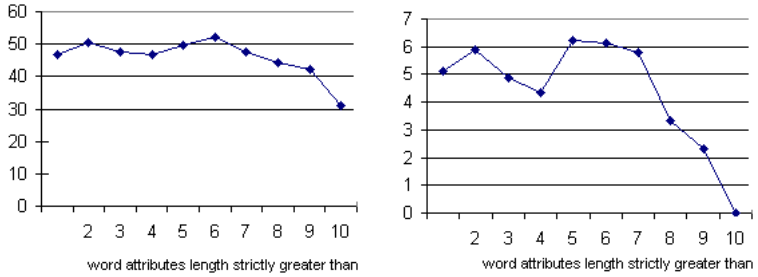
$$dominance\_number(y; x_1, x_2, \ldots x_n) = \sum_{i=1}^{n} better\_solution(y, x_i)$$

Let us consider a data set and the F-measure value for each parameter value. We computed the dominace number (definition 2) for each parameter value. The overall evaluation of each parameter value is made by averaging the dominace number for all data sets considered.

4.4. **Results.** We do that independently for bigram and unigram feature and for each word length parameter. The results for bigram feature is presented in figure 3

and the results for unigram feature are presented in figure 4. On x axis, graphic representations contain the results for each value used as length parameter. The y axis correspond to the average of the evaluation measure.

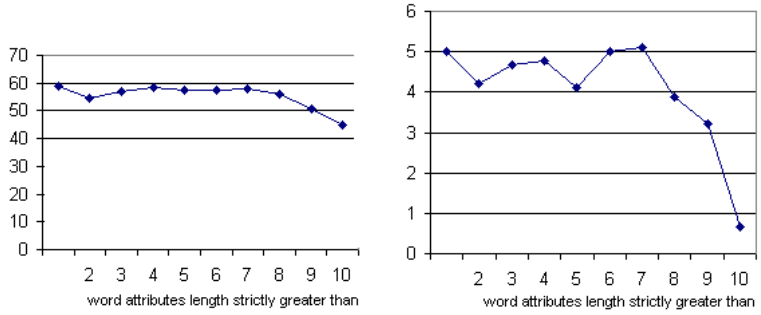
The use of the average of dominance number values and the results are given in figures 3(b) and 4(b).



(a) average of FMeasures

(b) average of dominace numbers corresponding to FMeasures

FIGURE 3. Evaluation for bigram word features



(a) average of FMeasures

(b) average of dominace numbers corresponding to FMeasures

FIGURE 4. Evaluation for unigram word features

It can easily be observed that the best results for bigram type of features are achieved for length parameter with value six if we are giving credit to the evaluation that use average of F-measure values (figure 3(a)). If we are using the evaluation technique based on dominance number (figure 3) we should say that we get better results if we select the words longer than five characters. As the two different

evaluation techniques get slightly different results, we could only claim that best results are obtained for length parameter with value five or six.

When using unigrams, first of the two evaluation methods indicates as best results those obtained without using the length parameter (or choosing the value 0 for this parameter), closely followed by results obtained for length parameter with value four (figure 4(a)). The second evaluation indicates selection of word feature by the length greater than seven as getting best results (figure 4(b)).

5. Conclusions

The experiments confirm the intuition that the results are better when using longer words as features. The length parameter value for which the results are better, cover the set $\{4, 5, 6, 7\}$. The non-achievement of this study is that we couldn't indicate a unique best value for the word length parameter.

Some future studies about other features which could improve word sense discrimination results and the possibilities to integrate these conclusions in existing WSD methods are in this moment in our attention.

References

[1] D. Avram: *Extragerea informatiilor de tip semantic din texte folosind clasificare nesupervizata*. PhD. Thesis, Babes-Bolyai University Cluj-Napoca (in Romanian)(in preparation)

[2] *dexonline - Romanian **EX**planatory **D**ictionary* , 2004.
http://www.dexonline.ro/

[3] A. Kulkarni, T. Pedersen: *SenseClusters: Unsupervised Clustering and Labeling of Similar Contexts*. Proceedings of the ACL Interactive Poster and Demonstration Sessions, pp. 105–108, 2005
http://www.aclweb.org/anthology/P/P05/P05-3027

[4] C. Manning, H. Schutze: *Foundation of statistical natural language processing*. MIT, 1999

[5] R. Mihalcea: *Word Sense Disambiguation with pattern learning and automatic feature selection*. Natural Language Engineering, 1:1–15, 2002

[6] G.Miller, W.G. Charles: *Contextual corelates of semantic similarity*. Language and Cognitive Processes, 6:1–28, 1991

[7] T. Pedersen, R. Bruce, J. Wiebe: *Sequential Model Selection for Word Sense Disambiguation*. Proceedings of the Fifth Conference on Applied Natural Language Processing (ANLP-97), 1997
http://www.d.umn.edu/∼tpederse/Pubs/anlp97.ps

[8] T. Pedersen: *A decision tree of bigrams is an accurate predictor of word sense*. In Proceedings of the Second Annual Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL-01), 2001
http://www.d.umn.edu/∼tpederse/Pubs/naacl01.pdf

[9] A. Purandare, T. Pedersen: *SenseClusters - Finding Clusters that Represent Word Senses*. In Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI), 2004
http://www.d.umn.edu/∼tpederse/Pubs/AAAI04PurandareA.pdf

[10] A. Purandare, T. Pedersen: *Word Sense Discrimination by Clustering Contexts in Vector and Similarity Spaces*. Proceedings of the Conference on Computational Natural Language

Learning (CoNLL) 2004
http://www.d.umn.edu/∼tpederse/Pubs/conll04-purandarep.pdf
[11] A. Purandare: *Unsupervised Word Sense Discrimination by Clustering Similar Context*. PhD. Thesis, University of Minnesota, 2004
[12] T. Pedersen and A. Kulkarni: *Identifying Similar Words and Contexts in Natural Language with SenseClusters*. Proceedings of the Twentieth National Conference on Artificial Intelligence, 2005
http://www.d.umn.edu/∼tpederse/Pubs/aaai2005-demo-sc.pdf
[13] T. Pedersen: *Language Independent Methods of Clustering Similar Contexts*. Eurolan 2005 Summer School. Tutorials, 2005
[14] H. Schutze: *Automatic Word Sense Discrimination*. Computational linguistics, 24(1):97–123, 1998
[15] H. Schutze: *Dimensions of meaning*. Proceedings of Supercomputing '92, pp.787–796, 1992
[16]
[17] T. Pedersen, A. Purandare, A. Kulkarni: *Senseclusters home page*. 2005.
http://senseclusters.sourceforge.net/
[18] W. Stadler: *A Survey of Multicriteria Optimization, or the Vector Maximum Problem*. Journal of Optimization Theory and Applications 29:1–52, 1979
[19] R.E. Steuer: *Multiple Criteria Optimization: Theory, Computation and Application*. New York: Wiley, 1986
[20] D. Tatar: *Word sense disambiguation; a short survey*. Studia Univ. Babes-Bolyai, XLIX(2):17–27, 2004

Babes-Bolyai University, Faculty of Mathematics and Computer Science, Department of Computer Science, Cluj-Napoca, Romania
E-mail address: davram@cs.ubbcluj.ro, dtatar@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume L, Number 2, 2005

THE LAW OF WORD LENGTH IN A VOCABULARY

DANA AVRAM LUPSA, RADU LUPSA

Abstract. In the literature we can find linguistics laws that are then exploited by many applications. This paper presents an empirical law that describes the frequency of the words of a given length in a language's vocabulary, as well as the length of distinct words in a corpus. This is a law that applies to any language.

1. Introduction

In linguistics there are some general laws that have no imediate consequences for computational linguistics. There are also some laws that are a gold mine and are exploited by many applications on computational linguistics.

From the first category are Zipf law and Heaps law. In linguistics, Zipf law [14] states that while only a few words are used very often, many or most are seldom used. The frequency of a word ranked the n-th (notated P_n) is given by the next relation: $P_n \approx \frac{1}{n^\alpha}$, where α is almost 1. This means that a word that occurs 10 times more frequently than another word, it is ranked 10 less. Heaps law [13] is an empirical law which describes the portion of a vocabulary which is represented by an instance document (or set of instance documents) consisting of words chosen from the vocabulary V. This can be formulated as $V_R(n) = K \times n^\beta$, where V_R is the subset of the vocabulary V represented by the instance text of size n. K and β are free parameters determined empirically. With English text corpora, typically K is between 10 and 100, and β is between 0.4 and 0.6. Heaps' law means that as more instance text is gathered, there will be diminishing returns in terms of discovery of the full vocabulary from which the distinct terms are drawn.

In the second category is placed the distributional hypothesis introduced by Harris [5] and which is widely used in NLP applications ([1], [6], [8] and the list can continue). The basic idea is that *we should know a word by the company it keeps* ([4]).

Received by the editors: November 5, 2005.

2000 *Mathematics Subject Classification.* 68R15, 68T50.

1998 *CR Categories and Descriptors.* G12 [**Mathematics of Computing**]: NUMERICAL ANALYSIS – *Approximation – Special function approximations*; I27 [**Computing Methodologies**]: ARTIFICIAL INTELLIGENCE– *Natural Language Processing – Text analysis* .

A quite complete and recent survey on linguistic principles and their applications can be found in [1], [2], [12].

This paper describes a law that we found. Section 2 presents the parametrized function that describes the frequency of the words length in a language's vocabulary. It is an empirical law which is stated as being general, in the sense that it applies for any language. To the best of our knowledge, this is the first time the distinct word length frequency in a vocabulary is stated as a law.

On the other hand, the distinct words in a corpus are an approximation of that language vocabulary (see Heap's law in [13]). In consequence, in section 3 we verify if the frequences of distinct word lengths in a corpus are described by the same law.

In this paper, our study is applied for two languages: Romanian and English. The justification of the fitting the data with the given law is done by computing the relative error.

2. The Law that Describe the Frequency of the Words of a Given Length in a Language's Vocabulary

This section presents the empirical law that describes the relation between the length of words and its absolute frequency (i.e. the number of distinct words for each possible word length) in a language vocabulary. The law takes into account the dictionary forms only, and each of them is counted once.

In the following experiments we rely of the fact that the dictionary word forms (also named basic forms) are found as entries in a dictionary and they are also the forms of the words in Wordnet synsets.

2.1. **The Law.** The absolute frequency of lengths of the words in a language vocabulary states that the absolute frequency of words of a given length is approximated by the function:

$$LV(x; c, k, \theta) = c \times x^k \times e^{-\frac{x}{\theta}} \tag{1}$$

where k, θ and c are parameters determined experimentally and $LV(x; c, k, \theta)$ is the law that describe the absolute frequency of the dictionary form of the words with length x in a language vocabulary.

2.2. **Experimental Data. The Vocabulary.**

2.2.1. *Romanian Experimental Data.* We took the Internet version of the Romanian explanatory dictionary ([3]), 1998 edition, off-line version, that we shall call *dex98*. Its database contains 41466 entries of 39531 distinct word forms.

2.2.2. *English Experimental Data.* We extracted the English words from the Wordnet for English. We rely on the fact that the English vocabulary is formed by all the words that appears in the Wordnet synsets. We also consider that synsets contains only the base form of the words. There were 74331 distinct words.

2.3. **The Hypothesis and Romanian Vocabulary.** In this section we present the way we approximate the parameters for LV function. We also compute the relative error with which the LV function approximates the Romanian vocabulary. Considering that the computed values leave room for further improvement, we also refine our search for values of the parameters and we point out that this leads to improvement.

2.3.1. *LV Parameters Estimation.* We selected all the distinct basic word forms that are found as entries in the dictionary. We grouped them by their length. In order to verify the hypothesis, we determined the values for c, k and θ, and we computed the approximation error of the parametrized LV function. We estimated the best values for c, k and θ in the following set of possible values:

$$\begin{array}{ll} c \in \{1, 2, \ldots, 100\} & , \\ k \in \{0.1, 0.2 \ldots, 9.9, 10.0\} & , \\ \theta \in \{0.1, 0.2, 0.3, \ldots, 9.9, 10.0\} & . \end{array} \tag{2}$$

For all these possible values of c, k and θ we computed the following sum:

$$\sum_{i=1}^{n} |LV(i, c, k, \theta) - d_i|$$

where n is the number of distinct lengths of the words, and d_i is the number of the words with length i. We took as the best parameters those that minimize the above sum, that is:

$$(c, k, \theta) \leftarrow argmin_{c,k,\theta} \sum_{i=1}^{n} |LV(i, c, k, \theta) - d_i| \tag{3}$$

The best parameters for LV describing the absolute words length frequency according to the Romanian dictionary are:

$$\begin{array}{ll} c = 1 & , \\ k = 9 & , \\ \theta = 0.8 & . \end{array} \tag{4}$$

The graphical representation of the distinct word length absolute frequency, along with the function from (1) with the parameters from (4) is shown in figure 1. Experimental data computed from *dex98*are given as vertical lines. The approximation function of (1) is given as the continous curve.

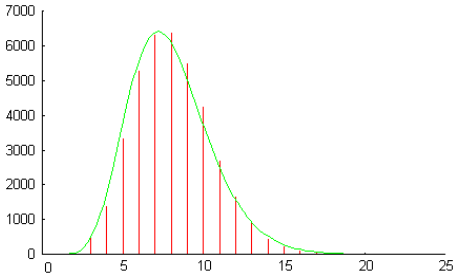


FIGURE 1. Distinct word length absolute frequency as of *dex98* approximated by LV with parameters from (4)

2.3.2. *Error of the Approximation.* Using the notations d_i for the absolute frequency of length i words and e_i the (theoretical) aproximation ($e_i = LV(i)$), we computed the absolute error (that we notate $AbsErr$):

$$AbsErr = \sum_{i=1}^{n} |d_i - e_i| = \sum_{i=1}^{21} |d_i - e_i| \approx 2848$$

The relative error (notated $RelErr$) is considered:

$$RelErr = \frac{AbsErr}{\sum_{i=1}^{n} |\max(d_i, e_i)|} \tag{5}$$

This relative error formula has the next properties:

- the minimum error value is 0
- the error is 0 if and only if $d_i = e_i$, $\forall i \in 1, 2 \ldots, n$
- the error is undefined if and only if $d_i = 0$ and $d_i = 0$ $\forall i \in 1, 2 \ldots, n$
- the maximum error value is 1
- the error is 1 if and only if
  - we are not in the next case: $d_i = 0$ and $d_i = 0$ $\forall i \in 1, 2 \ldots, n$
  - $d_i = 0$ or $e_i = 0$, $\forall i \in 1, 2 \ldots, n$

In our case, $d_i > 0$ and $e_i > 0$, $\forall i \in 1, 2 \ldots, n$, so the next relation holds:

$$0 < RelErr < 1 \ .$$

More precisely, for function from (1) and with parameter values from (4), the relative error is:

$$\begin{aligned} RelErr &= \frac{AbsErr}{\sum_{i=1}^{n} |\max(d_i, e_i)|} \\ &\approx \frac{2848}{40670} \approx 0.0700 = 7.00\% \end{aligned} \tag{6}$$

In the next section (2.3.3) we will see that the parameters can be approximated better by using smaller steps, and that, in this case, the relative error is reduced.

2.3.3. *Fine Tuning the Parameters.* The first five ranked estimated parameters (in section 2.3.1) for (c, k, θ) are presented in (7).

$$\begin{array}{llll} c = 1.00 & k = 8.50 & \theta = 0.90 & ; \\ c = 3.00 & k = 7.90 & \theta = 0.90 & ; \\ c = 1.00 & k = 9.00 & \theta = 0.80 & ; \\ c = 2.00 & k = 8.10 & \theta = 0.90 & ; \\ c = 4.00 & k = 7.80 & \theta = 0.90 & . \end{array} \tag{7}$$

Starting with the remark that the first five ranked parameters satisfy (8):

$$\begin{array}{lll} c & \in & \{1, 2, 3, 4\} \quad , \\ k & \in & [7.8, 9.00] \quad , \\ \theta & \in & \{0.8, 0.9\} \quad , \end{array} \tag{8}$$

we searched the best values for c, k, and θ in the following set of possible values:

$$\begin{array}{ll} c \in \{0.10, 0.11, \ldots, 4.99, 5.00\} & , \\ k \in \{7.00, 0.11, \ldots, 10.00\} & , \\ \theta \in \{0.10, 0.11, \ldots, 3.00\} & . \end{array} \tag{9}$$

The best identified parameters for (c, k, θ) are:

$$\begin{array}{ll} c = 0.32 & , \\ k = 9.89 & , \\ \theta = 0.75 & \end{array} \tag{10}$$

In this case, the relative error (computed with formula (5)) is:

$$\begin{array}{rl} RelErr & = \frac{\sum_{i=1}^{n} |d_i - e_i|}{\sum_{i=1}^{n} |\max(d_i, e_i)|} \\ & \approx 0.0262 = 2.62\% \end{array} \tag{11}$$

The new identified parameter are better; there is a decrease of the error by 62.4%.

The graphical representation of the distinct word length absolute frequency, along with the best approximation function of the form (1) with the parameters from (10) is shown in figure 2. Experimental data computed from *dex98*are given as vertical lines. The approximation function of (1) is given as the continous curve.

2.4. **The Hypothesis and English Vocabulary.** We estimated the LV parameter values for English vocabulary (see section 2.2.2) in the same way as in sections 2.3.1 and 2.3.3 for Romanian data.

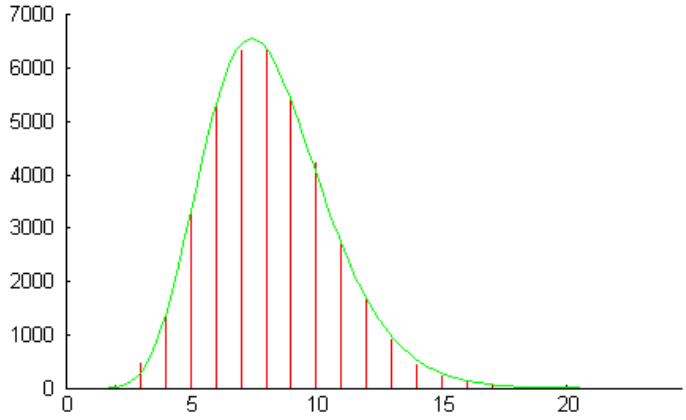


FIGURE 2. Distinct word length absolute frequency as of *dex98* approximated by LV with parameters from (10)

2.4.1. *LV Parameters Approximation.* By using the same method as in section 2.3.1 and searching possible parameter values as indicated in list (2), the best parameters we computed for (c, k, θ) are:

$$\begin{array}{ll} c = 3 & , \\ k = 7.8 & , \\ \theta = 1 & . \end{array} \tag{12}$$

2.4.2. *Error of the Approximation.* We computed the relative error as in section 2.3.2, formula (5). The relative error for parameter values from (12) is:

$$\begin{aligned} RelErr &= \frac{\sum_{i=1}^{n} |d_i - e_i|}{\sum_{i=1}^{n} |\max(d_i, e_i)|} \\ &= \frac{\sum_{i=1}^{27} |d_i - e_i|}{\sum_{i=1}^{27} |\max(d_i, e_i)|} \\ &\approx \frac{6698}{80023} \approx 0.0837 = 8.37\% \end{aligned} \tag{13}$$

2.4.3. *Fine Tuning the LV Parameters.* Best ranked five parameters are:

$$
\begin{array}{lll}
c = 6.00 & k = 7.10 & \theta = 1.10 \quad ; \\
c = 4.00 & k = 7.60 & \theta = 1.00 \quad ; \\
c = 2.00 & k = 8.40 & \theta = 0.90 \quad ; \\
c = 3.00 & k = 7.80 & \theta = 1.00 \quad ; \\
c = 7.00 & k = 7.00 & \theta = 1.10 \quad .
\end{array} \tag{14}
$$

Starting with the observation that the first five ranked parameters satisfy:

$$
\begin{array}{rcl}
c & \in & \{2, 3, 4, 6, 7\} \quad , \\
k & \in & [7.10, 8.40] \quad , \\
\theta & \in & \{0.9, 1.1\} \quad ,
\end{array} \tag{15}
$$

we estimated the best values for c, k, and θ in the following set of possible values:

$$
\begin{array}{l}
c \in \{0.10, 0.11, \ldots, 4.99, 8.00\} \quad , \\
k \in \{7.00, 0.11, \ldots, 10.00\} \quad , \\
\theta \in \{0.10, 0.11, \ldots, 3.00\} \quad .
\end{array} \tag{16}
$$

The best values identified for parameters (c, k, θ) are:

$$
\begin{array}{l}
c = 0.9 \quad , \\
k = 8.84 \quad , \\
\theta = 0.89 \quad .
\end{array} \tag{17}
$$

In this case, the relative error (see formula 5) is:

$$RelErr \approx 0.0572 = 5.72\% \tag{18}$$

There is a decrease of the error by 36.4%. This means that the new identified parameter are better.

The graphical representation of the distinct word length absolute frequency, along with the best approximation function of the form (1) is shown in figure 3. The function from (1) is represented by continuous line. The light color continuous line correspond to parameters from (12) and the dark color continuous line correspond to parameters from (17).

2.5. **Discussion.** We saw above that the distinct word length absolute frequency in a vocabulary is described by the parametrized function LV (see formula (1)). With the appropriate choice of the parameter values the absolute word length frequency in a vocabulary is aproximated by LV with precision of 97.38% for Romanian vocabulary (with parameter setting from (10)) and with precision of 94.28% for English vocabulary (with parameter setting from (17)).

We conclude that for a language there are c, k, and θ such that the frequency of the basic forms of the words of a given length is approximated by $LV(x, c, k, \theta)$, where x is the word length. This is *the law of the frequency of words Length in a Vocabulary.*

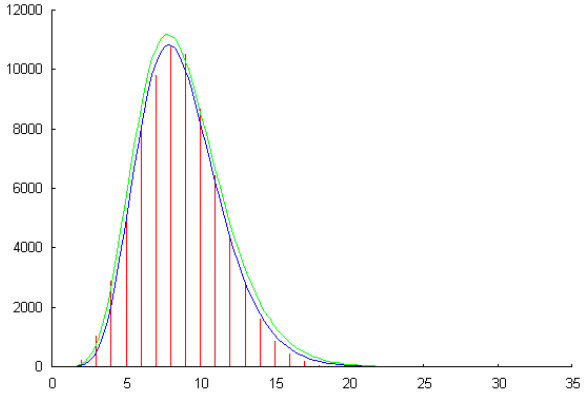


FIGURE 3. Distinct word length absolute frequency as of Wordnet approximated by LV with parameters from (10)

2.5.1. *A Short Comparison between Romanian and English Data.* We stated that the absolute word lengths frequency from vocabularies are approximated by the parametrised LV function (and we experimented that over Romanian and English vocabulary). If this is true, the shape of the histograms (in our case for Romanian and English) is the same.

We selected all the distinct basic word forms that are found in Romanian and English vocabulary (data presented in section 2.2) and grouped them by their length. The figure 4 represents the relative frequency of the distinct word lengths for both English and Romanian case. The length is represented on the x axis, and the relative frequency on the y axis. Note that the two histograms have the same shape.

3. The Frequency of the Distinct Word Forms of a Given Length in Texts

In this paragraph we shall see that the frequency of the distinct word lengths in texts is approximated by the same LV function.

Following the Heap's law [13], we could say that if a corpus is large enough, the distinct words from the corpus are an approximation for the words of the vocabulary of that language. That is why we expect that the frequencies of distinct words from a corpus, grouped by their length, to follow the same law. The difference between the distinct words in a corpus and the words in a vocabulary relies on the fact that the words which are entries in a dictionary (see section 2.2.1) are only the base form of the words. To one word from a dictionary usualy corresponds more than one word in a corpus; these are the flexioned forms of the word.

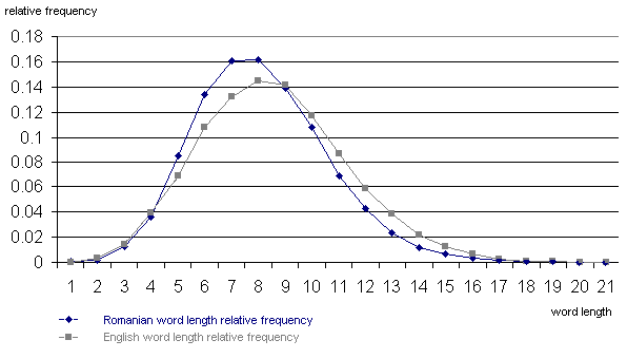


FIGURE 4. The relative frequency of distinct word lengths as of Romanian Dex and of English Wordnet.

We measured the length of each word in a given corpus, and, for each length, we counted all the distinct words of that length. We state that the number of distinct words of a given length is approximated by the LV function (see section 2.1).

We conducted the following experiment: we took all the different words from a Romanian language corpus, and we represented graphically the frequency of each word length as a function of the length.

3.1. **Romanian and English Corpora.** The Romanian corpus was automatically extracted from the Internet, by using a search by the words *limbaj* and *natural*. The corpus contains about 85000 words and 12500 among them are distinct. The English language corpus was constructed similarly to the Romanian one, by using a search by the words *natural* and *language*. The corpus contains about 50000 words and 5500 among them are distinct.

3.2. **The Approximation by LV Function Hypothesis.** The law of frequency of distinct words length in a corpus can be written as:

$$LV(x;k,\theta,c) = c \times x^k \times e^{-\frac{x}{\theta}} \tag{19}$$

where $LV(x;k,\theta,c)$ is the frequency of the distinct words of length x in the corpus (that is, the number of distinct words of length x in the corpus over the total number of distinct words in the same corpus), and k, θ, and c are experimentally determined parameters.

3.3. **LV Parameters Estimation.** Figures 5 and 6 show the approximated functions for the word length frequencies in texts. A discussion about the way the parameters are determined and the relative error of the approximation is given below.

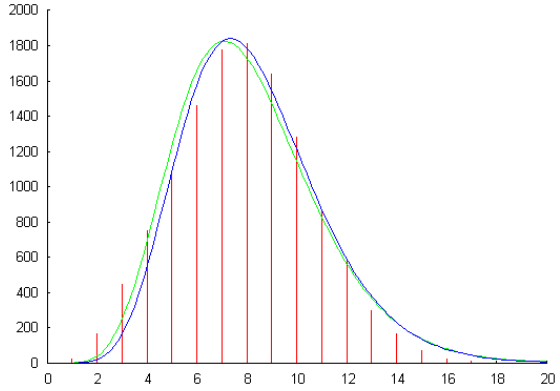


FIGURE 5. The approximation function for distinct word length absolute frequency in a Romanian corpus with two sets of parameters

We used steps from list (2) for the first approximation of LV parameters for Romanian texts. The estimated parameters were:

$$\begin{aligned} c &= 2 \quad , \\ k &= 7.10 \quad , \\ \theta &= 1.00 \quad . \end{aligned} \tag{20}$$

For those parameters, the relative error is 12.56%.

By verifing the values around the previously determined parameters by using smaller steps we determined the next parameters:

$$\begin{aligned} c &= 0.65 \quad , \\ k &= 7.99 \quad , \\ \theta &= 0.92 \quad . \end{aligned} \tag{21}$$

For those parameters we get better results; the relative error is 10.97%.

In figure 6, the light color represents LV function approximation with parameters from (20) and the dark color represents LV function approximation with parameters from (21).

We used steps from list (2) for the first approximation of LV parameters for English texts. The estimated parameters were:

$$\begin{aligned} c &= 12 \quad , \\ k &= 4.9 \quad , \\ \theta &= 1.3 \quad . \end{aligned} \tag{22}$$

For those parameters, the relative error is 7.36%.

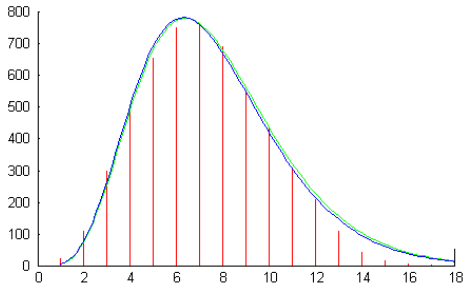


FIGURE 6. The approximation function for distinct word length absolute frequency in an English corpus with two sets of parameters

We verify the parameter values around the previously determined parameters by using smaller steps and we determined the next values:

$$\begin{aligned} c &= 12.75 \quad , \\ k &= 4.91 \quad , \\ \theta &= 1.28 \quad . \end{aligned} \tag{23}$$

For those parameters we get better results; the relative error is 6.55%.

In figure 6, the light color represents LV function approximation with parameters from (22) and the dark color represents LV function approximation with parameters from (23).

4. Conclusion and Future Research

This paper presents for the first time an empirical law which we call LV. It describes the frequency of the words of a given length in the vocabulary of a given language. It also approximates the frequency of distinct words length in a corpus. This is a law stated as being general, in the sense that it applies to any language. But we have studied it for two languages: Romanian and English. We intend to extend the verification of this law over other languages and we think to use EuroWordnet in order to do that.

The experiments indicates that most frequent in a vocabulary are words with length between six and ten. It is easy to see that words that do not carry semantic information of their own (as preposition, conjunction, auxiliary verbs) are among the shortest words in a language. We intend to use this remark to try to improve contexts clustering process by introducing a new clustering parameter which is word feature length. Language independent methods of clustering similar contexts ([7], [6], [10]) on which relies a multitude of other linguistic processing, as identifying similar words ([8]), name discrimination ([9]), word sense discrimination ([11]) do not use the length as a word feature parameter.

References

[1] S. Bordag. *Algorithms extracting linguistic relations and their evaluation.* 2005.
[2] S. Bordag and G. Heyer. *A structuralist framework for quantitative linguistics.* 2005.
[3] dexonline, Romanian **EX**planatory **D**ictionary 1998 (*dex98*). http://www.dexonline.ro/ (visited 2004).
[4] J.R. Firth. Modes of Meaning. In *Papers in Linguistics*, Oxford University Press, 1957.
[5] Z. Harris. *Mathematical structures of language.* Interscience Publishers, New York, 1968.
[6] A. Kulkarni and T. Pedersen. SenseClusters: Unsupervised clustering and labeling of similar contexts. In *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, 2005.
[7] T. Pedersen. Language independent methods of clustering similar contexts. In *Eurolan 2005 Summer School. Tutorials*, 2005.
[8] T. Pedersen and A. Kulkarni. Identifying similar words and contexts in natural language with senseclusters. In *AAAI 2005*, 2005.
[9] T. Pedersen, A. Purandare and A. Kulkarni. Name discrimination by clustering similar contexts. In *Proceedings of the Sixth International Conference on Intelligent Text Processing and Computational Linguistics (CICLING)*, 2005.
[10] T. Pedersen, A. Purandare and A. Kulkarni. Senseclusters home page, 2005. http://senseclusters.sourceforge.net/.
[11] A. Purandare and T. Pedersen. Word sense discrimination by clustering contexts in vector and similarity spaces. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL)*, 2004.
[12] B. Rieger. Computing granular word meanings. A fuzzy linguistic approach in computational semiotics, 2005.
[13] http://en.wikipedia.org/wiki/Zipf%27s_Law
[14] http://en.wikipedia.org/wiki/Heaps%27_law

Babes-Bolyai University, Faculty of Mathematics and Computer Science, Department of Computer Science, Cluj-Napoca, Romania

E-mail address: davram@cs.ubbcluj.ro, rlupsa@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume L, Number 2, 2005

ON TUBULAR SURFACES IN COMPUTER GRAPHICS

PAUL A. BLAGA

Abstract. We propose new approaches to the investigation of tubular and canal surfaces, regarded as swept surfaces, we give a parametric representation of the inverse of a canal surface and we suggest several applications of tubular surfaces in scientific vizualization.

1. Introduction

Tubular surfaces are among the surfaces which are easier to describe both analytically and "operationally". They are still under active investigation, both for finding best parameterizations (see, for instance, [9, 11]) or for application in different fields (for instance in medicine, see [6]).

We remind that, if C is a space curve, a *tubular surface* associated to this curve is a surface swept by a family of spheres of constant radius (which will be the radius of the tube), having the center on the given curve. Alternatively, as we shall see in the next section, for them we can construct quite easily a parameterization using the Frenet frame associate to the curve. The tubular surfaces are used quite often in computer graphics, but we think they deserve more attention for several reasons. For instance, there is the problem of representing the curves themselves. Usually, the space curves are represented by using *solids* rather then tubes. There are, today, several very good computer algebra system (such as Maple, or Mathematica) which allow the vizualisation of curves and surfaces, in different kind of representations. However, the graphical output they produce is not always of the best quality and it is, definitely, useful to be able to export the graphic in a format which is recognized by a professional software, such as RealStudio, or AutoCad, which, in turn, have superior rendering possibilities. Unfortunately, the DXF format, for instance, is based on meshes, therefore, for instance, if one produces a graphic containing both curves and surfaces, the curves will be "lost in translation" when the graphic is exported as a DXF file. The situation would be different, of course, if one would be able to choose to represent the curves as tubes, rather then solids. Another problem is related to the flexibility. Usually, the softwares for graphing curves and surfaces come equipped with a given

1998 *CR Categories and Descriptors.* I.3.5[**Computational Geometry and Object Modeling**]: Subtopic – *Curve, surface, solid, and object representations* .

Key words and phrases. tubular surfaces, canal surfaces.

set of thickness for the curves and it is not possible to prescribe the thickness of the curve at will. Moreover, if the space curves are represented as tubes, this helps also the intuition, it is possible to use different effects (shadow, transparency, etc.). It would be possible even to use the tubular surface for graphing the wireframe of a surface. There is, of course, a price that we have to pay: the speed. It is, clearly, more expensive to represent a curve as a tubular surfaces than to representing using solids. However, this problem can be partially solved using spline approximations. Moreover, this is really a problem only for the interactive visualization and it should be always possible to use the "tube approach" only when the "still" graphics is produces.

The aim of this paper is to discuss propose new approaches to the the tubular and canal surfaces (which, as we shall see, are natural generalization of tubular surfaces), providing explicit representation of them as *swept surfaces* as well as parameterization of the surfaces obtained by their transformation by inversion. Finally, we suggest some possible application of tubular surfaces in scientific vizualisation (for differential geometry and topology).

We mention that all the figures from the paper have been realized by using the C++ graphical library Open Geometry, based on OpenGL, elaborated at the Technical University of Wien by G. Glaeser and H. Stachel (see [8] for a recent survey).

2. Tubular surfaces

We shall give here the mathematical description of tubular surfaces associated to space curves. For all the geometrical notions, see [3].

Definition 2.1. Let $\mathbf{r} : I \to \mathbb{R}^3$ be a smooth, regular space curve. The *tubular surface* associated to $\mathbf{r}$, of radius a, is, by definition, the envelope of the family of spheres of radius a, with the center on the curve (see figure 2).

Remark 2.1. Clearly, if $\mathbf{r}$ is a straight line, then the tubular surface of radius a associated to it is just the circular cylinder of radius a, having $\mathbf{r}$ as symmetry axis. If, on the other hand, $\mathbf{r}$ is a circle, then the corresponding tubular surface is a *torus*.

We shall assume, hereafter that $\mathbf{r}$ is *biregular*, in other words, along the curve the following condition is fulfilled: $\mathbf{r}'(t) \times \mathbf{r}''(t) \neq 0$. We notice that this condition do not hold for straight lines. Then, to each point of the curve, we can associate the *Frenet frame*, i.e. the frame $\{\boldsymbol{\tau}, \boldsymbol{\nu}, \boldsymbol{\beta}\}$ formed by the following three vectors (see [3]):

(i) $\boldsymbol{\tau} = \dfrac{\mathbf{r}'}{\|\mathbf{r}'\|}$ – the unit tangent vector;

(ii) $\boldsymbol{\beta} = \dfrac{\mathbf{r}' \times \mathbf{r}''}{\|\mathbf{r}' \times \mathbf{r}''\|}$ – the unit binormal vector;

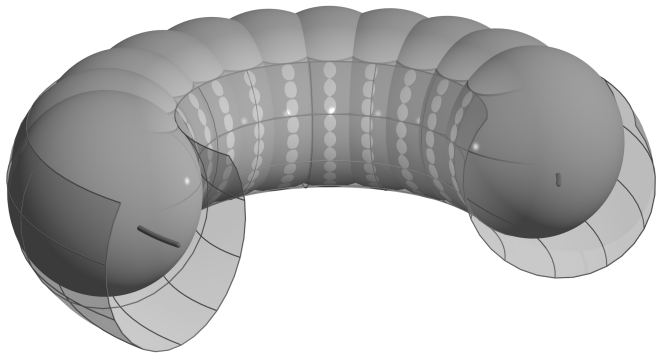


FIGURE 1. Tubular surfaces as envelopes of spheres

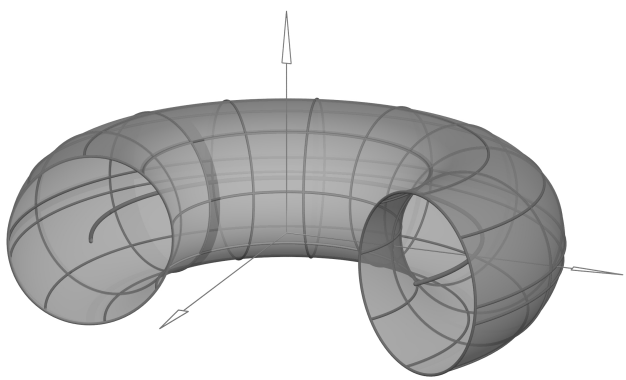


FIGURE 2. The torus as a tubular surface

(iii) $\boldsymbol{\nu} = \boldsymbol{\beta} \times \boldsymbol{\tau}$ – the unit principal normal vector.

Now, it is easy to see that the contact between the spheres from the family and the tubular surface is a great circle of the sphere, lying in the *normal plane* of the generating curve. Let us describe, then, a very simple method of parameterization of the tubular surface. Take the parameter along the generating curve to be one of the parameters and denote by $\mathbf{R}$ the position vector of a point on the surface. As we mentioned earlier, this point lies in the normal plan to the generating curve at a point t. On the other hand, it lies on a circle of radius a, situated in this plan, with the center at the point $\mathbf{r}(t)$ from the curve. Let us denote by $\boldsymbol{\rho}$ the vector connecting the point from the curve with the point from the surface. Then, clearly, we have

$$\mathbf{R} = \mathbf{r}(t) + \boldsymbol{\rho}. \tag{1}$$

The vector $\boldsymbol{\rho}$ itself lies in the normal plane. Let us denote by θ the angle between the vectors $\boldsymbol{\rho}$ and $\boldsymbol{\nu}$. Then, as one can see immediately, we have

$$\boldsymbol{\rho} = a(\boldsymbol{\nu}(t)\cos\theta + \boldsymbol{\beta}(t)\sin\theta). \tag{2}$$

Combining (1) and (2), we see that we obtained a parameterization of the tubular surface,

$$\mathbf{R}(t,\theta) = \mathbf{r}(t) + a\boldsymbol{\nu}(t)\cos\theta + a\boldsymbol{\beta}(t)\sin\theta. \tag{3}$$

Implementation issues. When using a computer algebra system, usually we can compute, formally, the derivatives of functions. However, generally, these facilities are not available. Apparently, we are left with the problem of evaluating numerically the derivatives in order to construct the Frenet frame. However, in practice, we usually do not follow this method. The solution is quite simple, if we use the geometrical interpretations of the tangent line and the osculating plane (the plane containing the tangent versor and the principal normal versor). Namely (see [3])

(1) the tangent line at a point of a curve is the limit position of the straight line determined by the given point and a neighboring point of the curve, when this one approaches the given point;
(2) the osculating plane at a point of the curve is the limit position of a plane determined by the given point and two neighboring points of the curve, when these ones approaches the given point.

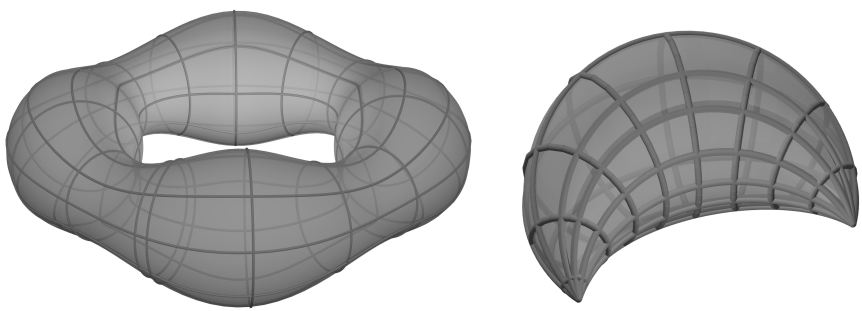
Thus, the algorithm for finding the Frenet at a point of the curve goes like that:

(1) Choose another point of the curve, which is close enough to the given one, by varying a little bit the parameter. These two points determine a straight line, which, within a precision limit which should be prescribed by the user, approximates the tangent line. We take then two points of the line. They determine a vector which, after normalization, can be considered as an approximation of $\boldsymbol{\tau}$ (the tangent versor)
(2) Choose yet another point of the curve, beside the one already chosen at the previous step, again, close enough to the initial point. The three points will determine a plane, which is an approximation of the osculating plane. The unit normal vector of this plane will be, the, an approximation of the binormal vector, $\boldsymbol{\beta}$.
(3) $\boldsymbol{\nu}$ (or, rather, its approximation) will be obtained by taking the cross product of the vectors constructed previously.

3. Canal surfaces

The tubular surfaces are particular cases of more general surfaces, called *canal surfaces*, which are envelopes of family of spheres, of *variable* radius, with the center on a given curve. It is very easy to see that they have a parameterization

of the form (3), only that now a is not any longer a constant. For instance, in the figure 3(a) we represented the canal surfaces obtained as the envelope of the spheres of radius $4 + \sin(2t)$, with the centers on the circle $\mathbf{r}(t) = (10\cos t, 10\sin t, 0)$, while in the figure 3(b) we represented a member of a family of surfaces which are very important for computer graphics, a *Dupin cyclide*. The canal surfaces (and, in particular, the tubular surfaces) have the property that they have a family of curvature lines which are circles (the contacts between the surfaces and the generating spheres). The Dupin cyclides have the property that *all* the curvature lines are circles and it is exactly this the reason why they are so useful in computer graphics, for blending more complex surfaces (see, for instance, the review paper by Boehm ([4]).



(a) A canal surface on a circle (b) A Dupin cyclide

FIGURE 3. Canal surfaces

4. Canal surfaces as swept surfaces

Swept surfaces (see, for instance, [10]) are, roughly speaking, surfaces obtained by "sweeping" a curve along another curve. More specifically, a swept surface is a surface obtained in the following way. We denote by $\mathbf{r}(t)$ the trajectory curve and by $\mathbf{C}(\theta)$ the section curve. Then a swept surface determined by the two curves is a surface of the form

$$\mathbf{S}(t,\theta) = \mathbf{r}(t) + M(t)\cdot\mathbf{C}(\theta), \tag{4}$$

where $M(t)$ is a 3×3 matrix, describing the transformations applied to the curve $\mathbf{C}$ along the trajectory. More precisely, M describes the rotations and the scalings applied to the section curve. In practice, usually both $\mathbf{T}$ and $\mathbf{C}$ are given as NURBS curves and the intention is to represent the surface (itself as a NURBS surface. Usually, for an arbitrary transformation matrix, this might not be possible, therefore we have to use approximations. In fact, only in the very particular case of *translation surfaces*, when the matrix M is the identity we have an exact

NURBS representation for the surface, if the two generating curves are given as NURBS curves. However, this case is not very interesting, as one can see immediately that the only translation surface that is also a tubular surface is the circular cylinder and, generally speaking, the translation surfaces which are canal surfaces are surfaces of revolution.

We shall deduce now the precise form of the equation of a tubular surface as a swept surface, in the sense that we shall indicate the form of the matrix M for the case of tubular surfaces. First of all, the curve $\mathcal{C}$ is a circle of radius a, which we assume to be situated in the xOy plane, in other words, its equation is of the form

$$\mathcal{C}(\theta) = (a\cos\theta, a\sin\theta, 0). \tag{5}$$

What we have to do now is to rotate the curve $\mathbf{C}$ in such a way that, after rotation, it will lye in the normal plane of the curve $\mathbf{r}$. Thus, for any t, the transformation matrix M should turn the xOy plane into the normal plane of the curve at the point $\mathbf{r}(t)$. The idea, of course, is to find a three-dimensional rotation that will turn the axes of the coordinate system (translated at the point of the curve!) into the axes of the Frenet frame, in such a way that the z-axis correspond to the tangent of $\mathbf{r}$, the x-axis – to the principal normal and the y-axis – to the binormal. But then (see [7]), we know that the columns of the rotation matrix should be nothing but the versors of the new direction. Thus, in this case, we have

$$M(t) = \begin{bmatrix} \boldsymbol{\nu}(t) & \boldsymbol{\beta}(t) & \boldsymbol{\tau}(t) \end{bmatrix}. \tag{6}$$

More generally, if we consider an arbitrary canal surface, instead of a tubular one, then an uniform scaling is, also, involved and then the transformation matrix will be the product of matrices:

(7)

$$M(t) = [\boldsymbol{\nu}(t)\ \boldsymbol{\beta}(t)\ \boldsymbol{\tau}(t)] \cdot \begin{bmatrix} f(t) & 0 & 0 \\ 0 & f(t) & 0 \\ 0 & 0 & f(t) \end{bmatrix} \equiv [f(t)\cdot\boldsymbol{\nu}(t)\ f(t)\cdot\boldsymbol{\beta}(t)\ f(t)\cdot\boldsymbol{\tau}(t)],$$

where $f(t)$ is the scaling factor.

Remark 4.1. It is very easy to check that, indeed, the equation (4), with the transformation matrix (6) and the sweeping curve (5) coincide with the equation of the tubular surface established earlier, namely the equation (3).

As we mentioned earlier, generally speaking, the swept surfaces are not, as such, NURBS surfaces, even if the generating curves are NURBS curves. The problem is due to the, generally complicated, structure of the transformation matrix (in particular, that of the rotation matrix). This claim is true, in particular, also for canal surfaces. There are several ways to approximate swept surfaces by NURBS. They are described, in details, in [10].

5. The inversion of tubular surfaces and canal surfaces

A geometrical transformation that it was not exploited yet properly in computer graphics is the *inversion*. We recall, (see, for instance [2]) that the inversion (in space), is a geometrical transformation, defined by a point P (called the *center of the inversion*) and a real number k,(different from zero), called the *power of the inversion*. The inversion $\mathcal{I}$ is defined on $\mathbb{R} \setminus P$, with values in the entire space, such that, for each point $M \neq P$, $\mathcal{I}(P)$ is a point that lies on the straight line PM and the following relation is fulfilled:

$$\overline{P\mathcal{I}(M)} \cdot \overline{PM} = k, \tag{8}$$

where with an overline we denoted the signed length of the segments. If k is positive, M and $\mathcal{I}(M)$ are on the same halfline with respect to P and the inversion is called *positive*. Otherwise, they are on opposite sides and the inversion is called *negative*. The sphere with center at P and with radius $\sqrt{|k|}$ is called the *inversion sphere* or *inverting sphere*.

We are going to find now explicitly the equation of the inverse of a canal surface. Let P be the center of inversion and k – its power. We assume that the radius vector of the center is $\mathbf{r}_0$ and M an arbitrary point in space, different from M, with the position vector $\mathbf{r}$. We denote by M' its inverse and by $\mathbf{r}'$ the radius vector of the inverse. We intend to find this radius vector, in terms of $\mathbf{r}$ and $\mathbf{r}_0$. First of all, as the points P, M and M' are colinear, the relation (8) is equivalent to

$$\overrightarrow{PM} \cdot \overrightarrow{PM'} = k \tag{9}$$

or, which is the same,

$$(\mathbf{r} - \mathbf{r}_0) \cdot (\mathbf{r}' - \mathbf{r}_0) = k. \tag{10}$$

On the other hand, as the three points are colinear, we also have

$$\mathbf{r}' - \mathbf{r}_0 = \lambda(\mathbf{r} - \mathbf{r}_0), \tag{11}$$

where λ is a constant to be determined. Combining (10) and (11), we get to the conclusion that

$$\mathbf{r}' = \mathbf{r}_0 + \frac{\mathbf{r} - \mathbf{r}_0}{(\mathbf{r} - \mathbf{r}_0)^2}. \tag{12}$$

Thus, the inverse of a canal surface with respect to a point of radius vector $\mathbf{r}_0$ will have the equation

$$\mathbf{R}'(t,\theta) = \mathbf{r}_0 + \frac{\mathbf{r}(t) - \mathbf{r}_0 + f(t)(\boldsymbol{\nu}(t)\cos\theta + \boldsymbol{\beta}(t)\sin\theta)}{(\mathbf{r}(t) - \mathbf{r}_0 + f(t)(\boldsymbol{\nu}(t)\cos\theta + \boldsymbol{\beta}(t)\sin\theta))^2}, \tag{13}$$

where $f(t)$ is the scaling function. Of course, if f is a constant, we get the equation of the inverse of tubular surface. The transformation by inversion should play a more important role in computer graphics, due to several appealing qualities (see, for instance, ([2]) for a discussion of the properties of inversion):

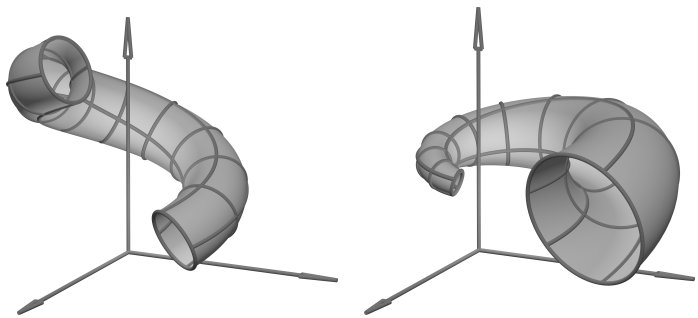


FIGURE 4. A tubular surface and its inverse

- it is a conformal transformation (leaves invariant the angles);
- leaves invariant the curvature lines of surfaces.

Also, a very important property is that the circles that do not pass through the inversion center are also transformed into circles. As a direct consequence of these properties, one can see easily that a torus is transformed into a Dupin cyclide (these are the only surfaces for which both families of curvature lines are circles (as it happens, also, for the torus, which is, also, a particular case of a cyclide)

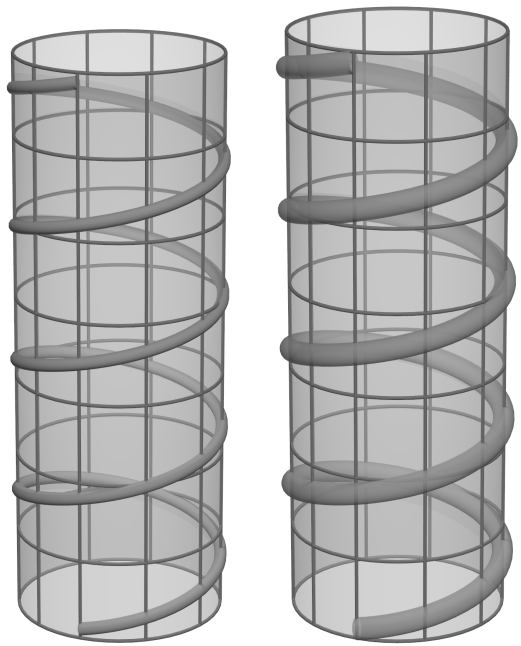
By inverting surface, with a suitable choice of the center of power of inversion, we can produce surfaces with interesting shapes, sometimes we can even imitate natural shapes. In the figure (5) we show a helical surface (obtained, for instance, as the envelope of a family of spheres of radii .5, with the center on a cylindrical helix of parameter .5, lying on cylinder of radius 1.4, having the z-axis as the symmetry axis, as well as the inverse of this surface with respect to the origin, with the power of inversion equal to 2.

6. Applications

We shall illustrate here two applications of the tubular surface for the visualisation of surfaces in differential geometry and topology.

We produced, first of all, two plots of a geodesic on a circular cylinder (see figure 6). Obviously, the method works also in more general circumstances, but we only want to emphasize the utility as such an approach, as opposed to the classical one. In the first graphic we have drawn a thinner tube, without transparency, in the second we took a bigger radius an also used transparency also for the tube, not only for the cylinder.

As a second example, we mention the illustration of the classification theorem for compact, orientable surfaces in $\mathbb{R}^3$. As it is known (see, for instance, [1]), any compact, orientable surface of genus g is homeomorphic to a sphere with g handles (see figure 6), or , which is the same, with a torus with g holes (see figure 6). The



(a) Thinner, no transparency

(b) Thicker, with transparency

FIGURE 5. A geodesic on the cylinder

images were constructed very easily, just as unions of a sphere and three tori, respectively as a union of three tori.

REFERENCES

[1] Agoston, M.: *Algebraic Topology: A First Course*, Marcel Dekker, 1976
[2] Audin, M.: *Geometry*, Springer, 2003
[3] Blaga, P.A.: *Lectures on Classical Differential Geometry*, Risoprint, Cluj-Napoca, 2005
[4] Boehm, W.: *On Cyclides in Geometric Modelling*, Computer Aided Geometric Design, **7** (1990), 243-255.
[5] Boehm, W., Prautsch, H.: *Geometric Concepts for Geometric Design*, A.K. Peters, 1994
[6] Bornik, A., Reitinger, B., Beichel, R.:*Simplex-Mesh based Surface Reconstruction and Representation of Tubular Structures*, in Proceedings of BVM2005, Springer, 2005
[7] Foley, J.D., van Dam, A., Feiner, S.K., Hughes, J.F.: *Computer Graphics, Principles and Practice*, second edition, Addison-Wesley, 1990
[8] Glaeser, G., Schröcker, H.-P.: *Handbook of Geometric Programming Using Open Geometry GL*, Springer, 2001

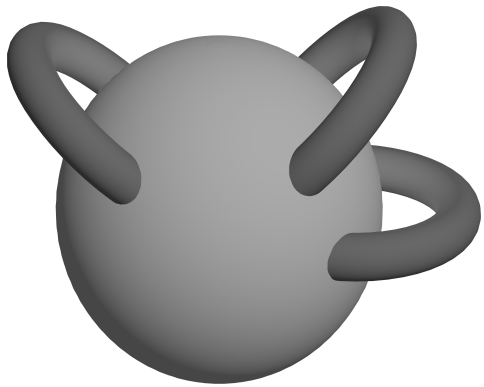


FIGURE 6. A sphere with three handles

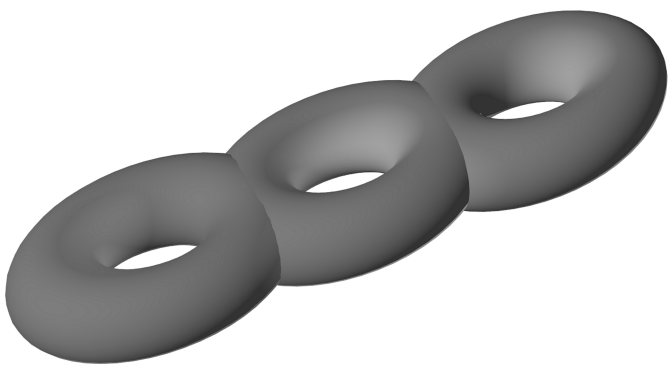


FIGURE 7. A torus with three holes

[9] Landsmann, G., Schicho, J., Winkler, F: *The Parametrization of Canal Surfaces and the Decomposition of Polynomials into a Sum of Two Squares* J. Symb. Comput. **32**(1/2)(2001), 119-132
[10] Piegl, L., Tiller, W.:*The NURBS Book*, second editionn, Springer, 1997
[11] Schicho, J.: *Proper Parametrization of Real Tubular Surfaces* J. Symb. Comput. **30**(5) (2000), 583-593 (2000)

"BABEŞ-BOLYAI" UNIVERSITY, FACULTY OF MATHEMATICS AND COMPUTER SCIENCE, 1, KOGĂLNICEANU STREET, CLUJ-NAPOCA, ROMANIA
E-mail address: pablaga@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume L, Number 2, 2005

A NEW APPROACH IN FRAGMENTATION OF DISTRIBUTED OBJECT ORIENTED DATABASES USING CLUSTERING TECHNIQUES

ADRIAN SERGIU DARABANT

Abstract. Horizontal fragmentation plays an important role in the design phase of Distributed Databases. Complex class relationships: associations, aggregations and complex methods, require fragmentation algorithms to take into account the new problem dimensions induced by these features of the object oriented models. We propose in this paper a new method for horizontal partitioning of classes with complex attributes and methods, using AI clustering techniques. We provide quality and performance evaluations using a partition evaluator function and we prove that fragmentation methods handling complex interclass links produce better results than those ignoring these aspects.

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 as well data partitioning and allocation to the nodes of the system. Horizontal fragmentation, in Object Oriented Database Systems, distributes class instances into fragments. Each object 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. Horizontal fragmentation is usually subdivided in primary and derived fragmentation.

Many of the existing Object Oriented (OO) fragmentation approaches are usually inspired from the relational fragmentation techniques. 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. The OO model is inherently more complex than the relational model. Inheritance, polymorphism, class aggregation and association all induce complex relations between classes in an object oriented database. In order to cope with the increased complexity of the OO model, one can divide class features as follows: *simple attributes* – attributes with scalar types; *complex attributes* – attributes with complex types (other classes), sets, bags, etc. as their

Received by the editors: December 5, 2005.

domain; *simple methods* – methods accessing only local class simple attributes; *complex methods* - methods that return or refer instances of other classes.

In this paper we approach the horizontal fragmentation problem of classes with complex attributes and methods. We rely on AI clustering as an alternative to the current state of the art fragmentation techniques derived from the relational approaches.

1.1. **Related Work.** Fragmentation methods for OODB environments, or flat data models have been generally considered in Karlapalem [1], [4], [5], Ezeife [2]. 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. She is using the algorithm developed for horizontal fragmentation in relational data models. Bellatreche et al. [9] propose a method that emphasizes the role of queries in the horizontal fragmentation.

We have already discussed an alternative AI clustering fragmentation method for OO models with simple attributes and simple methods in [12].

1.2. **Contributions.** We propose a new technique for horizontal fragmentation in object-oriented databases with complex attributes and methods. Fragmentation in complex OO hierarchies is usually performed in two steps: primary fragmentation and derived fragmentation. Primary fragmentation groups class instances according to a set of class conditions [12] imposed on their simple attributes. Derived fragmentation takes into account the class relationships (aggregation, association, complex methods). It groups instances of a class in fragments according to the fragmentation of the related classes. There are generally two approaches in derived fragmentation: *left order derived fragmentation* (*parent first*) and *right order derived fragmentation* (*child first*). They differ in the order in which two related classes are fragmented. In the left order derived fragmentation, the referring class is fragmented first and determines a partitioning of the instance set of the referred class. In the right order derived fragmentation, the referred class is fragmented first and determines the partitioning of the instances of the referring class.

We propose an algorithm that unifies the two fragmentation steps: *primary* and *derived* into a single step. Both class conditions and class relationships are modeled together in a vector space. Each object is represented as a vector and we use the k-means clustering algorithm for separating clusters (fragments) of objects.

The paper is organized as follows. The next section of this work presents the object data model and the constructs used in defining the object database and expressing queries. Section 3 introduces the vector space model we use to compare objects, methods for constructing the object characteristic vectors and similarity metrics over this vector space. Section 4 presents our fragmentation algorithm. In section 5 we present a complete fragmentation example over a class hierarchy

and we evaluate the quality of our fragmentation scheme by using a variant of the Partition Evaluator [12].

2. Data Model

We use an object-oriented model with the basic features described in the literature [8], [11]. Object-oriented databases (OODB) represent data entities as objects supporting features like inheritance, encapsulation, polymorphism, etc. 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 object identifier (OID). Each class can be seen in turn as a class object. Class objects are grouped together in metaclasses. This allows us to consider classes as being instances of higher-level classes that describe the database schema. This way the database schema is self-describing.

Classes are organized in an inheritance hierarchy, in which a subclass is a specialization of its superclass. Although we deal here, for simplicity, only with simple inheritance, moving to multiple inheritance would not affect the fragmentation algorithm in any way, as long as the inheritance conflicts are dealt with into the data model. 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. Thus, in our model, the inheritance graph is a tree.

An *entry point* into a database is a metaclass 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 $C_1.A_1. \ldots A_n$, n$\geq$1 where: C_1 is an entry point in H, A_1 is an attribute of class C_1, A_i is an attribute of class C_i in H such that C_i is the domain of attribute A_{i-1} of class C_{i1} (1$\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).

As presented in [12], a *query* is a tuple with the following structure q=(Target class, Range source, Qualification clause), where:

- *Target class* – (query operand) specifies the root of the class hierarchy over which the query returns its object instances;
- *Range source* – a path expression starting from an entry point and specifying the source class hierarchy;
- *Qualification clause* – logical expression over the class attributes and/or class methods, in conjunctive normal form. The logical expression is constructed using atomic predicates: *path_expression* θ *value* where $\theta \in \{<, >, \leq, \geq, =, \neq, \in, \supset, \supseteq\}$.

3. Vector Space Modeling

3.1. **Primary Fragmentation Modeling.** We denote by $Q = \{q_1, \ldots, q_t\}$ the set of all queries in respect to which we want to perform the fragmentation. Let $Pred = \{p_1, \ldots, p_q\}$ be the set of all atomic predicates Q is defined on. Let $Pred(C) = \{p \in Pred \mid p$ imposes a condition to an attribute of class C or to an attribute of its parent$\}$. Given the predicate $p \equiv C_1.A_1. \ldots A_n\ \theta\ value$, $p \in Pred(C_n)$, if class C_i is the complex domain of A_{i-1}, $i = 2..n$, and A_n has a complex type or simple type.

Given *two* classes C and C', where C' is subclass of C, $Pred(C') \supseteq Pred(C)$. Thus the set of predicates for class C' comprises all the predicates directly imposed on attributes of C' and the predicates defined on attributes of its parent class C and inherited from it [12].

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 $Inst(C) = \{O_1, \ldots O_m\}$ is the set of all instances of class C, $Pred(C) = \{p_1, \ldots, p_n\}$:

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

$$w_{ij} = \frac{1}{m} \sum_{l=\overline{1..m}, a_{lj}=a_{ij}} [(a_{lj} | a_{lj} = 1) + (1 - a_{lj} | a_{lj} = 0)]$$

Each line i in $OCM(C)$ is the object-condition vector of O_i, $O_i \in Inst(C)$. We obtain from $OCM(C)$ the characteristic vectors for all instances of C. The characteristic vector for object O_i is $w_i = (w_{i1}, w_{i2}, \ldots, w_{in})$, 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 $CVM(C)$ [12].

3.2. **Attribute Induced Derived Fragmentation Modeling.** We have captured so far all characteristics of simple attributes and methods. We need to express 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. In the case of left derived fragmentation C_O will be fragmented first, followed by C_M. In the right derived fragmentation variant the order in which the two classes are fragmented is reversed. Each of the two strategies is suitable for different query evaluation strategies. For example, in reverse traversal query evaluation strategy, the right derived fragmentation variant gives the best results. We assume here, for space reasons, that right derived

fragmentation method is used. However, both: the algorithm and the vector space model remain the same when considering left derived fragmentation order.

Thus, in right derived fragmentation method, when fragmenting C_O we should take in account the fragmentation of C_M [13]. We want to place in the same fragment of C_O objects aggregating instances from a fragment of C_M. Objects of a fragment of C_O should aggregate as much as possible objects from the same fragment of C_M.

Let $\{\mathrm{F}_1, \ldots \mathrm{F}_n\}$ be the fragments of C_M. We denote by $Agg(O_i, F_j)$=$\{O^m \mid O^m \in F_j, O_i$ *references* $O^m\}$. Given the set of fragments for C_M, we define the *attribute-link induced object-condition vectors for derived fragmentation* as ad_i = $(ad_{i1}, ad_{i2}, \ldots, ad_{in})$, where each vector component is expressed by the following formula:

$$ad_{ij} = sgn(|Agg(O_i, F_j)|), \quad j = \overline{1,n} \tag{2}$$

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, we define the *attribute-link induced characteristic vectors for derived fragmentation* as $wad_i = (wad_{i1}, wad_{i2}, \ldots, wad_{in})$, where each vector component is expressed by the following formula:

$$wad_{ij} = \frac{|\{O_l \in Inst(C_O) | sgn(|Agg(O_l, F_j)|) = sgn(|Agg(O_i, F_j)|)\}|}{|Inst(C_O)|}, j = \overline{1,n} \tag{3}$$

Each wad_{ij} component gives the percentage of objects in C_O that aggregate/refer in the same way as O_i objects from F_j. Two objects O_i and O_l are said to aggregate F_j *in the same way* if they are both either linked or not linked with objects from F_j. According to this criterion, two objects are candidates 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.

3.3. **Method Induced Derived Fragmentation Modeling.** In the following paragraphs we model the class relationships induced by the presence of complex methods. Given a class with complex methods C(owner) that has to be fragmented, we need to take in account, when fragmenting it, the fragmentation of classes referred by its complex methods. In order to model the method reference dependencies in the fragmentation process we need to express this type of relationships in our vector space.

We denote by $MetComplex(C)$=$\{m_i \mid m_i$ *complex method of* $C\}$ – the set of all complex methods of class C.

Let $SetCRef(m, C) = \{C_R | C \neq C_R, C_R$ is referred by method $m \in MetComplex(C)\}$ be the set of classes referred by the complex method

m of class C. For a given instance of a class C with complex methods we denote as:
$SetORef(m, O_i, C_R)$=$\{O_r^{'} \in Inst(C_R) \mid C_R \in SetCRef(m, C), m \in MetComplex(C), O_r^{'}$ *is referred by method* $m\}$ – the set of instances of class C_R,referred by the complex method m of class C.

For each pair $(m_k, C_R) \in \{m_k \in MetComplex(C)\} \times SetCRef(m_k, C)$ we quantify the way each instance of C refers - through complex methods - instances from fragments of C_R. Given a class C_R referred by a complex method m_k of class C, and the fragments of class $C_R \rightarrow \{F_1, \dots F_n\}$, we define the *method-link induced object-condition vectors for derived fragmentation.* For each instance O_i of C let md_i= *(*md_{i1}, md_{i2}, *...,* md_{in}) be the *method-link induced object-condition vector.* Each vector component is defined by the following formula:

$$md_{ij} = sgn(|\{O_l \in Inst(C_R) | O_l \in F_j \cap SetORef(m_k, O_i, C_R)\}|), j = \overline{1,n} \tag{4}$$

Each md_{ij} evaluates to 1 when object O_i∈*Inst(C)* refers objects from fragment F_j of class C_R and 0 otherwise. For each object O_i we obtain, for each pair (m_k, C_R), one *method-link induced object-condition vector.* We derive from it the *method-link induced characteristic vector for derived fragmentation,* wmd_j =(wm_{i1}, wmd_{i2},*...*,wmd_{in}), where:

$$wmd_{ij} = \frac{|\{O_l \in Inst(C) | md_{lj} = md_{ij}\}|}{Inst(C)}, j = \overline{1,n}, l = \overline{1, |Inst(C)|} \tag{5}$$

Each wmd_{ij} quantifies the way objects of class C refer objects from fragments of C_j through complex methods.

When modeling relationships induced by the presence of complex methods, we obtain as many referring condition vectors (object-condition and characteristic), for each instance O_i of C, as the number of elements of the Cartesian product $\{m_k \in MetComplex(C)\} \times SetCRef(m_k, C)$.

3.4. **Derived Fragmentation Modeling.** As the number of elements in $\{m_k \in MetComplex(C)\} \times SetCRef(m_k, C)$ is usually large we need to use some heuristics in order to retain only the pairs with significant impact in the fragmentation. In order for a pair (m_k, C_R) to be kept it should satisfy the following combined restrictions:

- The number of calls to the method m_k should be significant compared to the contribution brought by all method calls made by applications running on the database;
- The number of instances of C_R referred by the method m_k should be significant compared to the number of instances of all classes generally referred by the applications.

The above conditions are expressed in the following formula (*significance factor*):

$$(6) \qquad Sig(m_k, C_R) = \frac{NrCalls(m_k)}{\sum_{m_l \in MetComplex(C)} NrCalls(m_l)} \times \frac{\sum_{O_i \in Inst(C)} |SetORef(m_k, O_i, C_R)|}{\sum_{C_p \in SetCRef(m_k)} \sum_{O_r \in Inst(C)} SetORef(m_k, O_r, C_p)}$$

In (6) the first factor gives the ratio between the number of calls to method m_k and the number of calls of all complex methods of class C. The second factor gives the ratio between the number of C_R instances referred by m_k and the number of all objects referred by m_k. In reality the actual method parameters would normally influence the set of objects referred by the method. Even more, the set of referred objects could be as well influenced by the internal state of the object. However, tracking all the possible combinations is computationally intractable – even in simple situations. The statistical heuristic proposed in (6) is still manageable and helps reducing the problem space dimensions.

Usually, the fragmentation of a class C is performed in two steps: primary fragmentation, according to query conditions, and derived fragmentation, according to the fragments of the member or owner classes. We merge the two phases into one single step capturing the semantic of both primary and derived fragmentations. For this we unify the characteristic vector, the attribute-link and method-link induced characteristic vectors for each object O_i of the class C, and we obtain the *extended characteristic vector*. Each extended characteristic vector quantifies all the information needed for fragmentation: the conditions imposed on the object and the relationships of the object with instances of related classes, induced either by complex attributes or by complex methods.

If the class C is related with classes $C_{A1}, C_{A2}, \ldots, C_{Ap}$ by means of complex attributes, and with classes $C_{M1}, C_{M2}, \ldots, C_{Mr}$ by means of complex methods, the *extended characteristic vector* we_i for object $O_i \in Inst(C)$ is obtained by appending the p attribute-link induced characteristic vectors and the number of $mc = |\{m_k \in MetComplex(C)\} \times SetCRef(m_k, C)|$ method-link characteristic vectors to the characteristic vector of O_i. However, as we have already mentioned above, we are using the *significance factor* to filter out non-relevant pairs (m_k, C_R) and vectors derived from them. As observed experimentally, a significance factor around 0.27 will filter out most of the inappropriate (m_k, C_R) pairs.

The *extended object-condition vector* ae_i for an object O_i is obtained in the same way by appending its attribute-link and method-link induced object-condition vectors to its object-condition vector. We denote by $EOCM(C)$ and $ECVM(C)$ the extended object-condition and characteristic matrices for class C.

3.5. **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 Euclidian and Manhattan metrics:

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

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

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

We use sim_E and sim_M in (8) to measure how similar two objects are. Both measures take values in [0,1] and are expressed using one of the two distances from (7). The distance functions and the similarity measures are inversely proportional in [0,1]. As the distance between two objects increases, their similarity decreases. We should note that all characteristic vectors have positive coordinates by definition.

4. K-means Centroid-based Fragmentation

We apply an algorithm we have used to fragment classes with simple attributes and methods: the k-means centroid based clustering algorithm [12]. The classical k-means algorithm takes the input parameter k and partitions a set of m objects into k clusters so that the resulting intra-cluster similarity is high but the inter-cluster similarity is low. Cluster similarity is measured in regard to the *mean* value of the objects in a cluster, which can be viewed as the cluster's *center of gravity* (*centroid*). First, the k-means algorithm randomly selects k of the objects, each of which initially represents a cluster mean or center. For each of the remaining objects, an object is assigned to the cluster to which is the most similar, based on the distance between the object and the cluster centroid. It then computes the new centroid for each cluster and redistributes all objects according to the new centroids. This process iterates until a criterion function converges. The criterion tries to make the resulting k clusters as compact and separate as possible.

Our version of the algorithm improves several aspects of the original algorithm with regard to the semantic of object fragmentation. First of all, we implement a variant where we choose as initial centroids the most representative objects in respect with fragmentation predicates, rather than choosing them arbitrarily. At each iteration, if an object should be placed in any of several clusters (same similarity with the centroid), we choose the cluster to which the object has maximum similarity with. We also choose as criterion function the degree of compactness/homogeneity H of all clusters. For a given cluster F, this value is the

difference between the maximum and minimum similarity of all pairs of objects in F:

$$H(F) = \max\{sim(a,b) \in F \times F, a \neq b\} - \min\{sim(a,b) \in F \times F, a \neq b\} \quad (9)$$

```
Algorithm k-meansFrag is:
Input: Class C, Inst(C) to be fragmented, the similarity function
sim : Inst(C) × Inst(C) → [0,1], m = |Inst(C), 1 < k ≤ m desired number of
fragments, OCM(C), CVM(C), threshold_value.
Output: the set of clusters F = {F_1, ..., F_f}, where f ≤ k.
Begin
   Centr={c_1, ..., c_k}=InitCentr(Inst(C), OCM(C), CVM(C), k);
   F = {F_i | F_i = {c_i}, c_i ∈ Centr, i = 1..k}; F' = ∅;
   // initial object allocation to clusters;
   For all objects O_i do
      F_candidates = {argmax_centr(sim(O_i, c_l), l = 1..k)};
      F_u* = argmax_sim(sim(O_i, f_c), f_c ∈ F_candidates); F_u* = F_u* ∪ {O_i};
   End For;
   While F' ≠ F and H(F)<threshold_value do
      For all F_i ∈ F recalculate centroid c_i;
      F' = F;
      For all objects O_i do
         F_candidates = {argmax_centr(sim(O_i, c_l), l = 1..k)}; (i)
         F_u* = argmax_sim(sim(O_i, F_c), F_c ∈ F_candidates); (ii)
         F'_v = F'_v − {O_i}, where O_i ∈ F'_v;
         F'_u* = F'_u* ∪ {O_i};
         F' = F' − {F'_l | F'_l = ∅}; // eliminate empty clusters
      End For;
   End While;
End.

Function InitCentr(Inst(C),OCM(C),CVM(C),k) is
Begin
   Centr=∅; n = |Pred(C)|;
   For i=1 to k do
      c_i = argmin[d_M(OCM(O_j), u_i)], Oj ∉ Centr, i ≤ n; (iii)
      c_i = argmin(sim(O_j, Centr)), O_j ∉ Centr, i > n; (iv)
      Centr = Centr ∪ {c_i};
   End for;
   Return Centr;
End Function;
```

Function *InitCentr* chooses the initial centroids as described above. In line (iii) u_i is the identity vector of degree i, which has 1 only on the i^{th} position and 0 on the other positions. Each u_i represents the corresponding predicate from $Pred(C)$. Line (iii) chooses as centroid the closest object to u_i, i.e. the most representative object for that predicate. We note that we can choose this way as many centroids as the number of predicates in $Pred(C)$. If we need more clusters than $|Pred(C)|$, we choose as their initial centroids the objects most dissimilar to the already chosen centroids (line (iv)). We try this way to minimize the impact of "losing" clusters in the following iterations. This occurs when all objects in a cluster relocate to other clusters because the initial centroid is not semantically representative to our set of predicates.

We use in lines (i) and (ii) the similarity of an object O_i with a cluster F_c, defined as (the average similarity with all objects of the cluster):

$$sim(O_i, F_c) = \frac{\sum_{a \in F_c} sim(O_i, a)}{|F_c|} \tag{10}$$

5. Results and Evaluation

In this section we illustrate the experimental results obtained by applying our fragmentation scheme on a test object database. 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. The problem with the evaluation method is that it is difficult to quantify a fragmentation result without allocating the fragments to the nodes of a distributed system. On the other side, the allocation problem must be solved in order to be able to evaluate the fragmentation. As resolving the allocation problem in the general case is not a trivial task, we need a simplified allocation model, yet a valid one. Our solution is to consider a distributed system running database applications (queries). Some of the nodes are part of the distributed object oriented DBMS as well. They hold fragments of the database and a database engine. All applications run with different frequencies on different nodes of the system. We chose to allocate each fragment on the node where is most needed (accessed).

Another issue that might affect the results is the fact that the order in which classes are fragmented is significant as it captures the semantic of query path expressions into the fragmentation process [13]. It might be possible to obtain better results by using a different order for fragmenting classes. We do not handle here the ordering problem, but we address it in [13].

The sample object database represents a reduced university database. The inheritance hierarchy is given in Figure 1 and a trimmed down version of the aggregation/association graph is shown in Figure 2.

The queries running on the classes of the database are given bellow:

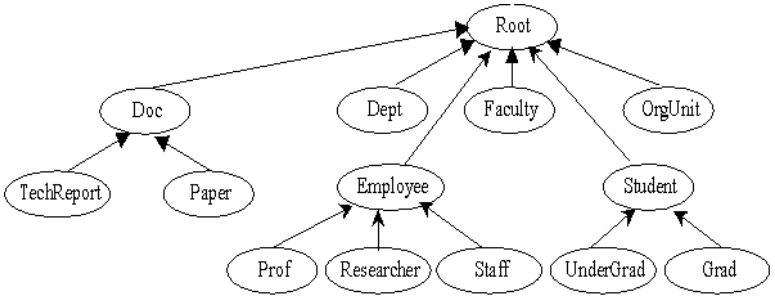


FIGURE 1. The database class hierarchy

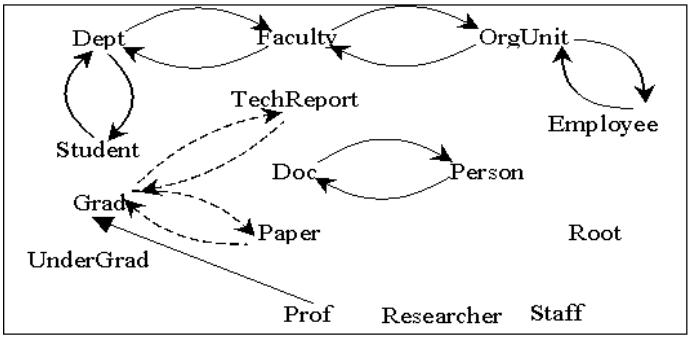


FIGURE 2. The database aggregation/association graph

q_1 =(Grad, Faculty.Dept.Student, Grad.Supervisor.OrgUnit.Name in ("ProgMeth", "InfSyst"))
q_2 =(UnderGrad, Faculty.Dept.Student, UnderGrad.Dept.Name like "CS%" and UnderGrad.Grade between 7 and 10)
q_3 =(UnderGrad, Faculty.Dept.Student, (UnderGrad.Dept.Name like "Math%" or UnderGrad.Dept.Name like "CS%") and UnderGrad.Age()≥24)
q_4 =(Researcher, Doc.Person, Researcher.count(Reasercher.doc) ≥2)
q_5 =(Prof, Faculty.OrgUnit.Employee, Prof.OrgUnit.Name in ("ProgMeth", "InfSyst") and Prof.salary≥40000)
q_6 =(Prof, Doc.Person., Prof.Paper.Publisher in ("IEEE", "ACM") and Prof.Position = "prof")
q_7 =(TechReport, Doc, TechReport.year>1999)
q_8 =(Set(Student.Dept), Person, Student.Grade<5)
q_9 =(Employee, Person, Employee.salary>35000)
q_{10} =(Grad, Person, Grad.count(Grad.Paper)≥1)
q_{11} =(Student, Person,Student.Dept.Name like "CS%")
q_{12} =(Student, Person,Student.Dept.Name like "Math%")

q_{13} =(Staff, Person, Staff.salary>12000)
q_{14} =(Person, Person, Person.Age()>30)

In Figure 2 the links between Doc and Person should be inherited by all subclasses of Person and Doc. This is graphically represented in the figure by the dotted arrows. Similar inherited links are present for other classes in this graph (links not represented here). The motivation for aggregation/association inheritance is presented in [13].

For measuring the fragmentation quality we determine the cost of remote accesses combined with the cost of local irrelevant accesses to each fragment. Remote accesses are made by applications running on a given node and accessing objects that are not stored on that node. Local irrelevant accesses are given by local processing incurred when a query accesses a fragment. Each access to a fragment implies a scan to determine objects that satisfy a condition. Irrelevant local access measure the number of local accesses to objects that will not be returned by the query. Intuitively, we want that each fragment be as compact as possible and contain as much as possible only objects accessed by queries running on the fragment's node. We use the following measure for calculating the fragmentation quality:

$$PE(C) = EM^2 + ER^2 \tag{11}$$

$$EM^2(C) = \sum_{i=1}^{M} \sum_{t=1}^{T} freq_{ts}^2 * |Acc_{it}| * \quad 1 - \frac{|Acc_{it}|}{|F_i|} \tag{12}$$

$$ER^2(C) = \sum_{t=1}^{T} \min \left\{ \sum_{s=1}^{S} \sum_{i=1}^{M} freq_{ts}^2 * |Acc_{it}| * \frac{|Acc_{it}|}{|F_i|} \right\} \tag{13}$$

The EM term calculates the local irrelevant access cost for all fragments of a class. ER calculates the remote relevant access cost for all fragments of a class. Acc_{it} represents the set of objects accessed by query t from fragment F_i. The value $freq_{ts}$ is the frequency of query t running on site s. In (12) s is the site where F_i is located, while in (13) s is any site not containing F_i. M is the number of clusters for class C, T is the number of queries and S is the number of sites [12]. The fragmentation is better when the local irrelevant costs and the remote relevant access costs are smaller. Each term of PE calculates in fact the average square error of these factors. Globally, PE measures how well fragments fit the object sets requested by queries.

Using the given query access frequency, the fragments above are allocated to 4 distributed sites. Query frequency at sites is presented in Table 1.

We qualitatively compare the results of our fragmentation method with a centralized and a full replicated database in Figure 3. The centralized version of the database is allocated to node S1, while in the replicated case each node holds a copy of the entire database. It can be seen that our fragmented database obtains smaller PE costs, with both measures, than the centralized and full replicated database. The full replicated case obtains the worst costs as the irrelevant access cost explodes in this case. Even though remote accesses tend to zero in the replicated case, the local irrelevant accesses increase considerably as each node holds an entire copy of the database, thus many

TABLE 1. Access frequencies of queries at distributed sites

Freq(q,s)	S1	S2	S3	S4
q1	0	10	5	20
q2	0	10	5	25
q3	20	0	15	10
q4	15	10	5	0
q5	25	20	0	20
q6	30	0	20	10
q7	30	25	0	10
q8	10	0	0	10
q9	20	20	10	0
q10	15	25	0	0
q11	5	10	5	0
q12	0	0	0	10
q13	15	0	0	5
q14	20	5	0	0

irrelevant objects for each query. In reality, the full replicated case performs well only when there are no updates to the database. The Manhattan similarity measure applied on *OCM* obtains the best results, followed by Manhattan similarity applied on characteristic vectors and by the Euclid measure at last.

In Figure 4 we present the *PE* costs induced on each fragmented class with each method. Here it can be seen that the Manhattan and Euclid measures behave approximately the same on all classes except *Undergrad*. In our example, classes have been fragmented in the order they appear in Figure 4, from left to right, Undergrad being the last fragmented class. Even if *PE* scores for other classes are approximately the same – the resulting fragments are not identical for different similarity measures.

This leads to the fact that when fragmenting Undergrad, the resulting fragments are influenced by the fragmentation of the related classes. Manhattan applied on *OCM* does the best fragmentation on the intermediate (related) classes, which leads to a better score when the last class (Undergrad) is fragmented.

Finally in Figure 5 we compare the results of the same fragmentation algorithm in two cases: when complex class relationships are considered and when complex class relationships are ignored, i.e primary only fragmentation. The *P-Euclid, P-Manhattan Charact. Vect.* and *P-Manhattan Obj. Conditions* denote the primary only versions of the fragmentation algorithm.

It can be seen that the fact of considering the complex class relationships improves quality. All similarity measures in this case behave better than the best case of the fragmentation without derived fragmentation.

6. Conclusions and Future Work

We proposed in this paper a new horizontal fragmentation method for object oriented distributed databases. Our method takes into account all complex relationships between

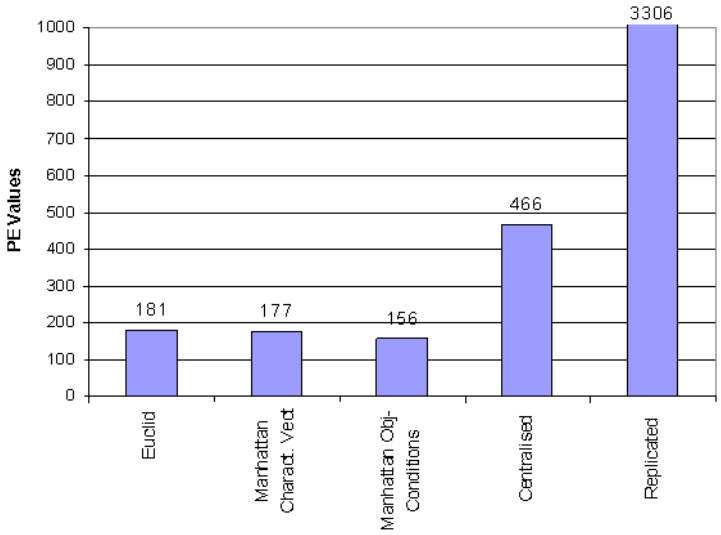


FIGURE 3. Comparative *PE* values for our fragmentation method, centralised and replicated databases

classes: aggregations, associations, and links induced by complex methods. Primary and derived fragmentations are modeled together and are performed in a single step. This reduces the complexity of our technique compared to traditional approaches that perform primary and derived fragmentation in two distinct steps, processing twice the entire database.

We have shown that taking complex relationships into account significantly improves fragmentation quality as opposed to methods considering only primary fragmentation. The order in which classes are fragmented is important as class relationships may induce mutual transitive class dependencies. There is always a fragmentation order that produces better results than the average of all other orders. We proposed an algorithm for determining the fragmentation order in [13].

We aim to find new ways of expressing inter-class relationships and compare their results and impact in the fragmentation process.

REFERENCES

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

[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) (1995) 247-272

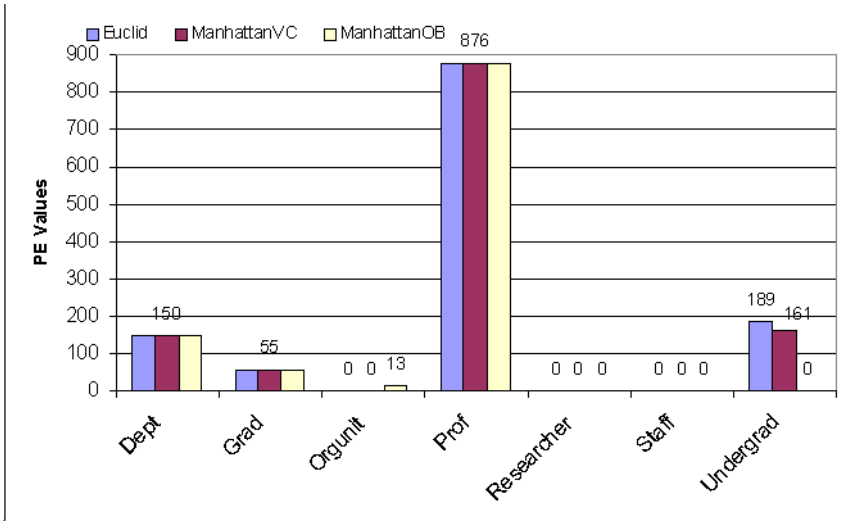


FIGURE 4. Comparative class *PE* values for each similarity measure

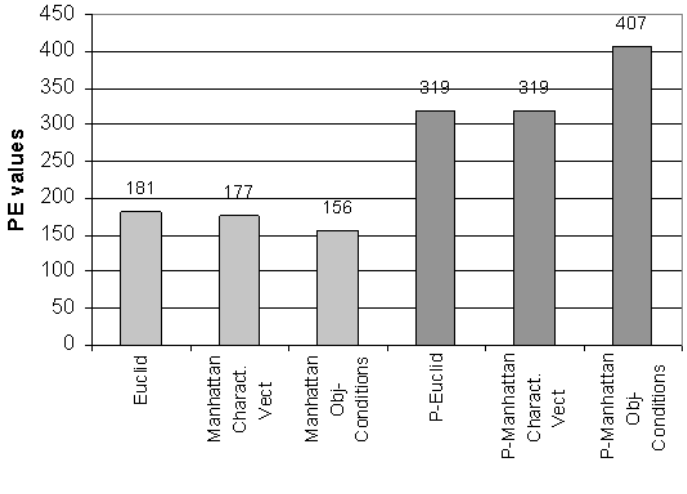


FIGURE 5. Comparative *PE* values for *primary only fragmentation* and our complex fragmentation method (*primary + derived fragmentation*)

[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, Taiwan (1995) 42–49
[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), Hong Kong (1996) 377–384
[6] Ravat, S.: La fragmentation d'un schema conceptuel oriente objet, In Ingenierie des systemes d'informaton (ISI), 4(2) (1996) 161–193
[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 (1994) 25-32
[8] Bertino, E., Martino, L.: Object-Oriented Database Systems; Concepts and Architectures, Addison-Wesley (1993)
[9] Bellatreche, L., Karlapalem, K., Simonet, A.: Horizontal Class Partitioning in Object-Oriented Databases, In Lecture Notes in Computer Science, volume 1308, Toulouse, France (1997) 58–67
[10] 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 (1996) 732-737
[11] Baiao, F., Mattoso, M.: A Mixed Fragmentation Algorithm for Distributed Object Oriented Databases, In Proc. Of the 9th Int. Conf. on Computing Information, Canada (1998) 141-148
[12] Darabant, A. S., Campan, A.: Semi-supervised learning techniques: k-means clustering in OODB Fragmentation, IEEE International Conference on Computational Cybernetics ICCC 2004, Vienna University of Technology, Austria, August 30 - September 1 (2004) 333-338
[13] Darabant, A.S, Campan, A.: Optimal Class Fragmentation Ordering in Object Oriented Databases, In Studia Universitatis Babes Bolyai Informatica, Volume XLIX, Number 1 (2004) 45-54

Babes-Bolyai University, Cluj-Napoca, Romania
E-mail address: dadi@cs.ubbcluj.ro