A Structure-Based Clustering on LDAP Directory Information

Vassiliki Koutsonikola, Athena Vakali,
Antonios Mpalasas, and Michael Valavanis

Department of Informatics
Aristotle University
54124 Thessaloniki, Greece
{vkoutson,avakali,antoniom,mvalavan}@csd.auth.gr

Abstract. LDAP directories have rapidly emerged as the essential framework for storing a wide range of heterogeneous information under various applications and services. Increasing amounts of information are being stored in LDAP directories imposing the need for efficient data organization and retrieval. In this paper, we propose the LPAIR & LMERGE (LP-LM) hierarchical agglomerative clustering algorithm for improving LDAP data organization. LP-LM merges a pair of clusters at each step, considering the *LD-vectors*, which represent the entries' structure. The clustering-based LDAP data organization enhances LDAP server's response times, under a specific query framework.

1 Introduction

Directory services provide a generic and appropriate framework for accessing a variety of information. They act as database repositories ensuring more efficient data retrieval mechanisms through the usage of Lightweight Directory Access Protocol (LDAP)[15]. LDAP is an open industry standard that gains wide acceptance due to its flexibility and the fact that it integrates with an increasing number of data retrieval and management applications [6].

To date, there are multiple applications that rely on LDAP servers. Most of the operating LDAP-based servers store information that describe user profiles and address books for messaging applications, configuration files of network devices and network security policies, under the Directory Enabled Networks (DEN) initiative [5]. Directory servers are also used to store certificates and revocation lists for PKI applications [1] as well as access control lists for authentication systems [13]. The new H.350 standard uses LDAP to provide a uniform way to store information related to video and voice over IP (VoIP) in directories [3]. Moreover, Grid computing has emerged as a very promising infrastructure for distributed computing, having its foundation and core on the distributed LDAP directories [2].

Due to the heterogeneity of LDAP data, finding appropriate data organization schemes such as clustering will ensure LDAP servers' functionality and performance. Earlier research efforts have focused either on proposing application-oriented LDAP schema definitions [12], or on introducing pure caching [7] and

A. An et al. (Eds.): ISMIS 2008, LNAI 4994, pp. 121–130, 2008.

indexing [8] approaches that can improve performance and scalability of directory based services. However, a framework that will propose a well-distributed and scalable LDAP data organization, regardless of the underlying application, enhancing at the same time system's performance, is necessary.

In this paper we propose the LPAIR & LMERGE (LP-LM) algorithm, an agglomerative structure-based clustering algorithm, for LDAP data organization. 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 [11]. Our work applies clustering analysis on LDAP entries and uses clustering results in order to define the LDAP data arrangement. More specifically, our main contributions can be summarized as follows:

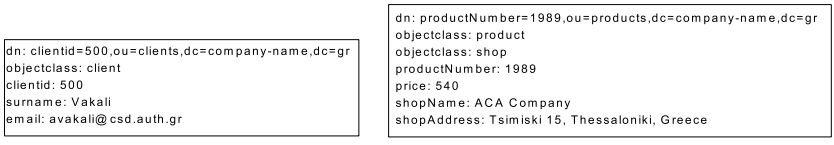
- We introduce the notion of LD-vectors to capture LDAP entries' structure
- We propose the structure-based LPAIR & LMERGE clustering algorithm which organizes LDAP data, regardless of the underlying applications.
- We carry out experiments to evaluate the LP-LM's efficiency as well as LDAP server's performance that adjusts its data organization to clustering results.

The rest of the paper is organized as follows: Section 2 discusses some basic concepts of LDAP data representation and the introduced LD-vector structures. Section 3 describes our problem formulation and the proposed LDAP clustering algorithm. Section 4 presents the experimentation while conclusions and future work insights are given in Section 5.

2 LDAP Background and Data Representation

Data is stored in LDAP directories in the form of entries arranged in hierarchical information trees. Each LDAP entry is identified by a distinguished name (DN) that declares its position in the hierarchy. The hierarchy's structure forms the directory information tree (DIT), which originates from a root (RootDN). In the basic LDAP notation, "dc" stands for domain component and "ou" for organizational unit. For example, the RootDN of the DIT that maintains clients' and products' data for a Greek company would be "dc=company-name, dc=gr", while the DN of the clients' and products' nodes would be "ou=clients, dc=company-name, dc=gr" and "ou=products, dc=company-name, dc=gr" respectively.

All information within a directory entry is stored as attribute-value pairs. The set of attributes that can appear in a given entry is determined by the objectclasses that are used to describe it. The definition of an objectclass specifies that some attributes may be mandatory while others optional. For example, the user defined objectclass "client", which can be used to describe a company's clients, would consider as mandatory the "surname" and "clientid" attributes while it would define as optional the "email" attribute. Moreover, the user defined "product" objectclass may consider as mandatory the "productNumber" and "price"



(a) A client's entry (b) A product's entry

Fig. 1. LDAP entries LDIF

attributes, while the objectclass "shop", the "shopName" and "shopAddress" attributes that refer to the shop that the described product exists. Figures 1(a) and 1(b) present the LDIF[1] of a sample client and product entry respectively.

The set of rules that define the objectclasses and the attributes they contain constitutes the LDAP schema. LDAP allows the definition of new objectclasses and attributes while it supports objectclasses inheritance and thus new objectclasses can be created to extend existing ones. In each case, the LDAP schema defines whether there is relation between pairs of objectclasses or objectlass-attribute pairs.

In this paper, we consider a particular framework where we have as source a set $E = \{e_1, \ldots, e_f\}$ of f LDAP entries. 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. As discussed above, the LDAP schema defines the related objectclasses pairs (due to inheritance) or the related objectlass-attribute pairs.

Definition 1 (LD-Pair of an entry)**.** *Given an LDAP entry $e_i \in E$, the **LD-Pair** of e_i is a set of pairs $LDP(e_i) = \{(d_x, d_y)\} : d_x \in O, d_y \in \{O \cup A\}, d_x \neq d_y$, 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_i.*

Example 1. Consider the two LDAP entries depicted in Figure 1. The LD-Pair for the client entry denoted as e_{client} is $LDP(e_{client})$={(client, clientid), (client, surname), (client, email)} while the LD-Pair for the product entry denoted as $e_{product}$ is $LDP(e_{product})$={(product, productNumber), (product,price), (shop, shopName), (shop, shopAddress)}. □

The definition of the entry's LD-Pair can also be extended to a set of entries.

Definition 2 (LD-Pair of an entries' set)**.** *Given a set $E^* \subseteqq E$ of f^* LDAP entries where $f^* \leq f$, the $LDP(E^*)$ is defined as $LDP(E^*) = \{\cup LDP(e_i)\}$, $\forall e_i \in E^*$.*

Example 2. We consider the set $E^* = \{e_{client}, e_{product}\}$ and the $LDP(e_{client})$, $LDP(e_{product})$ as discussed in the Example 1. According to Definition 2,

[1] LDIF (LDAP Data Interchange Format) is a standard for representing LDAP entries in human readable format.

$LDP(E^*)$= {(client, clientid), (client, surname), (client, email), (product, productNumber), (product,price), (shop, shopName), (shop, shopAddress)}. □

Next, we use the LD-Pair concept to define a vector data structure which represents the LDAP entries' structure.

Definition 3 (LD-VECTOR)**.** *Given the E^* set of f^* LDAP entries and its $LDP(E^*)$, we use l to denote the number of (d_x, d_y) pairs where $(d_x, d_y) \in LDP(E^*)$. Then $\forall e_i \in E^*$ we define the binary multivariate* ***LD-vector*** *$LDV(e_i,:)$ of l values as follows:*

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

Example 3. Given the set $E^* = \{e_{client}, e_{product}\}$ and its $LDP(E^*)$, then based on Defintion 3 the LD-vectors of e_{client} and $e_{product}$ are $LDV(e_{client})$=[1 1 1 0 0 0 0] and $LDV(e_{product})$=[0 0 0 1 1 1 1], respectively. □

3 LDAP Data Structure-Based Clustering

The binary nature of LD-vectors identifies them as categorical data. According to [4],[10], pure distance-based clustering algorithms may not be as effective on categorical data as link-based ones. The proposed clustering algorithm adopts the link-based perspective which groups entries in terms of their common neighbors, as expressed by the link value.

3.1 Problem Formulation

The representation of LDAP entries as binary vectors makes dissimilarity coefficients an appropriate choice for measuring distance between them. We choose Czekanowski (or Dice) dissimilarity coefficient as distance measure, instead of popular Jaccard, to give more gravity to the elements that two entries have in common. Given two binary vectors $LDV(e_i,:)$ and $LDV(e_j,:)$ of length l where $e_i, e_j \in E, i \neq j$, their distance $D(e_i, e_j)$ in terms of Czekanowski coefficient is defined as:

$$D(e_i, e_j) = \frac{b + c}{2a + b + c}$$

where, for $1 \leq t \leq l$, $a = |t| : \{LDV(e_i, t) = LDV(e_j, t) = 1\}$, $b = |t| : \{LDV(e_i, t) = 1 \wedge LDV(e_j, t) = 0\}$, $c = |t| : \{LDV(e_i, t) = 0 \wedge LDV(e_j, t) = 1\}$.

The values of D range between 0 and 1 with higher values indicating higher dissimilarity between the involved entries. The Czekanowski dissimilarity coefficient has been used to capture distances in various clustering approaches [14],[9].

Two entries are considered to be neighbors if their distance 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 $LN(e_i) = \{\cup e_j\} : D(e_i, e_j) \leq \theta, \forall e_i, e_j \in E$. Moreover,

for each of the entries belonging to one of the obtained clusters we compute the expected number of its neighbors as follows: Let C_i denote the $i-th$ of the k obtained clusters of size c_i. When $\theta = 0$, each entry belonging to a cluster C_i is expected to have only 1 neighbor, itself, while for $\theta = 1$ any other entry belonging to C_i is neighbor of e_i (their distance is always ≤ 1), resulting in c_i neighbors. For any other value of θ, where $0 < \theta < 1$, it is expected that higher values of θ will result in more neighbors of e_i. A quantity that applies to the above situation and can express the expected number of neighbors for an e_i entry is given by $c_i^{\frac{2\theta}{1+\theta}}$.

Furthermore, the link between two entries e_i and e_j expresses the number of their common neighbors and is calculated by $link(e_i, e_j) =\mid \mathrm{LN}(e_i) \cap \mathrm{LN}(e_j) \mid \forall e_i, e_j \in E$. Given the expected number of neighbors for each e_i entry in cluster C_i, the e_i contributes to a link value equal to $c_i^{\frac{4\theta}{1+\theta}}$ (one for each pair of its neighbors)[2]. Then, the total number of expected links in C_i cluster caused by all c_i in number entries will be $c_i^{1+\frac{4\theta}{1+\theta}}$.

The link-based clustering approach aims at maximizing the link between each pair of entries belonging to a single cluster. According to the Definition 3, there may be a set of LDAP entries that are represented by the same LD-vector, which our structure-based clustering algorithm will assign to the same cluster (due to their common structure). This could lead to unbalanced cluster and thus, inspired by [4], we define a criterion function $J(E)$ where the total number of links between a cluster's entries is divided by the expected link value for this cluster weighed by the the number of its entries.

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

Our goal is to maximize the link value of entries contained in a cluster. Therefore, we define the LDAP Clustering problem as follows:

Problem 1 (LDAP Clustering). Given a set E of f LDAP entries, an 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.

3.2 The LPAIR & LMERGE (LP-LM) Clustering Algorithm

The proposed LPAIR & LMERGE (LP-LM) algorithm is a hierarchical agglomerative clustering algorithm aiming at finding a solution to Problem 1. Since the goal of LP-LM is the maximization of the criterion function $J(E)$ (Equation 1),

[2] The total number of pairs that have e_i as neighbor is given by $2\binom{c_i^{\frac{2\theta}{1+\theta}}}{2} + c_i^{\frac{2\theta}{1+\theta}} = c_i^{\frac{4\theta}{1+\theta}}$ (each pair is measured twice, e.g. (e_x,e_y) and (e_y,e_x) while each entry e_x forms the (e_x,e_x) pair).

we need to specify the best pair of clusters to be merged at each step of the algorithm.

According to Equation 1, 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 defined as $link(C_i, C_j) = \sum_{e_x \in C_i, e_y \in C_j} link(e_i, e_j)$. Similarly to the definition of $J(E)$, in order to prevent the continuous merging of large-size clusters, we divide the $link(C_i, C_j)$ with an expected link value between C_i and C_j clusters. To compute the expected link value between two clusters we need to calculate the total link value of the two clusters 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. $c_j^{1+\frac{4\theta}{1+\theta}}$). We use this normalization factor as a heuristic to steer towards the maximization of the criterion function value. Therefore, the *merging criterion* $mc(C_i, C_j)$ of clusters C_i and C_j is defined as:

$$mc(C_i, C_j) = \frac{link(C_i, C_j)}{(c_i + c_j)^{1+\frac{4\theta}{1+\theta}} - c_i^{1+\frac{4\theta}{1+\theta}} - c_j^{1+\frac{4\theta}{1+\theta}}} \tag{2}$$

The pair of clusters that maximizes mc will be merged at each algorithm's step.

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 (distance threshold), $0 \leq \theta \leq 1$ and results in the assignment of LDAP entries to the k clusters.

Algorithm 1. The LPAIR & LMERGE algorithm

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

Output: Assignment of the LDAP entries in the k clusters, such that J is maximised.

```
1: /*Preprocessing*/
2: LDP = CreateLDPair(E)
3: LDV = CreateLDVectors(LDP)
4: D = ComputeDistance(LDV)
5: LN = ComputeNeighbors(D, θ)
6: link = ComputeLink(LN)
7: /*Clustering process*/
8: while NumClusters ≥ k do
9:    (C1, C2) = FindMergingClusters(link, mc)
10:   C* = merge(C1, C2)
11:   update(link, C*)
12: end while
```

Initially in LP-LM, a preprocessing takes place, where given the initial set E of LDAP entries the algorithm computes the entries' LD-Pairs (line 2) and then the respective LD-vectors (line 3). Using the Czekanowski coefficient (Section 3.1), the table D of 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 for the computation of the link value for each pair of entries, resulting in table *link* (line 6). After the preprocessing step, an iterative process follows which constitutes the main clustering process. This process lasts until k clusters are obtained (line 8). During each iteration [3] of this step, the LP-LM algorithm finds the best pair of clusters (C_1,C_2) to me merged according to the mc values (line 9). The two clusters C_1 and C_2 are merged and a new C^* cluster is obtained (line 10). The table *link* is updated (line 11) with the new link values in terms of C^* cluster.

4 Experimentation

Our data consists of around 10000 entries that describe DBLP data[4]. The DBLP dataset contains about 2000 entries for each of the following publication categories: articles, inproceedings, masterthesis, phdthesis and www. The LDAP schema involves a set of objectclasses (e.g. article, phdthesis) and a set of attributes (e.g. author, title, pages). For the experiments we have used the OpenLDAP directory server with Berkeley DB backend, setting cache size to zero (to obtain unbiased results) and having the "publicationid" attribute indexed.

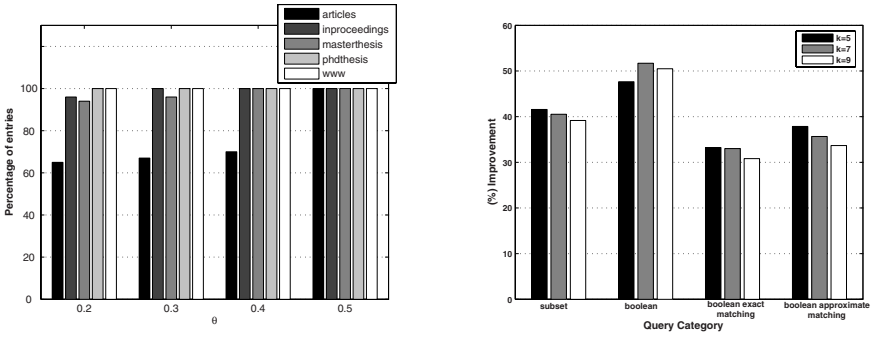
In the first section of our experimentation we study the LP-LM algorithm's performance for different values of θ and $k = 5$, given that our entries belong to 5 different categories, and we calculate the percentage of successfully clustered entries per category. The results for $\theta = 0.2, 0.3, 0.4$ and 0.5 are depicted in Figure 2(a). Lower values of θ demand higher similarity between two entries in order to be considered as neighbors, resulting in lower percentages of entries successfully assigned to clusters. Increasing the values of θ, the LP-LM manages to cluster successfully more entries while for $\theta = 0.5$ (and $\theta > 0.5$) there is no missclustered entry. The proper clustering is achieved for $\theta = 0.5$ and not for a lower value, because our dataset is described by a large number of distinct LD-vectors (e.g. there are 38 different LD-vectors capturing articles' structure). A higher number of LD-vectors indicates more dissimilar in structure LDAP entries and demands a higher value of θ to result in the proper clustering.

Moreover, as depicted in Figure 2(a), articles can not comprise a cluster for $\theta < 0.5$ because of the greater dissimilarity that exists between them compared to the other categories' entries. Furthermore, entries of both inproceedings and masterthesis categories can not be successfully clustered for low values of θ, eventhough lower percentages of their entries compared to articles, are not assigned properly. The LP-LM algorithmn assigns www and phdthesis entries correctly, even for low θ values. In all cases, the overall calculated percentage of successfully assigned entries is over 90% which proves the efficiency of the LP-LM algorithm.

In the second section of our experimentation we evaluate an LDAP server's performance, comparing its response times to a set of queries, in case of an un-

[3] In the first iteration, the pair of clusters is a pair of entries.

[4] DBLP data: http://www.sigmod.org/dblp/db/index.html The DBLP data were retrieved in XML format and converted to LDAP entries.



(a) Entries successfully assigned to clusters per category

(b) LDAP server's response time improvements

Fig. 2. Experimentation results

clustered data organization (flat DIT) and a clustered one, as indicated by the LP-LM clustering algorithm, for $\theta = 0.5$ and $k = 5, 7, 9$. In order to benefit from the data organization proposed, we apply a query processing which directs users' queries to specific LDAP data clusters. The idea behind the query processing operation is that, based on the users' queries keywords (expressed by the contained attributes and objectclasses) and a mapping scheme (which reveals the relations between clusters and sets of objectclasses and attributes), the responses' pathways in the LDAP DIT are determined. In the worst case scenario, the query's keywords will be found in all clusters and the search will have to start by the RootDN, resulting in response time equal to that of the unclustered scheme. In any other case, the search space would be reduced, resulting in improved response times. We have examined LDAP server's performance running a set of queries that belong to the following four categories:

- *Subset queries*: Queries that retrieve subsets of entries providing no query filter (e.g. all articles, all inproceedings).
- *Boolean queries*: Queries containing boolean expressions without involving specific attributes value (e.g. entries that have a booktitle but not an ISBN).
- *Boolean queries of exact matching*: Queries containing boolean expressions and filters specifying an exact matching (e.g. all inproceedings, www of 2002).
- *Boolean queries of approximate matching*: Boolean queries with approximate matching filters (e.g. articles, www containing in title the term "database").

It should be noted, that in all cases we executed the same set of queries for each category, and calculated the difference in response times. 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 2(b).

In case of "subset queries" we observe the same levels of improvements for the different values of k because the search space is reduced equally regarding the

clustered data. For instance, in an unclustered data organization, a query looking for all phdthesis, must search all 10000 entries in order to retrieve them while in the clustering-based data organization, the query processing locates the one cluster containing the phdthesis, reducing the search space to 2000 entries. For $k = 7$, the LP-LM has created one cluster for each of the articles, inproceedings, masterthesis and www entries while the phdthesis have been distributed in 3 clusters. The search space for phdthesis remains to 2000 entries while the transmission between clusters causes negligible delay to the response time. For $k = 9$, the LP-LM creates one cluster for each of the masterthesis and www categories, 3 clusters for phdthesis, 2 clusters for articles and 2 clusters for inproceedings without affecting the search space of the "subset queries".

The improvement observed in case of "boolean queries" depends significantly on the query and the obtained clusters. For example, a query asking for "ee" values of phdthesis will be directed to the one of the 3 phdthesis clusters which is the only one containing entries with "ee" attribute, reducing significantly the search space and resulting in high improvements of about 78% (for $k = 7, 9$). On the other hand, asking for "volume" values results in 14% improvement (for all k values), since "volume" is a common attribute contained in entries that belong to all categories except for masterthesis.

Similarly, for the boolean queries of exact and approximate matching, the obtained improvement is dependent of the query's keywords and the way they restrict the search space. For example, a query requesting articles that contain in their title the word "database" resulted in 75% improvement while a query retrieving articles, phdthesis, www and inproceedings published in one of the IEEE journals led to 15% improvement.

The above indicative (due to the lack of space) discussion clearly shows that the clustering-based data organization yields enhancement of LDAP server's performance, in terms of all LDAP query types. The recorded improvements depend on the search space defined by the query keywords (e.g. www or inproceedings) as well as the distinctiveness of attributes. The response time improvements are noticeable even in case of common attributes (e.g. "title") but they are remarkable when attributes appearing in restricted publication types (e.g. "ee") are involved. Moreover, the discussed improvements refer to a rather homogeneous dataset since publications are not described by a considerable variety of attributes and objectclasses. An LDAP server storing more heterogeneous data is expected to be more benefitted from the proposed clustering approach.

5 Conclusions and Future Work

This paper presents a structure-based clustering algorithm which is used to define the organization of the LDAP Data Information Tree. The proposed LPAIR & LMERGE (LP-LM) algorithm has been proved to perform efficiently in case of LDAP data described by different sets of objectclasses and attributes. Moreover, the LDAP server that adjusts its data organization to the clustering results presents improved response times, under a specific query framework. The

recorded improvements are especially high in case of queries containing keywords that correspond to distinctive clusters. Therefore, the LP-LM algorithm can be particularly beneficial for an LDAP server storing various information such as multimedia, network device configuration files and user profiles, enhancing the performance of the underlying applications.

For the future, we plan to incorporate knowledge about data's content to the overall clustering process and experiment with more distance metrics.

References

1. Chadwick, D.: Deficiencies in LDAP when used to support PKI. Communications of the ACM 46, 99–104 (2003)
2. Fan, Q., Wu, Q., He, Y., Huang, J.: Optimized Strategies of Grid Information Services. In: Proc. of the First Int. Conf. on Semantics, Knowledge, and Grid, p. 90 (2005)
3. Gemmill, J., Chatterjee, S., Miller, T., Verharen, E.: ViDe.Net Middleware for Scalable Video Services for Research and Higher Education. In: ACM Southeastern Conf. GA. ACM 1-58113-675-7/030/03, pp. 463–468 (2003)
4. Guha, S., Rastogi, R., Shim, K.: ROCK: A Robust Clustering Algorithm For Categorical Attributes. In: Proc. 15th Int. Conf. Data Eng., pp. 512–521 (1999)
5. Howes, T., Smith, M.: LDAP: Programming Directory-Enabled Applications with Lightweight Directory Access Protocol. Macmillan Technical Publishing, Basingstoke (1997)
6. Koutsonikola, V., Vakali, A.: LDAP: Framework, Practices, and Trends. IEEE Internet Computing 8, 66–72 (2004)
7. Kumar, A., Gupta, R.: Edge Caching for Directory Based Web Applications: Algorithms and Performance. In: Proc. of the 8th international workshop in Web content caching and distribution, pp. 39–56 (2004)
8. Lee, H., Mun, S.-G., Huh, E.-N., Choo, H.: Efficient Data Indexing System Based on OpenLDAP in Data Grid. In: Int. Conf. on Computational Science, vol. 1, pp. 960–964 (2006)
9. Li, T.: A Unified View on Clustering Binary Data. Machine Learning 62, 199–215 (2006)
10. Lian, W., Cheung, D., Mamoulis, N., Yiu, S.-M.: An Efficient and Scalable Algorithm for Clustering XML Documents by Structure. IEEE Trans. on Knowledge and Data Engineering 16, 82–96 (2004)
11. Liang, J., Vaishnavi, V., Vandenberg, A.: Clustering of LDAP directory schemas to facilitate information resources interoperability across organizations. IEEE Trans. on Systems, Man and Cybernetics, Part A 36, 631–642 (2006)
12. Lim, S., Choi, J., Zeilenga, K.: Design and Implementation of LDAP Component Matching for Flexible and Secure Certificate Access in PKI. In: Proc. of the 4th Annual PKI R&D Workshop, pp. 41–51 (2005)
13. Park, J., Sandhu, R., Ahn, G.-J.: Role-based access control on the web. ACM Trans. on Information and System Security (TISSEC) 4, 37–71 (2001)
14. Ponaramenko, J., Bourne, P., Shindyalov, I.: Building an Automated Classification of DNA-binding Protein Domains. Bioinformatics 18, S192–S201 (2002)
15. Whal, M., Howes, T., Kille, S.: Lightweight Directory Access Protocol (v3). IETF RFC 2251 (1997)