Collaborative Clustering of XML Documents

Sergio Greco, Francesco Gullo, Giovanni Ponti, Andrea Tagarelli

Dept. of Electronics, Computer and Systems Sciences (DEIS), University of Calabria
Via P. Bucci, 41C, 87036 Arcavacata di Rende (CS), Italy

Abstract

Clustering XML documents is extensively used to organize large collections of XML documents in groups that are coherent according to structure and/or content features. The growing availability of distributed XML sources and the variety of high-demand environments raise the need for clustering approaches that can exploit distributed processing techniques. Nevertheless, existing methods for clustering XML documents are designed to work in a centralized way.

In this paper, we address the problem of clustering XML documents in a collaborative distributed framework. XML documents are first decomposed based on semantically cohesive subtrees, then modeled as transactional data that embed both XML structure and content information. The proposed clustering framework employs a centroid-based partitional clustering method that has been developed for a peer-to-peer network. Each peer in the network is allowed to compute a local clustering solution over its own data, and to exchange its cluster representatives with other peers. The exchanged representatives are used to compute representatives for the global clustering solution in a collaborative way. We evaluated effectiveness and efficiency of our approach on real XML document collections varying the number of peers. Results have shown that major advantages with respect to the corresponding centralized clustering setting are obtained in terms of runtime behavior, although clustering solutions can still be accurate with a moderately low number of nodes in the network. Moreover, the collaborativeness characteristic of our approach has revealed to be a convenient feature in distributed clustering as found in a comparative evaluation with a distributed non-collaborative clustering method.

Keywords: collaborative distributed clustering, XML, P2P networks, XML structure and content information, transactional data

Email addresses: greco@deis.unical.it (Sergio Greco), fgullo@deis.unical.it (Francesco Gullo), gponti@deis.unical.it (Giovanni Ponti), tagarelli@deis.unical.it (Andrea Tagarelli)

1. Introduction

The clustering problem is central in data management as it refers to unsupervised learning of the inherent structure of relationships in the data. The discovered relationships are expressed as a set of groups, or clusters, where data objects within the same cluster are similar to each other while dissimilar from objects in different clusters.

Text databases represent a fruitful research area in data clustering. With the growing availability of large document collections, there has been an increasing demand for fast and accurate organization of such data. In the last years, research on *document clustering* has focused on the development of approaches and methods that aim to address the special requirements for clustering large document collections, such as high dimensionality, ease for browsing, meaningfulness of cluster descriptions [26, 36, 12, 30, 21]. Moreover, text data available from most informative sources, primarily over the Web but also in digital libraries and scientific repositories, have a semistructured nature. As the connection point between the natural language text and the rigidly structured tuples of typed data, semistructured text data enables the modeling of complex real-world objects and their relationships.

Within this view, XML has become the preeminent way for effectively representing such data, thanks to its extensible markups and document type descriptors. As a meta-language for markup, XML allows the definition of customized tags describing the data enclosed by them. This flexibility in the XML syntax simplifies the deployment of arbitrary languages for domain-specific markup: just to mention a few domain applications, XML has been used in Web content syndication and rendering, multimedia and networking, scientific data and literature, business processes and data exchange.

Particularly, the importance of XML is becoming more and more evident in high-demand environments, in which clustering large document sets is challenging as it has to face tight requirements on both processing power and space resources. One example is represented by some Web news services that need to apply clustering algorithms to articles in XML format spanning over thousands of news sources with a frequency of few minutes. In this case, a distributed clustering approach would divide the capability of processing over several nodes as opposed to concentrating performance on a single workstation, where traditional centralized approaches would fail since transferring all data to a central clustering service is prohibitive in large-scale systems. In the case of news articles, the objective of a clustering system would be most likely to discover groups of articles discussing similar contents, regardless of their structures. Actually, many other real scenarios are even more complicated since XML documents from heterogeneous sources are typically organized using different logical structures, and this also holds for sources that may have similar contents. As an example, consider users in a peer-to-peer network who want to share information about software encoded in XML format, such as software name, developers' name, latest release date, platform, license, reviews and ratings. All such structural fields would be encoded using the markup vocabulary authored by any spe-

cific source, and contents might be interleaved with structure in different ways. For instance, using a text-centric representation approach, an XML document might contain the full descriptions of the various reviews, including ratings, in repeated occurrences of the element `review`. By contrast, using a data-centric representation approach, a different XML document may follow a more complex substructure rooted in the element `reviews` which would include a number of sub-elements containing a short description for each of selected aspects relevant to the review (e.g., positive and negative comments, rating, recommendation). In such a scenario, the partial matchings between different structures (and their combinations with text values) could be identified based on an XML similarity detection approach properly devised for taking into account heterogeneous structure as well content information. This way, users would be allowed to easily access an integrated and more complete information, hence to extract interesting knowledge patterns.

The scenario described above is just one among the many existing in XML distributed applications, which range from scientific literature and data to personal profiles, from book or music reviews to product documentations. As a matter of fact, XML is being extensively used in peer-to-peer (P2P) networks [19, 1, 28, 5], due to the natural combination of a standard way for representing and exchanging information with a technology for sharing and locating distributed data which has proven to enable innovative services [29]. However, despite this synergistic coupling of XML and P2P networks, existing methods for clustering XML data are designed to work only on a centralized environment. This partly depends on an inherent difficulty in devising representation models of both XML structure and content information that are able to effectively support summarization of XML data, thus favouring the development of clustering methods that maintain feasibility in large-scale systems. Moreover, most clustering strategies cannot easily be distributed, since there is an additional level of complexity due to the design and implementation of scalable and effective protocols for communication that allow nodes to minimize exchanged data. In this respect, a related issue concerns the type and the form of the information that need to be selected and exchanged among the nodes, which impacts on the significance of the obtained clustering solutions.

Contribution

Our proposal is focused on the development of a distributed framework for efficiently clustering XML documents. Assuming the distributed environment as a P2P network, the underlying idea is to enable each node in the network to access a portion of a given document collection and to communicate with all the other nodes to perform a clustering task in a collaborative fashion. To the best of our knowledge, we bring for the first time the problem of collaborative distributed clustering in the XML data domain.

The proposed framework borrows the approach to modeling and clustering XML documents from our earlier works [33, 32]. Following the lead of these works, XML documents are transformed into *transactional data* based on the

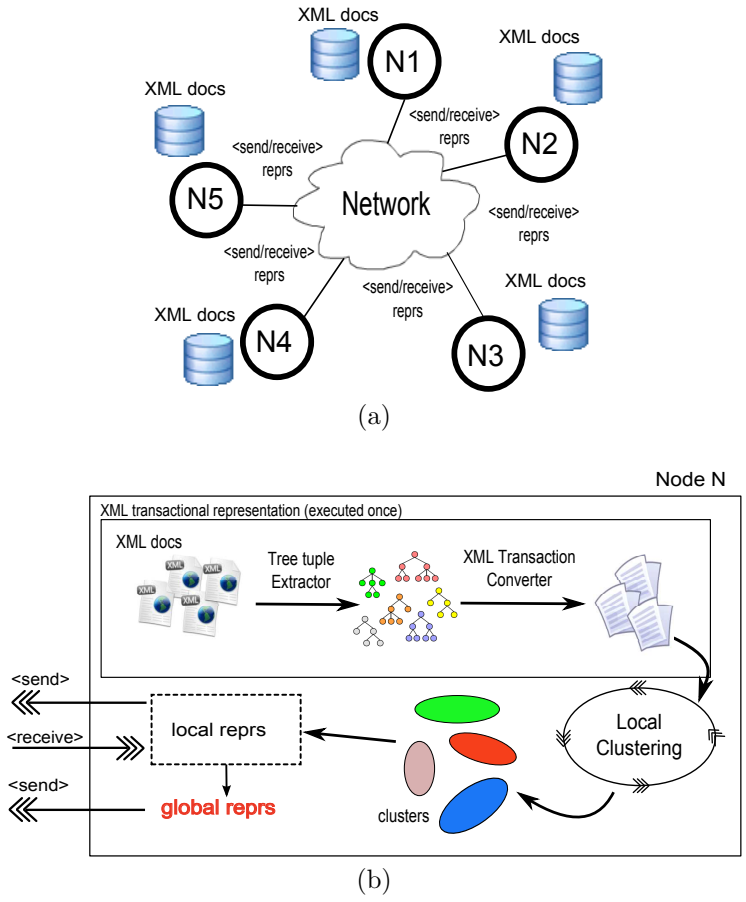


Figure 1: Overview of the collaborative distributed clustering of XML documents: (a) an example XML distributed environment as a P2P network, and (b) a schematization of data flows in a single node of the network

notion of *tree tuple*. XML tree tuples enable a flat, relational-like XML representation that is well-suited to meet the requirements for clustering XML documents according to structure and content information.

We resort to the well-known paradigm of *centroid-based partitional clustering* [17] to conceive our distributed, transactional clustering framework. It should be emphasized that such a clustering paradigm is particularly appealing to a distributed environment. Indeed, the availability of a summarized description of the clustered data provided by the cluster representatives is highly desirable especially when the input data is spread across different peers. Cluster representatives are hence used to describe portions of the document collection and can conveniently be exchanged with other nodes on the network.

Figure 1 provides an overview of our collaborative distributed clustering framework. A number of XML information sources is spread over a P2P network

(Figure 1(a)). Each node in the network has its own local XML repository and communicates with the other nodes sending and receiving summarized information about the local clustering process (i.e., cluster representatives). Figure 1(b) shows the main processes involved in each single node. A preprocessing phase produces a transactional representation of the local XML documents based on tree tuples. At each iteration of the collaborative algorithm, each node yields a local clustering solution (i.e., a partition of its own set of XML data). For each local cluster, the corresponding (local) representative is obtained and sent out to nodes that are in charge of computing the "global" representatives. More precisely, every node computes a subset of the global representatives; the i-th node computing the global representative for a set of clusters, receives from each other node the representative for the corresponding local cluster. Once computed, the global representatives are finally sent back to all the nodes to update their local clusters.

Major features and advantages offered by our approach can be summarized as follows:

- High level of resource distribution — our approach is totally distributed since both data and (clustering) processes are distributed over several nodes.
- Collaborativeness — as typical in a P2P network, whose main strength is its independence of dedicate infrastructure and centralized control [29], collaborativeness leads to a distributed environment that presents several advantages mainly in terms of: reliability (no centralized index server needs to be maintained), resource sharing (i.e., every node locally shares its resources and administrates its client-server environment), efficiency and effectiveness (i.e., processing power increases as demands increase, and transmission rate is higher than a client-server network since resources can be made available from multiple nodes connected to each other as peers).
- Limited network load — our notion of XML cluster representative is well-suited for representing structure and content information in XML data, and ensures an efficient exchange of information.
- Ease of implementation — the logics adopted by every node for processing the information exchanged with other nodes is simple. This allows an easy implementation of the processes performed by nodes: indeed, through the definition of global representatives, this logics exploits the repeated application of a procedure very similar to that used for summarizing information exchanged among the nodes (local representatives).

We conducted experiments on large, real-world collections of XML documents, which are particularly suitable for assessing the ability of the proposed framework in performing collaborative clustering of XML documents by structure and content. Documents in each of these collections were distributed over a P2P network, where the number of peers was varied. Results have shown that, although the final clustering accuracy is typically reduced with respect to

the centralized case, the parallelism due to a relatively small number of collaborating nodes in the network leads to a drastic reduction of the overall runtime needed for the clustering task. Besides this major strength with respect to a centralized clustering solution, further experiments have unveiled a remarkable beneficial impact of the collaborativeness feature of our approach compared to a non-collaborative distributed clustering method.

Plan of the paper

The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 provides preliminaries for XML transactional representation, including the notions of tree tuple and transaction for the XML data domain. Section 4 describes our XML transactional similarity measure and collaborative clustering algorithm. Section 5 reports experimental evaluation on the framework from both effectiveness and efficiency viewpoint. Finally, Section 6 presents concluding remarks and pointers for future research.

2. Related Work

Distributed XML data management has received an increasing attention in the last few years. As usual in emerging database applications, early proposals have been developed in order to primarily enact efficient distributed query processing and optimization. In [2], the Active XML (AXML) language, a logical language based on the embedding of service calls within XML documents, is extended to enable the declarative specification and deployment of XML applications across distributed complex processes. In [5], the distribution of XML documents across a P2P network is exploited to speed-up query processing. In contrast to related research studies (e.g., [28]), a clustering-based distribution scheme is designed to ensure a more homogeneous assignment of documents to peers, according to the clusters identified in the set of stored XML documents.

The attractiveness of publishing information sources in XML for organizations that want to easily interoperate has also fed an increasing interest in developing solutions for collaborative creation and editing of XML documents. For instance, in [16], an approach for the reconciliation of XML documents in a decentralized P2P environment is presented. In such environment, users can work off-line on their document versions and, as they reconnect to the network, synchronize their changes with other users. Consistency over the concurrent edits on XML documents is maintained by merging XML structures using a tombstone operational transformation based approach.

As previously mentioned, current methods for clustering XML documents are designed to offer centralized solutions. In this respect, a first problem arises in the definition of an XML representation model that is able to effectively handle both structure and content information in XML data. Representing semistructured and XML data has been traditionally addressed by labeled rooted trees. Consequently, dealing with such data has leveraged results from research on tree matching, including a number of algorithms for computing tree edit distances

(e.g., [25]). However, due to complexity issues, edit distance based approaches are infeasible for large data collections. To overcome these issues, summarization models have been proposed in order to (i) concisely represent XML data while preserving some structural relationships between XML elements, and (ii) fast compute XML similarity thus making it efficient in case of large-scaled XML documents [4, 24, 23, 9, 27]. In [23], an efficient graph-based summarization model, called s-graph, defines a concise XML representation that can be generalized to sets (clusters) of XML documents; however, the s-graph model may incur loose-grained similarity, as two documents may share the same s-graph prototype and still have significant structural differences (e.g., hierarchical relationships between elements). For instance, in [24], a compact structure is introduced to summarize the distinct nodes at each level of an XML document, and a notion of structural match between elements is defined according to the level information of each tree object. Like [24], the summarized structure proposed in [4] is organized as a vector of levels as well, although it considers the distinct edges at each level of an XML document. Moreover, it is also able to preserve the structural relationships between nodes of consecutive levels in the form of edge lists, which is in principle useful for distinguishing between semantically/structurally different XML documents.

A different category is provided by subtree mining and matching algorithms [3, 20]. Such algorithms exploit a tree representation of XML documents, however they compute XML similarity in terms of coverage of frequent substructures (e.g., subtrees, paths) at a specified support level, instead of calculating the tree edit distance between any pair of XML documents. The complexity of algorithms that belong to this category lies on the complexity of mining the frequent sub structures, which might turn out to be inefficient in case of large XML document sets.

The development of vector-space models to represent XML data has also attracted great attention, especially in XML information retrieval contexts [34, 13, 35, 8]. In [35], feature generation concerns properties on the paths, such as the path length, the root node label, and the number of path nodes. In [8], XML documents are transformed into sets of attribute-values according to various tree relationships among the document nodes, such as parent-child and next-sibling relationships, and path occurrences. In [13], both the XML element names and their text content values texts are taken into account to form two distinct feature sets. XML documents are hence represented based on these feature sets and the K-means algorithm is applied. $\Longrightarrow$**chiarire DL06**$\Longleftarrow$ Another $\Longrightarrow$**???**$\Longleftarrow$ hybrid clustering algorithm is described in [34], where the content features are modeled as a vector of terms weighted by their frequency within documents, and the structural features are modeled as a vector of distinct complete paths weighted by their appearance within documents. The Euclidean distance is used to compute the dissimilarity between the corresponding vector representations of the XML documents.

In our earlier works [33, 32], we originally introduced an XML representation model that allows for mapping XML document trees into *transactional data*. In a generic application domain, a transaction dataset is a multi-set of variable-

length sequences of objects with categorical attributes; in the XML domain, we devise a transaction as a set of items, each of which embeds a distinct combination of structure and content features from the original XML data. Within this view, XML documents are not directly transformed to transactional data, rather they are initially decomposed on the basis of the notion of *tree tuple*. Intuitively, given any XML document, a tree tuple is a tree representation of a complete set of distinct concepts that are correlated according to the structure semantics of the original document tree. Tree tuples extracted from the same tree maintain similar or identical structure while reflect different ways of associating content with structure as they can be naturally inferred from the original tree.

Traditional clustering techniques assume data is memory-resident. However, this assumption does not hold in many large-scale systems. In this respect, the development of clustering methods in parallel and distributed environments is becoming important since clustering and, in general, data mining tasks often require huge amounts of resources in storage space and computation time. Moreover, data is often inherently distributed into several databases, making a centralized analysis of such data inefficient and prone to security risks.

Among the few proposals to parallelize the clustering process in generic data domains [22], a parallel implementation of the K-means clustering algorithm is defined in [11], where a multi-processor architecture is assumed based on the message passing paradigm. Each processor has its own local memory, while the access to the other processes' memory is ensured by exploiting a standardized Message Passing Interface (MPI) library. The dataset is partitioned into equal-sized blocks among the processes. After selecting k objects as initial cluster representatives, each process carries out the basic K-means procedure to cluster its local objects. At the end of each iteration, processes exchange and sum up the local Sum of Squared-Errors (SSEs) to obtain the global SSE and compute new cluster representatives. The algorithm stops when the global SSE does not change in the next iteration.

One of the earliest studies on distributed data mining is proposed in [18], where an agent-based architecture is defined in such a way that each agent has a local model of the world and agents cooperate to improve solutions. The problem of document clustering in a distributed peer-to-peer network has been addressed recently. For instance, in [14], the significance of centroid-based partitional clustering like K-means is leveraged as an efficient approach to distributed clustering of documents. In [15], the authors originally propose a collaborative approach to distributed clustering of unstructured documents. The key idea underlying that work is to improve the local clustering solutions by exploiting the distributed environment on the basis of recommendations exchanged by the various peers. Also, document cluster summaries are modeled in form of keyphrases. Our work shares with [15] the adoption of a collaborative approach to distributed document clustering. However, our work is significantly different in that:

- XML documents are far more complex than structure-free texts, since

the property of being content-bearing belongs to textual elements that are interleaved with (and contextually dependent on) logical structure tags; this requires a data representation model capable of embedding both structure and content information.

- XML cluster summarization needs to go beyond the relative simple extraction of representative key-phrases that belong to plain documents within each cluster; our XML cluster summaries are defined as cluster representative transactions, which are conceived to contain highly representative items of structure and content information present in the within-cluster XML document set.

- XML information exchanged among peers is not supplied in the form of recommendations, but in a simpler way that exploits the definition of "meta-representatives" for the computation of the global clustering solution.

3. XML Transactional Representation

3.1. Preliminaries on XML trees and paths

A *tree* T is a tuple $T = \langle r_T, N_T, E_T, \lambda_T \rangle$, where $N_T \subseteq \mathbb{N}$ denotes the set of nodes, $r_T \in N_T$ is the distinguished root of T, $E_T \subseteq N_T \times N_T$ denotes the (acyclic) set of edges, and $\lambda_T : N_T \mapsto \Sigma$ is a function associating a node with a label in the alphabet Σ. Let Tag, Att, and Str be alphabets of tag names, attribute names, and strings respectively. An *XML tree* XT is a pair $XT = \langle T, \delta \rangle$, such that: *i)* T is a tree defined on the alphabet $\Sigma = Tag \cup Att \cup \{\mathtt{S}\}$, where symbol $\mathtt{S} \notin Tag \cup Att$ is used to denote the `#PCDATA` content model; *ii)* given $n \in N_T$, $\lambda_T(n) \in Att \cup \{\mathtt{S}\} \Leftrightarrow n \in Leaves(T)$; *iii)* $\delta : Leaves(T) \mapsto Str$ is a function associating a string to a leaf node of T.

An *XML path* p is a sequence $p = s_1.s_2 \ldots .s_m$ of symbols in $Tag \cup Att \cup \{\mathtt{S}\}$. Symbol s_1 denotes the tag name of the document root element. An XML path can be categorized into two types: *tag path*, if $s_m \in Tag$, or *complete path*, if $s_m \in Att \cup \{\mathtt{S}\}$. We denote by $\mathcal{P}_{XT}$ the set of all the complete paths in XT and by $\mathcal{TP}_{XT}$ the set of all the maximal tag paths in XT, i.e., $\mathcal{TP}_{XT} = \{s_1 \ldots .s_{m-1} | s_1 \ldots .s_{m-1}.s_m \in \mathcal{P}_{XT}\}$. The length of the longest path in $\mathcal{P}_{XT}$ determines the depth of XT, denoted as $depth(XT)$.

Let $XT = \langle T, \delta \rangle$ be an XML tree, and $p = s_1.s_2 \ldots .s_m$ be an XML path. The application of p to XT identifies the set $p(XT) = \{n_1, \ldots, n_h\}$ of all nodes such that, for each $i \in [1..h]$, there exists a sequence of nodes, or *node path*, $np_i^p = [n_{i_1}, \ldots, n_{i_m}]$ with the following properties: $n_{i_1} = r_T$ and $n_{i_m} = n_i$; $n_{i_{j+1}}$ is a child of n_{i_j}, for each $j \in [1..m\text{-}1]$; and, $\lambda(n_{i_j}) = s_j$, for each $j \in [1..m]$.

The application of a given path to an XML tree is called *answer*. Formally, given an XML tree XT and a path p, the answer of p on XT is defined as either $\mathcal{A}_{XT}(p) \equiv p(XT)$ (i.e., the set of node identifiers $p(XT)$) if p is a tag path, or $\mathcal{A}_{XT}(p) = \{\delta(n) \mid n \in p(XT)\}$ (i.e., the set of string values associated to the leaf nodes identified by p) if p is a complete path.

3.2. XML tree tuples

Tree tuple resembles the notion of tuple in relational databases and has been proposed to extend functional dependencies to the XML setting [6]. In a relational database, a tuple is a function assigning each attribute with a value from the corresponding domain. Given an XML tree XT, a maximal subtree of XT is an *XML tree tuple* τ if the answer of each (tag or complete) path p in XT on τ has size not greater than 1, i.e., $|\mathcal{A}_\tau(p)| \leq 1$.

We hereinafter denote the set of tree tuples from any given tree XT as $\mathcal{T}_{XT}$, and the set of tree tuples from the tree collection $\mathcal{XT}$ simply as $\mathcal{T}$. Also, following the notation introduced in Section 3.1, we use $\mathcal{P}_\tau$ to denote the set of all the complete paths in a tree tuple τ.

Example 1. Figure 2(a) shows a simplified XML document (from the DBLP archive) concerning two conference papers. Such a document is graphically represented by the XML tree in Figure 2(b). In the tree, any internal node has a unique label denoting a tag name. Each leaf node corresponds to either an attribute or `#PCDATA` content, and is labeled with either name and value of the attribute, or symbol `S` and the string corresponding to `#PCDATA`. As examples of path answers, (tag) path dblp.inproceedings.title yields the set of node identifiers $\{n_8, n_{20}\}$, whereas (complete) path dblp.inproceedings.author.S yields the set of strings {'M. J. Zaki', 'C. C. Aggarwal'}.

As shown in Figure 3, three tree tuples can be extracted from the tree of Figure 2(b). One tree tuple is from the right subtree rooted in the dblp element (Figure 3(c)). Two distinct tree tuples are extracted from the left subtree rooted in dblp, as in this subtree there are two paths dblp.inproceedings.author, each of which yields a distinct path answer corresponding to one author of a paper. Suppose now that node n_3 is pruned from the subtree of Figure 3(a): in this case, the resulting tree is no more a tree tuple as it is not a maximal subtree.

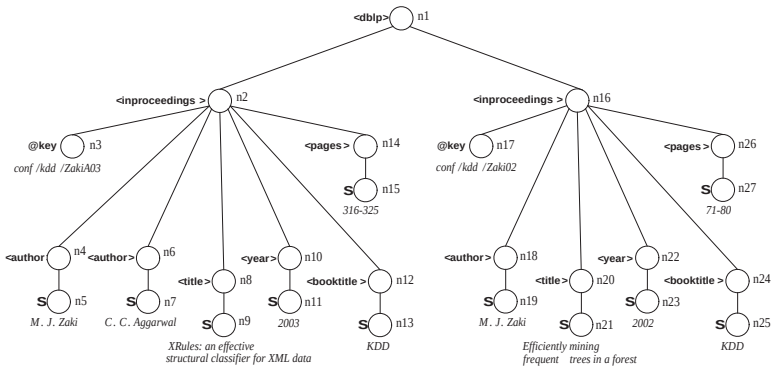
3.3. A transactional model for XML tree tuples

In the generic categorical domain, a transactional dataset is a multi-set of transactions over a set of categorical values, or items. In our XML setting, the item set is built over all the leaf elements in a given collection of XML tree tuples, hence it corresponds to the set of answers of complete paths applied to the tree tuples. A transaction is then modeled with the set of items associated to the leaf elements of any specific tree tuple. Formally, given an XML tree tuple τ, the *XML transaction* corresponding to τ is the set $\mathcal{I}_\tau = \{\langle p, \mathcal{A}_\tau(p)\rangle \mid p \in \mathcal{P}_\tau\}$, where each pair $\langle p, \mathcal{A}_\tau(p)\rangle$ is referred to as an *XML tree tuple item*. The rationale behind this model is that each path applied to a tree tuple yields a unique answer, thus each item in a transaction indicates information on a concept that is distinct from that of other items in the same transaction. We also denote with $\mathcal{S}$ the *XML transaction set* for a given collection $\mathcal{XT}$ of XML trees, which is defined as $\mathcal{S} = \bigcup_{XT \in \mathcal{XT}} \mathcal{S}_{XT}$, where $\mathcal{S}_{XT} = \{\mathcal{I}_\tau \mid \tau \in \mathcal{T}_{XT}\}$.

Example 2. In order to model XML tree tuples as transactions, we can decompose each tree tuple into its distinct paths and respective answers, as shown in

```
<dblp>
  <inproceedings key="conf/kdd/ZakiA03">
    <author>M. J. Zaki</author>
    <author>C. C. Aggarwal</author>
    <title>XRules: an effective structural
      classifier for XML
    </title>
    <pages>316-325</pages>
    <year>2003</year>
    <booktitle>KDD</booktitle>
  </inproceedings>
  <inproceedings key="conf/kdd/Zaki02">
    <author>M. J. Zaki</author>
    <title>Efficiently mining
      frequent trees in a forest
    </title>
    <pages>71-80</pages>
    <year>2002</year>
    <booktitle>KDD</booktitle>
  </inproceedings>
</dblp>
```

(a)

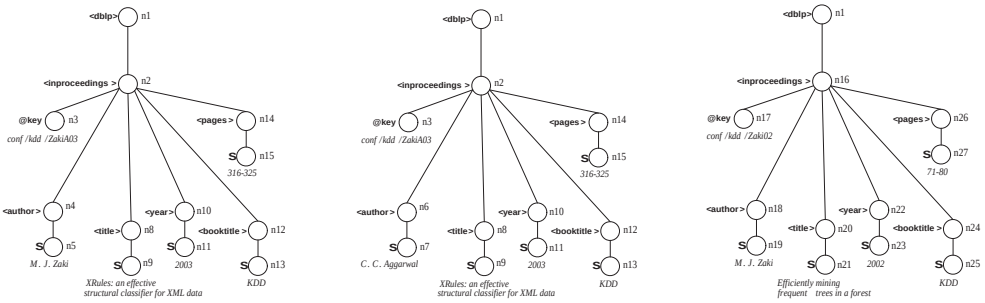


(b)

Figure 2: Example DBLP XML document and its tree

Figure 4(a). For example, in tree tuple τ_1, the application of path dblp.inproceedings.@key yields the attribute value 'conf/kdd/ZakiA03' corresponding to node n_3. Then, item e_1 is associated to the above pair path-answer. Yet, the answer of path dblp.inproceedings.booktitle.S is the string 'KDD' corresponding to two nodes, n_{13} of tree tuples τ_1 and τ_2, and n_{25} of tree tuple τ_3.

Once the item domain has been completely defined, a transaction is assigned with each tree tuple by mapping its pairs path-answer into the corresponding items. A transactional representation of the tree tuples of Figure 3 is shown in Figure 4(c). Notice that, in the example, all the transactions contain the same number of tree tuple items, as their corresponding tree tuples have the same number of leaf nodes. Clearly, transactions might be differently sized, depending on the specific structure of the associated tree tuples.



(a) (b) (c)

Figure 3: The tree tuples extracted from the XML tree of Figure 2(b)

4. XML Transactional Clustering

In this section, we describe how XML tree tuples modeled as transactions can be compared to each other and clustered by applying a centroid-based partitional algorithm suitably designed for a collaborative environment.

4.1. XML tree tuple item similarity

XML transactions are compared according to both their structure and content features, by computing the similarity between their respective tree tuple items. Given two tree tuple items e_i and e_j, the *tree tuple item similarity* is computed by the function:

$$sim(e_i, e_j) = f \times sim_S(e_i, e_j) + (1 - f) \times sim_C(e_i, e_j), \tag{1}$$

where sim_S (resp. sim_C) denotes the structural (resp. content) similarity between the items, and $f \in [0, 1]$ is a factor that tunes the influence of the structural part to the overall similarity.

Moreover, two XML tree tuple items e_i and e_j are said to be γ-*matched* if

$$sim(e_i, e_j) \geq \gamma \tag{2}$$

where $\gamma \in [0, 1]$ is a similarity threshold introduced to set the minimum degree of matching of the combinations of structure and content features embedded in the two tree tuple items.

Similarity by Structure

Structural similarity between two tree tuple items e_i and e_j is evaluated by comparing their respective tag paths.

Computing the similarity between any two paths is essentially accomplished by referring to it as a simple case of string matching of their respective element names, and finally averaging the (weighted) matchings. To this end, given any two tags t and t', the Dirichlet function (Δ) is applied in such a way that $\Delta(t, t')$ is equal to one if the tags match, otherwise $\Delta(t, t')$ is equal to zero.

path (p)	$\mathcal{A}_{\tau_1}(p)$	*node ID*
dblp.inproceedings.@key	'conf/kdd/ZakiA03'	n_3
dblp.inproceedings.author.S	'M. J. Zaki'	n_5
dblp.inproceedings.title.S	'XRules: an effective ...'	n_9
dblp.inproceedings.year.S	'2003'	n_{11}
dblp.inproceedings.booktitle.S	'KDD'	n_{13}
dblp.inproceedings.pages.S	'316-325'	n_{15}

path (p)	$\mathcal{A}_{\tau_2}(p)$	*node ID*
dblp.inproceedings.@key	'conf/kdd/ZakiA03'	n_3
dblp.inproceedings.author.S	'C. C. Aggarwal'	n_7
dblp.inproceedings.title.S	'XRules: an effective ...'	n_9
dblp.inproceedings.year.S	'2003'	n_{11}
dblp.inproceedings.booktitle.S	'KDD'	n_{13}
dblp.inproceedings.pages.S	'316-325'	n_{15}

path (p)	$\mathcal{A}_{\tau_3}(p)$	*node ID*
dblp.inproceedings.@key	'conf/kdd/Zaki02'	n_{17}
dblp.inproceedings.author.S	'M. J. Zaki'	n_{19}
dblp.inproceedings.title.S	'Efficiently mining ...'	n_{21}
dblp.inproceedings.year.S	'2002'	n_{23}
dblp.inproceedings.booktitle.S	'KDD'	n_{25}
dblp.inproceedings.pages.S	'71-80'	n_{27}

(a)

item ID	*associated node IDs*
e_1	n_3
e_2	$n_5,\ n_{19}$
e_3	n_9
e_4	n_{11}
e_5	$n_{13},\ n_{25}$
e_6	n_{15}
e_7	n_7
e_8	n_{17}
e_9	n_{21}
e_{10}	n_{23}
e_{11}	n_{27}

(b)

tr_1	$e_1\ e_2\ e_3\ e_4\ e_5\ e_6$
tr_2	$e_1\ e_7\ e_3\ e_4\ e_5\ e_6$
tr_3	$e_8\ e_2\ e_9\ e_{10}\ e_5\ e_{11}$

(c)

Figure 4: Transactional representation of the tree tuples of Figure 3: (a) paths and answers, (b) item domain, and (c) transaction set

Given two XML tree tuple items e_i and e_j, let $p_i = t_{i_1}.t_{i_2}.\ \ldots\ .t_{i_n}$ and $p_j = t_{j_1}.t_{j_2}.\ \ldots\ .t_{j_m}$ be their respective tag paths. The *structural similarity* between e_i and e_j is defined as

$$sim_S(e_i, e_j) = \frac{1}{n+m}\left(\sum_{h=1}^{n} s(t_{i_h}, p_j, h) + \sum_{k=1}^{m} s(t_{j_k}, p_i, k)\right) \tag{3}$$

such that, for each tag t and path $p = t_1.t_2.\ \ldots\ .t_L$, $s(t, p, a) = \max_{l=1..L}(1 + |a - l|)^{-1} \times \Delta(t, t_l)$.
Above, the tag matchings are corrected by a factor which is inversely proportional to the absolute difference of location of the tags in their respective paths. Essentially, this factor penalizes the similarity of two paths that have the same tags but are differently located.

It should also be noted that information on structural similarity could be semantically enriched with the support of a knowledge base, like in our previous

works; however, in this work, we deliberately intended to consider only syntactic similarity aspects to concentrate on the clustering phase and on the investigation of the benefits deriving from a collaborative distributed approach. Therefore, we leave this point as a future development of the proposed framework.

Similarity by Content

We refer to a *textual content unit* (for short, TCU) as the preprocessed text[1] of a tree tuple item, i.e., a `#PCDATA` element content or an attribute value. To weight the relevance of terms in TCUs, we defined a function which represents an adaptation of the popular *tf.idf* (term frequency - inverse document frequency) to our XML transactional domain.

Given a collection $\mathcal{XT}$ of XML trees , let w_j be an index term occurring in a TCU u_i of a tree tuple $\tau \in \mathcal{T}$ extracted from a tree $XT \in \mathcal{XT}$. The *ttf.itf* (*Tree tuple Term Frequency - Inverse Tree tuple Frequency*) weight of w_j in u_i with respect to τ is defined as

$$ttf.itf(w_j, u_i|_\tau) = tf(w_j, u_i) \times \exp\left(\frac{n_{j,\tau}}{N_\tau}\right) \times \frac{n_{j,XT}}{N_{XT}} \times \ln\left(\frac{N_\mathcal{T}}{n_{j,\mathcal{T}}}\right)$$

where $tf(w_j, u_i)$ denotes the number of occurrences of w_j in u_i, and the other symbols denote the number of TCUs appearing in τ (N_τ) and in the portion containing w_j ($n_{j,\tau}$), in XT (N_{XT}) and in the portion containing w_j ($n_{j,XT}$), in $\mathcal{T}$ ($N_\mathcal{T}$) and in the portion containing w_j ($n_{j,\mathcal{T}}$). Note that, the *ttf.itf* weight increases by increasing each of the factors in the function, i.e., the term frequency within the specific TCU, the term popularity across the TCUs of the same XML transaction and across the TCUs of the same document tree, and the term rarity across the whole collection of TCUs.

Content similarity between two tree tuple items is computed by measuring the text similarity of their respective TCUs. We adopt a vector-space model to represent the TCUs, therefore any TCU u_i is modeled with a vector $\vec{u}_i$ whose j-th component corresponds to an index term w_j and contains the *ttf.itf* relevance weight. The size of TCU vectors is equal to the size of the vocabulary $\mathcal{V}$, i.e., the set of index terms extracted from all TCUs in the collection $\mathcal{T}$ of tree tuples. Clearly, from a point of view of data structure implementation, proper structures can be exploited to drastically reduce the actual dimensionality of each TCU vector, since TCU vectors are typically sparse. To measure the similarity between TCU vectors, the well-known *cosine similarity* [31] is used.

4.2. The CXK-means clustering algorithm

XML tree tuples modeled as transactions are efficiently clustered by carrying out a partitional algorithm devised for the XML transactional domain. Generally, given a set of objects and a positive number k, a partitional clustering

[1]Text preprocessing is usually accomplished by means of language-specific operations such as lexical analysis, removal of stopwords and word stemming [7].

algorithm identifies k non-empty, disjoint groups each containing a homogeneous subset of objects. An important class of partitional approaches is based on the notion of *representative*, or *centroid*, of cluster: each object is assigned to a cluster C according to its distance from a specific data point c, which is the representative of C.

In [33, 32], we developed a centroid-based partitional clustering algorithm, which is essentially a variant of the K-means algorithm for the XML transactional domain. From clustering strategy viewpoint, this algorithm works as a traditional centroid-based method to compute $k+1$ clusters: starts choosing k objects as the initial cluster representatives, then iteratively reassigns each remaining object to the closest cluster until all cluster representatives do not change. The $(k+1)$-th cluster, called *trash cluster*, is created to contain unclustered objects.

Two major aspects in the XML transactional clustering algorithm are (i) the notion of proximity used to compare XML transactions and (ii) the notion of cluster representative.

In generic transactional domains, a widely used proximity measure is the Jaccard coefficient, which determines the degree of matching between any two transactions as directly proportional to their intersection (i.e., number of common items) and inversely proportional to their union. However, computing exact intersection between XML transactions is not effective, since XML tree tuple items may share structural or content information to a certain degree even though they are not identical. For this purpose, the notion of standard intersection between sets of items is enhanced to capture non-exact similarities in structure and content XML features. Let us now introduce our notion of enhanced intersection between XML transactions.

Given two XML transactions tr_1, tr_2, and a similarity threshold $\gamma \in [0,1]$, the set of *γ-shared items* between tr_1 and tr_2 is defined as

$$match^{\gamma}(tr_1, tr_2) = match^{\gamma}(tr_1 \rightarrow tr_2) \ \cup \ match^{\gamma}(tr_2 \rightarrow tr_1),$$

where

$$match^{\gamma}(tr_i \rightarrow tr_j) = \{e \in tr_i \mid \exists e_h \in tr_j, \ sim(e, e_h) \geq \gamma, \\ \nexists e' \in tr_i, sim(e', e_h) > sim(e, e_h)\}.$$

The set of γ-shared items hence resembles the intersection between transactions at a degree greater than or equal to a similarity threshold γ. Being defined this notion of enhanced intersection, we define the *XML transaction similarity* function between tr_1 and tr_2 as

$$sim_J^{\gamma}(tr_1, tr_2) = \frac{|match^{\gamma}(tr_1, tr_2)|}{|tr_1 \cup tr_2|} \tag{4}$$

We now present our proposed XML transactional clustering algorithm for a collaborative distributed environment, called *CXK-means*. Figure 5 sketches the

Global Input:
A set $\mathcal{S}$ of XML transactions distributed over m nodes;
The desired number k of clusters; A similarity threshold γ.
Global Output:
A partition $\mathcal{C}$ of $\mathcal{S}$ in k clusters distributed over m nodes.

Process N_0
Method:
define a partition of $\{1, \ldots, k\}$ into m subsets $Z_1, \ldots, Z_m$;
send $(\{Z_1, \ldots, Z_m\}, k, \gamma)$ to N_i, $\forall i \in [1..m]$;

Process N_i
Input:
A set $\mathcal{S}^i \subset \mathcal{S}$ of XML transactions.
Output:
A partition $\mathcal{C}^i = \{C_1^i, \ldots, C_k^i\}$ of $\mathcal{S}^i$ into k clusters.
Method:
receive $(\{Z_1, \ldots, Z_m\}, k, \gamma)$ from N_0;
let $Z_i = \{j_1, \ldots, j_{q_i}\}$, with $0 \leq q_i \leq k$, $\sum_{i=1}^{m} q_i = k$;
/ select q_i initial global clusters */*
select $\{tr_1, \ldots, tr_{q_i}\}$ from $\mathcal{S}^i$ coming from distinct original trees;
let $g_{j_s} = tr_s$, $\forall s \in [1..q_i]$;
$C_j^i = \{\}$;
repeat
  send (broadcast) $\{g_j | j \in Z_i\}$ to $N_1, ..., N_m$;
  receive $\{g_j | j \in Z_h\}$ from N_h;
  repeat */* transaction relocation */*
    $C_{k+1}^i = \{tr \in S^i | sim_J^\gamma(tr, g_j) = 0\}$; $\{Eq.\ (4)\}$
    for each $j \in [1..k]$ **do**
      $C_j^i = \{tr \in S^i \setminus C_{k+1}^i | sim_J^\gamma(tr, g_j) \geq sim_J^\gamma(tr, g_t)\}, \forall t \in [1..k]\}$; $\{Eq.\ (4)\}$
      $\ell_j^i = \mathsf{ComputeLocalRepresentative}(C_j^i)$;
    end for
  until no transaction is relocated;
  if no ℓ_j^i changes **then**
    send (broadcast) $(\{\}, V_i = done)$;
  else
    send $(\{(\ell_j^i, |C_j^i|) | j \in Z_{i'}\}, V_i = continue)$ to all other $N_{i'}$, $i' \neq i$;
  receive $(\{(\ell_j^h, |C_j^h|) | j \in Z_i\}, V_{i'})$ from all other $N_{i'}$, $i' \neq i$;
  if $(\exists h \in [1..m]$ s.t. $V_h = continue)$ **then**
    $g_j = \mathsf{ComputeGlobalRepresentative}(\{(\ell_j^1, |C_j^1|), \ldots, (\ell_j^m, |C_j^m|)\})$;
until $V_1 = \cdots = V_m = done$;

Figure 5: The *CXK-means* algorithm

main phases of the algorithm, and Figure 6 shows the main functions involved in the algorithm execution. Major characteristics of *CXK-means* are described in the following.

The input set $\mathcal{S}$ of all XML transactions is distributed over m nodes. Each node stores a local subset $\mathcal{S}^i$ and communicates with all the other nodes sending

Function ComputeLocalRepresentative(C) : rep;
  $I_C = \{e \mid e \in tr \wedge tr \in C\}$;
  $P_C = \{\langle p, h\rangle \mid \exists h$ items $(p, u) \in I_C\}$;
  for each $e \in I_C$ **do**
    $rank_S(e) = sum\{h \mid \exists e' = (p', u') \in I_C \wedge \langle p', h\rangle \in P_C \wedge\ sim_S(e, e') \geq \gamma\}/|P_C|$;
    $rank_C(e) = sum_{e' \in I_C}\{(\vec{u} \cdot \vec{u}')/(\|\vec{u}\| \times \|\vec{u}'\|)\}$, where $\vec{u}$ and $\vec{u}'$ are the TCU vectors of e and e', respectively;
    $rank(e) = f \times rank_S(e) + (1 - f) \times rank_C(e)$; $\{Eq.\ (1)\}$
  end for
  let $\overline{I}_C$ be the list containing the elements in I_C ordered by $rank$ values;
  return GenerateTreeTuple($\overline{I}_C, C$);

Function ComputeGlobalRepresentative(T) : rep;
  Given any set $X = \{(x_1^1, x_1^2), \ldots, (x_S^1, x_S^2)\}$ of pairs,
    let $X[q]$ be the projection $\{x_1^q, \ldots, x_S^q\}$ of X, with $q \in \{1, 2\}$;
  let $T = \{t_i, \ldots, t_m\}$, where $t_i = (tr_i, w_i), i \in [1..m]$;
  $I_T = \{(e, \overline{w}) \mid \exists (tr, w) \in T\ \wedge\ e \in tr\ \wedge\ \overline{w} = sum\{w' \mid (tr', w') \in T \wedge e \in tr'\}\}$;
  $P_T = \{\langle p, h\rangle \mid \exists h$ items $(p, u) \in I_T[1]\}$;
  for each e s.t. $(e, w) \in I_T$ **do**
    $g\_rank_S(e) = sum\{h \mid \exists e' = (p', u') \in I_T[1] \wedge \langle p', h\rangle \in P_T \wedge sim_S(e, e') \geq \gamma\}/|P_T|$;
    $g\_rank_C(e) = sum_{e' \in I_C}\{(\vec{u} \cdot \vec{u}')/(\|\vec{u}\| \times \|\vec{u}'\|)\}$, where $\vec{u}$ and $\vec{u}'$ are the TCU vectors of e and e', respectively;
    $g\_rank(e) = w \times (f \times g\_rank_S(e) + (1 - f) \times g\_rank_C(e))$; $\{Eq.\ (1)\}$
  end for;
  let $\overline{I}_T[1]$ be the list containing the elements in $I_T[1]$ ordered by g_rank values;
  return GenerateTreeTuple($\overline{I}_T[1], T[1]$);

Function GenerateTreeTuple(I_C, C) : rep;
  let $|tr_{max}|$ be the maximum length of transaction within C;
  $rep' = \emptyset$; $s' = 0$;
  repeat
    let $I_C^* \subseteq I_C$ be the set of items in I_C with the highest rank;
    $rep = rep'$; $s = s'$;
    $rep' = conflateItems(rep \cup I_C^*)$;
    $s' = sum_{tr \in C}\{sim_J^{\gamma}(tr, rep')\}$; $\{Eq.\ (4)\}$
    $I_C = I_C \setminus I_C^*$;
  until ($I_C = \emptyset\ \vee\ |rep| > |tr_{max}| \vee s' < s$)
  return rep;

Figure 6: Functions employed by the *CXK-means* algorithm

"local" representatives and receiving "global" representatives. An initial process corresponding to a node N_0 defines a partition of the set $\{1, \ldots, k\}$ of cluster identifiers into m subsets $Z_j, j \in [1..m]$. Each set Z_j contains the identifiers of the clusters for which the node N_j has the responsibility of computing the global representatives. It should be noted that the presence of node N_0 does not contrast the collaborative nature of the proposed CXK-means. Indeed, N_0 is not responsible of summarizing the information coming from the various peers $N_1, \ldots, N_m$ and, therefore, does not act as a coordinator; rather, N_0 performs

only trivial startup operations which, in principle, can be performed by any peer.

Each node N_i ($i \in [1..m]$) is in charge of computing local clusters $C_1^i, \ldots, C_k^i$ and local representatives $\ell_1^i, \ldots, \ell_k^i$, but also a subset $\{g_j | j \in Z_i\}$ of the global representatives (using the local representatives computed by all nodes). Each node has a process that executes a classical K-means-like partitional clustering scheme on its local data in $\mathcal{S}^i$. The clustering process employs global representatives received from each other node in the network and terminates when transaction assignments to local clusters do not change.

For each node N_i, the local representative of a cluster C_j^i (function ComputeLocalRepresentative) is computed by starting from the set of γ-shared items among all the transactions within C_j^i. More precisely, for each transaction in C_j^i, the union of the γ-shared item sets with respect to all the other transactions in C_j^i is obtained; this guarantees no dependence of the order of examination of the transactions. Then, the set of γ-shared items is involved into function GenerateTreeTuple to compute a representative having the form of a tree tuple. According to such a function, a raw representative is firstly defined by selecting the items from these union sets with the highest frequency: the raw representative, however, may not have the form of a tree tuple, as some items therein may refer to the same path but with different answers. Any raw representative is transformed into a tree tuple by $conflateItems$ procedure. This procedure is applied to a set I of items and yields a tree tuple composed by all the distinct paths p involved into the items in I; the content associated to each path p is the union of the contents of the items in I having p as a path. A greedy heuristic refines the current representative by iteratively adding the remaining most frequent items until the sum of pair-wise similarities between transactions and representative cannot be further maximized. By involving again $conflateItems$ procedure, any refinement ensures that the resulting representative is actually a tree tuple.

The global representative g_j of a cluster C_j (function ComputeGlobalRepresentative) is computed in a way similar to that employed for local representatives. A major difference is that global representatives exploit the m local representatives $\ell_j^1, \ldots, \ell_j^m$ along with their respective weights $|C_j^1|, \ldots, |C_j^m|$, in order to take also into account the size of the clusters summarized by each node. The rationale is that the greater is the weight $|C_j^i|$ (i.e., the greater the number of transactions belonging to the cluster C_j stored into the local repository $\mathcal{S}^i$ at node i), the greater is the information in the local representative ℓ_j^i in summarizing cluster C_j.

Nodes communicate their local state by sending a flag to other nodes in the network. In particular, a node sends a termination signal (i.e., "*done*") if, at the end of its local clustering process, all its local cluster representatives do not change with respect to the ones computed in the previous execution. In this way, the collaborative clustering process continues until each node N_i in the network reaches a stable clustering solution (i.e., each flag $V_{i'}$ is "*done*", $\forall i' \in [1..m]$).

4.3. Complexity

In the following, we discuss the computational complexity of the proposed *CXK-means*, by analyzing the costs of the *(i)* similarity functions exploited by the algorithm, *(ii)* main memory operations, and *(iii)* communications among nodes. We conclude this section by providing a comparison between centralized and distributed cases, paying special attention to the impact of the network size and the dataset size on efficiency improvements.

Complexity of similarity functions. The cost of the various similarity functions exploited by *CXK-means* are summarized in the following.

- The cost C_S^e of evaluating function sim_S (Eq. (3)), which computes the structural similarity between any two items $e_i = \langle p_i, u_i \rangle$ and $e_j = \langle p_j, u_j \rangle$, is as follows. It is bounded by $\mathcal{O}(|p_i| \times |p_j| \times \mathrm{C}_\Delta)$, where $|p_i|$ and $|p_j|$ are the lengths of paths p_i and p_j, respectively, and C_Δ is the cost of computing the Dirichlet function between any two tags. Since the lengths of paths p_i and p_j are bounded by $depth(XT)$, i.e., the depth of the XML tree XT from which the input transactions are extracted (cf. Section 3.1), it results that $\mathrm{C}_S^e = \mathcal{O}((depth(XT))^2 \times \mathrm{C}_\Delta) = \mathcal{O}((depth(XT))^2)$, since C_Δ can be reasonably assumed to be a constant.

- Evaluating content similarity between any two items e_i and e_j by means of function sim_C consists in performing cosine similarity between TCU vectors $\vec{u}_i$ and $\vec{u}_j$ of e_i and e_j, respectively; this cost, denoted as C_C^e, is linear with respect to to the dimensionalities $|\vec{u}_i|$ and $|\vec{u}_j|$ of vectors $\vec{u}_i$ and $\vec{u}_j$, respectively, i.e., $\mathrm{C}_C^e = \mathcal{O}(|\vec{u}_i| + |\vec{u}_j|)$. More generally, this cost is bounded by the maximum TCU size $|u_{max}|$ over all the input transactions, i.e., it holds that $\mathrm{C}_C^e = \mathcal{O}(|u_{max}|)$. This bound depends on the content of the TCUs either initially contained within the input transactions or generated during the computation of representatives. Clearly, $|u_{max}|$ is $\mathcal{O}(|\mathcal{V}|)$, where $|\mathcal{V}|$ denotes the size of the vocabulary (cf. Section 4.1), but in practice it holds that $|u_{max}| \ll |\mathcal{V}|$.

- The cost $\mathrm{C}^e = \mathrm{C}_S^e + \mathrm{C}_C^e$ of computing function sim (Eq. (1)) is bounded by $\mathcal{O}(|u_{max}| + (depth(XT))^2) = \mathcal{O}(|u_{max}|)$, as it typically holds that $depth(XT) \ll |u_{max}|$.

- The cost C^γ of computing function sim_J^γ between transactions tr_1 and tr_2 (Eq. (4)) depends on the set of γ-shared items between tr_1 and tr_2. This is computed by evaluating the similarity between each pair $e_i \in tr_1$, $e_j \in tr_2$ of items according to function sim; thus, it is bounded by $\mathcal{O}(|tr_1| \times |tr_2| \times \mathrm{C}^e) = \mathcal{O}(|tr_{max}|^2 \times |u_{max}|)$, where $|tr_{max}|$ is the maximum length of a transaction in $\mathcal{S}$.

Complexity of main memory operations. We analyze in the following the complexity of the main memory operations performed by *CXK-means* algorithm in each node N_i, $i \in [1..m]$. Clearly, the global complexity depends on the

number of iterations performed; however, this number is usually a small constant (in our experiments, for instance, it was always smaller than 10). To this end, our focus here is on the study of the complexity of a single iteration.

Essentially, each iteration of *CXK-means* involves *(i)* a *relocation phase*, in which the transactions are assigned to the closest local representative, and *(ii)* the computation of local and global representatives (ComputeLocalRepresentative and ComputeGlobalRepresentative functions, respectively). The costs of such phases are summarized next.

- The relocation phase requires a comparison between each transaction within $\mathcal{S}^i$ (i.e., the subset of transactions stored into node N_i) and each of the k global representatives by means of function sim_J^γ; thus, the cost of such a phase is bounded by $\mathcal{O}(k \times |\mathcal{S}^i| \times \mathrm{C}^\gamma) = \mathcal{O}(k \times |\mathcal{S}^i| \times |tr_{max}|^2 \times |u_{max}|)$.
- In the ComputeLocalRepresentative function, two sub-phases can be distinguished: (a) computing item ranking, and (b) generation of the tree tuple representatives:
  - given any cluster C_j^i (j-th cluster in the i-th node), phase (a) requires the computation of structural and content rankings. Structural ranking ($rank_S$) involves the computation of the structural similarity $sim_S(e_i, e_j)$, $\forall e_i, e_j \in I_{C_j^i}$ ($I_{C_j^i}$ is the set containing the items belonging to all the transactions within C_j^i). The cost of this operation is in principle bounded by $\mathcal{O}(|I_{C_j^i}|^2 \times \mathrm{C}_S^e)$; however, as the input XML tree XT is fixed, one can pre-compute the structural similarity between every pair of maximal tag paths of XT only once, and exploit these pairwise similarities for computing structural ranking. This leads to a reduced cost bounded by $\mathcal{O}(|\mathcal{TP}_{XT}|^2 \times \mathrm{C}_S^e + |I_{C_j^i}| \times |P_{C_j^i}|) = \mathcal{O}(|\mathcal{TP}_{XT}|^2 \times (depth(XT))^2 + |I_{C_j^i}| \times |\mathcal{TP}_{XT}|)$, where $|P_{C_j^i}|$ is $\mathcal{O}(|\mathcal{TP}_{XT}|)$ and $\mathcal{TP}_{XT}$ denotes the set of maximal tag paths in XT (cf. Section 3.1). As $|I_{C_j^i}|$ is $\mathcal{O}(|C_j^i| \times |tr_{max}|)$ and $|\mathcal{TP}_{XT}|$ can be reasonably assumed to be $\mathcal{O}(|tr_{max}|)$, the cost of structural ranking is bounded by $\mathcal{O}(|tr_{max}|^2 \times (depth(XT))^2 + |C_j^i| \times |tr_{max}|^2)$. On the other hand, the complexity of content ranking ($rank_C$) is $\mathcal{O}(|I_{C_j^i}|^2 \times \mathrm{C}_C^e) = \mathcal{O}(|C_j^i|^2 \times |tr_{max}|^2 \times |u_{max}|)$. It can be noted that the cost $\mathcal{O}(|I_{C_j^i}| \times \log |I_{C_j^i}|) = \mathcal{O}(|C_j^i| \times |tr_{max}| \times \log(|C_j^i| \times |tr_{max}|))$ for sorting the set $I_{C_j^i}$ is not considered as it is dominated by the cost of content ranking;
  - phase (b) is performed by the function GenerateTreeTuple, which consists of three main operations: *(i)* selecting the items with the highest rank, which is bounded by $\mathcal{O}(|I_{C_j^i}|) = \mathcal{O}(|C_j^i| \times |tr_{max}|)$, *(ii)* joining the TCUs of items having the same path ($conflateItems$ procedure), whose cost is bounded by $\mathcal{O}(|I_{C_j^i}| \times |u_{max}|) = \mathcal{O}(|C_j^i| \times |tr_{max}| \times |u_{max}|)$, and *(iii)* computing the sum of similarities between the cur-

rent representative and all the transactions within the cluster, whose cost is bounded $\mathcal{O}(|I_{C_j^i}| \times |C_j^i| \times \mathrm{C}^\gamma) = \mathcal{O}(|tr_{max}|^3 \times |C_j^i|^2 \times |u_{max}|)$.

As the costs of phase (a) are dominated by those of phase (b), the overall complexity of performing ComputeLocalRepresentative function over a single cluster C_j^i is $\mathcal{O}(|tr_{max}|^3 \times |C_j^i|^2 \times |u_{max}|)$. Since this function is called for all k clusters, we have a global cost of $\mathcal{O}(\sum_{j=1}^{k} |tr_{max}|^3 \times |C_j^i|^2 \times |u_{max}|$ $= \mathcal{O}(|tr_{max}|^3 \times |u_{max}| \times \sum_{j=1}^{k} |C_j^i|^2) = \mathcal{O}(|tr_{max}|^3 \times |u_{max}| \times |S^i|^2)$.

- The analysis of ComputeGlobalRepresentative function is similar to that carried out for local representatives. The only difference is that, at each step, this function is performed over a set of m transactions (i.e., the local representatives computed by all the m nodes), rather than a set of size $|C_j^i|$. Hence, the cost of computing a single global representative by a node N_i can be trivially obtained by replacing $|C_j^i|$ with m in the formula expressing the cost of computing local representatives. Therefore, this cost is bounded by $\mathcal{O}(|tr_{max}|^3 \times m^2 \times |u_{max}|)$. Assuming that the responsibilities of computing global representatives are uniformly distributed over the nodes, the number $|Z_i| = q_i$ of global representatives computed by node N_i is bounded by $\mathcal{O}(\lceil k/m \rceil)$. Therefore, the global cost by node N_i is bounded by $\mathcal{O}(k \times m \times |tr_{max}|^3 \times |u_{max}|)$.

In conclusion, we can state that the global complexity of the main memory operations performed by *CXK-means* in each node N_i is equal to the sum of the complexities discussed above; therefore, it is $\mathcal{O}(|tr_{max}|^2 \times |u_{max}| \times (|\mathcal{S}^i| \times k + (\sum_{j=1}^{k} |C_j^i|^2 + k \times m) \times |tr_{max}|))$. In this respect, it is reasonable to assume that $|\mathcal{S}^i| \geq k$, otherwise the clustering process is trivial; this leads to $|\mathcal{S}^i| \times k \leq \sum_{j=1}^{k} |C_j^i|^2$ and, therefore, to the following overall complexity of main memory operations performed by *CXK-means*:

$$\begin{aligned} C_{mem} &= \mathcal{O}\left(|tr_{max}|^3 \times |u_{max}| \times \left(\sum_{j=1}^{k} |C_j^i|^2 + k \times m\right)\right) = \\ &= \mathcal{O}(|tr_{max}|^3 \times |u_{max}| \times (|\mathcal{S}^i|^2 + k \times m)) \end{aligned}$$

Complexity of communications. To analyze the cost of communications, we note that the cost of transferring a single transaction from any node to another one is bounded by $\mathcal{O}(|tr_{max}| \times (|u_{max}| + depth(XT))) = \mathcal{O}(|tr_{max}| \times |u_{max}|)$. Indeed, any transaction has at most $|tr_{max}|$ elements and any element is composed by a path having size at most equal to $depth(XT)$ and a $|u_{max}|$-dimensional TCU vector. Assuming again that the responsibilities of computing global representatives are uniformly distributed over the nodes, we note that any node N_i sends out at each iteration of *CXK-means*:

- $q_i = \lceil k/m \rceil$ transactions (i.e., global representatives) to all other $m-1$ nodes, with a cost of $\mathcal{O}((m-1)/m \times k \times |tr_{max}| \times |u_{max}|)$;
- $(m-1)/m \times k$ transactions (i.e., local representatives) to a single node, with a cost of $\mathcal{O}((m-1)/m \times k \times |tr_{max}| \times |u_{max}|)$. Indeed, any node N_i

sends each local representative only to the node having the responsibility of computing the corresponding global representative; the total number of local representatives sent by N_i is $(m-1)/m \times k$ as we do not have to consider the representatives of the clusters for which N_i is responsible for computing global representatives.

Analogously, each node N_i receives from all other nodes $(m-1)/m \times k$ transactions (i.e., global representatives) at a cost of $\mathcal{O}((m-1)/m \times k \times |tr_{max}| \times |u_{max}|)$, along with $(m-1)/m \times k$ transactions (local representatives) at a cost of $\mathcal{O}((m-1)/m \times k \times |tr_{max}| \times |u_{max}|)$. Therefore, we can state that the global complexity of communications by each node N_i is equal to

$$C_{comm} = \mathcal{O}\left(\frac{m-1}{m} \times k \times |tr_{max}| \times |u_{max}|\right)$$

It should be noted that, although $(m-1)/m$ is obviously $\mathcal{O}(1)$, we report the fraction $(m-1)/m$ in the above formula to highlight the differences between centralized and distributed cases; indeed, for the centralized case, $m = 1$ implies no communications.

Distributed vs. centralized *CXK-means*. Once derived the complexity of main memory operations and communications by each node, we focus on the analysis of time consumptions, distinguishing between the centralized ($m = 1$) and distributed ($m > 1$) cases. To this end, we denote by t_{mem} and t_{comm} the time needed to perform a single main memory operation and the time needed for a single communication between any pair of nodes, respectively. Thus, the time needed by each node for performing *CXK-means* is bounded by $\mathcal{O}(C_{mem} \times t_{mem} + C_{comm} \times t_{comm})$, i.e., $\mathcal{O}(|tr_{max}|^3 \times |u_{max}| \times (\sum_{j=1}^{k} |C_j^i|^2 + k \times m) \times t_{mem} + |tr_{max}| \times |u_{max}| \times k \times (m-1)/m \times t_{comm}) = \mathcal{O}(|tr_{max}| \times |u_{max}| \times (|tr_{max}|^2 \times (\sum_{j=1}^{k} |C_j^i|^2 + k \times m) \times t_{mem} + k \times (m-1)/m \times t_{comm}))$. As we can reasonably assume that $k \times m$ is $\mathcal{O}(\sum_{j=1}^{k} |C_j^i|^2)$—both k and m are usually much lower than $|\mathcal{S}^i| = \sum_{j=1}^{k} |C_j^i|$—we can state that the global time spent over all the m nodes is bounded by $\mathcal{O}(|tr_{max}| \times |u_{max}| \times (|tr_{max}|^2 \times m \times (\sum_{j=1}^{k} |C_j^i|^2) \times t_{mem} + k \times (m-1) \times t_{comm}))$. To better comprehend this formula, two limit cases can be considered:

- *Case 1:* $|C_j^i|$ is $\mathcal{O}(|\mathcal{S}^i|/k)$ (i.e., clusters have roughly the same size), that is $\sum_{j=1}^{k} |C_j^i|^2$ is $\mathcal{O}(|\mathcal{S}^i|^2/k) = \mathcal{O}(|\mathcal{S}|^2/(k \times m^2))$.
- *Case 2:* $\exists j$ *such that* $|C_j^i| = \mathcal{O}(|\mathcal{S}^i|)$ (i.e., there exists a cluster containing most of the transactions), that is $\sum_{j=1}^{k} |C_j^i|^2$ is $\mathcal{O}(|\mathcal{S}^i|^2) = \mathcal{O}(|\mathcal{S}|^2/m^2)$.

Within this view, we can finally express the global time consumption by *CXK-means* as $\mathcal{O}(f(m))$, where:

$$f(m) = |tr_{max}| \times |u_{max}| \times \left(\frac{|tr_{max}|^2 \times |\mathcal{S}|^2 \times t_{mem}}{h \times m} + k \times t_{comm} \times (m-1)\right)$$

where $1 \leq h \leq k$ takes into account the distribution of transactions over the various clusters. It can be easily noted that the above function $f(m)$ is a sum of an hyperbolic function and a linear function; thus, it has a global minimum located in:

$$m = \frac{|\mathcal{S}|}{\sqrt{h}} \times \sqrt{\frac{|tr_{max}|^2 \times t_{mem}}{k \times t_{comm}}}$$

This enables us to draw the following conclusions:

- The global minimum of function $f(m)$ represents an upper-bound for the number m to guarantee efficiency improvements with respect to the centralized case; this essentially means that distributing transactions over m nodes is in general more convenient than having a single node storing the whole set of transactions until m becomes equal to the global minimum of function $f(m)$.

- While the upper-bound for m is not reached, it holds that the larger the number m of nodes, the larger the "efficiency gain" of distributed *CXK-means* with respect to centralized *CXK-means*. In particular, the improvement of the efficiency follows an hyperbolic trend: it is more evident for small m values, while decreasing as m approaches the upper-bound.

- The value of the upper-bound for m (i.e., the global minimum of function $f(m)$) is directly (resp. inversely) proportional to the size $|\mathcal{S}|$ of the input dataset of transactions (resp. the parameter $h \in [1..k]$ that takes into account the distribution of the cluster sizes). Thus, the larger the set and/or the smaller the distribution of transactions over the various clusters, the larger the value of the upper-bound for m and, therefore, the greater the (maximum reachable) efficiency improvements of the distributed case with respect to the centralized case.

5. Experimental Evaluation

5.1. Experimental setting

We assessed the proposed framework in performing clustering according to structure, content, or both information. We hereinafter refer to these kinds of solutions as *structure-driven*, *content-driven*, and *structure/content-driven* clustering, respectively. The first two types of clustering concern the detection of groups of XML that are homogeneous by either structure or content. The third type (i.e., structure/content-driven clustering) includes a variety of scenarios, ranging from detecting common structures across different topics, or conversely, to identifying classes of tree tuples that both cover common topics and belong to the same structural category.

The three types of clustering correspond to different settings of the parameters f and γ, which control the XML transaction similarity function (cf. Eqs. (1)-(2)). We varied f within [0,1] with step 0.1, and γ within [0.5,1) with

step 0.05—we chose $\gamma = 0.5$ as the maximum tolerance threshold in computing similarities. Also, since the setting of f depends on the clustering goal, we decided to partition the (discrete) interval [0,1] as follows: [0,0.3] for content-driven clustering, [0.4,0.6] for structure/content-driven clustering, and [0.7,1] for structure-driven clustering.

Network topology was characterized by the type and number of nodes. In particular, the architecture of each node was composed by an Intel Itanium 2 64bit 1,400 MHz (dual core), 4 GB memory RAM and GigaBit network interface, running Debian New Linux 4.0 64bit. We performed experiments by varying the number of nodes from 1 to a maximum of 19; note that a network size equal to 1 refers to centralized clustering, which represents the baseline case.

Data partitioning is a crucial aspect in distributed environments. For this reason, we considered two scenarios in our experiments: the first consists in partitioning data in such a way that the entire set $\mathcal{S}$ is equally distributed over the m nodes (i.e., $|\mathcal{S}^i| = |\mathcal{S}|/m, \forall i \in [1..m]$); in the second scenario, a half of the nodes hold a portion of the data that is the half of the one held by the remaining ones (i.e., there are $m/2$ nodes with $4|\mathcal{S}|/3m$ transactions and other $m/2$ nodes with $2|\mathcal{S}|/3m$ transactions).

5.2. Data description

We used four real word document collections for the evaluation. A short description for each of these datasets is given next—further information, including the XML structures as DTDs, can be found in [33].

The *DBLP* collection is a subset of the popular DBLP digital bibliography on computer science.[2] *DBLP* is comprised of 3,000 documents which correspond to 5,884 transactions and 8,231 distinct items. It contains short text descriptions (e.g., author names, paper titles, conference names), and covers 4 main categories, namely "journal articles" (article), "conference papers" (inproceedings), "books" (book), and "book chapters" (incollection). Six topical classes are instead identified, which are "multimedia", "logic programming", "web and adaptive systems", "knowledge based systems", "software engineering", and "formal languages". If both content and structure information are taken into account, 16 classes are identified.

The *IEEE* dataset is the IEEE collection version 2.2, which has been used as a benchmark in the INEX document mining track 2008.[3] *IEEE* consists of 4,874 articles originally published in 23 different IEEE journals from 2002 to 2004. Documents in this collection conform to a complex schema which includes front matter, back matter, section headings, text formatting tags and mathematical formulas. For our experiments, stylistic and other non-logical markups were filtered out. In our XML transactional domain, the *IEEE* collection has 211,909 transactions and 135,869 distinct items. Also, the number of leaf nodes is 228,869, the maximum fan out is 43, and the average depth

[2] http://dblp.uni-trier.de/xml/

[3] http://www.inex.otago.ac.nz/documentcollection.asp

is about 5. In *IEEE*, the article journals determine the categories that were used to partition the collection, which strictly follow the original INEX categorization. Precisely, two structural categories correspond to "transactions" and "non-transactions" articles, whereas the classification by content organizes the articles by the following 8 topic-classes: "computer", "graphics", "hardware", "artificial intelligence", "internet", "mobile", "parallel", and "security". Moreover, 14 hybrid classes are identified according to these structural and content classes.

The *Shakespeare* collection is a subset of Shakespeare 2.00,[4] an archive of Shakespeare's plays in XML format. *Shakespeare* is comprised of seven (long) documents which correspond to the plays Henry the Sixth (Part 1, 2 and 3), Henry the Eighth, Hamlet, Macbeth and Othello. All lines corresponding to the same speech in the original document were concatenated to form a unique speech.line element. Three structural classes were identified according to the presence/absence of discriminatory paths, namely personae.pgroup, act.prologue, and act.epilogue. Moreover, as found in [33], tree tuples were preferably grouped into 5 classes for content-driven clustering, and into 12 classes for structure/content-driven clustering.

Finally, *Wikipedia* is a subset of 10,000 documents from the Wikipedia XML Corpus used in the INEX contest 2007 [10]. This collection contains very long articles, which are organized in 21 thematic categories, each corresponding to a Wikipedia portal [10]. Analogously to *IEEE*, we removed non-logical markups from the documents. Due to the absence of evident or frequent structural differences among the individual Wikipedia articles, we mainly used this set for content-driven clustering; for purposes of clustering evaluation, we referred to a 21-class thematic organization [33].

5.3. Cluster validity measures

To assess the quality of clustering solutions for the datasets, we exploited the availability of reference classifications for XML documents. The objective was to evaluate how well a clustering fits a predefined scheme of known classes (natural clusters). For this purpose, we resorted to the well-known *F-measure* [21], which is defined as the harmonic mean of values that express two notions from Information Retrieval, namely *Precision* and *Recall*. F-measure ranges within $[0, 1]$, where higher values refer to better quality results. Since we perform tree tuple decomposition of XML documents and then transactional modeling, the evaluation process takes into account the set $\mathcal{S}$ of XML transactions.

Given a set $\mathcal{S} = \{tr_1, \ldots, tr_m\}$ of XML transactions, let $\Gamma = \{\Gamma_1, \ldots, \Gamma_H\}$ be the reference classification of the transactions in $\mathcal{S}$, and $\mathcal{C} = \{C_1, \ldots, C_K\}$ be the output partition yielded by a clustering algorithm. *Precision* of cluster C_j with respect to class Γ_i is the fraction of the transactions in C_j that has been correctly classified, whereas *Recall* of cluster C_j with respect to class Γ_i is

[4] http://metalab.unc.edu/bosak/xml/eg/shaks200.zip

the fraction of the objects in Γ_i that has been correctly classified. Formally,

$$P_{ij} = \frac{|C_j \cap \Gamma_i|}{|C_j|} \ , \ R_{ij} = \frac{|C_j \cap \Gamma_i|}{|\Gamma_i|} \ , \ F_{ij} = \frac{2P_{ij}R_{ij}}{P_{ij} + R_{ij}}$$

In order to score the quality of C with respect to Γ by means of a single value, the overall F-measure $F(\mathcal{C}, \Gamma)$ is computed using the weighted sum of the maximum F_{ij} score for each class Γ_i.

$$F(\mathcal{C}, \Gamma) = \frac{1}{|\mathcal{S}|} \sum_{i=1}^{H} |\Gamma_i| \max_{j \in [1..K]} F_{ij}$$

5.4. Evaluation goals

As the problem of collaborative distributed clustering of XML documents is addressed for the first time in this work, there is no strictly competing method to be compared to our *CXK-means*. For this purpose, our experimental evaluation was mainly conceived to evaluate the performance of *CXK-means* with respect to the centralized case (which arises when the number m of peer nodes is equal to one), as well as to a non-collaborative distributed approach, in terms of both efficiency and effectiveness. In this respect, we identified the following main evaluation goals:

1. *Efficiency*: Evaluation of the runtime performance of *CXK-means* with respect to the centralized case, by varying the number of nodes in the distributed environment (P2P network). According to the study on the computational complexity reported in Section 4.3, it is expected that, as the number of nodes increases, the computation time required in each node decreases, but also the network traffic (i.e., exchange of cluster representatives) increases. This behavior leads to the identification of a certain number of nodes that acts as a "saturation point", meaning that further increasing the number of nodes does not guarantee any significant efficiency gain; within this view, a major objective is to evaluate the saturation (stabilization) point in every executed test. Note also that such a saturation point should in principle be close to the global minimum of function $f(m)$ discussed in Section 4.3.
2. *Effectiveness*: Evaluation of accuracy of *CXK-means* with respect to the centralized case by varying the number of nodes. The algorithm performance is expected to be inversely proportional to the number of nodes, since increasing this number leads to a reduction of the distribution ratio of the transactions over the nodes; as a consequence, each node produces, at each step of the distributed algorithm, a local clustering solution over a small portion of data, which cannot really represent the final overall solution. In this respect, it is crucial to assess the loss of accuracy of *CXK-means* with respect to the centralized case when the number of nodes is equal (or close) to the number of nodes recognized as a stabilization point in the efficiency evaluation.

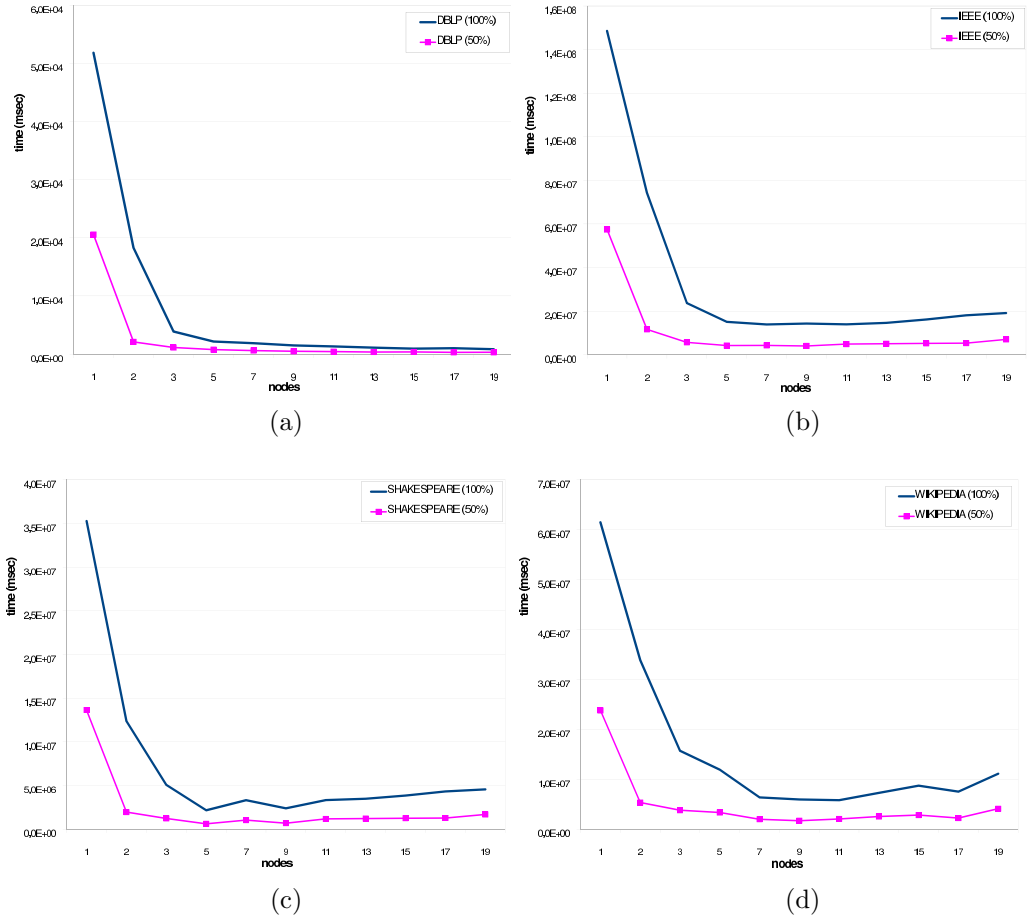


Figure 7: Clustering time performances varying the number of nodes and the dataset size: (a) *DBLP*, (b) *IEEE*, (c) *Shakespeare*, and (d) *Wikipedia*

3. *Impact of collaborativeness*: We conducted a further experimental session to compare our *CXK-means* with an existing parallel/distributed related work, which was suitably adapted to handle XML transactional data.

5.5. Results

Efficiency. Figure 7 shows time performances on the four evaluation sets by increasing the number of nodes and also varying the size of the datasets. Results refer to structure/content-driven clustering experiments (i.e., $f \in [0.4, 0.6]$) and equally distributed in the network.

These results highlight the major advantage of *CXK-means* with respect to a centralized setting, which concerns a better runtime behavior. In fact, a higher number of nodes in the network leads to more parallelism, which results in a

drastic reduction of the overall time needed for the clustering task. However, as highlighted by the complexity analysis reported in Section 4.2, when the number of nodes grows up, the data exchanged among nodes grows up as well. This fact clearly affects negatively the network traffic (i.e., exchange of cluster representatives) which might not be negligible anymore. Indeed, as we can see in Figure 7 for all datasets, after a drastic reduction of the runtime due to just a few nodes, the runtime remains roughly constant for a certain range, then it starts to slightly increase when the number of nodes becomes significantly higher. It should be noted that the trends shown in Figure 7 are close to those expected, i.e., those theoretically derived by the complexity analysis in Section 4.2; in fact, after an initial hyperbolic decreasing behavior, the efficiency follows an increasing linear function.

Concerning the evaluation of the stabilization (saturation) points, we observed that time performances on *IEEE* tend to stabilize for 6 and 4 nodes, respectively in the case of full and halved datasets; similar trends are found for *Wikipedia* (8 and 6) and *Shakespeare* (9 and 5). On *DBLP*, time performances tend to stabilize for a smaller number of nodes (4 and 2, respectively) which is probably due to a smaller size of *DBLP* with respect to *IEEE*, in terms of both transactions and vocabulary of terms.

Another important remark is that as the dataset size is halved, the minimum number of nodes to bring down the clustering times tends to decrease. This further supports our study on the computational complexity of *CXK-means* reported in Section 4.2, in that the advantage of the distributed collaborative approach with respect to the centralized one tends to become less significant as the dataset size is reduced.

Effectiveness. Tables 1(a)–(c) report on accuracy results obtained on the various datasets by *CXK-means* when data is equally partitioned over the nodes. We varied the number of nodes and the type of clustering setting (i.e., structure-, content-, and structure/content-driven clustering). For the sake of presentation, here we show results for a maximum number of nodes equal to 9, since accuracy results for nodes from 11 to 19 followed similar trends.

For each dataset and clustering setting, results refer to multiple (10) runs of the algorithm and correspond to F-measure scores averaged over the range of f values specific of the clustering setting. Moreover, the best setting of parameter γ was found to be close to high values (typically above 0.85), for each dataset and type of clustering [33].

As it is reasonable to expect, the centralized case (i.e., one node) corresponds to an upper bound in terms of clustering quality for the collaborative distributed approach. While our focus is not on the effectiveness evaluation of the centralized case—the interested reader can find details in [33]—it can be noted how the clustering accuracy decreases as the number of nodes increases, regardless of the set and the type of clustering. However, this performance degradation remains relatively acceptable for a distributed environment, which is partly due to our model of cluster representative in achieving good quality summaries for the clusters. Indeed, loss of accuracy of *CXK-means* with respect

Table 1: Clustering accuracy results for data equally distributed over the nodes: (a) $f \in [0, 0.3]$ (content-driven similarity), (b) $f \in [0.4, 0.6]$ (structure/content-driven similarity), (c) $f \in [0.7, 1]$ (structure-driven similarity)

(a)

set	*# of clusters*	*# of nodes*	*F-measure (avg)*
DBLP	6	1	0.795
		3	0.730
		5	0.701
		7	0.639
		9	0.574
IEEE	8	1	0.629
		3	0.552
		5	0.514
		7	0.440
		9	0.396
Shakespeare	5	1	0.964
		3	0.902
		5	0.861
		7	0.832
		9	0.790
Wikipedia	21	1	0.834
		3	0.793
		5	0.768
		7	0.724
		9	0.698

(b)

set	*# of clusters*	*# of nodes*	*F-measure (avg)*
DBLP	16	1	0.803
		3	0.750
		5	0.716
		7	0.641
		9	0.585
IEEE	14	1	0.598
		3	0.524
		5	0.478
		7	0.423
		9	0.375
Shakespeare	12	1	0.772
		3	0.734
		5	0.701
		7	0.682
		9	0.659

(c)

set	*# of clusters*	*# of nodes*	*F-measure (avg)*
DBLP	4	1	0.991
		3	0.971
		5	0.935
		7	0.855
		9	0.751
IEEE	2	1	0.655
		3	0.572
		5	0.527
		7	0.453
		9	0.406
Shakespeare	3	1	0.681
		3	0.653
		5	0.638
		7	0.599
		9	0.572

to the centralized setting was always lower than 0.2 in relation to the number of nodes leading to the stabilization of efficiency performance determined in the previous paragraph (i.e., 4, 6, 9, and 8 for *DBLP*, *IEEE*, *Shakespeare*, and *Wikipedia*, respectively); precisely, the decrease in accuracy was roughly equal to 0.08 (*DBLP*), 0.14 (*IEEE*), 0.17 (*Shakespeare*), and 0.13 (*Wikipedia*).

We also evaluated clustering accuracy in case of data unequally distributed over the nodes in the network. As shown in Tables 2(a)–(c), results followed similar trends to those observed in the case of equal distribution of data over the nodes. For each set and type of clustering, we observed a slight degradation of accuracy with respect to the corresponding results achieved in the equally distributed case (Tables 1(a)–(c)). This can be explained since the local execution of *CXK-means* on nodes with few transactions produces a clustering solution that is less accurate with respect to the one produced by nodes having a higher number of transactions. However, as this performance degradation remains pretty small (from about 0.01 to 0.10), there is evidence to suggest that the global representative function is still able to produce high-accuracy cluster

Table 2: Clustering accuracy results for data unequally distributed over the nodes: (a) $f \in [0, 0.3]$ (content-driven similarity), (b) $f \in [0.4, 0.6]$ (structure/content-driven similarity), (c) $f \in [0.7, 1]$ (structure-driven similarity)

(a)

set	*# of clusters*	*# of nodes*	*F-measure (avg)*
DBLP	6	1	0.795
		3	0.657
		5	0.631
		7	0.575
		9	0.516
IEEE	8	1	0.629
		3	0.541
		5	0.504
		7	0.432
		9	0.388
Shakespeare	5	1	0.964
		3	0.857
		5	0.818
		7	0.790
		9	0.751
Wikipedia	21	1	0.834
		3	0.737
		5	0.714
		7	0.673
		9	0.649

(b)

set	*# of clusters*	*# of nodes*	*F-measure (avg)*
DBLP	16	1	0.803
		3	0.675
		5	0.645
		7	0.577
		9	0.527
IEEE	14	1	0.598
		3	0.514
		5	0.468
		7	0.414
		9	0.367
Shakespeare	12	1	0.772
		3	0.697
		5	0.667
		7	0.648
		9	0.626

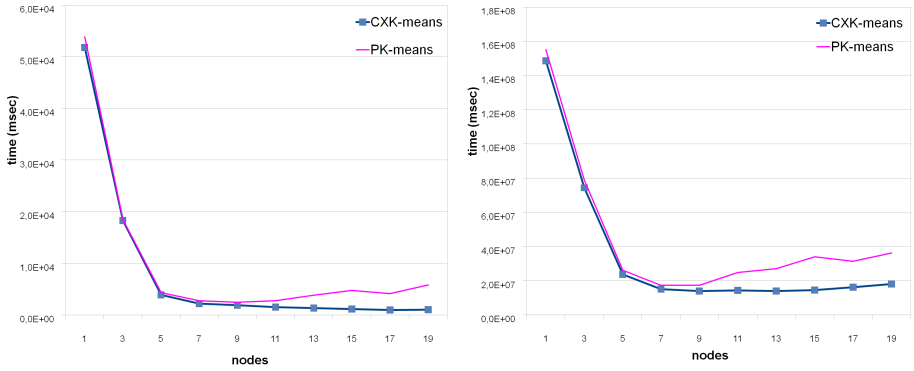
(c)

set	*# of clusters*	*# of nodes*	*F-measure (avg)*
DBLP	4	1	0.991
		3	0.874
		5	0.841
		7	0.769
		9	0.676
IEEE	2	1	0.655
		3	0.560
		5	0.516
		7	0.444
		9	0.398
Shakespeare	3	1	0.681
		3	0.620
		5	0.606
		7	0.569
		9	0.543

representatives even for nodes associated with a small portion of the data; a key role in this respect is played by the local cluster sizes (i.e., weights) that are taken into account by function ComputeGlobalRepresentative along with the local representatives outputted by each node.

In order to give just a brief summary of the accuracy results for nodes from 11 to 19, they continued to follow a decreasing trend, but the degradation was quite small with respect to the ones achieved by our *CXK-means* in a network with 9 nodes. In particular, the loss of accuracy for 19 nodes with respect to 9 nodes was of about 0.10 on average for all datasets.

Comparison with a non-collaborative distributed approach. Since our proposal is, to the best of our knowledge, the first that addresses a distributed collaborative approach to clustering XML documents, we resorted to parallel partitional clustering to select a competitor for this analysis. Specifically, we referred to the parallel K-means algorithm [11] as a baseline, non-collaborative method. While this algorithm and our *CXK-means* share the clustering strategy (i.e., centroid-based partitional clustering), we needed to adapt the former to



(a) (b)

Figure 8: Clustering time performances of *CXK-means* and *PK-means* by varying the number of nodes on (a) *DBLP* and (b) *IEEE*

allow it *(i)* to handle XML transactions and *(ii)* to cluster them in a P2P network.

To enable the parallel K-means (for short, *PK-means*) to deal with XML transactions, the algorithm was equipped with the notions of XML transaction similarity (instead of Euclidean distance) and XML cluster representative computation (instead of simple mean of vectors). As far as the second aspect, we adapted *PK-means*, which has been designed for multi-processor systems, to be executed in a distributed environment, particularly a P2P network. For this purpose, the multi-process architecture was mapped to the network nodes, each of which was associated with a local memory to store the data (to simulate the distributed memory environment), whereas the message passing paradigm adopted by the algorithm was implemented by exploiting the network communications.

Moreover, to ensure the performance of the two algorithms were compared fairly, the same initial configuration of clustering was set while varying the other parameters. Specifically, for each node in the network, the initial local cluster representatives were randomly chosen among the transactions in the local dataset, then both *CXK-means* and *PK-means* were fed with such transactions.

Figure 8 shows time performances of our *CXK-means* and *PK-means* on *DBLP* and *IEEE* by varying the number of nodes in the network. Results refer to the structure/content-driven clustering case (i.e., $f \in [0.4, 0.6]$) and data equally distributed in the network. In the figure, we can observe that our *CXK-means* behaved better than *PK-means* on both sets. The time performances of the two algorithms remained quite comparable for a relatively small number of nodes (i.e., from 1 to 11 on *DBLP*, and from 1 to 9 on *IEEE*), whereas the gap became larger when the number of nodes in the network increased (i.e., from 13 to 19 on *DBLP*, and from 11 to 19 on *IEEE*). This result emphasizes that the higher amount of information exchanged among the nodes when *PK-means* is carried out has a remarkable impact on the runtimes; in fact, the performance

degradation of *PK-means* is mainly due to the higher network traffic required for the communications. In addition, time profiles (on both the datasets) of our *CXK-means* followed a decreasing trend, which did not change significantly for a higher number of nodes; on the contrary, *PK-means* time performances had a notably increasing trend for network configurations with many nodes, which limits the algorithm execution to relatively smaller networks.

While efficiency analysis highlighted the advantage of our *CXK-means* with respect to *PK-means*, the accuracy results revealed the two algorithms are substantially comparable—actually, *CXK-means* performed slightly better than *PK-means*, with an average improvement of 0.03 over all datasets and network configurations. This result confirms that the collaborative strategy adopted by *CXK-means* in exchanging information among the nodes is extremely advantageous over a non-collaborative distributed approach.

6. Conclusion and Future Work

We presented a collaborative distributed framework for clustering XML documents; to the best of our knowledge, this is the first collaborative approach to clustering XML documents by structure and content in a distributed P2P environment. We developed a distributed, centroid-based partitional clustering algorithm, where cluster representatives are used to describe portions of the document collection and can conveniently be exchanged with other peers on the network. Each peer yields a local clustering solution over its own set of XML data, and exchanges the cluster representatives with other nodes. This sort of recommendation is used to compute global representatives, thus finally obtaining an overall clustering solution in a collaborative way. Experimental evidence has shown that the collaborative distributed approach outperforms the corresponding centralized clustering setting in terms of runtime behavior, paying a limited loss of accuracy.

We plan to extend our collaborative framework to deal with semantic information of both structural and content type from XML data according to the study provided in [33]. Also, it would be interesting to investigate how the proposed distributed clustering approach can help in the integration and classification of heterogeneous XML sources.

References

[1] Abiteboul, S., Manolescu, I., Polyzotis, N., Preda, N., Sun, C., 2008. XML Processing in DHT Networks. In: Proc. IEEE Int. Conf. on Data Engineering (ICDE). pp. 606–615.

[2] Abiteboul, S., Manolescu, I., Taropa, E., 2006. A Framework for Distributed XML Data Management. In: Int. Conf. on Extending Database Technology (EDBT). pp. 1049–1058.

[3] Aggarwal, C. C., Ta, N., Wang, J., Feng, J., Zaki, M., 2007. XProj: a Framework for Projected Structural Clustering of XML Documents. In: Proc. ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD). pp. 46–55.

[4] Antonellis, P., Makris, C., Tsirakis, N., 2008. XEdge: Clustering Homogeneous and Heterogeneous XML Documents using Edge Summaries. In: Proc. ACM Symposium on Applied Computing (SAC). pp. 1081–1088.

[5] Antonellis, P., Makris, C., Tsirakis, N., 2009. Utilizing XML Clustering for Efficient XML Data Management on P2P Networks. In: Proc. Int. Conf. on Database and Expert Systems Applications (DEXA). pp. 68–82.

[6] Arenas, M., Libkin, L., 2004. A Normal Form for XML Documents. ACM Trans. on Database Systems (TODS) 29 (1), 195–232.

[7] Baeza-Yates, R., Ribeiro-Neto, B., 1999. Modern Information Retrieval. ACM Press Books. Addison Wesley.

[8] Candillier, L., Tellier, I., Torre, F., 2005. Transforming XML Trees for Efficient Classification and Clustering. In: INEX Workshop. pp. 469–480.

[9] Costa, G., Manco, G., Ortale, R., Tagarelli, A., 2004. A Tree-based Approach to Clustering XML Documents by Structure. In: Proc. European Conf. on Principles and Practice of Knowledge Discovery in Databases (PKDD). pp. 137–148.

[10] Denoyer, L., Gallinari, P., 2008. Report on the XML Mining Track at INEX 2007: categorization and clustering of XML documents. Tech. rep.

[11] Dhillon, I. S., Modha, D. S., 1999. A Data-Clustering Algorithm on Distributed Memory Multiprocessors. In: ACM SIGKDD Workshop on Large-Scale Parallel KDD Systems. pp. 245–260.

[12] Dhillon, I. S., Modha, D. S., 2001. Concept Decompositions for Large Sparse Text Data Using Clustering. Machine Learning 42 (1/2), 143–175.

[13] Doucet, A., Lehtonen, M., 2006. Unsupervised Classification of Text-Centric XML Document Collections. In: INEX Workshop.

[14] Eisenhardt, M., Muller, W., Henrich, A., 2003. Classifying Documents by Distributed P2P Clustering. In: GI Jahrestagung (2). pp. 286–291.

[15] Hammouda, K., Kamel, M., 2006. Collaborative Document Clustering. In: Proc. SIAM Int. Conf. on Data Mining (SDM). pp. 451–461.

[16] Ignat, C.-L., Oster, G., 2008. Peer-to-peer collaboration over XML documents. In: Proc. Int. Conf. on Cooperative Design Visualization and Engineering (CDVE). pp. 66–73.

[17] Jain, A., Dubes, R., 1988. Algorithms for Clustering Data. Prentice-Hall advanced reference series. Prentice-Hall.

[18] Kargupta, R., Hanzaoglu, I., Stafford, B., 1997. Distributed data mining using an agent based architecture. In: Proc. European Conf. on Principles and Practice of Knowledge Discovery in Databases (PKDD). pp. 211–214.

[19] Koloniari, G., Pitoura, E., 2005. Peer-to-Peer Management of XML Data: Issues and Research Challenges. SIGMOD Record 34 (2), 6–17.

[20] Kutty, S., Tran, T., Nayak, R., Li, Y., 2008. Clustering XML Documents Using Closed Frequent Subtrees: A Structural Similarity Approach. In: Fuhr, N., Kamps, J., Lalmas, M., Trotman, A. (Eds.), Focused Access to XML Documents. Vol. 4862 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg, pp. 183–194.

[21] Larsen, B., Aone, C., 1999. Fast and effective text mining using linear-time document clustering. In: Proc. ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD). pp. 16–22.

[22] Li, J., Liu, Y., k. Liao, W., Choudhary, A., 2007. Parallel Data Mining Algorithms for Association Rules and Clustering. In: Rajasekaran, S., Reif, J. (Eds.), Handbook of Parallel Computing: Models, Algorithms and Applications. CRC Press.

[23] Lian, W., Cheung, D., Mamoulis, N., Yiu, S., 2004. An Efficient and Scalable Algorithm for Clustering XML Documents by Structure. IEEE Trans. on Knowledge and Data Engineering (TKDE) 16 (1), 82–96.

[24] Nayak, R., Xu, S., 2006. XCLS: A Fast and Effective Clustering Algorithm for Heterogeneous XML Documents. In: Proc. Pacific-Asia Conf. on Knowledge Discovery and Data Mining (PAKDD). pp. 292–302.

[25] Nierman, A., Jagadish, H., 2002. Evaluating Structural Similarity in XML Documents. In: Proc. ACM SIGMOD Int. Workshop on the Web and Databases (WebDB). pp. 61–66.

[26] Oikonomakou, N., Vazirgiannis, M., 2006. A Review of Web Document Clustering Approaches. In: Data Mining and Knowledge Discovery Handbook. Springer, Ch. 43, pp. 921–943.

[27] Polyzotis, N., Garofalakis, M., 2002. Structure and Value Synopses for XML Data Graphs. In: Proc. Int. Conf. on Very Large Data Bases (VLDB). pp. 466–477.

[28] Rao, P. R., Moon, B., 2009. Locating XML Documents in a Peer-to-Peer Network Using Distributed Hash Tables. IEEE Trans. on Knowledge and Data Engineering (TKDE) 21 (12), 1737–1752.

[29] Rodrigues, R., Druschel, P., October 2010. Peer-to-Peer Systems. Communications of the ACM 53 (10), 72–82.

[30] Steinbach, M., Karypis, G., Kumar, V., 2000. A Comparison of Document Clustering Techniques. In: Proc. of the KDD'00 Workshop on Text Mining.

[31] Strehl, A., Ghosh, J., Mooney, R., 2000. Impact of Similarity Measures on Web-page Clustering. In: Proc. AAAI Workshop on AI for Web Search. pp. 58–64.

[32] Tagarelli, A., Greco, S., 2006. Toward Semantic XML Clustering. In: Proc. SIAM Int. Conf. on Data Mining (SDM). pp. 188–199.

[33] Tagarelli, A., Greco, S., 2010. Semantic Clustering of XML Documents. ACM Trans. on Information Systems (TOIS) 28 (1).

[34] Tran, T., Nayak, R., Bruza, P., 2008. Combining Structure and Content Similarities for XML Document Clustering. In: Proc. of the 7th Australasian Data Mining Conference (AusDM). Vol. 87. pp. 219–226.

[35] Vercoustre, A. M., Fegas, M., Gul, S., Lechevallier, Y., 2005. A Flexible Structured-based Representation for XML Document Mining. In: INEX Workshop. pp. 443–457.

[36] Zhao, Y., Karypis, G., 2004. Empirical and Theoretical Comparison of Selected Criterion Functions for Document Clustering. Machine Learning 55 (3), 311–331.