Neurocomputing 121 (2013) 234–247

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

OClustR: A new graph-based algorithm for overlapping clustering

Airel Pérez-Suárez [a,b,*], José F. Martínez-Trinidad [a], Jesús A. Carrasco-Ochoa [a], José E. Medina-Pagola [b]

[a] Instituto Nacional de Astrofísica, Óptica y Electrónica, Luis Enrique Erro ♯1, Sta. María Tonantzintla, Puebla, CP: 72840, Mexico
[b] Centro de Aplicaciones de Tecnologías de Avanzada, 7ma A ♯21406, Playa, CP: 12200, Havana, Cuba

ARTICLE INFO

Article history:
Received 17 August 2012
Received in revised form
30 November 2012
Accepted 25 April 2013
Communicated by M. Sato-Ilic
Available online 11 June 2013

Keywords:
Data mining
Overlapping clustering
Graph-based algorithms

ABSTRACT

Clustering is a Data Mining technique, which has been widely used in many practical applications. From these applications, there are some, like social network analysis, topic detection and tracking, information retrieval, categorization of digital libraries, among others, where objects may belong to more than one cluster; however, most clustering algorithms build disjoint clusters. In this work, we introduce OClustR, a new graph-based clustering algorithm for building overlapping clusters. The proposed algorithm introduces a new graph-covering strategy and a new filtering strategy, which together allow to build overlapping clusterings more accurately than those built by previous algorithms. The experimental evaluation, conducted over several standard collections, showed that our proposed algorithm builds less clusters than those built by the previous related algorithms. Additionally, OClustR builds clusters with overlapping closer to the real overlapping in the collections than the overlapping generated by other clustering algorithms.

1. Introduction

Clustering is one of the most important techniques in Pattern Recognition and Data Mining. This technique aims at grouping a set of objects into a set of classes called *clusters*, such that objects belonging to the same cluster are more similar than objects belonging to different clusters [1].

There are several research areas where clustering has been applied successfully, for example: image segmentation [2], fingerprint classification [3], large scale data analysis [4] and social network analysis [5], among others. Although several clustering algorithms have been proposed in recent years, most of them build disjoint clusters; *i.e.*, they do not allow objects to belong to more than one cluster. However, there are some applications like social network analysis [6,7], information retrieval [8], text segmentation [9] and news stream analysis [10], among others, where it is common that objects belong to more than one cluster. For these kinds of applications, overlapping clustering is useful and important.

Although several clustering algorithms have been reported in the literature addressing the problem of overlapping clustering [5–7,10–27], all of them have some limitations which could reduce their application scope or their usefulness in practical problems. For this reason, in this work, the problem of overlapping clustering is addressed.

The main contribution of this paper is a new overlapping clustering algorithm, which is based on graph theory. The proposed algorithm, called OClustR (Overlapping Clustering based on Relevance), introduces a new graph-covering strategy and a new filtering strategy, which together allow obtaining a small set of overlapping clusters and reducing the limitations of the previous algorithms. These characteristics make the OClustR algorithm suitable for handling overlapping clustering in real applications like information organization [8], web document clustering [15] and text segmentation [9], among others.

We conducted several experiments, over several standard data collections, in order to compare OClustR against other algorithms of the state-of-the-art. These experiments show that our proposed algorithm builds more accurate clusters, according to the FBcubed evaluation measure [28], than those built by the previous overlapping clustering algorithms.

The remainder of this paper is organized as follows: in Section 2, the related work is outlined. In Section 3, we introduce the OClustR algorithm. The experimental evaluation, showing the performance of our proposed algorithm on several data collections, is presented in Section 4. Finally, the conclusions and some ideas about future work are presented in Section 5.

* Corresponding author at: Instituto Nacional de Astrofísica, Óptica y Electrónica, Luis Enrique Erro #1, Sta. María Tonantzintla, Puebla, CP: 72840, Mexico. Tel.: +537 272 1676; fax: +537 273 0045.
E-mail addresses: airel@ccc.inaoep.mx, airel26@gmail.com, asuarez@cenatav.co.cu (A. Pérez-Suárez), fmartine@inaoep.mx (J.F. Martínez-Trinidad), ariel@inaoep.mx (J.A. Carrasco-Ochoa), jmedina@cenatav.co.cu (J.E. Medina-Pagola).

0925-2312/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.neucom.2013.04.025

2. Related work

In recent years, several algorithms for overlapping clustering have been reported in the literature [5–7,10–27]. These algorithms are different wrt. the model they use to represent a data collection and the criteria they use to build clusters. The DHS algorithm, reported in [29], was not included as related work because it addresses the problem of overlapping hierarchical clustering, which is out of the scope of this paper.

Star [12], Estar [13,14], Gstar [19], ACONS [20], ICSD [24], SimClus [26] and DCS [27] are *graph-based* clustering algorithms, which are able to build overlapping clusterings. All these algorithms represent the collection of objects as a *thresholded similarity graph* $G_\beta = \langle V, E_\beta \rangle$, and they build clusterings through a *cover* of G_β [12]; *i.e.*, a vertex cover. For covering G_β, Star, Estar, ACONS, ICSD, DCS and SimClus use a special kind of sub-graph, called *star-shaped sub-graph* (s-graph) [12]. On the other hand, for covering G_β, Gstar defines a new kind of sub-graph, called *generalized star-shaped sub-graph* (g-graph) [19]. In this context, each s-graph or g-graph selected is interpreted as a cluster.

All these graph-based algorithms follow a *greedy* heuristic for covering the similarity graph; however, each one of them employs a different criterion for ordering and selecting the sub-graphs used for covering G_β. The ICSD and DCS algorithms use the same strategy for building the clustering but ICSD is an incremental algorithm and DCS is a dynamic algorithm. Besides, the SimClus algorithm uses the same strategy and criterion for building the clustering as the one used in Estar algorithm proposed by Gil-García et al. in [14]. After performing the covering process, there are algorithms like Gstar, ACONS, ICSD and DCS that process the selected sub-graphs in order to remove those having all their vertices covered by other sub-graphs; for these algorithms, the remaining sub-graphs constitute the final clustering. The computational complexity of Star, ICSD and DCS is $O(n^2)$; the computational complexity of Estar, Gstar, and ACONS is $O(n^3)$.

These algorithms have several limitations. First of all, as we will demonstrate in our experiments, they produce a large number of clusters. It is known that, in real problems, the correct number of clusters is unknown. However, when we use a clustering algorithm in order to discover hidden relations among objects in a collection, the number of clusters obtained should be small wrt. the number of objects in the collection. Note that, if the number of clusters grows, then analyzing those clusters could be as difficult as analyzing the whole collection.

Another limitation of these algorithms, as we will show in our experiments, is that they build clusterings with high overlapping. As it was mentioned in the introduction, to obtain overlapping clusters is very useful for several applications; however, when the overlapping is high, it could be difficult to obtain something useful about the structure of the data. Besides, there are applications like document segmentation by topic, where a high overlapping could be a signal of a bad segmentation [9]. A similar example can be found also in the context of social networks analysis [21,23].

Finally, the Gstar algorithm has an additional limitation; it needs to store a high amount of information for each vertex in G_β; therefore, it could be inefficient for large collections. Besides, as it was showed in [19], Estar could leave uncovered objects; *i.e.*, objects that do not belong to any cluster.

Another algorithm able to build overlapping clusters is STC [11], which was developed for clustering *snippets*. The Snippets are small texts used by systems like Google to describe the results of a web search. For building a clustering, STC builds a *suffix tree* [30] containing the suffixes of all the snippets in the collection. All nodes of this tree, containing two or more snippets, are used as seeds for building the clusters. Finally, STC builds the clusters by iteratively merging some of the previously detected seeds, following a strategy similar to that used by the Single-link algorithm [31]. Even when the authors of STC claim that the computational complexity of STC is $O(n \cdot \log n)$, as it was mentioned in [15], the complexity of STC could get to be exponential, depending on the number of suffixes contained in all the snippets.

The main limitation of STC is related to the construction of the suffix tree. Although the construction of this tree depends on the amount of snippets to cluster, generally, it could be very expensive when the number of snippets grows. Besides, STC was specifically designed to work with text strings which limits its applicability.

Another example of an overlapping clustering algorithm is the Incremental Strong Compact algorithm (ISC) [10]. ISC is a graph-based algorithm that represents a collection of objects as a *maximal thresholded similarity graph* $G_{max-\beta} = \langle V, E_{max-\beta} \rangle$; unlike graphs used by the previous overlapping graph-based algorithms, $G_{max-\beta}$ is a directed graph. For building a clustering, first of all, ISC builds the set of all connected components of $G_{max-\beta}$, disregarding the orientation of the edges. Afterwards, it extracts the *strong compact sets* (sc-set) covering each connected component; in this context, each sc-set constitutes a cluster. The computational complexity of ISC is $O(n^2)$.

ISC has the same limitations of the previous graph-based algorithms; this means, ISC builds a large number of clusters with high overlapping.

Another algorithm addressing the problem of overlapping clustering is SHC [15]. SHC is an incremental clustering algorithm based on the concept of *Histogram Ratio* of a cluster. The histogram ratio of a cluster is a measure of cluster cohesiveness. For clustering an object collection, SHC processes the objects in an incremental way. For each object o, SHC computes the most suitable clusters in which o should be added to; for this purpose, SHC analyzes how much the histogram ratio of each cluster varies if o is added to it. Finally, after checking all clusters, if o was not added to any cluster then o constitutes a new cluster. The computational complexity of SHC is $O(n^2)$.

SHC has several limitations. First, it needs to tune the values of several parameters (β, ϵ and HR_{min}), whose values depend on the collection to process. In general, the users do not have any a priori knowledge about the collection they want to cluster; therefore, to tune up several parameters could be a difficult task. Second, as we will show in our experiments, SHC builds clusterings with high overlapping.

Other overlapping clustering algorithms have been reported in the context of overlapping community detection in complex networks [5–7,16–18,21–23,25]. In this research area, a network is represented as a graph (undirected or directed, depending the specific problem), where vertices are the objects of study and edges are links between the objects.

One of the first algorithms proposed in this area was introduced by Palla et al. in 2005, for studying social and biological networks [16]. This algorithm works on binary networks (*i.e.*, with undirected and unweighted edges) and it defines a community or cluster as the union of all the *k-cliques* (*i.e.*, complete sub-graphs of size k), that can be reached from each other, through a series of adjacent *k-cliques* (where adjacency means sharing $k-1$ vertices). Thus, this method first locates all the *cliques* (*i.e.*, maximal complete sub-graphs) of the network and then, it identifies the communities by carrying out a standard component analysis of the clique–clique overlap matrix [32]. The computational complexity of this method is exponential [16]. The main limitation of this method is its computational complexity, which makes it inapplicable for real problems involving large networks.

Other algorithms for overlapping community detection are RaRe-IS [18] and LA-IS^2 [17]. These algorithms represent a network as an undirected and unweighted graph, whose vertices represent individuals, web pages, etc.; and edges represent communications or links between the vertices.

RaRe-IS [18] consists of two steps: initialization and improvement. In the initialization step, this algorithm applies a method,

named RaRe, which builds a set of *seed* clusters. For building these seed clusters, RaRe begins by ranking the vertices in the graph, according to a predefined criterion (*e.g.*, vertex degree or Page Rank [33]). Afterwards, some vertices with high rank are iteratively removed until the graph became decomposed in connected components with size between an upper and a lower predefined bounds. After that, each one of the removed vertices is added to a component only if its density (or any other metric) is improved. Once the seed clusters have been built, in the improvement step, the algorithm applies a method named IS, over each seed cluster. The IS method updates each seed cluster by adding or removing one vertex at a time, as long as a metric, over the seed cluster, strictly improves its value (*e.g.*, the cluster density). The computational complexity of RaRe-IS is $O(n^2)$. The main limitation of this algorithm is that it needs to tune the values of several parameters: *t*, *min* and *max* for RaRe method; and *max_fail* for IS method. Besides, as it was pointed out in [7], RaRe-IS may produce clusterings with high overlapping.

The algorithm LA-IS^2 follows the same idea of RaRe-IS, but it introduces new methods for the initialization and improvement steps. For building the seed clusters, in the initialization step, LA-IS^2 applies a method named LA (from *Link Aggregate*). This method starts by ordering the vertices by decreasing Page Rank [33]. Then, LA uses a strategy in which the ordered vertices are processed sequentially and added to a cluster only if this addition improves the cluster density. If a vertex was not added to any cluster, it constitutes a new cluster. Once the seed clusters have been built, LA-IS^2 sequentially processes each seed by applying a method named IS^2. This method updates a seed by adding or removing one vertex at a time, as long as the cluster density strictly improves its value. Unlike IS used by the RaRe-IS algorithm [18], IS^2 adds to a seed cluster only those vertices adjacent to vertices in the seed. The computational complexity of LA-IS^2 is $O(n^2)$.

The main limitation of LA-IS^2 is that it makes irrevocable assignment of vertices to clusters; that is, once a vertex has been assigned to a cluster it cannot be removed and added to another cluster even if doing it improves the clustering quality.

Another overlapping clustering algorithm is LA-CIS, proposed by Goldberg et al. in [7]. This algorithm follows the same idea of LA-IS^2 [17], but instead of applying IS^2 in the improvement step, it applies a variation of the IS method (see above), called CIS. There are two main differences between IS and CIS. First, before updating a seed cluster, CIS sorts the vertices according to their degree. Second, in CIS the connectivity of a cluster is examined every time a whole scan finishes. If the cluster consists of multiple connected components, then it is replaced by the connected component having the highest density. The computational complexity of LA-CIS is $O(n^2)$. Like LA-IS^2, the main limitation of this algorithm is that it makes irrevocable assignment of vertices to clusters. Besides, it may produce clusterings with high overlapping.

Other overlapping clustering algorithms, used for complex networks analysis, are CONGA [21] and CONGO [23]. These algorithms represent a network as an undirected and unweighted graph, and they are variations of the GN algorithm [34]. The GN algorithm builds a set of disjoint clusters using a strategy of four steps: (1) calculate *edge betweenness* (EB, for short) for each edge in the network, (2) find the edge having the highest EB and remove it, (3) recalculate the EB for all remaining edges, and (4) repeat steps 2 and 3 until no edges remain.

The CONGA [21] algorithm extends the GN algorithm, allowing overlapping clusterings, by introducing the concept of *split betweenness* (SB, for short) of vertices; this concept provides a way to decide when to split a vertex and which vertex to split. Thus, CONGA modifies the GN algorithm so that, in step 1, CONGA calculates the EB of each edge as well as the SB of each vertex. Afterwards, in step 2, CONGA removes the edge with the highest EB or the vertex with the highest SB. In the third step, the EB of each edge and the SB of each vertex are recalculated; the fourth step remains equal. The computational complexity of CONGA is $O(n^6)$ for standard networks and it could get to be $O(n^3)$ for sparse networks.

The main limitation of this method is its computational complexity, which makes it unapplicable for real problems involving networks with high number of vertices and edges. Besides, CONGA needs to know in advance the number of clusters to build; however, this number is commonly unknown in real problems.

The CONGO [23] algorithm extends CONGA by introducing the concept of *local betweenness*. This concept helps, at step 3 of CONGA, neither recomputing EB for all the edges nor recomputing the SB for all the vertices. Instead, only a small region around the removed edge or the split vertex is recomputed. Steps 1, 2 and 4 in CONGO algorithm are the same as in CONGA algorithm. The computational complexity of this algorithm is $O(n^4)$ for standard networks and it could get to be $O(n^2 \cdot \log\ n^2)$ for sparse networks, but making some hard assumptions as it is shown in [23]. The main limitation of the CONGO algorithm is its high computational complexity. Besides, as in CONGA, CONGO needs to know in advance the number of clusters to build.

Another algorithm able to produce overlapping clustering is H-FOG [6]. This algorithm transforms the problem of clustering the set of vertices in a network into the problem of clustering the set of links in a network; in this way, it allows overlapping among clusters. H-FOG is based on probability and it uses a strategy similar to the Single-link algorithm [35]. The computational complexity of H-FOG is $O(n^6)$. The main limitation of this method is its computational complexity. Besides, it needs to know in advance the number of clusters to build.

Another overlapping clustering algorithm is RRW [25], developed for discovering complexes and pathways within large-scale protein networks. This algorithm starts by building for each vertex in the network an *affinity vector*, through the random walk technique. After that, it uses the vector representing each vertex in order to build the strong connected component of the graph; in this context, each component constitutes a cluster. Afterwards, it sorts the clusters according to their statistical significance and it post-processes the resulting set of clusters by removing those clusters that do not satisfy a predefined overlap threshold. The computational complexity of this algorithm is $O(n^2)$. The main limitation of RRW is that it needs to tune the values of at least four parameters.

Another algorithm which builds overlapping clustering is SSDE-Cluster [5]. This algorithm was developed for social networks analysis and it represents the network as an undirected and weighted graph. For building a clustering this algorithm follows three steps. In the first step, SSDE-Cluster uses Sampled Spectral Distance Embedding (SSDE) to approximately embed the graph in $d << n$ dimensions [36]; where n is the number of vertices of the graph representing the network. In the second step, the vertices are clustered through a Gaussian Mixture Model (GMM), trained using the E-M algorithm. The GMM posterior probabilities are then used, in the third step, to compute overlapping clusters. The computational complexity of this algorithm is $O(n \cdot k \cdot d)$; k being the predefined number of clusters. The main limitation of SSDE-Cluster is that it needs to tune the values of at least four parameters, including the number of clusters to build.

Another clustering algorithm for detecting overlapping communities in complex networks was proposed by Zhang et al. in [22]. This algorithm uses *spectral mapping* over the adjacency matrix of the network, in order to compute K eigenvectors representing the information of the network. Then, the Fuzzy C-Means algorithm is used for clustering these eigenvectors in k clusters, where $2 \leq k \leq K$; all these fuzzy partitions are converted into overlapping clusterings using a parameter λ. Finally, the overlapping clustering which maximizes a variation of the modular function proposed in [34] is selected as the

final clustering. A key limitation of this algorithm is its computational complexity which, in the best scenario, is $O(n^2 \cdot K \cdot h)$, h being the iterations of the Fuzzy C-Means algorithm. Another limitation of this algorithm is that it needs to tune the values of several parameters.

As it can be seen from the related work, there are many algorithms addressing the problem of overlapping clustering. However, as we pointed out in this section, they have several limitations. These limitations are mainly related to: (a) the production of a large number of clusters, (b) the necessity of tuning several parameters whose values depend on the collection to be processed, and (c) the production of clusterings with high overlapping. Besides, there are several algorithms having a high computational complexity; this makes these algorithms worthless for many real problems.

The algorithm proposed in this work solves the aforementioned limitations while it has an acceptable computational complexity. Moreover, as we will show in our experiments, our algorithm builds more accurate clusterings than those built by the previous algorithms.

3. Overlapping clustering based on *relevance*

In this section, a new clustering algorithm for building overlapping clustering is introduced. This algorithm, called OClustR (Overlapping Clustering based on Relevance), represents a collection of objects as a *weighted thresholded similarity graph* $\tilde{G}_\beta$ and it builds an overlapping clustering in two steps. In the first step, an initial set of clusters is generated through a cover of $\tilde{G}_\beta$, using a special kind of sub-graphs named *weighted star-shaped sub-graphs* (ws-graphs). For building this cover, OClustR introduces a new criterion that allows selecting the ws-graphs needed for covering $\tilde{G}_\beta$. Afterwards, in order to build the final clustering, the initial clusters are improved in a second step. For improving the clusters, OClustR introduces a strategy, which aims at reducing both the number of clusters and the overlapping.

The presentation of OClustR is divided in four parts. First of all, in Section 3.1, we give some basic concepts needed for introducing our algorithm. After, in Section 3.2, we explain how OClustR obtains the initial set of clusters and, in Section 3.3, we explain how OClustR improves the initial clusters in order to build the final clustering. Finally, in Section 3.4 the complexity of the proposed algorithm is analyzed.

3.1. Basic concepts

Let $O = \{o_1, o_2, \ldots, o_n\}$ be a collection of objects, $\beta \in [0, 1]$ a given parameter and $S(o_i, o_j)$ a similarity function, such that $\forall o_i, o_j \in O, o_i \neq o_j$, $S(o_i, o_j) = S(o_j, o_i)$.

A *weighted thresholded similarity graph* is an undirected and weighted graph $\tilde{G}_\beta = \langle V, \tilde{E}_\beta, S\rangle$, such that $V = O$ and there is an edge $(v, u) \in \tilde{E}_\beta$ iff $S(v, u) \geq \beta$; each edge $(v, u) \in \tilde{E}_\beta, v \neq u$ is labeled with the value of $S(v, u)$.

The *set of adjacent vertices* of a vertex $v \in V$, denoted as $v.Adj$, is the set of vertices $u \in V$, such that there is an edge $(v, u) \in \tilde{E}_\beta$. Any vertex v having $v.Adj = \varnothing$ will be known as *isolated*. In addition, the cardinality of $v.Adj$ is known as the *degree* of v.

Let $\tilde{G}_\beta = \langle V, \tilde{E}_\beta, S\rangle$ be a weighted thresholded similarity graph. A *weighted star-shaped sub-graph* (*ws-graph*) in $\tilde{G}_\beta$, denoted by $G^\star = \langle V^\star, E^\star, S\rangle$, is a sub-graph of $\tilde{G}_\beta$, having a vertex $c \in V^\star$, such that there is an edge between c and all the other vertices in $V^\star \setminus \{c\}$. The vertex c is called the *center* of the ws-graph and the remaining vertices are called *satellites*. Isolated vertices are considered *degenerated* ws-graphs.

Let $\tilde{G}_\beta = \langle V, \tilde{E}_\beta, S\rangle$ be a weighted thresholded similarity graph and $W = \{G_1^\star, G_2^\star, \ldots, G_k^\star\}$ be a set, such that each $G_i^\star = \langle V_i^\star, E_i^\star, S\rangle$, $\forall i = 1 \ldots k$, is a ws-graph of $\tilde{G}_\beta$. The set W is a *cover* of $\tilde{G}_\beta$ iff it meets that $V = \cup_{i=1}^{k} V_i^\star$. In addition, we will say that a ws-graph $G_i^\star$ *covers* a vertex v iff $v \in V_i^\star$.

Let $G_v^\star = \langle V_v^\star, E_v^\star, S\rangle$ be a non-degenerated ws-graph, having v as its *center*. The intra-cluster similarity of $G_v^\star$, denoted as $Intra\_sim(G_v^\star)$, is the average similarity between all pair of vertices belonging to $V^\star$ [37], that is

$$Intra\_sim(G_v^\star) = \frac{\sum_{z,u \in V_v^\star, z \neq u} S(z, u)}{\frac{|V_v^\star| \cdot (|V_v^\star| - 1)}{2}} \tag{1}$$

Computing $Intra\_sim(G_v^\star)$ using (1) is $O(n^2)$. Thus, using this equation for computing the intra-cluster similarity of the ws-graph determined by all vertices in $\tilde{G}_\beta$ becomes $O(n^3)$. Since we want to reduce the computational complexity of OClustR, we should find an efficient approximation of (1) in order to use it in our algorithm.

From the definition of weighted thresholded similarity graph, we have that all edges (v, u) such that $S(v, u) < \beta$, do not belong to $\tilde{G}_\beta$. Since any ws-graph $G_v^\star$ is a sub-graph of $\tilde{G}_\beta$, these edges do not belong to $G_v^\star$ either. Therefore, if these edges are not taken into account in the computation of $Intra\_sim(G_v^\star)$, we could rewrite (1) as follows:

$$Aprox\_Intra\_sim(G_v^\star) = \frac{\sum_{(z,u) \in E_v^\star} S(z, u)}{\frac{|V_v^\star| \cdot (|V_v^\star| - 1)}{2}} \tag{2}$$

Even when we save some calculations, Eq. (2) is still $O(n^2)$. However, it can be noticed from (2) that, for a given ws-graph $G_v^\star$, the value of the numerator of $Aprox\_Intra\_sim(G_v^\star)$ only depends on the number of edges of $G_v^\star$, as well as on their weights. Therefore, based on the definition of ws-graph, we know that the lowest value of $Aprox\_Intra\_sim(G_v^\star)$ is reached when the edges in $E_v^\star$ are only those between the center v and the satellites of $G_v^\star$. Assuming this worst scenario, we could rewrite (2) as follows:

$$Aprox\_Intra\_sim(G_v^\star) = \frac{\sum_{u \in V_v^\star, u \neq v} S(v, u)}{\frac{|V_v^\star| \cdot (|V_v^\star| - 1)}{2}} \tag{3}$$

Computing $Aprox\_Intra\_sim(G_v^\star)$ using (3) is $O(n)$. Thus, we can use (3) for computing $Aprox\_Intra\_sim(G_v^\star)$ for each ws-graph $G_v^\star$ in $\tilde{G}_\beta$, for saving computational time. However, the denominator of (3) grows faster than its numerator. Therefore, a minor increase in the cardinality of $V^\star$ could represent a big decrease in the value of $Aprox\_Intra\_sim(G_v^\star)$. Moreover, the above mentioned characteristic of (3) makes the value of $Aprox\_Intra\_sim(G_v^\star)$ to be near to 0, for larger values of $|V_v^\star|$. This could represent a bias for ws-graphs with many satellites (*i.e.*, big ws-graphs) and also, it makes difficult the comparison between values of the *Aprox_Intra_sim* for two big ws-graphs. Based on this, we substitute the denominator of (3) by $(|V_v^\star| - 1)$; that is, to approximate the value of $Aprox\_Intra\_sim(G_v^\star)$ as the average weight of the edges existing in $G_v^\star$ between v and the satellites. Then, we could rewrite (3) as follows:

$$Aprox\_Intra\_sim(G_v^\star) = \frac{\sum_{u \in V_v^\star, u \neq v} S(v, u)}{|V_v^\star| - 1} \tag{4}$$

3.2. Building the initial clusters

We can build a covering of $\tilde{G}_\beta$ by finding the *minimum vertex cover* of $\tilde{G}_\beta$. The *minimum vertex cover* of $\tilde{G}_\beta = \langle V, \tilde{E}_\beta, S\rangle$ is the minimum set of vertices $M \subseteq V$, such that any vertex in $\tilde{G}_\beta$ belongs to M or it has an adjacent vertex belonging to M. However, finding the minimum vertex cover of $\tilde{G}_\beta$ is a NP complete problem [13]; therefore, we propose to approximate the minimum vertex cover of $\tilde{G}_\beta$ by covering $\tilde{G}_\beta$ using ws-graphs.

As it can be seen from the previous subsection, a ws-graph is determined by its center; thus, the problem of building a set $W = \{G_{c_1}^\star, G_{c_2}^\star, \ldots, G_{c_k}^\star\}$ of ws-graphs, such that W is a cover of $\tilde{G}_\beta$,

can be seen as the problem of building a set $C=\{c_1,c_2,\ldots,c_k\}$ such that $c_i \in C$ is the center of $G^{\star}_{c_i} \in W$, $\forall i=1,\ldots,k$. Since each vertex in $\tilde{G}_\beta$ can form a ws-graph, all vertices in $V^{\star}$ should be analyzed to build the set C. For pruning the search space and to establish a criterion for selecting the vertices that should be included in C, OClustR introduces the concept of *relevance* of a vertex. For defining the *relevance* of a vertex v, first we will define the *relative density* and the *relative compactness* of a vertex v.

An idea for reducing the number of ws-graphs needed for covering $\tilde{G}_\beta$ would be to iteratively select the vertices with higher degree. In this way, we try maximizing the number of vertices that is added to the cover of $\tilde{G}_\beta$ on each iteration. However, suppose that the vertices $v_1,v_2,\ldots,v_k$ were selected in the first k iterations. Let u be the vertex, among those non-selected vertices, having the highest degree in the iteration $k+1$. If u is adjacent to d vertices of the k previously selected vertices and u is added to C, only $|u.Adj|-d$ vertices would be added to the cover of $\tilde{G}_\beta$; the other vertices adjacent to u were added to the cover of $\tilde{G}_\beta$ in the previous iterations. But, if in the iteration $k+1$ would exist another vertex z such that: (i) $|z.Adj|<|u.Adj|$, (ii) z is adjacent to d_1 of the previously selected vertices, and (iii) $|u.Adj|-d<|z.Adj|-d_1$, then selecting vertex z in the iteration $k+1$ would be a better choice than selecting u.

Now, suppose that in the iteration $k+1$, there are two vertices v and u, such that v can add six vertices to the covering and u can add four vertices to the covering. Also suppose that $|v.Adj|=10$ and $|u.Adj|=5$. If we make our decision based on the number of vertices that each vertex can add to the covering, the choice would be the vertex v. Nevertheless, vertex u can cover a greater percent of its adjacent vertices than vertex v (4/5 versus 6/10). Motivated by the aforementioned ideas, we introduce the concept of *relative density* of a vertex.

The *relative density* of a vertex $v \in V$, denoted as *v.densityR*, is computed as follows:

$$v.densityR = \frac{v.density}{|v.Adj|}$$

where *v.density* denotes the number of adjacent vertices of v having a degree not greater than the degree of v. The density of v expresses how many adjacent vertices could include v in the cover of $\tilde{G}_\beta$. The relative density determines the ratio between the density of v and the number of adjacent vertices of v; this measure takes values in the interval [0,1]. The higher the value of *v.densityR*, the greater the number of adjacent vertices that v could include in the cover of $\tilde{G}_\beta$ and therefore, the better v is for covering $\tilde{G}_\beta$.

It is important to highlight that the vertices that were not counted in *v.density* are those selected in the previous iterations; therefore, a high value of *v.densityR* also means that the ws-graph determined by v has a low overlapping with the previous selected ws-graphs. In this way, we also check the overlapping among the selected ws-graphs.

Relative density can be used for reducing the number of ws-graphs needed for covering $\tilde{G}_\beta$ and for selecting ws-graphs with low overlapping. However, since this property is mainly based on the degree of vertices, using the relative density could lead to select ws-graphs with a high number of satellites but with a low average similarity among them. For solving this issue we define the concept of *relative compactness* of a vertex, which also takes into account the *Aprox_Intra_sim* of a ws-graph (computed using Eq. (4)).

The *relative compactness* of a vertex $v \in V$, denoted as *v.compactnessR*, is computed as follows:

$$v.compactnessR = \frac{v.compactness}{|v.Adj|}$$

where *v.compactness* denotes the number of vertices $u \in v.Adj$ such that $Aprox\_Intra\_sim(G^{\star}_v) \geq Aprox\_Intra\_sim(G^{\star}_u)$, being $G^{\star}_v$ and $G^{\star}_u$ the ws-graphs determined by v and u, respectively. The compactness of v expresses, from a different point of view, how many adjacent vertices could include v in the cover of $\tilde{G}_\beta$. Similar to *v.densityR*, the relative compactness takes values in [0,1] and the higher the value of *v.compactnessR*, the greater the number of adjacent vertices that v could include in the cover of $\tilde{G}_\beta$ and therefore, the better v for covering the graph.

The *relevance* of a vertex v, denoted as *v.relevance*, combines the relative density and relative compactness as the average of *v.densityR* and *v.compactnessR*. Since both *v.densityR* and *v.compactnessR* take values in [0,1], the relevance of a vertex also takes values in [0,1]. Besides, from this definition it can be inferred that a high value of relevance will correspond with vertices having high values of *v.densityR* and/or *v.compactnessR*; therefore, the higher the value of *v.relevance* the better v is for covering $\tilde{G}_\beta$. From the above comment, we can conclude that we must select the vertices in descending order according to their relevance. Besides, since vertices having $v.relevance=0$ are those vertices having $v.densityR=0$ and $v.compactnessR=0$, we do not have to verify these vertices in order to build the covering of $\tilde{G}_\beta$. Thus, we reduce the number of vertices that should be revised for building the cover of $\tilde{G}_\beta$.

The strategy OClustR proposes for covering $\tilde{G}_\beta$ has three steps. In the first step, all vertices having relevance greater than zero are added to a *list of candidates L*; isolated vertices are included directly in C since they are degenerated ws-graphs. Afterwards, in the second step, the list L is sorted in decreasing order, according to the relevance of the vertices. Finally, in the third step the vertices of L are iteratively visited in this order. Each vertex $v \in L$ is added to C if it satisfies at least one of the following conditions:

(a) v is not covered yet.
(b) v is already covered but it has at least one adjacent vertex u which is not covered yet. This condition avoids the selection of ws-graphs having all their satellites covered by previously selected ws-graphs.

After this process, the ws-graph formed by each vertex included in C constitutes an initial cluster.

3.3. Improving the initial clusters

Once the set C is built, we analyze C in order to remove the vertices forming less useful ws-graphs. In this context, the usefulness of a ws-graph $G^{\star}$ will be assessed based on how many of its satellites it shares with other selected ws-graphs.

The strategy used for covering $\tilde{G}_\beta$ is *greedy*; thereby, it could happen that some vertices initially added to C are no longer necessary for covering $\tilde{G}_\beta$; *i.e.*, they could be removed from C and the remaining vertices still cover $\tilde{G}_\beta$ completely. For instance, if there is one vertex $v \in C$ such that all the satellites of $G^{\star}_v$, as well as v belong to at least another previously selected ws-graph, then v could be removed from C and the remaining vertices in C still cover $\tilde{G}_\beta$ completely. On the contrary, if v belongs to at least another previously selected ws-graph but its satellites do not belong, v cannot be removed from C. Notice that, removing v from C in this last situation will leave the non-shared satellites of $G^{\star}_v$ uncovered. Let $v \in C$ be a vertex which determines the ws-graph $G^{\star}_v$ in the covering of $\tilde{G}_\beta$. Additionally, let *v.Shared* be the set of satellites that $G^{\star}_v$ shares with other selected ws-graphs and let *v.Non_shared* be the set of satellites belonging only to $G^{\star}_v$. In this work, we will understand that $G^{\star}_v$ is not *useful* for covering $\tilde{G}_\beta$ iff the following conditions are met: (1) there is at least another selected ws-graph containing v as a satellite, and (2) $|v.Shared|>|v.Non\_shared|$.

Non-useful ws-graphs increase the overlapping of the initial set of clusters; therefore, removing these ws-graphs would help to reduce the number of clusters as well as their overlapping. However, for removing these ws-graphs we need to add all their non-shared satellites to other clusters.

Since a non-useful ws-graph $G_v^\star$ meets that its center v belongs to at least another ws-graph $G^\star$, the non-shared satellites of $G_v^\star$ could be added to the cluster defined by $G^\star$. If there are more than one ws-graph covering vertex v, then the non-shared satellites will be added to the ws-graph having the greatest number of satellites among the candidates; thus, we allow the creation of clusters with many objects. If there is a tie then any of the ws-graphs having the greatest number of satellites can be selected. In this work, for simplicity, we select the first of these ws-graphs. When the non-shared satellites of a non-useful ws-graph must be added to another ws-graph $G_u^\star$, we add those satellites to a list named *u.Linked*; thus, the cluster determined by the ws-graph $G_u^\star$ now will include also the vertices in *u.Linked*. Hereinafter, the vertices added to *u.Linked* will be known as the *linked satellites* of $G_u^\star$. It is important to understand that we use the term *linked* in an illustrative way and that we already do not create any edge between u and any of the vertices in *u.Linked*.

The strategy proposed for removing the non-useful ws-graphs is composed of two steps. In the first step, we mark all the vertices in C as *not-analyzed* and sort C in decreasing order, according to the degree of the vertices. After, in the second step each vertex $v \in C$ is visited for removing from *v.Adj* all the vertices forming non-useful ws-graphs. For doing this, each vertex $u \in v.Adj$ is processed as follows: if u already belongs to C and it is yet marked as *not-analyzed*, then we check if $G_u^\star$ is not useful for covering $\tilde{G}_\beta$, according to the above definition. The vertices belonging to C and marked as *analyzed* are those which were determined as useful in a previous iteration and therefore, they do not need to be verified again. It is important to notice that, since u belongs to $G_v^\star$, it already meets condition (1). Thus, we only need to verify condition (2) on $G_u^\star$ and this is $O(n)$. Besides, we would like to highlight that, since vertices in C were ordered in descending order of their degree, $G_v^\star$ is the ws-graph having the greatest number of satellites that covers u. Finally, if $G_u^\star$ is not useful, u is removed from C and the vertices in *u.Shared* are added to *v.Linked*; otherwise, if $G_u^\star$ is useful, vertex u is marked as *analyzed*. Once all vertices in *v.Adj* were visited, the set formed by vertex v, together with vertices in *v.Adj* and *v.Linked*, constitutes a cluster in the final clustering.

The pseudocode of OClustR algorithm is showed in Algorithm 1.

Algorithm 1. OClustR algorithm.

Input: $O = \{o_1, o_2, \ldots, o_n\}$ - a collection of objects, β - a similarity threshold
Output: SC - a set of clusters

```
// Building candidate list
"Build G̃_β from collection O, using the similarity threshold β";
"Compute the relevance of each vertex in G̃_β";
C := {v ∈ V | v.Adj = ∅};
L := {v ∈ V | v.relevance > 0};
// Covering graph G̃_β
"Sort L in descending order by relevance";
foreach vertex v ∈ L do
    if v meets condition a) or b) then C := C ∪ {v};
end
// Removing non-useful ws-graphs
SC := ∅;
"Sort C in descending order by degree";
"Mark each vertex in C as not-analyzed";
foreach vertex v ∈ C do
    v.Linked := ∅;
    foreach vertex u ∈ v.Adj do
        if u ∈ C and u is marked as not-analyzed then
            if the ws-graph determined by u is not useful then
                v.Linked := v.Linked ∪ u.Non_shared;
                C := C \ {u};
            end
            else "Mark u as analyzed";
        end
    end
    SC := SC ∪ {{v} ∪ v.Adj ∪ v.Linked};
end
```

3.4. Complexity analysis

For determining the computational complexity of Algorithm 1, we will analyze each one of its steps. Let n be the size of the collection $O = \{o_1, o_2, \ldots, o_n\}$; *i.e.*, $|O| = n$. For building $\tilde{G}_\beta$ in step 1, we should compute the similarity between all pairs of objects in O. Therefore, the total number of operations done in this step is $T_1(n) = n^2$; thus, $T_1(n)$ is $O(n^2)$. It is important to mention that, during this step, we can compute the value of $Aprox\_Intra\_sim(G^\star)$ for each ws-graph $G^\star$ in $\tilde{G}_\beta$, using expression (4), without affecting $T_1(n)$.

For computing the relevance of each vertex $v \in V$ in step 2, both *v.densityR* and *v.compactnessR* must be computed first. Analogously, for computing *v.densityR* and *v.compactnessR*, we must compute *v.density* and *v.compactness*, respectively. Both *v.density* and *v.compactness* can be computed in the same loop. For computing *v.density* we need to compare the degree of v against the degree of each vertex in *v.Adj*. On the other hand, for computing *v.compactness* we need to compare the value of $Aprox\_Intra\_sim(G_v^\star)$ against the value of $Aprox\_Intra\_sim(G_u^\star)$, for each vertex $u \in v.Adj$. In the worst scenario, $|v.Adj| = n$. Therefore, the total number of operations done in step 2 is $T_2(n) = n^2$; thus, $T_2(n)$ is $O(n^2)$.

Both steps 3 and 4 can be done in the same loop, by checking the degree and the relevance of each vertex in V; hence, the total number of operations for these steps is $T_{3\text{–}4}(n) = n$. Thereby, $T_{3\text{–}4}(n)$ is $O(n)$. Step 5 can be done in $O(n \cdot \log\, n)$ using the merge sort algorithm [38].

In steps 6–8 where the candidates of L are analyzed, for verifying if a vertex $v \in L$ meets condition (a) or condition (b), it is necessary to visit all vertices in *v.Adj*; thus, the total number of operations done in this process is $T_{6\text{–}8}(n) = |L| \cdot n$. In the worst scenario, $|L| = n$. Thus, the total number of operations done in steps 6–8 is $T_{6\text{–}8}(n) = n^2$; therefore, $T_{6\text{–}8}(n)$ is $O(n^2)$. Since the degree of a vertex v is an integer in $[0, n-1]$, following the *pigeonhole principle*, the sorting process of step 10 can be done using a number of operations $T_{10}(n) = n$; hence, $T_{10}(n)$ is $O(n)$. The total number of operations done in step 11 is $T_{11}(n) = |C|$. In the worst scenario, $|C| = n$. Thus, $T_{11}(n) = n$; hence, $T_{11}(n)$ is $O(n)$.

For estimating the total number of operations done in steps 12–24, hereinafter referred to as *loop-A*, a deeper analysis is required. Let $|C| = q_0$ be the number of vertices selected in steps 6–8. In the first iteration of *loop-A*, during steps 14–22, hereinafter referred to as *loop-B*, each vertex $u \in \{v.Adj \cup C\}$ marked as *not-analyzed* is deleted from C or marked as *analyzed*, when u is processed in steps 15–21. Thus, we could divide the set of vertices $u \in \{v.Adj \cup C\}$ that are processed in *loop-B*, during the first iteration of *loop-A*, into two sets: M_1 and R_1. The set M_1 contains the vertices marked as *analyzed*. The set R_1 contains the vertices that formed non-useful ws-graphs and, consequently, were removed from C. In addition, after the execution of *loop-B* in the first iteration of *loop-A*, there remains in C a set Z_1 of vertices marked as *not-analyzed*.

From the aforementioned analysis, we can conclude that after the execution of *loop-B* in the first iteration of *loop-A*, $|M_1|$ vertices of C were marked as *analyzed*, $|R_1|$ vertices of C were removed from C, and there remain $|Z_1|$ vertices in C marked as *not-analyzed*; where $|M_1| + |R_1| + |Z_1| = q_0$. Since there are only $|Z_1|$ vertices in C marked as *not-analyzed*, any vertex that could be processed during the execution of *loop-B*, in the second iteration of *loop-A*, belongs to Z_1; that is, we have that $|M_2| + |R_2| + |Z_2| = |Z_1|$. Following a similar reasoning, in the third iteration we have that $|M_3| + |R_3| + |Z_3| = |Z_2|$. Generalizing, for $i > 1$, we have that $|M_i| + |R_i| + |Z_i| = |Z_{i-1}|$.

The number of operations done in an iteration i of *loop-A* is $F_i(n) = P_1 + P_2 + P_3$, where P_1 is the total number of operations done in *loop-B* for all the vertices in $u \in \{v.Adj \cup C\}$ marked as

not-analyzed; P_2 is the remaining number of operations done in *loop-B*; and P_3 is the number of operations done in step 23. For knowing if the ws-graph determined by u (G_u^*) is not useful, we need to visit each vertex in *u.Adj*. As it was mentioned before, in the worst scenario $|u.Adj| = n$. From the above analysis, we know that the number of vertices marked as *not-analyzed* during *loop-B* was $|M_i| + |R_i|$; thus, $P_1 = n \cdot (|M_i| + |R_i|)$. Besides, since for vertices in *v.Adj* that do not meet the conditions of step 15 no more operations are done, $P_2 = n - (|M_i| + |R_i|)$. Additionally, since sets *v.Adj* and *v.Linked* do not have any vertex in common, we should visit at most n vertices for building the cluster associated to vertex v in step 23; therefore, $P_3 = n$.

Up to here, we know that the number of operations done in the first iteration of *loop-A* is $F_i(n) = n \cdot (|M_i| + |R_i|) + n - (|M_i| + |R_i|) + n$. If the number of iterations done for *loop-A* is $q \leq q_0$, then the total number of operations done in steps 12–24 is

$$T_{12-24}(n) = \sum_{i=1}^{q} F_i(n) = \sum_{i=1}^{q} (n \cdot (|M_i) + |R_i|) + n - (|M_i| + |R_i|) + n|$$

$$T_{12-24}(n) = \sum_{i=1}^{q} (n \cdot (|M_i| + |R_i|)) + \sum_{i=1}^{q} 2 \cdot n - \sum_{i=1}^{q} (|M_i| + |R_i|)$$

Thus,

$$T_{12-24}(n) = n \cdot \sum_{i=1}^{q} (|M_i| + |R_i|) + 2 \cdot n \cdot q - \sum_{i=1}^{q} (|M_i| + |R_i|) \quad (5)$$

As it is known from the previous analysis, the following equations are true:

$$\begin{aligned} &|M_1| + |R_1| + |Z_1| = q_0 \\ &|M_2| + |R_2| + |Z_2| = |Z_1| \\ &\vdots \\ &|M_q| + |R_q| + |Z_q| = |Z_{q-1}| \end{aligned} \quad (6)$$

adding all equations in (6), we obtain

$$\sum_{i=1}^{q} (|M_i| + |R_i|) + \sum_{i=1}^{q} |Z_i| = q_0 + \sum_{i=1}^{q-1} |Z_i|$$

$$\sum_{i=1}^{q} (|M_i| + |R_i|) + |Z_q| = q_0$$

thus,

$$\sum_{i=1}^{q} (|M_i| + |R_i|) = q_0 - |Z_q| \quad (7)$$

If we substitute (7) in (5), we have that

$$T_{12-24}(n) = n \cdot (q_0 - |Z_q|) + 2 \cdot n \cdot q - (q_0 - |Z_q|)$$
$$T_{12-24}(n) = n \cdot q_0 - n \cdot |Z_q| + 2 \cdot n \cdot q - q_0 + |Z_q|$$

grouping terms having equal sign

$$T_{12-24}(n) = n \cdot q_0 + 2 \cdot n \cdot q + |Z_q| - (n \cdot |Z_q| + q_0)$$

Since $(n \cdot |Z_q| + q_0) \geq 0$ and q_0, q and $|Z_q|$ are integers in [0,n], we have that

$$T_{12-24}(n) 2n \cdot q_0 + 2 \cdot n \cdot q + |Z_q| 2n^2 + 2 \cdot n^2 + n$$

hence,

$$T_{12-24}(n) \leq 3 \cdot n^2 + n \quad (8)$$

From (8), we can conclude that T_{12-24} is $O(n^2)$. Finally, based on the aforementioned analysis, the total number of operations done in Algorithm 1 is $T(n) = T_1(n) + T_2(n) + T_{3-4}(n) + T_5(n) + T_{6-8}(n) + T_{10}(n) + T_{11}(n) + T_{12-24}(n)$. By the rule of the sum, $T(n)$ is $O(\max(T_1(n), T_2(n), T_{3-4}(n), T_5(n), T_{6-8}(n), T_{10}(n), T_{11}(n), T_{12-24}(n)))$. Therefore, $T(n)$ is $O(n^2)$ and, consequently, the computational complexity of OClustR is $O(n^2)$.

4. Experimental results

In this section, the results of some experiments testing the performance of OClustR algorithm are presented.

The experiments were conducted over several overlapping collections. In these experiments, we contrasted our results against those obtained by Star [12], ISC [10], SHC [15], Estar [13], Gstar [19], ACONS [20], ICSD [24] and DCS [27] algorithms. The experiments were focused on comparing the algorithms according to: (*a*) the quality of the clustering, (*b*) the number of clusters obtained, (*c*) the overlapping of the clustering, and (*d*) the time spent for clustering the collections.

All the algorithms used in the experiments were implemented in C++ and compiled using the g++ compiler. The experiments were performed on a PC with an Intel Core 2 Duo at 1.86 GHz CPU with 2 GB DDR2 RAM, running RedHat Enterprise Linux 5.3.

4.1. Collections used in the experiments

Since we are addressing the problem of overlapping clustering, we decided to evaluate the algorithms in the task of document clustering, where it is common that some documents belong to more than one topic.

The document collections used in the experiments were built from three benchmark text collections: AFP, Reuters-21578 and TDT2. The AFP benchmark was downloaded from http://trec.nist.gov and it contains news published by the AFP agency in 1994 and used in the TREC-5 conference. Reuters-21578 was downloaded from http://kdd.ics.uci.edu and it contains news published by Reuters during 1987. TDT2 was downloaded from http://www.nist.gov/speech/tests/tdt.html and it contains news collected from different sources from January 1998 to June 1998. AFP contains news in Spanish while the other two benchmarks contain news in English.

Ten document collections were built from the aforementioned benchmarks. The collection AFP was built from the AFP benchmark, using all its news. The collection Reu-Te was built from Reuters benchmark, using the news tagged as "Test" that have been associated with at least one topic. The collection Reu-Tr was built from Reuters benchmark, using the news tagged as "Train" that have been associated with at least one topic. The collection Reuters is the union of Reu-Te and Reu-Tr. The collection TDT was built from TDT2 benchmark, using the news that have been associated with at least one topic. Finally, five sub-collections called as TDT-1, TDT-2, TDT-3, TDT-4 and TDT-5 were built from the TDT collection. In order to build these five sub-collections, the news of TDT were randomly arranged into five folds. Afterwards, each sub-collection was built by randomly selecting three of these five folds.

The characteristics of the ten document collections used in our experiments are shown in Table 1. The columns labeled as "#Documents" and "#Terms" represent the number of documents and terms contained in each collection. The column labeled as "#Classes" represents the number of topics identified by human annotators for each collection. The column labeled as "Overlapping" represents the average number of classes in which a document is included [29].

In our experiments, documents were represented using the Vector Space Model (VSM) [39]. The index terms of the documents represent the lemmas of the words occurring at least once in the whole collection; these lemmas were extracted from the documents using the Tree-tagger.[1] Stop words such as: articles, prepositions and adverbs were removed.

[1] http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger

Table 1
Characteristics of the document collections.

Name	#Documents	#Terms	#Classes	Overlapping
AFP	695	11,785	25	1.023
Reu-Te	3587	15,113	100	1.295
Reu-Tr	7780	21,901	115	1.241
Reuter	11,367	27,083	120	1.258
TDT	16,007	68,019	193	1.188
TDT-1	8602	51,764	176	1.202
TDT-2	7404	44,610	178	1.173
TDT-3	10,258	53,706	174	1.189
TDT-4	10,074	53,036	172	1.182
TDT-5	11,328	55,923	182	1.180

The index terms of each document were statistically weighted using term frequency normalized by the *maximum term frequency*. The *maximum term frequency* is the highest frequency of a term in a given document [40]. The cosine measure was used to compute the similarity between two documents [41].

4.2. Evaluation measures

Several measures have been proposed to evaluate the clusters obtained by a clustering algorithm [35,42,43]. There are three types of evaluation measures: *external measures*, *relative measures* and *internal measures* [44]. From these three kinds of measures, the most widely used are the external measures. The external measures evaluate a clustering solution based on how much this clustering solution resembles a set of classes, commonly known as *ground-truth*, which has been manually tagged by human experts. The more similar the clustering solution is to the *ground-truth*, the better the clustering algorithm is.

Many external evaluation measures have been proposed in the literature, such as Purity and Inverse Purity [35], Jaccard coefficient [42], F1-measure [45], FM index [46], Entropy [47], Class Entropy [48] and V-measure [49], among others. These measures are different according to their mathematical foundations, their biases and their limitations. There are several works that have used measures based on *counting pairs*, like Fmeasure and Jaccard coefficient, for evaluating overlapping clusterings [19,20,24,50]. Other works have used measures based on *set matching*, such as F1-measure, for evaluating their results [13,29]. However, none of the external measures reported so far have been developed, at least explicitly, for evaluating overlapping clustering; that is, these measures fail at reflecting the fact that, in a perfect overlapping clustering, objects sharing n classes should share n clusters.

To the best of our knowledge, there are only two works which propose and discuss measures for evaluating overlapping clusterings [28,51]. In [28], Amigo et al. proposed a new external measure for evaluating overlapping clusterings, called FBcubed, which is computed using variations of the Bcubed precision and recall measures [52].

Let $D(o')$ be the set of objects, including o', which shares at least one cluster with an object o'. The new Bcubed precision of an object o' is defined as follows:

$$Bcubed_{pre}(o') = \frac{\sum_{e \in D(o')} MBcubed_{pre}(o', e)}{|D(o')|}$$

where $MBcubed_{pre}(o', e)$ is the *multiplicity Bcubed precision* of o' wrt. e and it is computed as follows:

$$MBcubed_{pre}(o', e) = \frac{MIN(|C(o') \cap C(e)|, |L(o') \cap L(e)|)}{|C(o') \cap C(e)|}$$

where $C(o')$ and $L(o')$ are the clusters and classes to which object o' belongs; $C(e)$ and $L(e)$ are defined in the same way as $C(o')$ and $L(o')$.

Let $H(o')$ be the set of objects, including o', which shares at least one class with an object o'. The new Bcubed recall of an object o' is defined as follows:

$$Bcubed_{rec}(o') = \frac{\sum_{e \in H(o')} MBcubed_{rec}(o', e)}{|H(o')|}$$

where $MBcubed_{rec}(o', e)$ is the *multiplicity Bcubed recall* of o' wrt. e and it is computed as follows:

$$MBcubed_{rec}(o', e) = \frac{MIN(|C(o') \cap C(e)|, |L(o') \cap L(e)|)}{|L(o') \cap L(e)|}$$

The overall Bcubed precision, denoted as $Bcubed_{pre}$, is computed as the average of the Bcubed precision of all objects in the collection; the overall Bcubed recall, denoted as $Bcubed_{rec}$, is defined analogously but using the Bcubed recall of all objects. Finally, the FBcubed measure is computed as the harmonic mean of $Bcubed_{pre}$ and $Bcubed_{rec}$, that is

$$FBcubed = \frac{2 \cdot Bcubed_{pre} \cdot Bcubed_{rec}}{Bcubed_{pre} + Bcubed_{rec}} \quad (9)$$

It is important to notice that the way in which the Bcubed precision and recall of an object are defined allows the FBcubed measure to detect situations in which:

(a) The clustering algorithm does not capture completely the relationship between two items. This situation happens when there are two objects sharing more classes than clusters. In this case, the overall Bcubed recall will decrease and consequently, the FBcubed measure will decrease too.
(b) The clustering algorithm is introducing more information than necessary. This situation happens when there are two objects sharing more clusters than classes. In this case, the overall Bcubed precision will decrease and consequently, the FBcubed measure will decrease too.

Additionally, the FBcubed measure meets four formal constraints which allow it to evaluate several desirable characteristics in a clustering [28]. These four formal constraints are:

(a) *Cluster homogeneity*, this constraint states that clusters should not mix objects from different classes (see Fig. 1a).
(b) *Cluster completeness*, this constraint is the counterpart of the previous one and it states that items belonging to the same class should be grouped together in the same cluster (see Fig. 1b).
(c) *Rag bag*, this constraint states that it is preferable to have clean clusters plus a noisy cluster than having clusters with a dominant class plus additional noise (see Fig. 1c).
(d) *Cluster size versus quantity*, this constraint states that to separate one object from its class of $n > 2$ members is less harmful than to fragment n binary classes (see Fig. 1d).

As we can see from Fig. 1, these constraints are intuitive and they express important characteristics that should be evaluated by an external evaluation measure. Moreover, in [28] the authors showed that none of the most used external evaluation measures satisfy all the four formal constraints.

In [51], Ramírez et al. proposed three external measures for evaluating overlapping clusterings: GFM, PCMP, and Clustering Recall (R_U). The first, GFM, is a generalization of the known Fowlkes–Mallows index (FM index) [46], which is a measure based on counting pairs. On the other hand, PCMP (from Partial Class Match Precision) is also a measure based on counting pairs but, different from GFM, it measures the probability of randomly

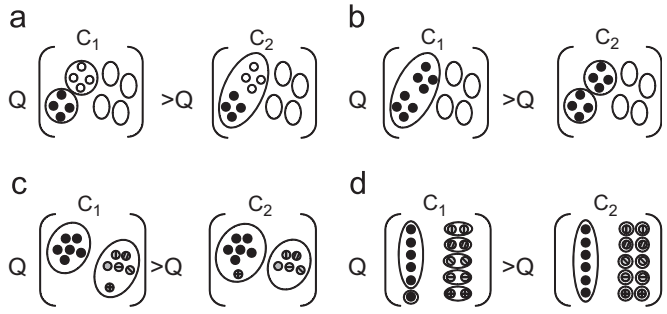

Fig. 1. Illustrating the four formal constraints [28]. (a) Cluster homogeneity. (b) Cluster completeness. (c) Rag bag (d) Cluster size versus quantity.

selecting two objects from the same class taken from a randomly sampled cluster. As was stated in [51], both previous measures work under the assumption that the overlapping is restricted to pairs of classes; this restriction does not necessarily hold in real problems. Besides, as was showed in [28], the measures based on counting pairs do not meet constraints *c* and *d*. The R_U measure is based on the concept of "recall", it was taken from the Information Retrieval area, and measures the average number of objects of each class that was included in any cluster. This is a measure based on set matching which, like purity [35], does not meet constraints *b* and *c*. Moreover, none of the three measures proposed in [51] are able to evaluate the fact that, in a perfect overlapping clustering, objects sharing *n* classes should share *n* clusters.

Based on the aforementioned analysis and in order to conduct a fair comparison among the algorithms, we will use the FBcubed measure for evaluating the quality of the clusterings built by each algorithm. A more detailed explanation about the FBcubed measure, together with a case of study and a deeper analysis, can be found in [28].

4.3. Quality

In this experiment, we compared the clustering algorithms according to the quality of the clusters they built for each collection. For evaluating the clusters we used the FBcubed measure [28], described in Section 4.2. For a better understanding of how we conducted this experiment, we will explain in detail the procedure we followed with the AFP collection.

First, all algorithms were executed over the AFP collection, using values of β in [0.10, 0.50], with an increment of 0.01; that is, we used $\beta = 0.10, 0.11, 0.12$ and so on. In the case of the SHC algorithm, we tested different values for the parameters ϵ and HR_{min}; finally, the values that produced the best results were chosen. Afterwards, we computed the FBcubed of the clusterings built by each algorithm for each value of β. Since the algorithms ISC and Estar do not depend on the data order, we selected their highest FBcubed values, over all β values, as their best performance over AFP. For Star, SHC, Gstar, ACONS, ICSD, DCS and OClustR, which depend on the data order, we repeat the experiment 20 times, for each value of β, varying the order of the documents. For these algorithms, their highest average FBcubed value, over all β values, was selected as their best performance over AFP.

The same above described procedure was used for computing the best performance of the clustering algorithms over the remaining collections. From all this evaluation process, we observed the following:

- For values of β greater than 0.50 or smaller than 0.10, the quality value of the clusterings built by all algorithms decreases. For this reason, we do not use values of β out of the above mentioned interval.
- Even when the results of Star, SHC, Gstar, ACONS, ICSD, DCS and OClustR depend on data order, the standard deviation of the FBcubed values of the clusterings built for each different order of the documents was less than 0.01 for each value of β. Therefore, we can rely in the average FBcubed value, for each value of β used, as the best performance for each of these β values.

In Table 2, we show the best performances, according to the FBcubed measure, attained by each algorithm over each collection.

As it can be noticed from Table 2, OClustR outperforms all tested algorithms, according to the FBcubed measure, in all the collections used in this experiment. Additionally, in Table 3 we show, for each collection, the percentage of improvement in the value of the FBcubed measure that our proposed algorithm attained, considering the FBcubed values obtained by the other algorithms. The ISC and SHC algorithms were excluded from this analysis since their performance was poor wrt. the other algorithms.

As it can be seen from Table 3, our proposed algorithm attained improvements, in the value of the FBcubed measure, up to 41.14%. Moreover, if we consider the average value of the improvements, OClustR outperforms the results of Star, Estar, GStar, ICSD, ACONS and DCS algorithms in 8.19%, 26.48%, 27.66%, 29.10%, 28.93% and 29.10% respectively.

4.4. Number of clusters

This experiment was focused on comparing the algorithms regarding the number of clusters they built for each document collection, when they attained their best performance according to the FBcubed measure. Table 4 shows the number of clusters built by each algorithm for each collection.

As it can be seen from Table 4, our proposed algorithm builds, in all collections, less clusters than the other clustering algorithms used in the comparison. It is important to highlight that these results of OClustR are obtained without affecting the quality of the clusterings (see Table 2); in this way, OClustR builds clusterings that could be easier to analyze than those built by the other algorithms used for comparison.

4.5. Overlapping

In this experiment, we compared the algorithms according to the overlapping they produce for each document collection, when they attained their best performance according to the FBcubed measure. The overlapping of a clustering is computed as the average number of clusters in which an object is included [29]. In Table 5, we show the overlapping of the clusterings built by each algorithm for each collection.

As it can be seen from Table 5, OClustR builds clusterings which have less overlapping than the clusterings built by the other algorithms. This way, OClustR allows overlapping among the clusters but it controls the overlapping in order to avoid building clusters with high overlapping that could be interpreted as the same cluster. This characteristic could be useful for applications like document segmentation by topic [9] or web document organization [11,15], where low overlapping is desirable.

Additionally, we carried out another experiment for knowing how far the overlapping produced by each clustering algorithm is from the overlapping existing in the *ground-truth* of each collection.

For doing this, for each collection *D*, we divide the overlapping of the clustering built by each algorithm (showed above in Table 5) and the overlapping existing in the *ground-truth* (see the column labeled as "Overlapping" in Table 1); hereinafter, we will refer to this division as *relative overlap* (RO). A relative overlap value of 1 means that both the algorithm and the *ground-truth* have the

Table 2
Best performance of each algorithm, according to the FBcubed value, for each collection. The highest value per collection appears bold-faced.

Coll.	Algorithms								
	Star	ISC	Estar	Gstar	ICSD	ACONS	SHC	DCS	OClustR
AFP	0.69	0.20	0.63	0.63	0.61	0.62	0.27	0.61	**0.77**
Reu-Te	0.45	0.05	0.39	0.40	0.39	0.40	0.20	0.39	**0.51**
Reu-Tr	0.42	0.03	0.36	0.36	0.36	0.36	0.19	0.36	**0.43**
Reuter	0.42	0.02	0.34	0.35	0.35	0.36	0.19	0.35	**0.43**
TDT	0.43	0.06	0.37	0.35	0.35	0.34	0.15	0.35	**0.48**
TDT-1	0.45	0.09	0.39	0.38	0.38	0.38	0.16	0.38	**0.48**
TDT-2	0.47	0.10	0.40	0.40	0.40	0.39	0.17	0.40	**0.52**
TDT-3	0.46	0.07	0.40	0.40	0.39	0.39	0.17	0.39	**0.51**
TDT-4	0.46	0.07	0.40	0.39	0.39	0.39	0.17	0.39	**0.50**
TDT-5	0.46	0.07	0.39	0.37	0.37	0.37	0.16	0.37	**0.50**

Table 3
Improvement percentage obtained by OClustR algorithm.

Collections	Algorithms					
	Star	Estar	Gstar	ICSD	ACONS	DCS
AFP	11.14	22.99	21.83	26.49	24.70	26.49
Reu-Te	13.70	30.75	29.24	30.27	27.73	30.27
Reu-Tr	4.24	22.01	20.90	21.82	20.71	21.82
Reuter	2.24	27.69	23.22	25.04	22.07	25.04
TDT	10.71	31.63	37.86	39.24	41.14	39.24
TDT-1	5.66	22.09	25.68	26.13	26.61	26.13
TDT-2	8.77	28.59	29.01	29.37	31.81	29.37
TDT-3	8.76	27.07	27.77	29.81	30.16	29.81
TDT-4	7.68	23.87	26.33	27.46	28.04	27.46
TDT-5	9.02	28.12	34.80	35.35	36.28	35.35
Ave.	**8.19**	**26.48**	**27.66**	**29.10**	**28.93**	**29.10**

Table 4
Number of clusters built by each algorithm for each collection. The best performance per collection appears bold-faced.

Coll.	Algorithms								
	Star	ISC	Estar	Gstar	ICSD	ACONS	SHC	DCS	OClustR
AFP	123	334	98	90	104	129	85	104	**52**
Reu-Te	507	1785	600	711	621	798	273	621	**102**
Reu-Tr	471	3936	904	849	853	857	561	853	**166**
Reuter	583	5726	659	1532	1183	1420	815	1183	**211**
TDT	2019	8250	1854	1653	1657	1663	1203	1657	**769**
TDT-1	1184	4425	1207	1075	1078	1077	643	1078	**377**
TDT-2	970	3743	1074	948	950	954	579	950	**388**
TDT-3	1338	5253	1355	1187	1190	1196	758	1190	**594**
TDT-4	1104	5154	1303	1158	1160	1163	731	1160	**434**
TDT-5	1425	5816	1451	1291	1293	1295	837	1293	**614**

Table 5
Overlapping of the clustering built by each algorithm for each collection. The lowest value per collection appears bold-faced.

Coll.	Algorithms								
	Star	ISC	Estar	Gstar	ACONS	ICSD	SHC	DCS	OClustR
AFP	1.71	1.65	2.52	2.31	2.48	2.53	2.43	2.53	**1.18**
Reu-Te	3.41	1.79	7.40	6.73	6.66	7.64	13.13	7.64	**1.40**
Reu-Tr	5.54	1.84	12.14	13.08	12.65	13.32	29.33	13.32	**1.56**
Reuter	5.46	1.82	15.92	15.47	15.47	19.25	47.55	19.25	**1.53**
TDT	4.81	1.88	59.41	69.43	66.38	70.97	80.74	70.97	**1.50**
TDT-1	3.38	1.84	44.22	49.08	46.40	49.63	47.91	49.63	**1.43**
TDT-2	3.38	1.80	35.11	37.85	37.46	37.85	39.82	37.85	**1.39**
TDT-3	3.81	1.84	42.40	46.08	44.83	46.22	53.79	46.22	**1.53**
TDT-4	4.08	1.83	43.34	47.59	46.24	48.72	56.04	48.72	**1.45**
TDT-5	3.98	1.88	44.03	51.46	48.44	52.23	59.79	52.23	**1.45**

same overlapping. On the other hand, if the relative overlap is greater than 1, then the algorithm produces more overlapping than the real overlapping of the collection. Otherwise, if the relative overlap is lower than 1, then the algorithm produces less overlapping than the real overlapping of the collection. Thus, in order to know how many times far is the overlapping produced by an algorithm A for each collection D, from its real overlapping, we compute the absolute value of the difference between 1 and the RO produced by the algorithm A; that is, we compute the absolute value of ($RO-1$). The closer to zero this absolute value, the better the algorithm A performs. We will refer to this absolute value as the *absolute relative overlap* (ARO). Table 6 shows the ARO produced by each algorithm for each collection.

As it can be seen from Table 6, OClustR was the best for all tested collections. Moreover, OClustR performs, on an average, at least twice times better than the ISC algorithm, which was the second best.

4.6. Runtime

In this experiment, we compare the algorithms according to their runtime. For doing this, we execute each algorithm over Reu-Te, Reu-Tr, Reuter and TDT-1 collections, using different values of β.

Table 6
Absolute relative overlap (ARO) produced by each algorithm for each collection. The best performance per collection appears bold-faced.

Coll.	Algorithms								
	Star	ISC	Estar	Gstar	ACONS	ICSD	SHC	DCS	OClustR
AFP	0.67	0.61	1.47	1.26	1.43	1.47	1.38	1.47	**0.15**
Reu-Te	1.63	0.38	4.71	4.20	4.14	4.90	9.14	4.90	**0.08**
Reu-Tr	3.46	0.48	8.78	9.54	9.20	9.73	22.63	9.73	**0.26**
Reuter	3.34	0.44	11.65	11.29	11.30	14.31	36.80	14.31	**0.22**
TDT	3.05	0.58	49.01	57.45	54.88	58.74	66.97	58.74	**0.26**
TDT-1	1.81	0.53	35.79	39.83	37.60	40.29	38.86	40.29	**0.19**
TDT-2	1.88	0.53	28.93	31.27	30.94	31.27	32.95	31.27	**0.19**
TDT-3	2.21	0.55	34.66	37.75	36.70	37.87	44.24	37.87	**0.28**
TDT-4	2.45	0.55	35.67	39.26	38.12	40.22	46.41	40.22	**0.23**
TDT-5	2.38	0.59	36.31	42.61	40.05	43.26	49.67	43.26	**0.23**
Ave.	2.29	0.53	24.70	27.45	26.44	28.21	34.90	28.21	**0.21**

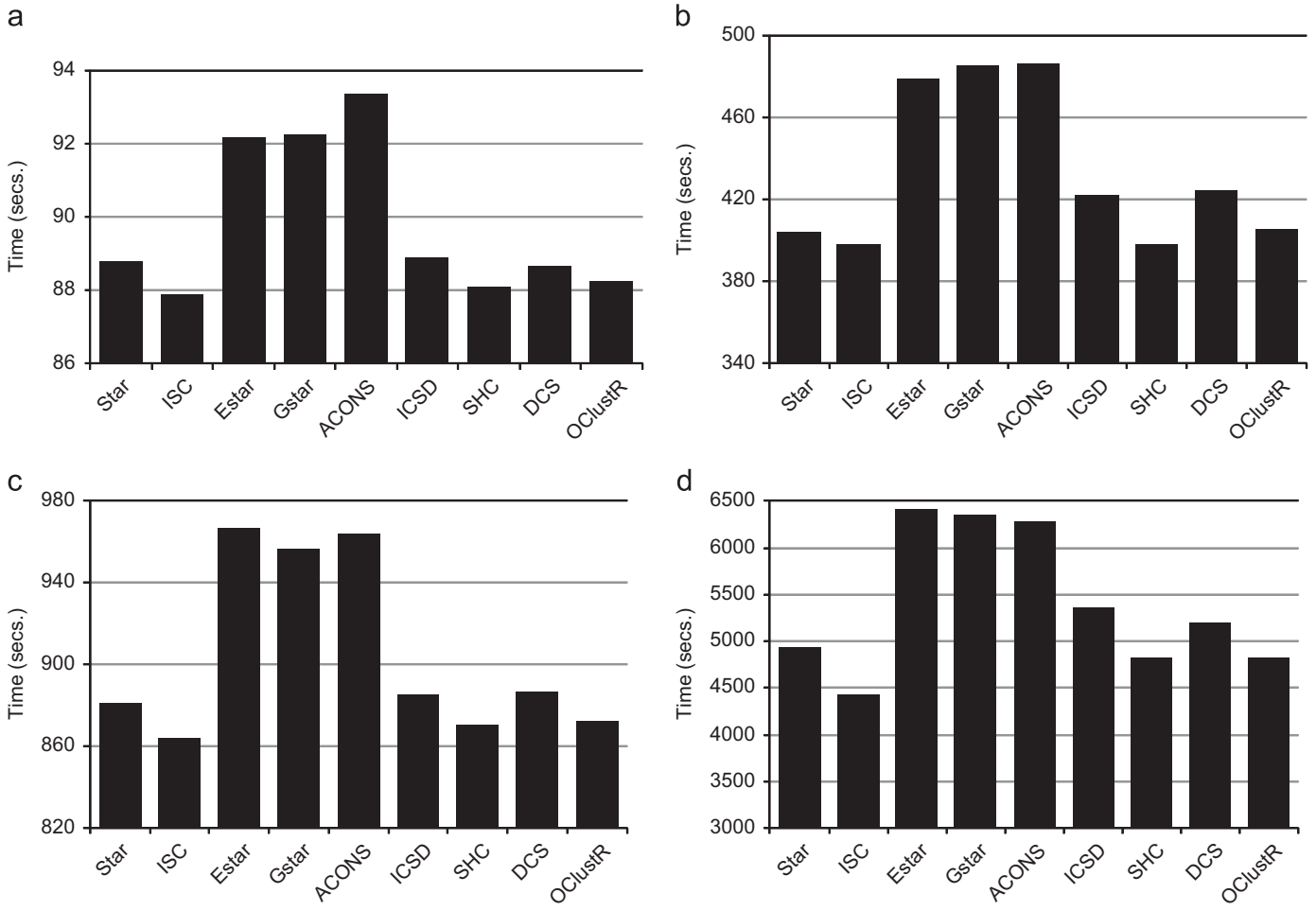

Fig. 2. Runtime for each algorithm over (a) Reu-Te, (b) Reu-Tr, (c) Reuter and (d) TDT-1 collections, for $\beta = 0.25$.

For algorithms like Star, Gstar, ACONS, ICSD, SHC, DCS and OClustR, which depend on data order analysis, we repeated this experiment 20 times, randomly varying the order of the documents. In Figs. 2–4 we show the average runtime of each algorithm over Reu-Te, Reu-Tr, Reuter and TDT-1 collections, for $\beta = 0.25$, 0.30 and 0.25, respectively.

As it can be seen from Figs. 2–4, our proposed algorithm has better performance than Estar, Gstar, ACONS, ICSD, DCS and Star algorithms. Although ISC and SHC algorithms have slightly better performance than OClustR, it is important to remember that our proposed algorithm builds clusterings with higher quality than those clusterings built by ISC and SHC algorithms. Moreover, OClustR also outperforms these two algorithms according to the number of clusters and their overlapping. We conducted other experiments over the remaining collections, using different values for β, and we observed the same behavior.

5. Conclusions

In this paper, we introduce OClustR, a new clustering algorithm for building overlapping clusters. OClustR represents a collection of objects as a *weighted thresholded similarity graph* $\tilde{G}_\beta$, where β is a predefined parameter. OClustR builds an overlapping clustering in two steps. In the first step, OClustR builds an initial set of clusters through a covering of $\tilde{G}_\beta$, using *weighted star-shaped sub-graphs* (ws-graphs). For building this covering, OClustR introduces a new criterion for ordering the ws-graphs and a new graph-covering strategy for selecting those ws-graphs needed for covering $\tilde{G}_\beta$. In the second step, OClustR improves the set of initial clusters, in order to build the final clustering. For improving the clusters, OClustR introduces a strategy, which aims at reducing both the number of clusters and their overlapping.

The proposed algorithm was compared against other overlapping clustering algorithms reported in the literature, using several standard overlapping collections. The experiments were focused on comparing the algorithms according to the quality of the clusterings they build, the number of clusters produced, the overlapping of the obtained clusterings and the time they spent for clustering some experimental collections. From these experiments, we can conclude OClustR builds clusterings more accurately, according to the FBcubed evaluation measure, than those built by all the other overlapping clustering algorithms. Additionally, unlike all the tested algorithms, OClustR builds less clusters with less overlapping than those built by the other tested algorithms. Moreover, from these experiments we can conclude that our proposed algorithms have better trade off between quality and efficiency than the other tested algorithms.

Additionally, our proposed algorithm solves the limitations presented in the state-of-the-art algorithms (see Section 2) and it has an acceptable $O(n^2)$ computational complexity. From the above comments, we can conclude that OClustR is a better option for overlapping clustering than previously reported algorithms.

As future work, we will develop a version of OClustR algorithm able to process additions, deletions as well as modifications of objects in an already clustered collection, without rebuilding the

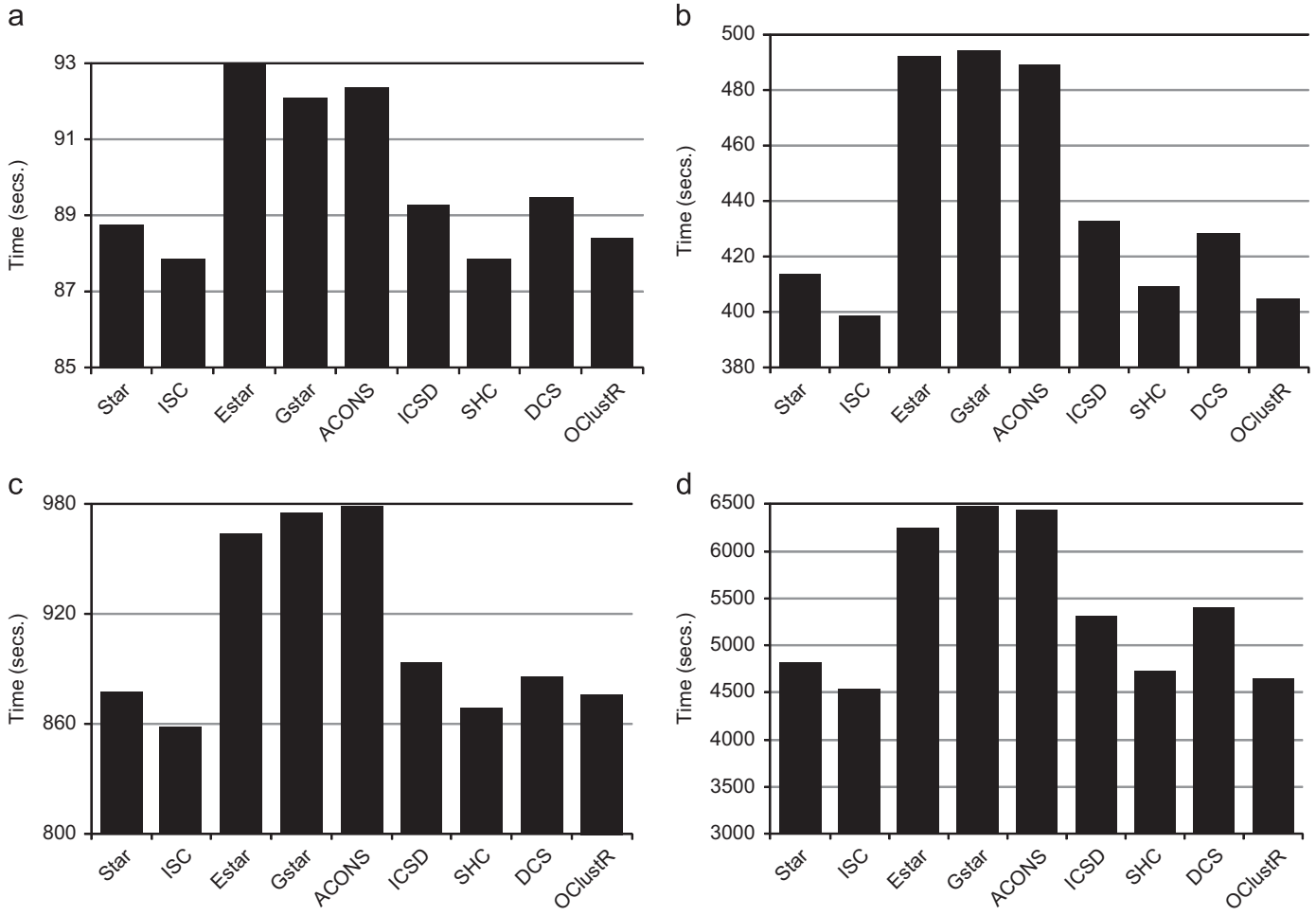

Fig. 3. Runtime for each algorithm over (a) Reu-Te, (b) Reu-Tr, (c) Reuter and (d) TDT-1 collections, for $\beta = 0.30$.

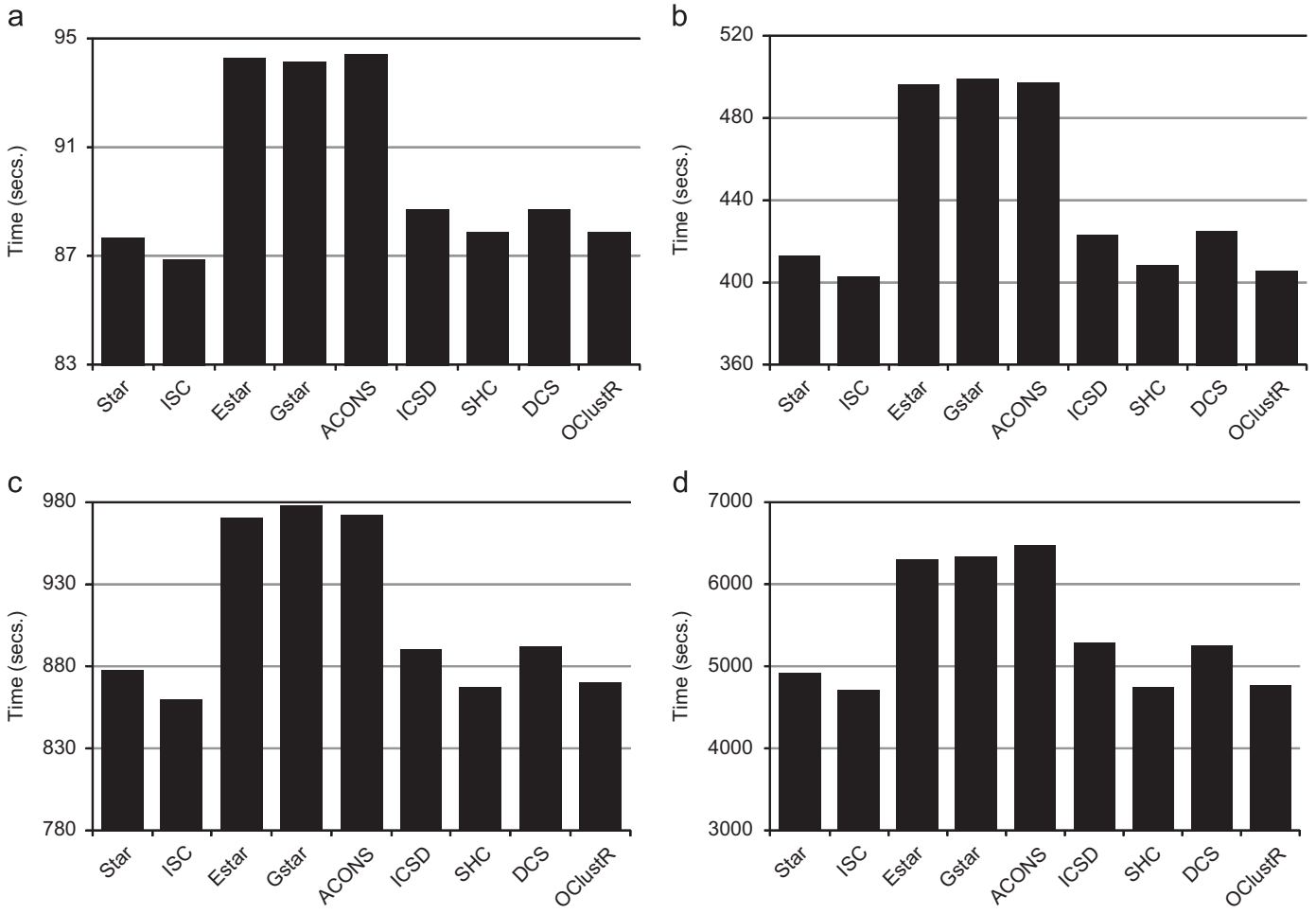

Fig. 4. Runtime for each algorithm over (a) Reu-Te, (b) Reu-Tr, (c) Reuter and (d) TDT-1 for $\beta = 0.35$.

clustering starting from scratch. It will become OClustR in a Dynamic clustering algorithm, thus increasing its application scope.

Acknowledgment

We thank the National Council on Science and Technology of Mexico (CONACyT) for its support to this research through the project Grants CB2008-106443 and CB2008-106366.

References

[1] A.K. Jain, M.N. Murty, P.J. Flynn, Data clustering: a review, ACM Comput. Surv. 31 (3) (1999) 264–323.
[2] Y. Li, H. Shi, L. Jiao, R. Liu, Quantum evolutionary clustering algorithm based on watershed applied to SAR image segmentation, Neurocomputing 87 (2012) 90–98 http://dx.doi.org.10.1016/j.neucom.2012.02.008..
[3] M.U. Munir, M.Y. Javed, S.A. Khan, A hierarchical k-means clustering based fingerprint quality classification, Neurocomputing 85 (2012) 62–67, http://dx.doi.org/10.1016/j.neucom.2012.01.002.
[4] C. Alzate, J.A.K. Suykens, Sparse kernel spectral clustering models for large-scale data analysis, Neurocomputing 74 (2011) 1382–1390, http://dx.doi.org/10.1016/j.neucom.2011.01.001.
[5] M. Magdon-Ismail, J. Purnell, SSDE-cluster:fast overlapping clustering of networks using sampled spectral distance embedding and GMMs, in: Proceedings of SocialCom2011, 2011, pp. 756–759.
[6] G. Davis, K. Carley, Clearing the FOG: fuzzy, overlapping groups for social networks, Soc. Netw. 30 (3) (2008) 201–212.
[7] M. Goldberg, S. Kelley, M. Magdon-Ismail, K. Mertsalov, A. Wallace, Finding overlapping communities in social networks, in: Proceedings of SocialCom2010, 2010, pp. 104–113.
[8] J. Aslam, E. Pelekhov, D. Rus, The star clustering algorithm for static and dynamic information organization, J. Gr. Algorithms Appl. 8 (1) (2004) 95–129.
[9] R. Abella-Pérez, J.E. Medina-Pagola, An incremental text segmentation by clustering cohesion, in: Proceedings of HaCDAIS 2010, 2010, pp. 65–72.
[10] A. Pons-Porrata, J. Ruiz-Shulcloper, R. Berlanga-Llavorí, Y. Santiesteban-Alganza, Un algoritmo incremental para la obtención de cubrimientos con datos mezclados, in: Proceedings of CIARP2002, 2002, pp. 405–416.
[11] O. Zamir, O. Etziony, Web document clustering: a feasibility demonstration, in: Proceedings of the 21st Annual International ACM SIGIR Conference, 1998, pp. 46–54.
[12] J. Aslam, K. Pelekhov, D. Rus, Static and dynamic information organization with star clusters, in: Proceedings of the Seventh International Conference on Information and Knowledge Management, 1998, pp. 208–217.
[13] R.J. Gil-García, J.M. Badía-Contelles, A. Pons-Porrata, Extended star clustering algorithm, in: Proceedings of CIARP2003, 2003, pp. 480–487.
[14] R.J. Gil-García, J.M. Badía-Contelles, A. Pons-Porrata, Parallel algorithm for extended star clustering, in: Proceedings of CIARP2004, 2004, pp. 402–409.
[15] K.M. Hammouda, M.S. Kamel, Efficient phrase-based document indexing for web document clustering, IEEE Trans. Knowl. Data Eng. 16 (10) (2004) 1279–1296.
[16] G. Palla, I. Derényi, I. Farkas, T. Vicsek, Uncovering the overlapping community structure of complex networks in nature and society, Nature 435 (7043) (2005) 814–824.
[17] J. Baumes, M. Goldberg, M. Magdon-Ismail, Efficient identification of overlapping communities, in: Proceedings of ISI 2005, 2005, pp. 27–36.
[18] J. Baumes, M. Goldberg, M. Krishnamoorty, M. Magdon-Ismail, N. Preston, Finding communities by clustering a graph into overlapping subgraphs, in: Proceedings of IADIS Applied Computing, 2005, pp. 97–104.
[19] A. Pérez-Suárez, J. E. Medina-Pagola, A clustering algorithm based on generalized stars, in: Proceedings of MLDM 2007, 2007, pp. 248–262.
[20] A. Gago-Alonso, A. Pérez-Suárez, J.E. Medina-Pagola, ACONS: a new algorithm for clustering documents, in: Proceedings of CIARP2007, 2007, pp. 664–673.
[21] S. Gregory, An algorithm to find overlapping community structure in networks, in: Proceedings of the PKDD 2007, 2007, pp. 91–102.

[22] S. Zhang, R.S. Wang, X.S. Zhang, Identification of overlapping community structure in complex networks using fuzzy c-means clustering, Physica A: Stat. Mech. Appl. 374 (1) (2007) 483–490.
[23] S. Gregory, A fast algorithm to find overlapping communities in networks, in: Proceedings of the 12th ECML KDD, 2008, pp. 408–423.
[24] A. Pérez-Suárez, J.F. Martínez-Trinidad, J.A. Carrasco-Ochoa, J.E. Medina-Pagola, A new incremental algorithm for overlapped clustering, in: Proceedings of CIARP2009, 2009, pp. 497–504.
[25] K. Macropol, T. Can, A.K. Singh, RRW: repeated random walks on genome-scale protein networks for local cluster discovery, BMC Bioinformatics 10(1):283, 2009. http://dx.doi.org/10.1186/1471-2105-10-283.
[26] M. Al-Hasan, S. Salem, M.J. Zaki, SimClus: an effective algorithm for clustering with a lower bound on similarity, Knowl. Inf. Syst. 28 (3) (2011) 665–685 http://dx.doi.org.10.1007/s10115-010-0360-6..
[27] A. Pérez-Suárez, J.F. Martínez-Trinidad, J.A. Carrasco-Ochoa, J.E. Medina-Pagola, A dynamic clustering algorithm for building overlapping clusters, Intell. Data Anal. 16 (2) (2012) 211–232.
[28] E. Amigó, J. Gonzalo, J. Artiles, F. Verdejo, A comparison of extrinsic clustering evaluation metrics based on formal constraints, Inf. Retr. 12 (2009) 461–486.
[29] R.J. Gil-García, A. Pons-Porrata, Dynamic hierarchical algorithms for document clustering, Pattern Recognit. Lett. 31 (6) (2010) 469–477.
[30] D. Gusfield, Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology, Cambridge University Press, 1997. (Ch. 6).
[31] R. Sibson, An optimally efficient algorithm for the single link cluster method, Comput. J. 16 (1973) 30–34.
[32] M.G. Everett, S.P. Borgatti, Analyzing clique overlap, Connections 21 (1998) 49–61.
[33] L. Page, S. Brin, R. Motwani, T. Winograd, The pagerank citation ranking: bringing order to the web, Working Paper, Stanford Digital Libraries.
[34] M.E.J. Newman, M. Girvan, Finding and evaluating community structure in networks. Phys. Rev. E, 69(2), 2004. http://dx.doi.org/10.1103/PhysRevE.69.026113.
[35] Y. Zhao, G. Karypis, Criterion Functions for Document Clustering: Experiments and Analysis, Technical Report 01-40, Department of Computer Science, University of Minnesota, Minneapolis, MN, 2001.
[36] A. Civril, M. Magdon-Ismail, E. Bocek-Rivele, SSDE: Fast graph drawing using sampled spectral distance embedding, in: Graph Drawing, 2007, pp. 30–41.
[37] T. Jo, M. Lee, The evaluation measure of text clustering for the variable number of clusters, in: Proceedings of the 4th International Symposium on Neural Networks: Part II—Advances in Neural Networks, 2007, pp. 871–879.
[38] D.E. Knuth, The Art of Computer Programming, vol. 3, Addison-Wesley, 1973.
[39] G. Salton, A. Wong, C.S. Yang, A vector space model for automatic indexing, Commun. ACM 18 (11) (1975) 613–620 http://dx.doi.org.10.1145/361219.361220..
[40] E. Greengrass, Information Retrieval: A Survey, Technical Report TR-R52-008-001, 2001.
[41] M. Berry, Survey of Text Mining, Clustering, Classification and Retrieval, Springer-Verlag, 2004.
[42] M. Halkidi, Y. Batistakis, M. Vazirgiannis, On clustering validation techniques, J. Intell. Inf. Syst. 17 (2-3) (2001) 107–145.
[43] V. Roth, M.L. Braun, T. Lange, J.M. Buhmann, Stability-based model order selection in clustering with applications to gene expression data, in: Proceedings of the International Conference on Artificial Neural Networks, 2002, pp. 607–612.
[44] D. Pfitzner, R. Leibbrandt, D. Powers, Characterization and evaluation of similarity measures for pairs of clusterings, Knowl. Inf. Syst. 19 (3) (2009) 361–394.
[45] B. Larsen, C. Aone, Fast and effective text mining using linear-time document clustering, Knowl. Discovery Data Min. (1999) 16–22.
[46] M. Meila, Comparing clusterings by the variation of information, in: Proceedings of COLT/Kernel 2003, 2003, pp. 173–187.
[47] M. Steinbach, G. Karypis, V. Kumar, A comparison of document clustering techniques, Knowl. Discovery Data Min. (2000) 109–110.
[48] J. Bakus, M.F. Hussin, M. Kamel, A SOM-based document clustering using phrases, in: Proceedings of ICONIP'02, 2002, pp. 2212–2216.
[49] A. Rosenberg, J. Hirschberg, V-measure: A conditional entropy-based external cluster evaluation measure, in: Proceedings of EMNLP-CoNLL 2007, 2007, pp. 410–420.
[50] A. Banerjee, C. Krumpelman, S. Basu, R. Mooney, J. Ghosh, Model-based overlapping clustering, in: Proceedings of KDD2005, 2005, pp. 532–537.
[51] E.H. Ramírez, R. Brena, D. Magatti, F. Stella, Topic model validation, Neurocomputing 76 (2012) 125–133 http://dx.doi.org.10.1016/j.neucom.2011.04.032..
[52] A. Bagga, B. Baldwin, Entity-based cross-document coreferencing using the vector space model, in: Proceedings of COLING-ACL'98, 1998, pp. 79–85.

Airel Pérez-Suárez was born in 1979. He received his B.S. in Computer Science from the Havana University, Cuba, in 2002. He received his M.Sc. and Ph.D. degrees in Computational Sciences from the National Institute of Astrophysics, Optics and Electronics (INAOE), Mexico, in 2008 and 2011, respectively. He is currently an Aggregate Researcher of the Data Mining department at the Advanced Technologies Application Centre (CENATAV), Cuba.

José Fco. Martínez-Trinidad received his B.S. degree in Computer Science from Physics and Mathematics School of the Autonomous University of Puebla (BUAP), Mexico in 1995, his M.Sc. degree in Computer Science from the faculty of Computers Science of the Autonomous University of Puebla, Mexico in 1997 and his Ph. D. degree from the Center for Computing Research of the National Polytechnic Institute (CIC, IPN), Mexico in 2000. Professor Martinez-Trinidad edited/authored seven books and over one hundred and twenty journal and conference papers on subjects related to Pattern Recognition.

Jesús A. Carrasco-Ochoa received his Ph.D. degree in Computer Science from the Center for Computing Research of the National Polytechnic Institute (CIC-IPN), Mexico, in 2001. He works as a full time researcher at the National Institute for Astrophysics, Optics and Electronics of Mexico. He has published more than 100 papers on topics related to Pattern Recognition and Data Mining, and co-edited 7 books. His current research interests include Logical Combinatorial Pattern Recognition, Data Mining, Testor Theory, Feature and Prototype Selection, Text Analysis, Fast Nearest Neighbor Classifiers and Clustering.

José E. Medina-Pagola received a B.Sc. in Cybernetic Mathematics from the Havana University in 1977 and his Ph.D. from the Higher Polytechnic Institute "José A. Echeverría" (ISPJAE) in 1996. His research interests include but are not restricted to knowledge discovery and data mining, association rules, clustering, computational linguistic, information retrieval and text mining. He is currently a Senior Researcher and Research Deputy Director of the Advanced Technologies Application Centre (CENATAV), Cuba.