A Clustering-Driven LDAP Framework

VASSILIKI KOUTSONIKOLA and ATHENA VAKALI, Aristotle University

LDAP directories have proliferated as the appropriate storage framework for various and heterogeneous data sources, operating under a wide range of applications and services. Due to the increased amount and heterogeneity of the LDAP data, there is a requirement for appropriate data organization schemes. The LPAIR & LMERGE (LP-LM) algorithm, presented in this article, is a hierarchical agglomerative structure-based clustering algorithm which can be used for the LDAP directory information tree definition. A thorough study of the algorithm's performance is provided, which designates its efficiency. Moreover, the *Relative Link* as an alternative merging criterion is proposed, since as indicated by the experimentation, it can result in more balanced clusters. Finally, the *LP and LM Query Engine* is presented, which considering the clustering-based LDAP data organization, results in the enhancement of the LDAP server's performance.

Categories and Subject Descriptors: H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Clustering; information filtering*; *query formulation*; H.2.1 [**Database Management**]: Logical Design—*Schema and subschema*

General Terms: Algorithms, Performance

Additional Key Words and Phrases: LDAP services, DIT organization, clustering, merging criteria, query and retrieval engine

ACM Reference Format:
Koutsonikola, V. and Vakali, A. 2011. A clustering-driven LDAP framework. ACM Trans. Web 5, 3, Article 12 (July 2011), 34 pages.
DOI = 10.1145/1993053.1993054 http://doi.acm.org/10.1145/1993053.1993054

1. INTRODUCTION

The explosive growth of Web has increased the need for more robust and scalable distributed networks that are characterized by high performance, high capacity, secure, and reliable services which can be rapidly scaled and managed. The Lightweight Directory Access Protocol (LDAP) [Whal et al. 1997] is an important technology which enables data sharing under an extendable framework for the centralized storage and management of information that needs to be available for today's distributed systems and services. As the name suggests, LDAP is the lightweight version of the Directory Access Protocol and a direct descendent of the heavyweight X.500 [Chadwick 1994], the most common directory management protocol. It is an open industry standard that is gaining wide acceptance due to its flexibility in supporting the storage of heterogeneous data, providing at the same time optimized response times in read operations [Koutsonikola and Vakali 2004].

The original idea for this work was presented by the authors in their work "A structure-based clustering on LDAP directory information" at ISMIS'08.
Authors' address: V. Koutsonikola (corresponding author) and A. Vakali, Department of Informatics, Aristotle University, 54124 Thessaloniki, Greece; email: vkoutson@csd.auth.gr.

DOI 10.1145/1993053.1993054 http://doi.acm.org/10.1145/1993053.1993054

LDAP directories have been used to store a wide variety of information for enabling integration among applications and services on the network. In a typical framework, LDAP servers have been used to act as address books, providing information that describes user profiles and can be used by other applications such as mail services, authentication systems, etc. [Carter 2003; Hou et al. 2006]. Moreover, the usage of directory services was extended by the Directory-Enabled Networking initiative, a design philosophy whose goal is to enable applications to use directories in order to store complex objects describing the network components (such as printers and routers) and network security policies [Howes and Smith 1997; Maass 1997]. LDAP has also become the predominant protocol in support of PKIs accessing directory services for certificates and Certificate Revocation Lists (CRLs) [Chadwick 2003; Park et al. 2001], as well as of the new H.350 standard which uses LDAP to provide a uniform way to store information related to video and voice over IP (VoIP) in directories [Gemmill et al. 2003]. In addition, Grid computing has emerged as a very promising infrastructure for distributed computing, having its foundation and core on the distributed LDAP directories [Fan et al. 2005; Wu et al. 2006].

There is also a trend recorded lately towards proposing methods that integrate LDAP technology and semantic Web. These research activities involve the generation of RDF models from a directory information tree or an LDAP search query by mapping LDAP schema information into OWL ontologies and LDAP objects into RDF instance triples [Dietzold 2005]. Moreover, methods that combine LDAP and SPARQL are used to access RDF knowledge bases [Dietzold and Auer 2007a, 2007b], while an ontology-based method has been proposed for the creation of a yellow page directory service for end-users [Laukkannen et al. 2004]. The common characteristics that XML and LDAP share have motivated many research attempts to integrate these two technologies. To this context Directory Services Markup Language[1] (DSML) is an evolving specification that has been proposed to bridge the gap between directory services and XML-enabled applications, by representing directory information in XML. It is actually an XML version of the LDAP protocol that satisfies the need of interoperability between different LDAP directories vendors' which can be achieved with the XML adoption as the standard for their data exchange. An overview of existing work and tools in this field is presented in Koutsonikola and Vakali [2008]. Furthermore, a new direction towards designing Web services for the LDAP framework as well as using LDAP directories as Web services repository appears [Rodriquez 2007] which will ensure more flexibility in the LDAP communication interoperability.

LDAP's integration with various applications proves that LDAP can be used to store information from various heterogeneous data sources. Data is stored in LDAP directories in the form of LDAP entries which are organized in a hierarchical tree-like structure called the Directory Information Tree or DIT. The structure of DITs may differentiate in terms of the depth of hierarchy they present. For example, if we had to organize the books of a library in a directory service, then we could choose one of the following organization schemes: (i) book entries are organized in a flat tree as depicted in Figure 1, (ii) book entries are stored in a more hierarchical structure according to a basic categorization as presented in Figure 2, (iii) book entries form a three-level hierarchical tree structure on the basis of a further books categorization, as depicted in Figure 3. In the three former figures that depict in an abstract view three possible DIT organization schemes, the leafs of the DIT correspond to the actual book entries, while the tree structure depicts their organization which is characterized by different levels of hierarchy.

[1]DSML: http://xml.coverpages.org/dsml.html.

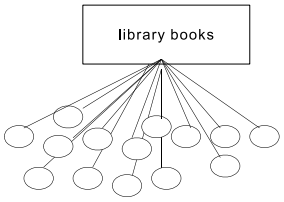


Fig. 1. Flat organization of book entries.

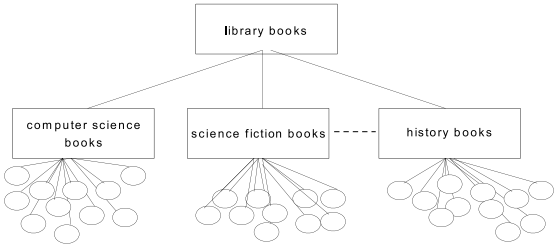


Fig. 2. Two-level hierarchical organization of book entries.

Finding an appropriate organization scheme has always been a challenge for systems' administrators, in their effort to provide better performance and quality of services. We may identify various application scenarios in which organizing LDAP data properly may be beneficial for their performance.

—*Academic environment*. LDAP directories usually serve as address books which store information describing academic staff and students. This information can, moreover, be used by authentication systems, PKI applications, roaming, and profiling services. An appropriate organization scheme of an academic environment's LDAP data could be based on the university, department, or title properties that describe a specific member's entry.
—*Enterprises environment*. In an enterprise framework, LDAP directories can be used to store information about both employees and products. Various applications such as e-commerce applications, authentication systems, Web, and mail services or search interfaces can, then, access the stored data. LDAP entries can be organized in terms of their properties which may refer to the title or company's department for the employees, or their type and special features with respect to products. This information may further be used in enterprises' services offered to their clients or other organizations under various application frameworks such as B2B marketing.
—*Sharing applications*. LDAP directories are used to store the data items that a specific application framework shares. These items may refer to images, audio or video files, movies, or books' entries, etc. The organization in the LDAP directories, employed by a sharing application, may consider the data nature (e.g., book or movie) and data content (social movies or thrillers).
—*Network devices operating framework*. Network components such as routers, hosts, and printers access directory servers to download information about their configuration settings and policies. The stored information, in this case, is organized according to the devices they describe or the networks that they belong to.

The variety of information employed by the preceding applications constitutes a significant challenge in defining appropriate data organization schemes. The purpose of

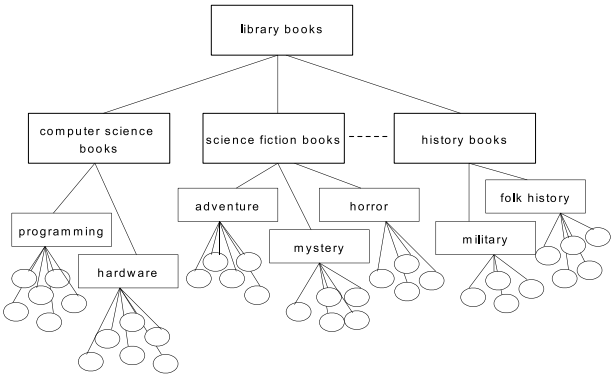


Fig. 3. Three-level hierarchical organization of book entries.

such tasks is to ensure efficient retrieval mechanisms in terms of both results accuracy and response times. Thus, we are looking for an automated LDAP data organization framework that will operate efficiently regardless of the underlying application and will facilitate the design of a query mechanism resulting in improved response times. Clustering analysis is by default a data organization method that is targeted to identify similarities between data items and propose their grouping according to the recorded relations [Vakali et al. 2004; Zeng et al. 2002]. Therefore, here, we address the problem of using clustering analysis to automatically define the LDAP Directory Information Tree when applied to large size datasets. The proposed LDAP data organization can result in the enhancement of LDAP server's performance and the improved quality of services.

1.1 Related Work

Existing research efforts usually employ application-oriented LDAP schema definitions in order to result in efficient storage frameworks as well as enhanced retrieval mechanisms. In these cases, the overall data organization follows the LDAP schema definition which is customized to meet a certain application's needs. Furthermore, query models adjusted to the LDAP schema arrangement are proposed to ensure efficient information retrieval. Such approaches have been proposed in different application frameworks such as PKI [Lim et al. 2005] and Grid computing [Hu and Du 2006], which extend the LDAP schema and query models with new definitions to support the applications' special features. These approaches facilitate mainly retrieval efficiency in terms of its accuracy and not the response time effectiveness.

On the other hand, various pure caching and indexing approaches have been proposed which can improve performance and scalability of directory-based services. The tuning of these two parameters may significantly improve the performance of a system that implements directory services, in terms of its latency and throughput. However, if not appropriately defined, it may result in performance deterioration, since, for example, the definition of many indices can make insertion and modification operations slower, while advanced caching policies are needed in case of dynamic environments. A summarization of approaches that attempt to enhance the quality of services provided by directory server is given in Table I.

Therefore, there are two main factors which contribute to the efficiency of an LDAP-based framework. The first one refers to the LDAP data organization which must be adjusted and meet a specific application's needs, while the second one is related to the overall system's performance. A framework that will propose an automatic,

Table I. Approaches Enhancing LDAP Applications' Performance

Application framework	Query model	Caching	Indexing	Methodologies
Public Key Infrastructure [Lim et al. 2005], Grid Computing [Hu and Du 2006]	✓			LDAP schema and query model extension, addition of new syntaxes and matching rules
Network Service Level Specifications [Wang et al. 2008]		✓	✓	Configuration settings adjustment, indexing and caching schemes
Directory Enabled Applications [Kapitskaia et al. 2000], [Kumar and Gupta 2003], [Amer-Yahia and Srivastava 2004]	✓	✓		Cache design based on query templates
XML Enabled Applications [Maron and Lausen 2001]		✓		Cache design, algorithm for internal data representation and query model

well-distributed, and scalable organization of LDAP servers, regardless of the underlying application, enhancing at the same time the system's performance is thus necessary.

1.2 Motivation and Contribution

Meeting the challenges for the development of an efficient LDAP organization and query framework would result in improved performance of a number of applications such as the ones discussed in Section 1. Our purpose is to employ appropriate data organization methods that will automatically define the DIT structure and render improved directory services performance, independently of the underlying application or the specific configuration settings. To this context, clustering can be used for the identification of related LDAP entries that can guide the data organization in subtrees. Clustering analysis is a well-known approach that is particularly appropriate for the exploration of interrelationships among the data points [Crabtree et al. 2005, 2006; Wang and Kitsuregawa 2001]. According to the authors' knowledge, LDAP and data clustering technologies have been barely combined. A clustering approach of LDAP metadata has been proposed to facilitate discovering of related directory objectclasses to better enable their reconciliation and reuse [Liang et al. 2006]. However, the development of a framework that will propose a well-distributed and scalable organization of an LDAP server's data, resulting in different levels of hierarchy, is necessary and can lead to improved directory services.

In this article we propose the use of clustering as a tool to indicate the LDAP data organization. Specifically, the LDAP entries' clusters will constitute the subtrees of the DIT structure, as depicted in Figure 4.

Moreover, the clustering-based LDAP data organization constitutes the basis on which a query engine, the *LP-LM Query Engine* presented in this article, operates and results in enhanced LDAP server's performance. This is accomplished by directing LDAP queries, based on the keywords they contain, to specific DIT subtrees which correspond to clusters extracted by the LP-LM clustering algorithm. The idea behind the proposed query engine is to take advantage of the organization indicated by clustering and accurately define a reduced search space for each LDAP query. The defined search space will include those DIT subtrees that are likely to contain entries related to the keywords of a specific LDAP query. A graphical representation of the overall idea is presented in Figure 5.

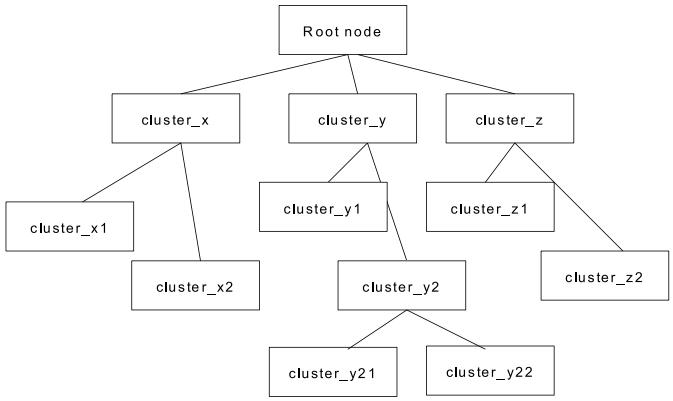


Fig. 4. LDAP entries' clusters correspond to DIT subtrees.

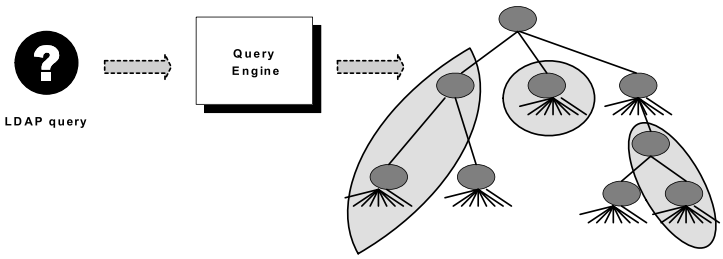


Fig. 5. Clustering-based LDAP Query Framework.

More specifically, in this article we perform a thorough study of the results obtained when the LPAIR and LMERGE (LP-LM) agglomerative structure-based clustering algorithm, introduced by the authors in Koutsonikola et al. [2008], is applied. Our main contributions can be summarized as follows.

— We describe the notions of LD-trees and LD-vectors and we provide the problem formulation in order to adequately describe the LDAP data structure that will be used in the clustering process.
— We examine the algorithm's behavior when combined with various distance measures in order to determine how their usage affects the algorithm, in terms of the degree of distinctiveness they offer.
— We propose a query framework implemented by the presented LP and LM Query Engine, which, considering the data organization proposed by the LP-LM algorithm, leads to an enhanced LDAP server's performance.
— We propose the *Relative Link* as an alternative merging criterion which identifies the clusters to be merged at each step and which, as indicated by the experimental results, leads to more balanced clusters.

The rest of the article is organized as follows: Section 2 discusses some basic concepts of LDAP data representation and the introduced LD-tree and LD-vector structures. Section 3 describes our problem formulation and the proposed LDAP clustering algorithm. Moreover the discussion about two employed merging criteria is provided. In Section 4 the organization of DIT according to the clustering results is described as well as the proposed query engine. Section 5 presents the experimentation while conclusions and future work insights are given in Section 6.

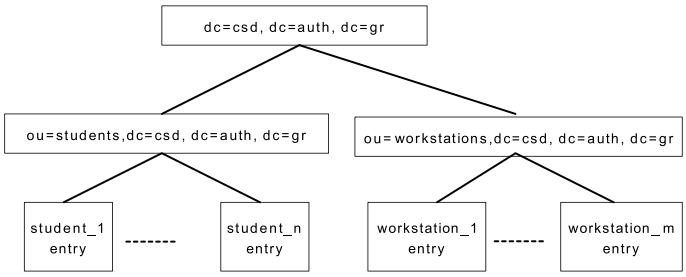
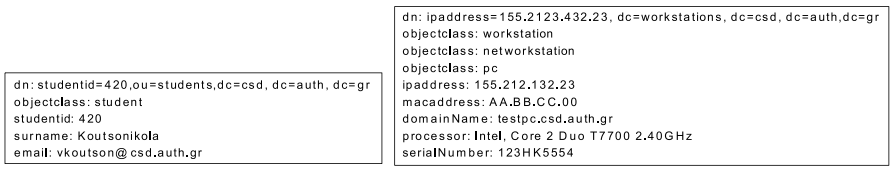


Fig. 6. LDAP DIT.



(a) A student's entry (b) A workstation's entry

Fig. 7. LDAP entries' LDIF.

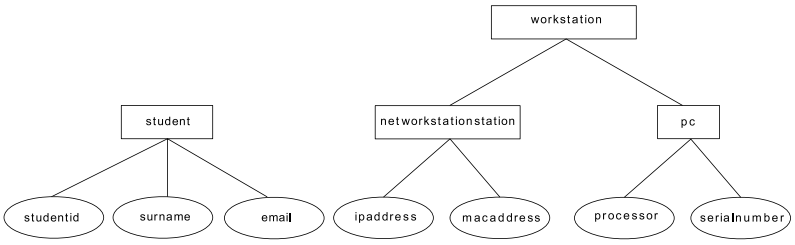
2. LDAP DATA REPRESENTATION

2.1 Basic LDAP Concepts

LDAP directories are databases arranged in a hierarchical tree-like structure called the Directory Information Tree (DIT). Entries are arranged within the DIT based on their Distinguished Name (DN) which unambiguously identifies a single entry, while the overall DIT originates from a root (RootDN). Given that in the basic LDAP notation, "dc" stands for domain component and "ou" for organizational unit, the RootDN of the DIT that describes the students and workstations data of the Computer Science Department of AUTH[2] would be "dc=csd, dc=auth, dc=gr" while the DNs of the nodes referring to the students and workstations would be "ou=students, dc=csd, dc=auth, dc=gr" and "ou=workstations, dc=csd, dc=auth, dc=gr", respectively. Figure 6 depicts the DIT of the described entities.

The basic LDAP storage unit is the directory entry, which is where information about a particular object resides. All information within a directory entry is stored as attribute-value pairs while the set of attributes that appear in a given entry is determined by the objectclasses that are used to describe it. For example, in the definition of the user-defined "student" objectclass, the attribute "studentid" and "surname" could be considered as mandatory while the "email" attribute could be defined as optional. Moreover, the user-defined "networkstation" objectclass, used to describe the workstations that belong to the computer science department, may consider as mandatory the "ipaddress", and "macaddress" attributes and "domainname" as optional, while the objectclass "pc" defines the "processor" and "serialnumber" attributes as required. Figures 7(a) and 7(b) present the LDIF[3] of a sample student and workstation entry, respectively.

[2]Aristotle University of Thessaloniki: http://www.auth.gr.
[3]LDIF (LDAP Data Interchange Format) is a standard for representing LDAP entries in human-readable format.



(a) A student's entry (b) A workstation's entry

Fig. 8. LDAP entries represented by tree structures.

The set of rules that define the objectclasses and the attributes they contain constitutes the LDAP schema. To preserve interoperability between different vendors' LDAP servers, a well-defined standard schema exists which is expected to be included in all LDAP servers. However, the LDAP schema is extensible, allowing users to define their own objectclasses and attributes to meet their applications' needs. LDAP schema also supports inheritance, meaning that one objectclass may inherit attributes from another. The relations between objectclasses and attributes, which describe an LDAP entry and are defined by the LDAP schema, can be sufficiently captured using a tree structure. Figure 8 depicts the trees that describe the student and workstation LDAP entries presented in Figure 7.

2.2 LDAP Data Representation

We consider a particular framework where we have as source a set $E = \{e_1, \ldots, e_f\}$ of f LDAP entries. Each of these entries is described using specific objectclasses and attributes. Let $O = \{o_1, \ldots, o_m\}$ denote the set of m objectclasses and $A = \{a_1, \ldots, a_n\}$ the set on n attributes used to describe E. Relations exist between specific objectclasses and attributes but also between objectclasses, as described in Section 2.1, due to the inheritance that the LDAP schema supports. As also described in Section 2.1, the internal structure of an LDAP entry can be described using a tree structure which captures the relations between objectclasses and attributes.

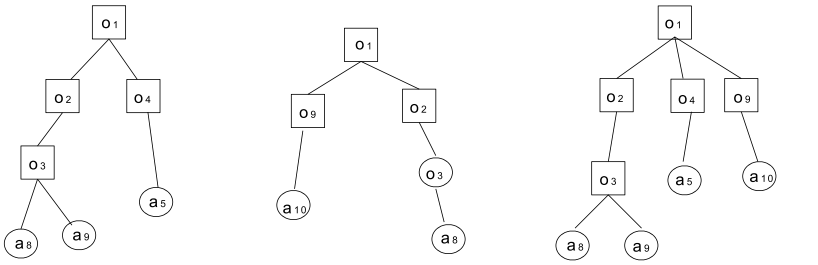
Definition 2.1 (*LD-Tree of an Entry*). Given an LDAP entry $e_x \in E$, the *LD-tree* of e_x is a tree structure $LDT(e_x) = (N; D)$ where $N = \{O \cup A\}$ the set of nodes and $D = \{(d_x, d_y) : d_x \in O, d_y \in \{O \cup A\}, d_x \neq d_y\}$ the set of edges, if and only if $\forall(d_x, d_y)$ pair, d_y is an objectclass that inherits objectclass d_x, or d_y is an attribute describing d_x in e_x.

The definition of an entry's LD-tree is extended to a set of entries as follows.

Definition 2.2 (*LD-Tree of an Entries' Set*). Given a set $E^* \subseteq E$ of f^* LDAP entries where $f^* \leq f$, the $LDT(E^*)$ is defined as $LDT(E^*) = \{\cup LDT(e_i)\}\ \forall e_i \in E^*$.

Figures 9(a) and 9(b) present the LD-trees of two distinct LDAP entries e_1 and e_2, respectively, where we can clearly see how some objectclasses are related (inherit) to other objectclasses and attributes. Figure 9(c) depicts the LD-tree describing both LDAP entries and containing the objectclasses and attributes of both entries.

In order to define data structures suitable for data processing, we proceed to the definition of a vector structure called LD-vector, which was first introduced by the authors in Koutsonikola et al. [2008] and sufficiently captures the structure of an LD-tree, withtout any loss of information.



(a) LDT(e_1) (b) LDT(e_2) (c) LDT(E^*), $E^* = \{e_1, e_2\}$

Fig. 9. LDAP tree structures.

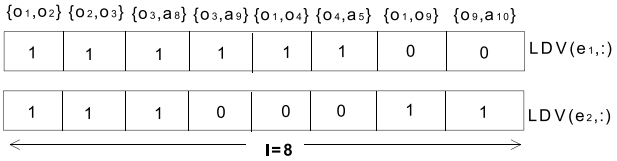


Fig. 10. LD-vectors of e_1 and e_2 entries.

Definition 2.3 (*LD-Vector*). Given the E^* set of f^* LDAP entries and its $LDT(E^*)$ LD-tree, we use l to denote the number of (d_x, d_y) pairs where $(d_x, d_y) \in LDT(E^*)$. Then $\forall e_i \in E^*$ we define the binary *LD-vector* $LDV(e_i, :)$ of e_i as follows.

$$\underset{1 \le i \le f^*, 1 \le r \le l}{LDV(e_i, r)} = \begin{cases} 1 \ \textit{if the r-th } (d_x, d_y) \textit{ pair of } LDT(E^*) \in LDT(e_i) \\ 0 \ \textit{otherwise} \end{cases}$$

Example 1. Let us consider the e_1 and e_2 entries represented by the LD-trees of Figures 9(a) and 9(b). In order to create their LD-vectors, we use the set of edges $D = \{(o_1, o_2), (o_2, o_3), (o_3, a_8), (o_3, a_9), (o_1, a_4), (o_4, a_5), (o_1, o_9), (o_9, a_1 0)\}$ of the $LDT(E^*)$ (Figure 9(c)). The presence of a node in the LD-vector of an entry is denoted by 1 and the absence by 0. Figure 10 presents the LD-vectors $LDV(e_1, :)$ and $LDV(e_2, :)$ of entries e_1 and e_2, respectively.

The preceding definition of the LD-vector concept provides a description of the LDAP entries structure which will be used by the clustering algorithm in order to identify similarities between LDAP data. This vector is suitable for clustering due to its simplicity and data representation adequacy which will not burden the algorithm's performance.

3. PAIRWISE CLUSTERING

The organization of data entries under the LDAP hierarchy is usually performed intuitively, based on the entities these entries describe (e.g., people, products). However, a framework that could indicate the data organization, with no a priori knowledge of the underlying application, and could, moreover, proceed to further data fragmentation in subtrees, would be more flexible. Our proposed framework employs clustering, which is a well-studied problem that aims to partition data objects into groups such that objects in the same group are more similar, while objects in different groups present more dissimilarities. In a clustering algorithm, the adopted similarity (dissimilarity) measure is closely associated to the data nature (e.g., numerical, categorical). According to Guha et al. [1999], in case of categorical data clustering approaches (as the one

discussed in this article), a link-based approach adopts a global perspective of the clustering problem, compared to the local one that is obtained when considering only the characteristics of the data themselves. This is due to the fact that a linked-based approach suggests that it would be beneficial to group points which share a large number of common neighbors, as expressed by the link values. Moreover, link-based clustering has proven quite beneficial in case of clustering XML documents [Lian et al. 2004].

3.1 Problem Formulation

In a link-based clustering approach applied on LDAP data, the relation between two LDAP entries is measured in terms of their common neighbors, as expressed by the link value. Specifically, two entries are considered to be neighbors if their distance D, which expresses their resemblance in terms of the number of attributes they share, is less than a user-defined threshold θ. The set $LN(e_i)$ contains the LDAP entries that are neighbors to $e_i \in E$ and is defined as follows.

$$LN(e_i) = \{e_j : D(e_i, e_j) \leq \theta\} \forall e_i, e_j \in E \tag{1}$$

As mentioned before, the link between two entries e_i and e_j expresses the number of their common neighbors and is given by

$$link(e_i, e_j) = \mid \mathrm{LN}(e_i) \cap \mathrm{LN}(e_j) \mid \forall e_i, e_j \in E. \tag{2}$$

Let C_i denote the $i-$th of the k clusters obtained by the clustering algorithm and c_i its size. Then, given that the user-defined threshold θ identifies neighbors, for $\theta = 0$ an e_i entry belonging to C_i cluster is expected to have only itself as a neighbor since there is no other entry e_j such that $D(e_i, e_j) = 0$. On the other hand, for $\theta = 1$, all c_i entries are expected to be neighbors of e_i since $D(e_i, e_j) \leq 1 \ \forall e_i, e_j \in E$. For $0 < \theta < 1$ it is expected that higher values of θ will result in more neighbors of e_i.

Inspired by the ideas of market basket analysis Guha et al. [1999] defined the quantity $c_i^{\frac{1-\theta}{1+\theta}}$ to provide the expected number of neighbors of an entry belonging to a cluster of size c_i. In case of Guha et al. [1999], similarities between entries are evaluated and θ expresses the threshold of similarity above which two entries are considered to be neighbors. However, in our approach, θ expresses the dissimilarity threshold for neighbors and thus the expected number of neighbors can be calculated by the expression $c_i^{1-\frac{1-\theta}{1+\theta}} = c_i^{\frac{2\theta}{1+\theta}}$. Furthermore, this quantity coincides with our expectations on the number of neighbors for the different values of θ, which was earlier discussed. Besides, as denoted in Guha et al. [1999], any possible errors in the estimation of the aforesaid quantity affects all the clusters similarly due to the objective function normalization.

LEMMA 3.1. *Given a cluster C_i of c_i size and the $c_i^{\frac{2\theta}{1+\theta}}$ expected number of neighbors for each $e_i \in C_i$ entry, the e_i contributes to an expected link value equal to $c_i^{\frac{4\theta}{1+\theta}}$.*

PROOF. To compute the link value caused by the entry $e_i \in C_i$, we need to compute the number of (e_x, e_y) $(e_i \neq e_x, e_y)$ pairs that have e_i as neighbor [Guha et al. 1999]. For this computation, we pose the following assumptions: (i) it holds that both the (e_x, e_y) as well as the (e_y, e_x) have e_i as neighbor and (ii) each $e_x \in C_i$ forms the (e_x, e_x) pair and has e_i as its neighbor. Thus, the total number of expected links $EL(e_i)$ caused by e_i entry is equal to

$$EL(e_i) = 2 \cdot \binom{c_i^{\frac{2\theta}{1+\theta}}}{2} + c_i^{\frac{2\theta}{1+\theta}} = 2 \cdot \frac{(c_i^{\frac{2\theta}{1+\theta}})!}{2! \cdot (c_i^{\frac{2\theta}{1+\theta}} - 2)!} + c_i^{\frac{2\theta}{1+\theta}} = c_i^{\frac{4\theta}{1+\theta}}. \qquad \square$$

Then, the total number of expected links in C_i cluster caused by all c_i in number entries, will be $c_i * EL(e_i) = c_i * c_i^{\frac{4\theta}{1+\theta}}$ which results in $c_i^{1+\frac{4\theta}{1+\theta}}$.

The goal of a link-based clustering approach is to maximize the sum of common neighbors (link) between each pair of entries belonging to a single cluster. However, aiming at the maximization of the $\sum_{e_x,e_y \in C_i} link(e_x, e_y)$ could result in the assignment of all entries to a single cluster. It is therefore important to define an objective function that will create clusters where the sum of links between members of the same cluster will be maximized and the sum of links between members of different clusters will be minimized. Moreover, the objective function defined must favor balanced clusters. Inspired by Guha et al. [1999], we define a criterion function J where the total number of links between the entries belonging to a cluster C_i is divided by the expected link value for this cluster, weighted by the size of cluster c_i. Dividing by the expected number of links in $J(E)$ prevents points with very few links between them from being put in the same cluster, since assigning them to the same cluster would cause the expected number of links for the cluster to increase more than the actual number of links and the result would be a smaller value for the criterion function. Thus, the objective function $J(E)$ is defined as

$$J(E) = \sum_{i=1}^{k} c_i * \sum_{e_x,e_y \in C_i} \frac{link(e_x, e_y)}{c_i^{1+\frac{4\theta}{1+\theta}}}. \tag{3}$$

Our goal is to maximize the link value of entries contained in a cluster. The LDAP clustering problem can now be defined as follows.

Problem 1 (*LDAP Clustering*). Given a set E of f LDAP entries, a user-defined integer value k, and the criterion function $J(E)$, find a CL clustering of E into k clusters such that the $J(E)$ is maximized.

The value of k which defines the number of clusters is set by users according to the degree they want entries to be distinguished in terms of their structure. Thus, users may decide upon k's value considering dataset's heterogeneity and desirable depth of DIT hierarchy.

3.2 Capturing Similarity Between LDAP Entries

The selection of the similarity or distance measure is important for every clustering process. In our framework, the representation of LDAP entries as binary vectors makes dissimilarity coefficients an appropriate choice for measuring distance between them. Given two binary vectors $LDV(e_x, :)$ and $LDV(e_y, :)$ of length l where $e_x, e_y \in E, x \neq y$, there are three fundamental quantities [Hohn 2005] that can be used to define the similarity between them.

— $a = |t| : \{LDV(e_x, t) = V(e_y, t) = 1\}$
— $b = |t| : \{LDV(e_x, t) = 1 \wedge V(e_y, t) = 0\}$
— $c = |t| : \{LDV(e_x, t) = 0 \wedge V(e_y, t) = 1\}$,

where $1 \leq t \leq l$.

The a, b, and c values are used to designate the common and different elements of two binary vectors. Given their values, Table II describes five popular dissimilarity coefficients that are employed in case of binary vectors [Li 2005].

All of the previous coefficients range between 0 and 1. Higher values of the coefficients indicate higher dissimilarity between the involved entries. The definition of the five coefficients shows that their difference lies mainly on the way they consider the

Table II. Dissimilarity Coefficients for Binary Data

Dissimilarity Coefficient	**Formula**
Jaccard coefficient	$\frac{b+c}{a+b+c}$
Czekanowski or Dice coefficient	$\frac{b+c}{2a+b+c}$
Rogers and Tanimoto coefficient	$\frac{b+c}{\frac{1}{2}\cdot a+b+c}$
Simpson	$1-\frac{a}{min(a+b,a+c)} = 1-\frac{a}{min(\lvert e_x\rvert,\lvert e_y\rvert)}$
Braun	$1-\frac{a}{max(a+b,a+c)} = 1-\frac{a}{max(\lvert e_x\rvert,\lvert e_y\rvert)}$

common and different attributes of two entries. More specifically, the definition of Jaccard, Dice, and Rogers coefficients differentiate on the gravity they give on the common elements of two LD-vectors, while according to their definitions, Simpson and Braun coefficients additionally take into account the size of the involved LD-vectors. These two coefficients favor LD-vectors with a smaller number of edges since they result in lower values of dissimilarity.

The suitability of the preceding dissimilarity coefficients is mainly determined by the underlying application and the involved data. The Braun and Simpson coefficients take into account more information about the compared entries, since they consider not only their common elements but also the total number of elements that their similarity is recorded (i.e., the entries' size). Incorporating the knowledge of the entries' sizes is more meaningful in case that a high number of LD-vectors is used to describe the LDAP entries. In that case, it is crucial to consider more information, since only the common and different elements may result in a less efficient distinction and unbalanced clusters. The higher the number of objectclasses and attributes used to describe LDAP entries, the more possible to result in more LD-vectors. Thus, in case of more heterogeneous datasets, the Braun and Simpson coefficients are recommended. With respect to the application scenarios described in Section 1, the Braun and Simpson dissimilarity coefficients would be more appropriate for the organization of sharing applications and network operation frameworks LDAP data, since in these cases, the diversity between entries is expected to be higher.

On the other hand, in case of datasets such as those describing users entries in an academic or enterprise environment, the used attributes are more standard and usually result in a lower number of LD-vectors. In these cases, considering the common and different elements between entries would sufficiently capture their dissimilarities. Therefore, for datasets characterized by higher homogeneity, the Jaccard, Dice, and Rogers coefficient can effectively be used.

The aforesaid coefficients have been employed to capture distances in various clustering approaches [Haranczyk and Holliday 2008; Murguia and Villasenor 2003; Ponaramenko et al. 2002].

Example 2. Let us consider the set $E^* = \{e_1, e_2, e_3\}$ of $l = 3$ LDAP entries and its corresponding LD-tree $LDT(E^*)$ which contains the D = { *(student, name), (student, surname), (student, email), (student, address)* } set of edges. Moreover, let the LD-vectors of the former three LDAP entries be defined as $LDV(e_1) = [1, 1, 0, 0]$, $LDV(e_2) = [1, 1, 1, 0]$ and $LDV(e_3) = [0, 1, 1, 1]$.

Table III presents the values of a, b, and c and the calculated distances for all pairs of entries, in terms of each of the defined dissimilarity coefficients. As it is clearly depicted in the Table III, in cases that the common edges are many (a is higher), as it happens for (e_1, e_2) and (e_2, e_3) where $a = 2$, the calculated dissimilarities are lower compared to those found for a smaller number of common edges (lower values of a) as in case of (e_1, e_3) where $a = 1$. Moreover, in the cases that the same number of common edges appears, the calculated dissimilarities are higher when the number of different

Table III. Dissimilarities between LDAP Entries

	Jaccard	**Dice**	**Rogers**	**Simpson**	**Braun**
e_1, e_2 ($a = 2,\ b = 0, c = 1$)	0.33	0.25	0.5	0	0.33
e_1, e_3 ($a = 1,\ b = 1, c = 2$)	0.75	0.6	0.85	0.5	0.66
e_2, e_3 ($a = 2,\ b = 1, c = 1$)	0.5	0.33	0.66	0.33	0.33

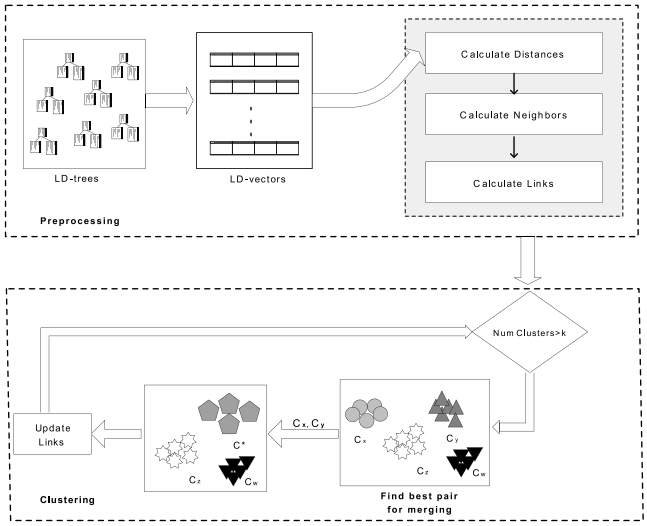


Fig. 11. The LP-LM clustering process.

edges between two entries is higher, and this happens for the (e_2, e_3) pair where the number of different edges is 2. For the (e_1, e_2) pair the respective number is 1.

3.3 The LPAIR and LMERGE (LP-LM) Clustering Algorithm

The proposed LPAIR and LMERGE (LP-LM) algorithm is a hierarchical structure-based algorithm which aims to find a solution to Problem 1. It is a two-step process which results in k clusters of LDAP entries characterized by high similarity. The similarity between LDAP entries is expressed by the their link value.

3.3.1 Clustering Phases. The two steps of the LP-LM algorithm are depicted in Figure 11.

At the beginning, the preprocessing takes place, where given a set of LDAP entries as described by their corresponding LD-trees, the set of LD-vectors is defined. Next, using one of the dissimilarity coefficients discussed in Section 3.2, dissimilarities between LD-vectors are calculated. Based on the dissimilarities, the set of neighbors $LN(e_i)$ of each e_i entry is found (Eq. (1)) and then, the link value between each pair (e_i, e_j) of LDAP entries is computed (Eq. (2)).

The link values "carry" the information about the similarity between entries and are used by the second algorithm's step, which constitutes the main clustering process. In this step, an iterative process takes place which terminates when k clusters are obtained. The proposed clustering algorithm is agglomerative in nature, and thus, at each step of the execution, the algorithm decides, based on a merging criterion, which is the best (C_x, C_y) pair of clusters to merge so as the values of the objective function $J(E)$ (Eq. (3)) will be maximized. In the definition of the merging criterion, the link

value between pairs of clusters is considered, as will be later discussed. The merging of the C_x and C_y clusters results in the C^* cluster. The link values with respect to the new C^* are computed, while the link values that referred to the C_x and C_y clusters are updated in terms of the C^* cluster. The whole process is repeated until only k clusters are remaining.

The LP-LM algorithm actually applies the ideas of ROCK [Guha et al. 1999] algorithm on a set of LDAP entries which are modeled as categorical data at the end of the preprocessing step. The proposed algorithm is also of the same logic with the S-GRACE [Lian et al. 2004] algorithm which applies ROCK on the s-graph representation of XML documents. Both LP-LM and S-GRACE algorithms adopt ROCK's ideas because ROCK handles the case that, even though some data points may not be close enough in distance but they share a large number of common neighbors, it would be beneficial to consider them belonging to the same cluster. However, appropriate adjustments are required in order to incorporate the LDAP and XML specifications and result in suitable data structure representations. For example, an xml document may be schema independent and thus result in arbitrary (graph) structure representation which cannot occur in case of LDAP entries that always obey the LDAP schema. Moreover, XML has proliferated as the standard for data representation while LDAP standard and directory services in general provide a whole framework for data storage and management.

3.3.2 The LP-LM Algorithm. The LP-LM algorithm takes as input a set E of f LDAP entries, the number k of clusters to be created, and a decimal θ, $0 \leq \theta \leq 1$, which defines the upper limit of distance between two LDAP entries that makes them neighbors. The algorithm's output is the assignment of LDAP entries to the k clusters.

At the beginning, the preprocessing takes place, where given the initial set E of f LDAP entries and the value of θ, the computation of basic structures for the clustering process follows. The algorithm initially constructs the entries' LD-trees (line 2) and then the respective LD-vectors (line 3). Using an appropriate dissimilarity coefficient, as those discussed in Section 3.2, the table D that records the distances between LDAP entries is computed (line 4) and then, based on D and θ, the algorithm calculates each entry's neighbors and stores them in table LN (line 5). The LN table is used by the algorithm to find the number of common neighbors between each pair of entries (link), and keeps them in the *link* table (line 6). Initially, each entry constitutes a separate cluster. For each cluster C_i (initially equal to e_i), the candidates for merging, that is, entries e_j for which $link(e_i, e_j) > 0$, are organized in the $CA(e_i)$ list (lines 7–9). Then, for each (e_i, e_j) candidate for merging pair of LDAP entries, the values of the merging criterion mc are calculated and stored in the triangle MCA table (lines 10–12).

After the preprocessing step, an iterative process follows which constitutes the main clustering process. This new step lasts until k clusters will be obtained (line 14). At each iteration of this step, the clusters C_x and C_y that constitute the best pair of clusters for merging, is the one with the highest merging criterion value, recorded in the MCA table (line 15). The merging of the former two clusters results in the new C^* cluster (line 16). The link values between the rest of the candidates for merging C clusters and the C_x and C_y are computed with respect to the new C^* cluster (line 18). The update of the CA table signifies the removal of the $CA(C_x)$ and $CA(C_y)$ lists and the addition of the reference to the C^* (line 19). Moreover, the update of the MCA table refers to the calculation of the merging criterion values that involve the new C^* cluster (line 20). This process iterates until the number of clusters is equal to k which consists the assignment of the f LDAP entries to k clusters.

ALGORITHM 1 The LPAIR and LMERGE algorithm.

Input: A set $E = \{e_1 \dots e_f\}$ of f LDAP entries, a threshold θ and the number of clusters k.

Ouput: Criterion function J and assignment of the LDAP entries in the k clusters, such that the criterion function value J is minimized.

```
1: /*Preprocessing*/
2: LDT(E) = CreateLDT(E)
3: LDV(E) = CreateLDV(LDT(E))
4: D = ComputeDistance(LDV(E))
5: LN = ComputeNeighbors(D, θ)
6: link = ComputeLink(LN)
7: for i := 1 to f do
8:    CA(e_i) = FindCandidates(e_i, link)
9: end for
10: for all e_i, e_j ∈ CA(e_i), e_i ≠ e_j do
11:    MCA(e_i, e_j) = FindMCValue(e_i, e_j, mc)
12: end for
13: /*Clustering process*/
14: while NumClusters > k do
15:    (C_x, C_y) = FindBestCandidates(MCA,mc)
16:    C* = merge(C_x, C_y)
17:    for all C ∈ CA(C_x) ∪ CA(C_y) do
18:       link(C, C*) = link(C, C_x) + link(C, C_y)
19:       update(CA)
20:       update(MCA)
21:    end for
22:    delete(CA(C_x)); delete(CA(C_y))
23: end while
```

3.4 Merging Criteria

In the proposed clustering framework, the merging criterion plays a major role, since it determines the pair of clusters that will be merged at each step of the agglomerative algorithm. The merging criterion must conform with the algorithm's objective which is the maximization of the objective function $J(E)$ defined in Eq. (3). In this section, two merging criteria are presented which are employed by the LP-LM algorithm. The first criterion is the *Expected Link* which is based on an estimation of the cross-link value between two candidates for merging clusters. The *Relative Link*, is proposed as an alternative merging criterion which is based on quantities computed during the algorithm's execution and is expected to guide the clustering process with more accuracy.

3.4.1 The Expected Link. At each step of the execution, the algorithm must decide upon the pair of clusters that will be merged so as the values of the objective function $J(E)$ (Eq. (3)) will be maximized. According to Eq. (3), the maximization of $J(E)$ signifies maximization of each cluster's link value. Thus, in each iteration, the best pair of (C_i,C_j) clusters candidate for merging is the one with the highest link value between them, defined as

$$link(C_i, C_j) = \sum_{e_x \in C_i, e_y \in C_j} link(e_x, e_y).$$

Considering only the cross links between two clusters does not ensure that, in case of unbalanced clusters, a large cluster will not "overwhelm" the whole process. Thus, it

is necessary to define a merging criterion that will favor the creation of more balanced clusters. Similarly to the definition of $J(E)$, in order to prevent the continuous merging of large size clusters, we divide the cross-links $link(C_i, C_j)$ between two candidates for merging C_i and C_j clusters, with the expected cross-link value between them. To compute the expected cross-link value between the two clusters we need to calculate the total link value if we considered them as one (i.e., $(c_i+c_j)^{1+\frac{4\theta}{1+\theta}}$) and subtract the link value of C_i (i.e., $c_i^{1+\frac{4\theta}{1+\theta}}$) and C_j (i.e., $c_j^{1+\frac{4\theta}{1+\theta}}$). Therefore, we can now define the Expected Link merging criterion $EL(C_i, C_j)$ of clusters C_i and C_j as

$$EL(C_i, C_j) = \frac{link(C_i, C_j)}{(c_i + c_j)^{1+\frac{4\theta}{1+\theta}} - c_i^{1+\frac{4\theta}{1+\theta}} - c_j^{1+\frac{4\theta}{1+\theta}}}. \quad (4)$$

The pair of clusters that maximize EL will be merged at each step of the algorithm's iteration.

3.4.2 Relative Link - An Alternative Merging Criterion. The Expected Link EL merging criterion defined in Eq. (4) (Section 3.3) is based on an estimation of the expected number of cross links between two candidates for merging clusters. Here, we propose the usage of a new merging criterion, the Relative Link *(RL)*, which is based on quantities calculated during the algorithm's execution. The definition of this criterion was inspired by the relative closeness defined in CHAMELEON [Karypis et al. 1999], and expresses the absolute link value of two clusters normalized with respect to the internal link of the two clusters.

$$RL(C_i, C_j) = \frac{link(C_i, C_j)}{\frac{c_i}{c_i+c_j} * InternalLink(C_i) + \frac{c_j}{c_i+c_j} * InternalLink(C_j)} \quad (5)$$

For the calculation of InternalLink values, we initially consider that the internal link for each e_x entry is equal to the number of its neighbors, that is,

$$InternalLink(e_x) = |\mathrm{LN}(e_x)| \forall e_x \in E.$$

As the algorithm proceeds, when two clusters C_i and C_j are merged, the internal link of the new cluster C^* is calculated as

$$InternalLink(C^*) = InternalLink(C_i) + InternalLink(C_j) + Link(C_i, C_j).$$

With the definition of the Relative Link merging criterion, the decision about the clusters that will be merged at each step is based on values computed during the algorithm's execution, and not estimated ones. Moreover, this criterion favors smaller clusters to be chosen for the merging.

4. CLUSTERING-BASED LDAP DATA ORGANIZATION

Defining appropriate LDAP data organization schemes constitutes a significant issue for LDAP administrators. In this article we deal not only with the development of an LDAP data clustering algorithm (as discussed in Section 3), we also employ the proposed clustering algorithm as a mechanism that will indicate LDAP organization, altering an initial flat DIT to a hierarchical one. Specifically, the application of the LP-LM clustering algorithm results in the creation of k clusters which contain similar LDAP entries in terms of their structure. These clusters will be the basis of the new LDAP data organization, that is, considering the clustering results we create one subtree for each of the obtained clusters.

For example, let us consider the set of LDAP entries that was described in Section 2.1, which contains the entries of the students and workstations of the computer

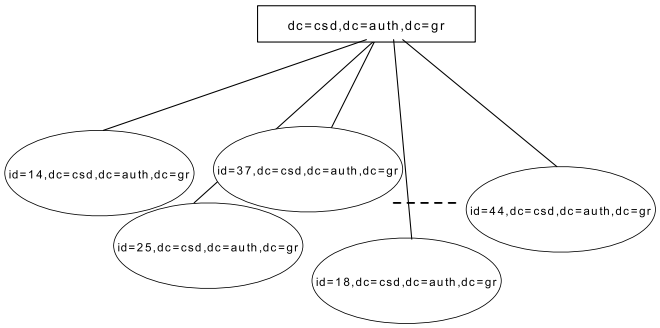


Fig. 12. Flat DIT ldap entries organization.

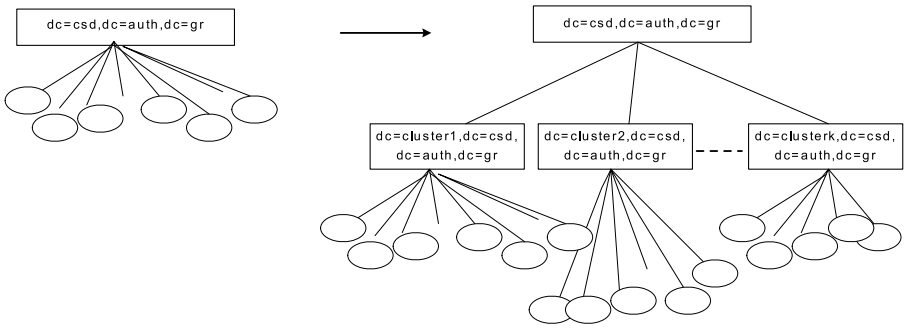


Fig. 13. Clustering-based DIT organization.

science department. The "dn" of the LDAP DIT will be "dc=csd, dc=auth, dc=gr" while the "dn" of each entry describing either a student or a workstation in a flat DIT will be "id=*idnumber*, dc=csd, dc=auth, dc=gr". Moreover, the flat organization of the LDAP entries in the DIT are depicted in Figure 12.

We assume that we apply the LP-LM clustering algorithm seeking for k clusters. Then, the dn for each subtree which corresponds to one cluster will be of the form "dc=$cluster_x$, dc=csd, dc=auth, dc=gr". Thus, the new LDAP organization is not restricted by any semantic that may rule LDAP data. To become more clear, if we applied LP-LM algorithm for $k = 4$ clusters, we could obtain 2 clusters containing students' entries and 2 clusters containing workstations' entries, or 3 clusters with students' and 1 with workstations' entries, etc. Figure 13 depicts how the initial flat DIT becomes hierarchical and the LDAP data are organized according to the clustering results.

The number of clusters k is a user-defined parameter which may be set according to the users' intuition derived from the problems setting or the distinctness they want between the obtained clusters. The higher the number of requested clusters, the more the clusters presenting greater heterogeneity will be split.

In this context, the LP-LM algorithm can be also applied to one of the obtained subtrees (clusters) resulting in further grouping of its entries and thus in greater depth of the DIT and more extensive separation of a specific subtree's entries. For example, if the algorithm was applied on the second subtree of the DIT for $k = 3$, then data would be organized according to Figure 14. This second perspective is useful in cases that we observe that clustering the whole set of entries results in some large size clusters. In such cases, we can perform clustering to the subtrees corresponding to those clusters in order to obtain a more balanced and hierarchical DIT.

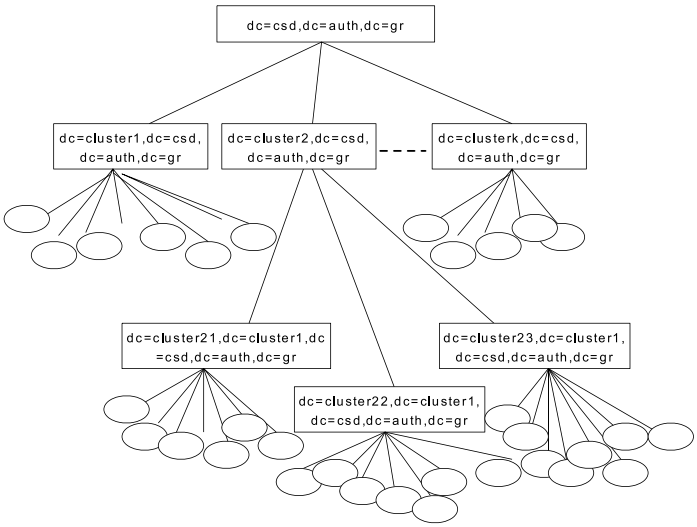


Fig. 14. Applying clustering on a subtree.

4.1 Implementing a Clustering-Based Query Engine

Clustering LDAP data aims to propose an appropriate LDAP data organization that can be beneficial for the overall LDAP server's performance. In this section we present a query engine that operates based on the LP-LM results. The purpose of the proposed *LP and LM Query Engine* is to take advantage of the clustering-based data organization and direct the user queries to the cluster or clusters that the requested data exists, reducing, thus, the search space. This is feasible through the underlying query mechanism provided by the LDAP protocol, since in the LDAP query model, a search request is composed by the following main parts.

— *BaseDn* determines the distinguished name of the node in directory information tree hierarchy that the search will begin.
— *Scope* defines the level in depth that the search operation will take place.
— *Filter* specifies the criteria an entry must match to be returned from a search. The search filter is a boolean combination of attribute value assertions. LDAP supports exact and approximate matching.
— *Attributes*. A list of the attributes to be returned from each entry, which matches the search filter.

Thus, if we manage to specifically define the subtree or subtrees that contain the answer to a particular LDAP query, we can direct appropriately the query, as depicted in Figure 5, and significantly reduce its response time.

The idea behind the query engine is the following: after the clustering process terminates, the LDAP data is organized according to the clustering results. The proposed LP and LM Query Engine operates on top of the LDAP query model and its purpose is to limit the scope of the LDAP queries by excluding subtrees (i.e., clusters) that do not contain entries related to the LDAP queries. Specifically, the operation of the LP and LM Query Engine is based on the functionality of the following software components.

— *The cluster-keywords extraction component* accesses the LDAP database and extracts the keywords that characterize each of the k obtained clusters. These keywords are the distinct objectclasses and attributes that exist in the entries of a

ALGORITHM 2 The CLUSTER-KEYWORDS EXTRACTION COMPONENT.

Input: the set of k clusters
Ouput: the sets of objectclasses' $O(C_i)$ and attributes' $A(C_i)$, $i = 1 \ldots k$, keywords describing the clusters.

```
for i := 1 to k do
  for all ldif(e_x) : e_x ∈ C_i do
    A(C_i) = extractAttributes(e_x);
    O(C_i) = extractObjectclasses(e_x);
  end for
end for
```

ALGORITHM 3 The QUERY-KEYWORD EXTRACTION COMPONENT.

Input: an LDAP query
Ouput: the set of keywords (objectclasses and attributes) the LDAP query contains.

```
queryFilter = ExtractFilter(LDAPQuery);
queryKeywords = extractKeywords(queryFilter);
```

cluster. The operation of the *cluster-keywords extraction component* is described in the form of pseudocode by Algorithm 2.

Example 3. If we assume that the *cluster-keywords extraction component* was applied on a cluster C_y with LDAP entries described by the LDIF of Figure 7(a) then the set of attributes and objectclasses would be $A(C_y)$ = {*studentid, surname, email*} and $O(C_y)$ = {*student*}, respectively.

— *The query-keyword extraction component* extracts the query's keywords, that is, it identifies the objectclasses and attributes contained in the query. The component's function is based on the identification and processing of the query's filter and it is described by Algorithm 3.

Example 4. An LDAP query looking for all students LDAP entries whose surname begins with "V" is the following:
ldapsearch -b "dc=csd,dc=auth, dc=gr" -s sub "(&(objectclass=student)(surname=V*))"
The -b parameter declares the baseDN of the query (the root node in this case), the -s parameter declares the scope of the query (sub indicates all levels including the baseDN) and the last part of the query constitutes the query filter. The query-keyword extraction component processes the query filter and extracts its keywords (objectclasses and attributes), that is, student and surname.

— *The mapping component* performs a dual functionality: (i) it accepts as input the sets of objectclasses' and attributes' keywords describing the clusters that were extracted by the cluster-keywords extraction component and creates a single file (mapping file) recording the correlation between clusters and keywords and (ii) given the query keywords of an LDAP query and the mapping file, it extracts the clusters that contain answers to the LDAP query and redefines it in order to limit its scope. The mapping component's function is described by Algorithm 4 while the clusters' extraction process is described by Algorithm 5.

Example 5. If we assume that the clustering algorithm has resulted in two clusters, one containing students' entries and the other one containing workstations'

ALGORITHM 4 The MAPPING COMPONENT: Clusters and keywords correlations.

Input: the sets of objectclasses' $O(C_i)$ and attributes' $A(C_i)$, $i = 1 \ldots k$, keywords describing the clusters.
Ouput: the mappingFile
  for $i = 1$ to k **do**
    keywords(C_i)=$A(C_i) \cup O(C_i)$
    print("cluster%d: %s", i, keywords(C_i)) >> mappingFile;
  end for

ALGORITHM 5 The MAPPING COMPONENT: Redefining the LDAP query.

Input: the LDAP query keywords, the mapping file
Ouput: the redefined LDAP query
  for all keyword ∈ queryKeywords **do**
    C_{query} = ExtractCluster(mappingFile, keyword);
    for all $c \in C_{query}$ **do**
      C^*_{query} = redefine(LDAPQuery, c);
    end for
  end for

entries, then the mapping file that would be created by the mapping component would contain lines of the following form:
cluster1: student, studentid, surname, name, email, telephone
cluster2: workstation, networkstation, pc, ipaddress, macaddress, domainName, processor, serialNumer
Next, considering that the LDAP query ldapsearch -b "dc=csd, dc=auth, dc=gr" -s sub "(&(objectclass=student)(surname=V*))" is executed, the mapping component identifies that the "student" and "surname" keywords exist in the cluster1 and thus redefines the LDAP query by adjusting the baseDN as follows:
ldapsearch -b "dc=cluster1, dc=csd, dc=auth, dc=gr" -s sub "(&(objectclass=student)(surname=V*))"
If the mapping component located the query's keywords in more than one clusters, for example, three clusters, then the LDAP query would be replaced by three LDAP queries that refer to specific baseDNs. This way, a reduction of the initial query's search space would be achieved with no loss in terms of the requested data accuracy.

It should be noted that the functionality of the cluster-keywords extraction component and mapping component with reference to the definition of relations between clusters and keywords is applied only once the LP-LM clustering algorithm performs the assignment of LDAP entries to clusters. On the other hand, the redefinition of an LDAP query by the mapping component and the extraction of query keywords by the query-keyword extraction component take place each time an LDAP query is executed.

The functionality of the proposed *LP and LM Query Engine* in a more abstract view is the following: after the clustering process terminates, the LDAP data is organized according to the clustering results. The cluster-keywords extraction component accesses the LDAP database and extracts the keywords that characterize each of the k obtained clusters. These keywords are the distinct objectclasses and attributes that exist in the entries of a cluster. The clusters' keywords are then passed to the mapping component, which records the correlation between clusters and keywords. When an LDAP query is sent by an LDAP-enabled application, it is first processed by the query-keyword extraction component that extracts the query's keywords, that is, it identifies

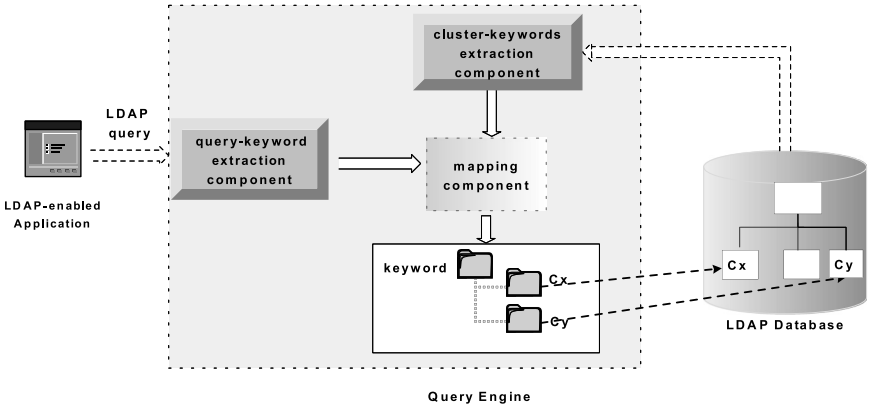


Fig. 15. The LP and LM Query Engine.

the objectclasses and attributes contained in the query. The extracted keywords are passed to the mapping component which, based on the recorded correlations, relates the query with the clusters that contain at least one of the keywords. The LDAP query is then directed by the LP and LM Query Engine, to the subtrees defined by the former clusters. Thus, the query's search space does not contain all the LDAP entries as it would in case of an unclustered (flat) DIT organization. The proposed LP and LM Query Engine limits the LDAP query's search space to the clusters that contain entries related to the query. In the worst case, the query's keywords will be located in all clusters and then the query will have to start by the RootDN. In this case, the response time would be equal to that of the unclustered scheme. In any other case, the search space would be reduced and thus better response times are expected. The overall architecture of the LP and LM Query Engine is depicted in Figure 15.

Example 6. Let us consider an LDAP framework which provides information about a departments's students and workstations. Then, applying the LP-LM clustering algorithm for $k = 2$, the two underlying clusters (one cluster referring to the students and the other to workstations) are extracted. The mapping component assigns the keywords "student", "surname", "email", and "studentid" to cluster $cluster_x$ (students' cluster) and the keywords "ipaddress", "macaddress", "processor", and "price" to cluster $cluster_y$ (workstations' cluster), depicted in Figure 15. When a user or an application performs an ldap query, the query keyword extraction component identifies the keywords and locates the clusters in which these are contained. For instance, if the query requests the ipaddresses of all workstations, then the extracted keywords are "ipaddress" and "workstation" which, according to the mapping component, are included in $cluster_y$. Thus, the query is directed by the LP and LM Query Engine to the LDAP subtree that refers to $cluster_y$, achieving a smaller search space and improved response time, since in case of an unclustered scheme, the search should always start by the RootDN and all entries should be searched.

5. EXPERIMENTATION

In this section we use the LP-LM algorithm to cluster a set of LDAP entries and organize the LDAP Directory Information Tree according to the obtained clusters. Both the algorithm's efficiency and the LDAP server's performance, which operates under the proposed LP and LM Query Engine's framework, are evaluated. Specifically, we perform a study on the clustering results obtained by employing the different distance coefficients, discussed in Section 3.2, in conjunction with different values of θ, in order

to see how they affect the algorithm's performance. Moreover, we present some experimental results that indicate the algorithm's performance when the Expected Link EL and the Relative Link RL (Eq. (5)) are used as the merging criterion. Finally, an experimentation section is provided, where the performance of the LDAP server that adjusts its data organization to the clustering results is examined. The LDAP server integrates with the proposed LP and LM Query Engine for the queries execution.

For the experiments we have used the OpenLDAP[4] directory server with Berkeley DB backend. The cache size was set to zero, in order to obtain unbiased results, and the entries "ids" were indexed. The algorithm was executed on an x86 architecture CPU running at 2.66 GHz clock frequency with 1GB RAM.

5.1 Data Workload

In our experimentation we have used two different datasets. The first dataset describes publications' entries of the DBLP database[5] while the second one contains entries from a movies database[6] which describes films, actors, and remakes. These two datasets are characterized by a different degree of homogeneity, since the number of objectclasses and attributes used to describe publications is more limited compared to the one used in case of the movies, actors, and remakes. Moreover, the second dataset is of larger size and is used in order to examine the algorithm's scalability. In both cases data were retrieved in XML format and were converted to LDAP entries. We will refer to these two datasets as DBLP and movies dataset, respectively.

The DBLP dataset consists of about 10000 entries that correspond to 5 groups of around 2000 entries for each of the following categories: articles, inproceedings, masterthesis, phdthesis, and www. The LDAP schema has been configured in a way that each entry is described by an objectclass (e.g., article) and a set of attributes (e.g., author, title, pages). According to the definition of objectclasses, there are some attributes which are used in case of more than one objectclasses while there are others that are defined only for a specific objectclass. For example, the attribute "author" can be used to describe both an "article" and an "inprocceding" entry while the attribute "month" is used only in case of articles. The main characteristic of this dataset is that the involved entries share many common attributes. On the other hand, the movies dataset contains about 20000 entries which correspond to a set of 7000 actor, 12000 film, and 1300 remake entries. These entries are described using more different objectclasses and attributes. For example, for the description of actors entries we use the objectclasses "actor" and "award" with included attributes such as "stagename", "familyname", "awardtype", and "studio". The movies dataset is thus characterized by more heterogeneity compared to the DBLP dataset.

5.2 Clustering Results Using Different Dissimilarity Coefficients

In this first section of our experimentation we use the LP-LM algorithm with the different dissimilarity coefficients, discussed in Section 3.2, in order to evaluate how each of them affects the clustering in terms of the distinctiveness it offers, while measuring the dissimilarity between entries. This is achieved by varying the values of θ which defines how "tight" we want the clustering to be. The higher the values of θ (near 1) the easier it will be for the clustering to identify neighbors but this, however, could lead the algorithm to mistakenly consider two dissimilar entries as neighbors. On the other hand, lower values of θ (near 0) require that entries will be quite close (similar) in order to be characterized as neighbors.

[4]OpenLDAP: http://www.openldap.org.
[5]DBLP data: http://www.sigmod.org/dblp/db/index.html.
[6]Movies database: http://infolab.stanford.edu/pub/movies/doc.html.

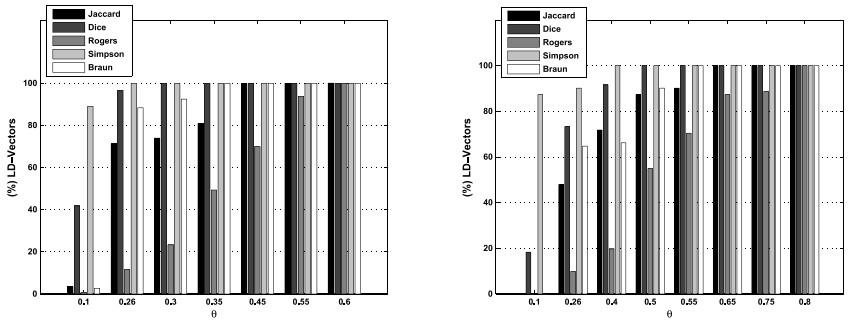
Table IV. Datasets Details

Dataset	Size	Classes	Number of LD-vectors	LD-vectors distribution
DBLP	10000	2000 articles 2000 inproceedings 2000 www 2000 masterthesis 2000 phdthesis	71	38 articles 13 inproceedings 10 www 7 masterthesis 3 phdthesis
movies dataset	20300	7000 actors 12000 movies 1300 remakes	146	124 actors 19 movies 3 remakes

The LP-LM is a structure-based algorithm which considers the LDAP entries structure as this is depicted in the extracted LD-vectors. The fact that LP-LM is applied on the set of derived LD-vectors and not the whole dataset reveals the way it addresses scalability issues. Therefore, we will mostly focus on the way the LD-vectors are assigned to clusters and then extend our discussion to the involved entries. After the preprocessing step, two sets of 71 and 146 distinct LD-vectors were created for the DBLP and movies datasets, respectively. More specifically, 38 of the 71 LD-vectors of the DBLP dataset corresponded to articles, 7 to masterthesis, 13 to inproceedings, 3 to phdthesis, and 10 to www. Thus, entries referring to articles differ more in their structure resulting in more LD-vectors, while phdthesis entries are represented by only 3 LD-vectors which means that there is not such a variance in their structure. Apparently, the number of the obtained LD-vectors is not indicative about the dissimilarity recorded between them. For example, most of the 38 articles' LD-vectors could differ only in one element, while the 3 LD-vectors of phdthesis could share only a few common attributes. In case of the movies dataset, the preprocessing resulted in 124 LD-vectors representing actors, 19 films, and 3 remakes. The number of actors LD-vectors is apparently overwhelming revealing the variety of elements that appear in the their entries. The details of both datasets and the extracted LD-vectors are summarized in Table IV.

Figure 16 depicts the percentages of the LD-vectors which are successfully assigned to clusters for the different θ values as well as dissimilarity coefficients. Our purpose is to examine which of the employed dissimilarity coefficients manages to successfully cluster the LD-vectors for the lower θ values which designates the required "tightness" in the clustering process. In Figure 16(a) we can see the assignments for the movies dataset. When the Simpson dissimilarity coefficient is used, the algorithm assigns successfully all the LD-vectors for the lower values of θ which is 0.26. Then, the Dice coefficient is the next dissimilarity coefficient that results in the appropriate clustering of the LD-vectors for θ = 0.3. For θ = 0.35, the Braun dissimilarity succeeds in the LD-vectors clustering while in case of Jaccard coefficient this happens for θ = 0.45. The Rogers dissimilarity coefficient needs the higher θ value to give the correct clustering assignment and this happens for θ = 0.6. Thus, the Simpson dissimilarity coefficient, when employed in the LP-LM algorithm, leads to the appropriate clustering assignment for the lower values of θ while the Rogers dissimilarity coefficient needs the highest θ values in order to achieve this goal.

The same conclusions were derived from the experimentation performed on the DBLP dataset. The results are depicted in Figure 16(b) where as we can see, the lower value of θ that results in all LD-vectors being correctly assigned is 0.4 and this happens in case of the Simpson coefficient. The next dissimilarity coefficient that needs



(a) movies dataset (b) DBLP dataset

Fig. 16. LD-vectors successfully assigned to clusters.

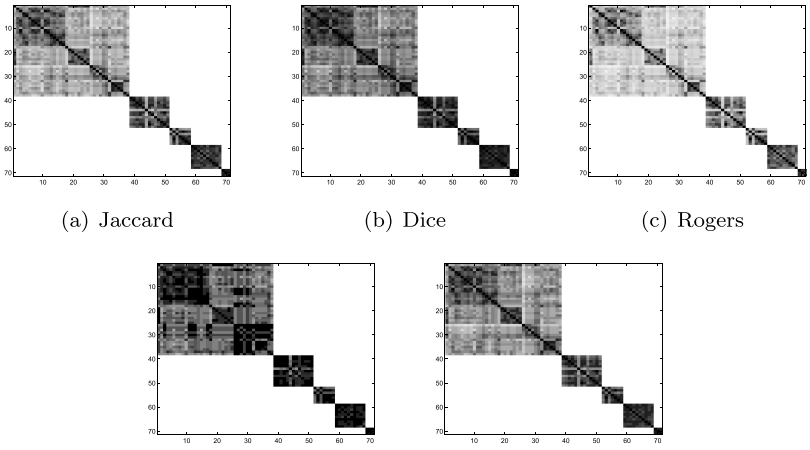
low θ values is Dice (θ = 0.5) and then follows Braun (θ = 0.55) and Jaccard (θ = 0.65). Rogers requires the highest θ value which is 0.8.

Considering the preceding results we observe that they are in accordance with the discussion on distance metrics performance that was presented in Section 3.2, where as it was noted, the various dissimilarity coefficients resulted in higher distance values in the order of Simpson, Dice, Braun, Jaccard, Rogers. Thus, our intuition outlined in Section 3.2 came true.

The performance of the LP-LM algorithm is apparently affected by the employed dissimilarity coefficient because the nature of each one of them may result in lower or higher dissimilarity values. For example, the Dice coefficient gives more gravity to the common attributes of two LDAP vectors compared to Jaccard and Rogers and that causes lower dissimilarity values. Moreover, both Simpson and Braun coefficients consider not only the common attributes that two LD-vectors share but also their size, that is, the number of attributes of the LD-Pairs. We have indicatively proceeded to the visualization of the clusters of DBLP dataset, in terms of their similarities and dissimilarities, in an attempt to represent graphically the way each of the dissimilarity coefficients "captures" relations between LD-vectors.

Figure 17 presents the similarities and dissimilarities between the LD-vectors for the different dissimilarity coefficients. Specifically, each plot depicts the distance table D of the 71 LD-vectors, whose rows and columns have been rearranged so that LD-vectors of the same cluster are put in consecutive rows (columns). Moreover, the darker the coloring of a cell (i,j), where $1 \leq i, j \leq 71$, the more similar the corresponding LD-vectors. Thus, given that clusters contain the most similar LD-vectors, the darker rectangles appear on the subplots diagonal and reveal the clusters of the DBLP dataset.

The discussion on the results of Figure 16(b) is in accordance with the similarities revealed by each coefficient visualized in Figure 17. The darker rectangles appear in case of the Simpson dissimilarity coefficient, indicating that when this coefficient is employed, a higher degree of similarity is identified among LD-vectors, and thus the algorithm succeeds for lower θ values. Regarding the rest of the coefficients, the rectangles that refer to the Dice coefficient are the darker ones and then comes those depicting similarities for the Braun and Jaccard coefficients. The lightest-colored rectangles appear in case of the Rogers dissimilarity coefficient, proving that in this case, higher θ values are required to obtain the correct clustering assignment. Moreover, the fact that the correct clustering is achieved to higher values of θ in case of the DBLP dataset, for each of the employed dissimilarity coefficients, shows that even though



(a) Jaccard (b) Dice (c) Rogers

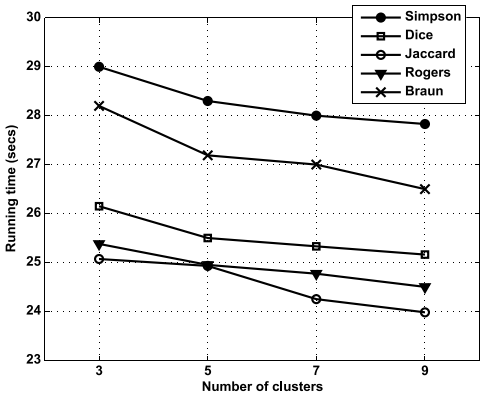
(d) Simpson (e) Braun

Fig. 17. The clustering outline of the DBLP dataset for $k = 5$.

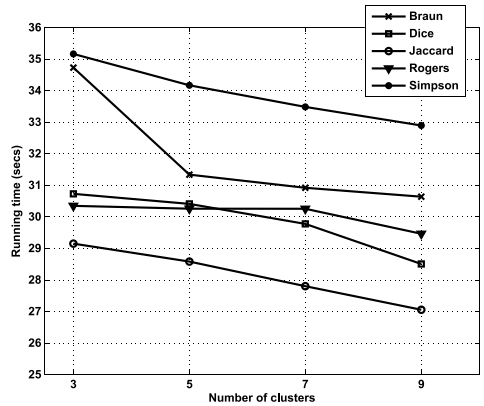
the number of the attributes that describe publication entries is smaller, the obtained LD-vectors are more different.

From the previous discussion it becomes evident that a dissimilarity coefficient guides the clustering process with the distinctiveness it provides and can contribute in good performance, which is signified by the successful assignment of LDAP entries to clusters. This can be achieved even when a "tight" clustering is required as defined by the θ value. According to the experimentation carried out on both datasets, the Simpson dissimilarity coefficient enables the algorithm to identify the underlying LD-vectors' relations and successfully cluster them, for the lowest θ value. Thus, the Simpson dissimilarity coefficient makes the algorithm more effective because setting lower θ values may, on the one hand, render the notion of neighbor "tighter" but on the other hand it provides more safety that the algorithm will manage to identify dissimilarities between LDAP entries. Higher values of θ create the risk that the algorithm may mistakenly consider two entries as neighbors. Consider the borderline case that θ is set to 1. Then, all entries are each other's neighbor. In general, the higher the values of θ the less importance we give to the entries' structural differences while lower θ values contribute to a more stern and accurate evaluation of entries' nearness.

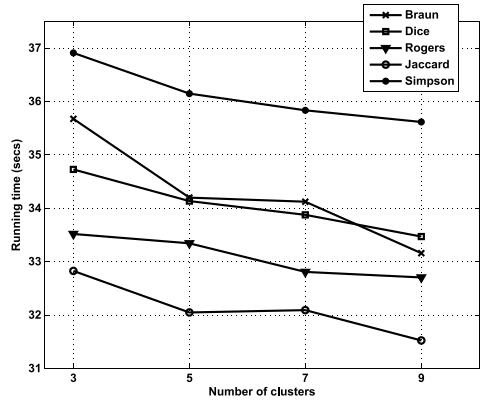
Moreover, we have evaluated the algorithm's running time for the various distances employed and the different number of clusters, while we have artificially extended our dataset in order to examine the scalability of the proposed approach. Specifically, from the original movies dataset with 20300 entries, we created two new datasets. The first one contains 50300 entries, that is, we added 30000 entries which are represented by 15 new and distinct LD-vectors. The second dataset contains 80300 entries, that is, the original dataset is extended with 60000 entries which correspond to 34 new LD-vectors. We consider as the algorithm's running time the interval between the formulation of LD-vectors and final clusters extraction, excluding, thus, the time it takes the algorithm to read LDAP entries. The obtained results for the movies dataset are depicted in Figures 18(a), 18(b), and 18(c) for the movies datatsets of 20300, 50300, and 80300 entries, respectively. As we can see, reducing the number of clusters causes more time for the algorithm to complete. This is expected since the LP-LM algorithm follows a hierarchical agglomerative process and needs more steps to result in smaller number of clusters. Furthermore, we see that when Jaccard, Dice, and Rogers distance



(a) 20300 entries



(b) 50300 entries



(c) 80300 entries

Fig. 18. LP-LM running time.

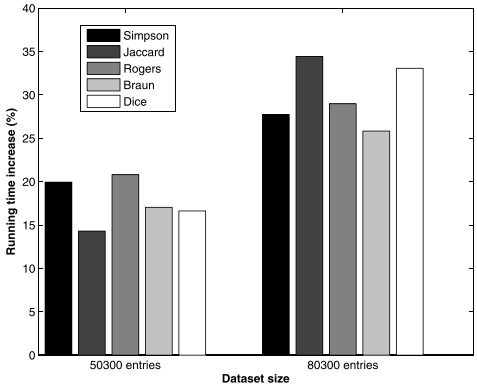


Fig. 19. Average running time increase.

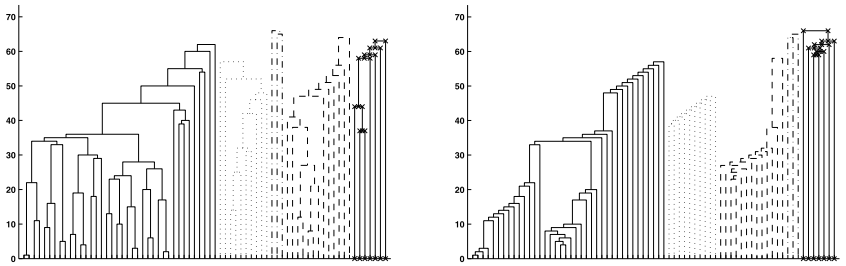
coefficients are employed, the algorithm needs less time to complete while in case of Braun and Simpson coefficients the execution time increases. This is due to the fact that the last two coefficients calculate distances between LD-vectors considering not only their common and different elements but also their length. Thus, in this case, some more computations need to take place resulting in some time expense.

Furthermore, we calculated the percentages of average deterioration in the algorithm's running time in case of the extended datasets and for the various distances. The results are depicted in Figure 19. As we can see, the depicted increases of the algorithm's running time varies depending on the dataset and the employed distance and correspond to a few seconds only (as we can see from Figure 18). This is due to the fact that after the LD-vectors' formation, the algorithm is based on them for the rest of the computations. Thus, the complexity is getting significantly lower. Considering, for example, the case of the dataset containing 80000 entries, the algorithm does not work with the 80300 entries but with the 180 LD-vectors that represent them. Moreover, the variation of the percentages of deterioration depend on the type of the new LD-vectors which are brought by the new entries, since LD-vectors with a higher number of attributes and objectclasses may result in more time-consuming calculations.

5.3 The Merging Steps Using the Expected and Relative Links

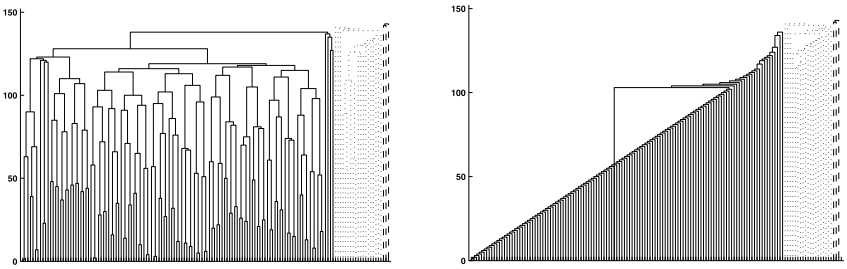
In this section of our experimentation we study the LP-LM algorithm's behavior when the two merging criteria, namely the Expected Link and the Relative Link are employed. As already discussed in Section 3.4.2, the Relative Link, contrary to the Expected Link, is based on quantities calculated during the execution of the algorithm and thus it is expected to have a better perspective on which clusters should be merged at each step. In order to compare these two merging criteria, we have indicatively proceeded to the graphical visualization of the successive mergings.

Figures 20 and 21 present the progress of the LP-LM agglomerative algorithm for both datasets when the Simpson dissimilarity coefficient was employed. Specifically, Figure 20 refers to the DBLP dataset where the 5 underlying clusters (articles, phdthesis, masterthesis, inproceedings, and www) are denoted using different colors. The dendrograms have been plotted in such a way that each point of the y-axis corresponds to a successive merging. According to Figure 20(a) which represents the merging steps in case the Relative Link is used, the algorithm usually chooses different pairs of clusters to merge at each step, resulting in smaller clusters which will at a later step be merged. On the contrary, using the Expected Link (Figure 20(b)) criterion the algorithm's behavior is different since, at each step, the cluster that was created from the



(a) using the *Relative Link* (b) using the *Expected Link*

Fig. 20. Clustering of the DBLP dataset.



(a) using the *Relative Link* (b) using the *Expected Link*

Fig. 21. Clustering of the movies dataset.

previous step is used for the next merging. This results to the creation of more unbalanced clusters.

Moreover, as depicted from the two former discussed subfigures, in case of the Expected Link criterion, the algorithm performs consecutive mergings of clusters that refer to the same data category. For example, it starts merging phdthesis (denoted by the magenta color), when the majority of the articles clusters (denoted by the blue color) have been merged. On the other hand, the usage of the Relative Link allows the algorithm to merge, at each step, pairs of clusters that belong to different categories.

The preceding observations also hold in case of the movies dataset. The progression of the algorithm in this case is depicted in Figure 21 where the Simpson dissimilarity coefficient has been used to capture dissimilarities between LDAP entries. Figure 21(a) shows the merging steps when the Relative Link was used while Figure 21(b) the respective progress for the Expected Link merging criterion. In both figures the 3 underlying clusters (actors, movies, remakes) are denoted by the different colors where the great number of LD-vectors corresponding to articles is verified.

As in case of the DBLP dataset, the LP-LM presents similar behavior when the two merging criteria are used. Here, it is even more obvious the fact that the usage of the Expected Link criterion makes the algorithm consider, as the basis for the next merging, the cluster that was obtained in the previous step. In case of the Relative Link, the algorithm still manages to create more balanced clusters at each step, as it does not always involve the cluster created at the previous phase. Moreover, the usage of the Relative Link makes the algorithm more flexible, in terms of merging clusters of different categories, whereas when the Expected Link is used, the algorithm proceeds to the clusters of another category when the majority of mergings that refer to the currently processed category finishes. In conclusion, the Relative Link has the

advantage over Expected Link, since, in cases the true number of clusters is not known, it results in a better quality and more balanced clustering scheme for the same value of k. This is important as usually the value of k is not known and thus, the Relative Link can conduce to improved clustering results.

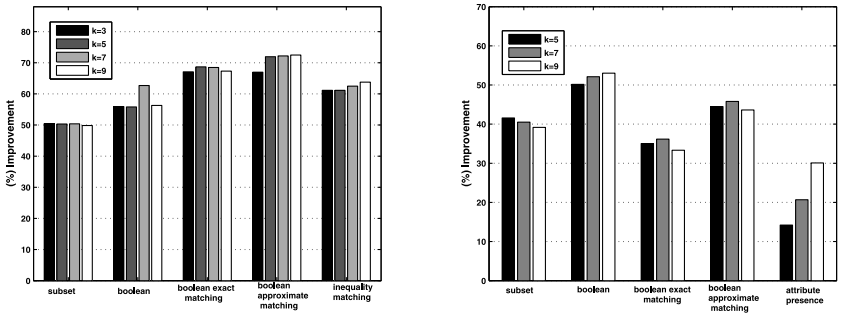
5.4 LDAP Server Performance under the Proposed Query Framework

Apart from the algorithm's efficiency it is important to study the impact of the clustered-based LDAP data organization on the system's performance. Here, we compare the response times of the LDAP server to a set of queries in case LDAP data was organized in flat DIT (unclustered scheme), and in case of a clustered DIT organization which is in accordance with the LP-LM algorithm's results. When the clustered LDAP scheme is employed, the queries are executed using the *LP and LM Query Engine*, described in Section 4.1, which determines the responses' pathways in the LDAP DIT, given the keywords extracted from the query. The queries that were executed can be categorized in the following categories.

— *Subset queries*. In this category, the queries retrieve subsets of entries providing no query filter. For example, a query of this category would ask for all articles or phdthesis.
— *Boolean queries*. Queries of this category contain boolean expressions without involving specific attributes values. For example, a boolean query would retrieve all entries that have a booktitle value but not an ISBN value or all phdthesis and masterthesis that have a volume value.
— *Boolean queries of exact matching*. These queries contain boolean expression and filters specifying an exact matching. For instance, a query of this category would request all inproceedings' entries of 2002.
— *Boolean queries of approximate matching*. This type of queries is similar to the Boolean queries of exact matching with the only difference that their query filter is approximate. For example, a query of this category would retrieve articles or phdthesis which contain the term "database".
— *Boolean queries of inequalities*. These queries contain boolean expressions and their filter is an inequality condition. A query that would ask for all authors that were born after 1970 would belong to this category.
— *Queries of attribute presence*. This category contains all queries that search for entries having as criterion the presence or absence of a specific attribute. For example, a query of this category would retrieve all entries that are described by a year value (e.g., year of publication, year of birth).

We have formed a set of queries which involved 20 queries per query category. The queries were defined in a way that they include every possible keyword (objectclass or attribute) and their combinations in order to result in more objective results. Moreover, in case of categories that involved a search filter the value of the filter was randomly selected from a set of values (e.g., a word contained in the title of a publication). For example, we have executed queries searching for movies whose author is Hitchcock, or actors who were born before 1980 and their origin is from USA, etc.

We have run the same set of queries for each of the aforementioned categories in the unclustered organization and in the clustered one, independently of the number of clusters that were obtained from the LP-LM algorithm (using the Dice dissimilarity coefficient). From the obtained response times we calculated the improvements, since for all query categories the response times were better in the clustering-based data organization. The obtained improvements were averaged and are depicted in Figure 22.



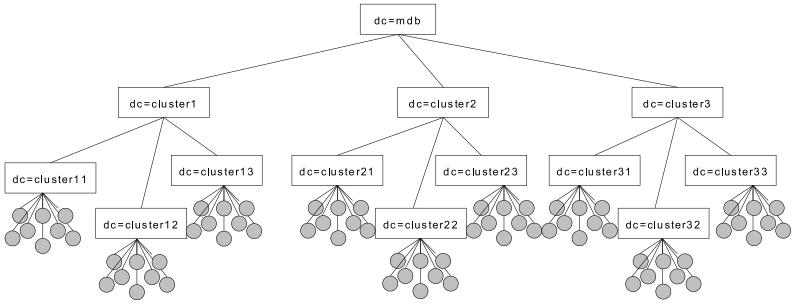
(a) movies dataset (b) DBLP dataset

Fig. 22. Improvements in the LDAP server's performance.

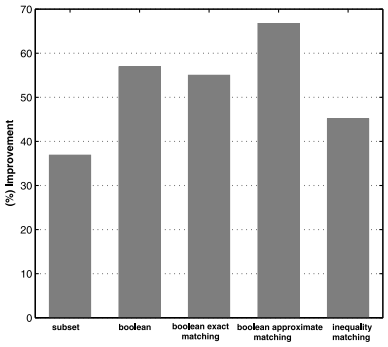
Figure 22(a) shows the improvements for the movies dataset, when the LDAP entries were organized according to the LP-LM algorithm's results for $k = 3, 5, 7, 9$, while Figure 22(b) presents the respective improvements fot the DBLP dataset and the same values of k. We have executed boolean inequality matching queries only on the movies dataset because it uses more attributes for which this type of query is meaningful (e.g., actor's date of birth, a film's cost or profit). On the other hand, we used queries of attributes presence only in case of the DBLP dataset, because its entries share many common attributes and the existence or absence of an attribute in the entries of the DBLP dataset is more important.

Furthermore, we evaluated the LDAP server's performance in case that we applied clustering in two levels of hierarchy, as discussed in Section 4. We indicatively present results for the movies dataset. LDAP entries were initially clustered into $k = 3$ clusters, and then, the LP-LM clustering algorithm was applied on each cluster to obtain $k = 3$ clusters. LDAP entries were organized in the DIT depicted in Figure 23(a). The same set of queries was executed and the obtained average improvements per category are presented in Figure 23(b). The improvements recorded in Figures 22 and 23 may differentiate for the same query because the obtained clusters are not the same. For example, for the results of Figure 22(a), LDAP data was organized in 9 clusters which are distributed to 4 clusters of actors, 2 clusters of movies, and 3 clusters of remakes. On the other hand, for the DIT organization of Figure 23(a) the 9 clusters correspond to 3 clusters of each of the former categories. Thus, a question addresses a cluster that exists in one of the above organization schemes, then it will result in greater improvements.

As it is clearly depicted in all of the mentioned subfigures, when a clustering-based LDAP organization is used, there is improvement in all types of queries. The recorded improvement depends on the search space defined by the query keywords which is in turn based on the keywords' distinctiveness. For example, a query asking for a common attribute such as the author of a publication or a film's actors results in lower improvements than a query asking for a publications's series or an actor's origin, which normally exists in a smaller subset of entries. If the clustering algorithm manages to identify an attribute's uniqueness and create a cluster containing the entries that contain it, then the search space will be significantly reduced and will result in considerably greater improvements. Moreover, the fact that the entries of the DBLP dataset share more common attributes is depicted on the average improvements which are higher for the movies dataset. An homogeneous dataset, which is described by a lower number of attributes, is more possible to result in less search space reduction and thus in lower response times improvements. To examine the effect in the response



(a) DIT organization

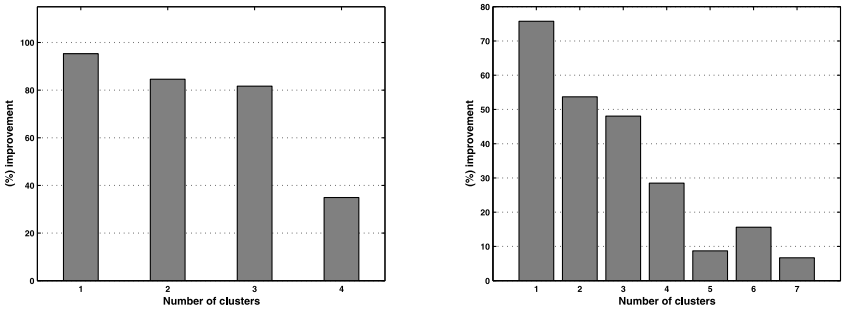


(b) improvements in the LDAP server's performance

Fig. 23. LDAP server's performance under a more hierarchical DIT.

time improvements of the search space as defined by the number of clusters, we have executed a set of queries whose responses exist in different number of clusters. The results are depicted in Figure 24 and refer to the clustering-based LDAP organization for $k = 9$ (Dice coefficient was employed for the DBLP dataset and Simpson for the movies dataset). More specifically, we have executed the same set of queries in case LDAP data were organized in a flat DIT and in case the DIT was adjusted to the clustering results. The improvements depicted in Figure 24(a) and Figure 24(b) refer to those queries (each bar is for one query) whose answer lies on $1-4$ or $1-7$ clusters, respectively. Thus, from the whole set of executed queries we chose those whose answers lie in different number of clusters in order to construct the different bars.

Figure 24(a) presents the improvements for the movies dataset while Figure 24(b) the respective improvements for the DBLP dataset. The homogeneity of the DBLP dataset is also proved by the fact that there are attributes that exist the maximum in 7 of the 9 requested clusters while in case of the movies dataset this happens for the maximum 4 of the 9 clusters. As we can see for both subfigures, the obtained improvements tend towards decreasing as the number of clusters that contain the requested keywords increases. This, however, may not hold in cases where an attribute exists in less clusters, but these clusters contain many entries. Such a case applies in the DBLP dataset for $k = 5$ and $k = 6$. The entries that contain the "ee" attribute are located in $k = 5$ clusters while the "volume" attribute exists in entries of $k = 6$ clusters. However, the calculated improvements are higher for the volume "attributes"



(a) movies dataset (b) DBLP dataset

Fig. 24. Response time improvements as a function of the search space (clusters).

because the total number of entries containing it is higher than the respective number of entries containing the "ee" attribute.

The experimentation results of this section show that a query framework that takes advantage of the clustering-based LDAP data organization results in the enhancement of the LDAP server's performance in search operations. The response time improvements are noticeable even in case of common attributes but they are remarkable when attributes appearing in restricted publication types are involved.

6. CONCLUSIONS-FUTURE WORK

In this article we perform a thorough study of the LPAIR-LMERGE hierarchical structure-based clustering algorithm which is proposed as a method for the appropriate LDAP Directory Information Tree definition. The LP-LM algorithm is used in conjunction with various distance metrics in order to examine the distinctiveness they offer in cases that a "tight" clustering is requested. Moreover, the LDAP server that adjusts its data organization to the clustering results presents improved response times, under a specific query framework implemented by the proposed LP and LM Query Engine. The recorded improvements are especially high in case of queries containing keywords that correspond to distinctive clusters. This is mostly observed in cases of heterogeneous datasets where there is a significant variance on the attributes used to describe LDAP entries. Finally, we propose the usage of the Relative Link as an alternative merging criterion which is based on quantities calculated during the execution process. The experimental results prove that using the Relative Link merging criterion, the algorithm results in more balanced clusters.

Our next step is to incorporate the knowledge about the content of the entries, that the LD-vectors describe, in the clustering process. Our purpose is to result in a more information enriched process and a deeper organization hierarchy which can result in further improvements of the LDAP server performance.

REFERENCES

AMER-YAHIA, S. AND SRIVASTAVA, D. 2004. Distributed evaluation of network directory queries. *IEEE Trans. Knowl. Data Engin. 16,* 4, 474–486.

CARTER, G. 2003. *LDAP System Administration*. OŔeilly.

CHADWICK, D. 1994. *Understanding X. 500: The Directory*. Chapman & Hall.

CHADWICK, D. 2003. Deficiencies in ldap when used to support pki. *Comm. ACM. 46*, 3, 99–104.

CRABTREE, D., GAO, X., AND ANDREAE, P. 2005. Improving Web clustering by cluster selection. *Proceedings on the IEEE/WIC/ACM International Conference on Web Intelligence*. 172–178.

CRABTREE, D., ANDREAE, P., AND GAO X. 2006. Query directed Web page clustering. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*. 202–210.

DIETZOLD, S. 2005. Generating rdf Models from ldap Directories. In *Proceedings of the SFSW'05 Workshop on Scripting for the Semantic Web*.

DIETZOLD, S. AND AUER, S. 2007a. Accessing rdf knowledge bases via ldap clients. In *Proceedings of 7th International Conference on Knowledge Management (I-KNOW'07)*.

DIETZOLD, S. AND AUER, S. 2007b. Integrating sparql endpoints into directory services. In *Proceedings of the ESWC'07 Workshop on Scripting for the Semantic Web*.

FAN, Q., WU, Q., HE, Y., AND HUANG, J. 2005. Optimized strategies of grid information services. In *Proceedings of the 1st International Conference on Semantics, Knowledge, and Grid*. ACM, 90.

GEMMILL, J., CHATTERJEE, S., MILLER, T., AND VERHAREN, E. 2003. ViDe.net middleware for scalable video services for research and higher education, vide.net middleware for scalable video services for research and higher education. In *Proceedings of the ACM Southeastern Conference*. ACM.

GUHA, S., RASTOGI, R., AND SHIM, K. 1999. Rock: A robust clustering algorithm for categorical attributes. In *Proceedings of the 15th International Conference on Data Engineering*. IEEE Computer Society. 512–521.

HARANCZYK, M. AND HOLLIDAY, J. 2008. Comparison of similarity coefficients for clustering and compound selection. *J. Chem. Inf. Model. 48*, 3, 498–508.

HOHN, M. 2005. Binary coefficients: A theoretical and empirical study. *Math. Geol. 8*, 2, 137–150.

HOU, H., WANG, X., AND WU, M. 2006. Hierarchical byzantine fault tolerant secure ldap. *IEEE International Conference on Systems, Man and Cybernetics*. IEEE Computer Society, 3844–3849.

HOWES, T. AND SMITH, M. 1997. *LDAP: Programming Directory-Enabled Applications with Lightweight Directory Access Protocol*. Macmillan Technical Publishing.

HU, H. AND DU, X. 2006. An ontology learning model in grid information services. In *Proceedings of the 1st International Conference on Innovative Computing, Information and Control*. IEEE Computer Society, 398–401.

KAPITSKAIA, O., NG, R., AND SRIVASTAVA, D. 2000. Evolution and revolutions in ldap directory caches. In *Proceedings of the 7th International Conference on Extending Database Technology: Advances in Database Technology*. Springer, 202–216.

KARYPIS, G., HAN, E., AND KUMAR, V. 1999. Chameleon: Hierarchical clustering using dynamic modeling. *IEEE Comput. 32*, 8, 68–75.

KOUTSONIKOLA, V. AND VAKALI, A. 2004. Ldap: Framework, practices, and trends. *IEEE Internet Comput. 8*, 5, 66–72.

KOUTSONIKOLA, V. AND VAKALI, A. 2008. *XML and LDAP Integration: Issues and Trends*. Vol. Open and Novel Issues in XML Database Applications: Future Directions and Advanced Technologies. IGI Global, Chapter II.

KOUTSONIKOLA, V., VAKALI, A., MPALASAS, A., AND VALAVANIS, M. 2008. A structure-based clustering on ldap directory information. In *Proceedings of the 17th International Symposium on Methodologies for Intelligent Systems*. Springer.

KUMAR, A. AND GUPTA, R. 2003. Edge caching for directory based Web applications: Algorithms and performance. In *Proceedings of the 8th International Workshop Web Content Caching and Distribution*. Kluwer, 39–56.

LAUKKANEN, M., VILJANEN, K., APIOLA, M., LINDGREN, P., MAKELA, E., SAARELA, S., AND HYVONEN, E. 2004. Towards semantic Web-based yellow page directory services. In *Proceedings of the 3rd International Semantic Web Conference*. Springer.

LI, T. 2005. A general model for clustering binary data. In *Proceeding of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*. ACM, 188–197.

LIAN, W., CHEUNG, D., MAMOULIS, N., AND YIU, S.-M. 2004. An efficient and scalable algorithm for clustering xml documents by structure. *IEEE Trans. Knowl. Data Engin. 16*, 1, 82–96.

LIANG, J., VAISHNAVI, V., AND VANDENBERG, A. 2006. Clustering of ldap directory schemas to facilitate information resources interoperability across organizations. *IEEE Trans. Syst. Man Cybernet. 36*, 4, 631–642.

LIM, S., CHOI, J., AND ZEILENGA, K. 2005. Design and implementation of ldap component matching for flexible and secure certificate access in PKI. In *Proceedings of the 4th Annual PKI R&D Workshop*. NIST Technical Publication.

MAASS, H. 1997. Location-Aware mobile applications based on directory services. In *Proceedings of the 3rd Annual ACM/IEEE International Conference on Mobile Computing and Networking*. ACM, 23–33.

MARON, P. AND LAUSEN, G. 2001. *HLCaches: An LDAP-Based Distributed Cache Technology for XML*. Albert-Ludwigs University at Freiburg.

MURGUIA, M. AND VILLASENOR, J. L. 2003. Estimating the effect of the similarity coefficient and the cluster algorithm on biogeographic classifications. *Ann. Bot. Fennici 40*, 6, 415–421.

PARK, J., SANDHU, R., AND AHN, G.-J. 2001. Role-Based access control on the Web. *ACM Trans. Inf. Syst. Secur. 4*, 1, 37–71.

PONARAMENKO, J., BOURNE, P., AND SHINDYALOV, I. 2002. Building an automated classification of dna-binding protein domains. *Bioinf. 18*, 2, S192–S201.

RODRIQUEZ, C. 2007. siledap: Easing Interactions with Directories. *Proceedings of TERENA Networking Conference*.

VAKALI, A., POKORNY, J., AND DALAMAGAS, T. 2004. An overview of Web data clustering practices. In *Proceedings of the 9th International Conference on Extending Database Technology (EDBT'04)*. Springer, 597–606.

WANG, X., SCHULZRINNE, H., KANDLUR, D., AND VERMA, D. 2008. Measurement and analysis of ldap performance. *IEEE/ACM Trans. Netw. 16*, 1, 232–243.

WANG, Y. AND KITSUREGAWA, M. 2001. Use link-based clustering to improve Web search results. In *Proceedings of the 2nd International Conference on Web Information Systems Engineering*. IEEE Computer Society, 115–124.

WHAL, M., HOWES, T., AND KILLE, S. 1997. Lightweight directory access protocol (v3). IETF RFC 2251.

WU, J., LEANGSUKSUN, C., AND RAMPURE, V. 2006. Policy-Based access control framework for grid computing. In *Proceedings of the 6th IEEE International Symposium on Cluster Computing and the Grid*. IEEE Computer Society, 391–394.

ZENG, H.-J., CHEN, Z., AND MA, W.-Y. 2002. A unified framework for clustering heterogeneous Web objects. In *Proceedings of the 3rd International Conference on Web Information Systems Engineering*. IEEE Computer Society, 161–170.

Received May 2008; revised July 2010; accepted September 2010