A New Incremental Algorithm for Overlapped Clustering

Airel Pérez Suárez[1,2], José Fco. Martínez Trinidad[1],
Jesús A. Carrasco Ochoa[1], and José E. Medina Pagola[2]

[1] National Institute for Astrophysics, Optics and Electronics (INAOE), Mexico
[2] Advanced Technologies Application Center (CENATAV), Cuba
{airel,fmartine,ariel}@ccc.inaoep.mx, jmedina@cenatav.co.cu

Abstract. In this paper, a new algorithm for incremental overlapped clustering, called Incremental Clustering by Strength Decision (ICSD), is introduced. ICSD obtains a set of dense and overlapped clusters using a new graph cover heuristic while reduces the amount of computation by maintaining incrementally the cluster structure. The experimental results show that our proposal outperforms other graph-based clustering algorithms considering quality measures and also show that ICSD achieves a better time performance than other incremental graph-based algorithms.

1 Introduction

Clustering is one of the most useful and common techniques in pattern recognition and data mining. Nevertheless, most clustering algorithms are non-incremental [1,2,3,4], which means that if new data is added to a training set, the algorithm must be applied again over the whole training set, without taking advantage of the previous clusters.

In environments like the World Wide Web, news streams and others, the data must be organized in overlapped clusters and these clusters must be frequently updated due to new data are continually added. Therefore, the problem of incremental overlapped clustering is addressed in this work.

The *graph-based* algorithms have received attention by the researchers in the last years because this kind of algorithms have features that increase their suitability for many applications where overlapped clustering is needed and they also have shown in this context a better performance that commonly used algorithms [1,2,3].

In this paper, a new incremental overlapped clustering algorithm named Incremental Clustering by Strength Decision (ICSD) is introduced. The novelty of the proposed algorithm is its ability to obtain a set of dense and overlapped clusters using a new graph cover heuristic while it reduces the amount of computation by maintaining clusters structured incrementally. The experimental evaluations showed that ICSD algorithm outperforms other graph-based algorithms [4,5,6].

The remainder of this paper is organized as follows. Section 2 presents related work. The ICSD algorithm is presented in section 3 and finally, conclusions and future work are given in section 5.

E. Bayro-Corrochano and J.-O. Eklundh (Eds.): CIARP 2009, LNCS 5856, pp. 497–504, 2009.

2 Related Work

There are different graph-based algorithms developed during the last years that deal with incremental clustering [5,6,7,8] or with overlapped clusters [1,2,3,4,5], but only a few of them faced both problems at the same time [5,6].

From the group of algorithms that build overlapped clusters [1,2,3,4,5], Cstar [4] has the best performance and outperforms two commonly used clustering algorithms: Single-link and UPGMA algorithms [9]. However, Cstar algorithm only groups static data and it has a computational complexity of $O(n^3)$.

Among all the incremental graph-based algorithms [5,6,7,8] only Star [5] and Strong Compact [6] algorithms faced the problem of overlapped clusters in an incremental context, while GLC[8] and Compact[7] build disjoint clusters; however, all these incremental algorithms have different drawbacks. The set of clusters built by GLC algorithm are connected components so they could have low cohesion. The Star, Compact and Strong Compact algorithms, on the other hand, build a lot of clusters, each one with a few prototypes. Additionally, Star algorithm builds clusters that depends on data order.

Star and Strong Compact algorithms, no matter how many prototypes are added to the dataset, update the set of clusters by adding those prototypes one by one and and they update the clusters after each addition. This is a constraint when a lot of new data must be added at the same time, since it diminishes the performance of these algorithms.

The algorithm proposed in this work introduces a new graph cover heuristic that produces overlapped clusters with high density. Our algorithm keeps the strengths of Cstar algorithm but with a lower computational complexity. The proposed algorithm also allows an efficient cluster update; in this way, the processing time is reduced.

3 Clustering by Strength Decision

The ICSD algorithm, introduced in this section, obtains a set of overlapped clusters through a cover of the *thresholded similarity graph* G_β by applying an heuristic based on the *strength* of vertices in G_β and using *star-shaped sub-graphs*[5].

Let $D = \{p_1, p_2, \ldots, p_n\}$ be a collection of prototypes, β a user-defined parameter and $S(p_i, p_j)$ a symmetric similarity function between prototypes p_i and p_j, a *thresholded similarity graph* is an undirected graph $G_\beta = \langle V, E_\beta \rangle$ where $V = D$ and $(p_i, p_j) \in E_\beta$ if and only if $S(p_i, p_j) \geq \beta$.

An *star-shaped sub-graph* is a sub-graph of $m + 1$ vertices with an special vertex c named *center* and m vertices called *satellites*. This sub-graph satisfies that there is an edge between the center and each satellite. When a star-shaped sub-graph only contains its center it is called *degenerated*. In this context, each star-shaped subgraph is interpreted as a cluster.

A cover of G_β, by this kind of sub-graphs, is determined by the set of centers C such that every vertex of G_β belongs to C or it is adjacent to at least one vertex in C. Usually, to obtain a cover of G_β, a greedy approach is applied [1,2,3,4,5].

The *strength* of a vertex v is calculated in two steps: in the first step, a vertex v receives one vote from each vertex u such that:

$$u \in v.Adj \wedge v.degree \geq u.degree$$

where $v.Adj$ is the set of adjacent vertices of v and $v.degree$ is the number of vertices contained in $v.Adj$ and $u.degree$ is defined in the same way that $v.degree$. The number of votes received by v in this first step will be denoted as $strength_{pre}$.

In a second step, $v.strength$ is calculated as follows:

$$v.strength = |\{u \in v.Adj \mid v.strength_{pre} \geq u.strength_{pre}\}|$$

The heuristic proposed in this work for building overlapped clusters considerers only the set Q of vertices with a non null value of strength; in this way, the set of vertices to be possibly included in C is reduced.

The set Q is sorted in descending order according to the strength and it is processed in that order for covering G_β as quick as possible. Each vertex $v \in Q$ is processed considering the following conditions:

1. if v is not covered in G_β then, it is added to the set of centers,
2. if v is covered in G_β then add v to C if and only if it has at least one not covered adjacent vertex. This condition raises from the fact that as overlap is allowed there would be vertices in Q which have all their satellites covered by previous selected vertices; thus selecting those vertices as center will not cover new vertices in G_β, therefore, they must not be included in C.

Once the set C has been built, a process, similar to that used by Cstar [4] to remove *redundant centers*, but using the degree of vertices instead of their voting-degree, is executed.

We decided to use degree of verties because of two main reasons: (a) since a cluster is defined by a center and its satellites then, the more degree a center has the more density[1] its cluster has, and (b) the vertex v, of a set of vertices M, with the highest voting-degree not always forms the densest cluster, because there could be another vertex $g \in M$, having a higher degree than v, which received a lower voting degree due to most of its adjacent vertices gave their vote to another adjacent vertex.

Finally, remaining vertices in C are marked as *center* and all other vertices in G_β are marked as *satellites*. As result, we will have a set of overlapped clusters, where the set of clusters is built from each vertex marked as center and its satellites.

Assuming that we have a set of overlapped clusters built using the heuristic presented above. Now we will analyze the clusters that are affected when new vertices or prototypes are added.

Let $G' = \langle V', E' \rangle$ be a connected component of G_β and $v \in V$, we will say that v *generates* the component G' if and only if $v \in V'$. A cluster should be updated

[1] In this paper density is defined as the average number of elements per cluster.

if its center, or at least one of its satellites, changes its strength after adding a new vertex. A vertex could changes its strength value only if it belongs to the component generated by some added vertex; therefore, only those components need to be re-clustered.

To update the clusters contained in a connected component it is important to know, first of all, how the insertions affect the current clustering built through the above described heuristic.

After some vertices are added to G_β the following could happen:

a) there are uncovered vertices; therefore, new vertices must be added to set C.
b) there is at least one non center vertex v which has an strength value greater than one of its adjacent centers or one center $c \in C$ that covers some vertices in $v.Adj$. All vertices like v should be considered to be included in C because they could improve the clustering's density.

For updating the clusters of a connected component $G' =< V', E' >$ the set of vertices V' is divided in the sets V'_s and V'_c which contain the set of all satellites having strength greater than zero and the set of all centers, respectively. Each one of these sets is processed independently to determine which vertices must be added to the candidate list Q.

Each satellite $s \in V'_s$ is processed considering the following conditions:

a) if s is uncovered or has at least one uncovered adjacent vertex then s is inserted into Q.
b) if s has at least one adjacent vertex v such that $s.strength > w.strength$, where w is the center having the higher value of strength among all adjacent centers of v then, s is inserted into Q and v is marked as *activated*; vertices marked as *activated* are useful when set V_c is processed.

Each center $c \in V'_c$ is processed considering the following conditions:

a) for each $v \in c.Adj$ such that $v.strength > c.strength$, insert v into Q and mark c as *weak*.
b) if c is marked as *weak* or it has at least one adjacent vertex marked as *activated* then, c is removed from V_c, it is marked as satellite and if $c.strength > 0$, c is inserted into Q.

Once the set Q has been built for the connected component G', it is processed using the heuristic proposed above for building overlapped clusters.

The pseudocode of ICSD is showed in Algorithm 1.

ICSD algorithm has a computational complexity of $O(n^2)$ (the proof was omitted due to space restrictions). ICSD unlike Star, Compact and Strong Compact algorithms, adds all new incoming vertices to G_β before updating the set of clusters, and it also applies an heuristic which only processes the clusters that have been actually affected due to insertions. These characteristics of ICSD make the algorithm to save time making it able to efficiently manage multiple insertions of prototypes.

Algorithm 1. ICSD algorithm

Input: G_β - a thresholded similarity graph
L - set of incoming prototypes
β - a similarity threshold
Output: G_β - updated thresholded similarity graph
SC - set of overlapped clusters

```
1  "Add each vertex in L to G_β and update G_β";
2  foreach new added vertex v do
3      if v is marked as not-processed then
4          "Build the connected component G' =< V', E' > associated with v";
5          if G' is an isolated vertex then "Mark v as center";
6          else
7              "calculate strength property for each vertex in G'";
8              "Build V'_s, V'_c, and Q sets";
9              while Q ≠ ∅ do
10                 u := arg max_x{x.strength | x ∈ Q};
11                 if u satisfies conditions 1) or 2) then "Add u to V'_c";
12                 Q := Q \ {u};
13             end
14             "Sort V'_c in ascending order by degree";
15             "Remove from V'_c all redundant centers";
16             "Mark vertices in V'_c as center and vertices in V' \ V'_c as satellite";
17         end
18         "Mark vertices in V' as processed";
19     end
20 end
21 "Mark vertices in V as not-processed";
22 "Build set SC";
```

4 Experimental Evaluation

In this section, an experimental evaluation of the proposed algorithm is presented. Since ICSD algorithm deals with overlapping clustering, the experiments were done over document collections where, as it was mentioned, some documents could belong to more than one cluster.

The document collections used in our experiments were extracted from three benchmark text collection: TREC-5, *Reuters*-21578 and TDT2. From these benchmarks, six document collections were formed: (1) AFP, built from TREC--5; (2) Reu-Te, built using the documents in *Reuters*-21578 tagged as "Test"; (3) Reu-Tr, built using the documents in *Reuters*-21578 tagged as "Train"; (4) Reu-To is the union of Reu-Te and Reu-Tr; (5) TDT2-v1 and (6) TDT2-v2 are two sub-collections of TDT2. The characteristics of all these collections are summarized in Table 1.

In the experiments, documents are represented using the Vector Space model where index terms represent the lemmas of words appearing in the collection; Stop words were removed. The terms of each document were statistically

Table 1. Characteristics of document collections

Collection	Documents	Topics	Overlapping	Terms
AFP	695	25	1.02	11785
Reu-Te	3587	100	1.30	15113
Reu-Tr	7780	115	1.24	21901
Reu-To	11367	120	1.26	27083
TDT2-v1	8603	176	1.17	51764
TDT2-v2	10258	174	1.19	53706

weighted using the logarithm of term's frequency. The cosine measure was used as similarity function.

For the experiments presented in this section, two measures commonly used to evaluate overlapped clustering were selected: Fmeasure (Fme) [10] and Jaccard--index (Jindex) [11]. Both measures evaluate quality based on how much the clustering resembles a set of classes manually labeled by experts; the higher the value of each measure is the better the clustering is.

The experiments were focused on comparing, through Fmeasure and Jaccard-index, the set of clusters obtained by Strong Compact (SComp), Star and Cstar algorithms against the clusters built by ICSD algorithm; Although Cstar is a non incremental algorithm, we decided to include it in this first experiment because it obtained the best quality results for overlapped clustering in different works [3,4]. Table 2 shows the best value of Fmeasure and Jaccard-index obtained by each algorithm for β values in [0.15,0.75].

Table 2. Results for each document collection

		AFP				*Reu-Te*				*Reu-Tr*			
Measures		SComp	Star	Cstar	ICSD	SComp	Star	Cstar	ICSD	SComp	Star	Cstar	ICSD
Fme	value	0.10	0.73	0.76	0.76	0.01	0.57	0.63	0.64	<0.01	0.56	0.56	0.57
	β	0.15	0.25	0.25	0.25	0.15	0.25	0.25	0.25	0.15	0.20	0.25	0.25
Jindex	value	0.05	0.57	0.61	0.61	0.01	0.40	0.46	0.47	<0.01	0.39	0.39	0.40
	β	0.15	0.25	0.25	0.25	0.15	0.25	0.25	0.25	0.15	0.20	0.25	0.25
		Reu-To				*TDT2-v1*				*TDT2-v2*			
Measures		SComp	Star	Cstar	ICSD	SComp	Star	Cstar	ICSD	SComp	Star	Cstar	ICSD
Fme	value	0.01	0.57	0.58	0.59	0.01	0.39	0.44	0.45	0.01	0.44	0.52	0.52
	β	0.15	0.20	0.25	0.25	0.15	0.30	0.30	0.30	0.15	0.30	0.30	0.30
Jindex	value	<0.01	0.40	0.41	0.41	0.01	0.24	0.28	0.29	<0.01	0.28	0.35	0.35
	β	0.15	0.20	0.25	0.25	0.15	0.30	0.30	0.30	0.15	0.30	0.30	0.30

As it can be noticed from Table 2, ICSD algorithm outperforms all the other algorithms in almost all cases.

Fig. 1 shows the results of a second experiment done in order to show the time spent by ICSD, Star and SComp, to cluster the largest tested collection (*Reu-To*) in an incremental way. In Fig. 1(A), curves ICSD, Star and SComp represent the time spent by each algorithm to update the clusters each time 1000

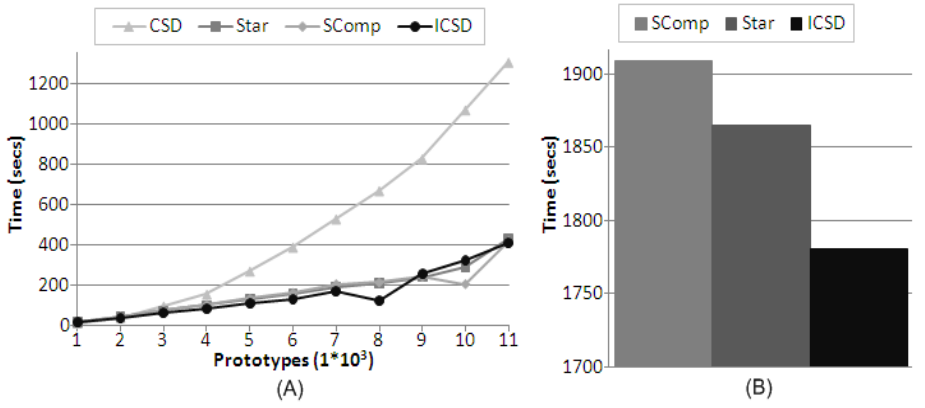

Fig. 1. (A) Behavior of incremental algorithm , B) total time for clustering the whole dataset in an incremental way

Table 3. Comparison considering number and density of clusters

	AFP		*Reu-Te*		*Reu-Tr*		*Reu-To*		*TDT2-v1*		*TDT2-v2*	
Alg.	Grp	Dty	Grp	Dty	Grp	Dty	Grp	Dty	Grp	Dty	Grp	Dty
SComp	413	3.4	1981	3.5	4366	3.6	6437	3.5	4834	3.5	5587	3.5
Star	54	29.1	157	110.8	203	224.2	255	294.4	241	278.8	254	331.5
Cstar	41	68.0	101	523.9	132	1420.5	177	1980.7	168	3349.5	172	3864.6
ICSD	41	69.9	104	546.2	136	1452.0	178	2030.6	172	3401.4	175	3938.5

documents are added[2] and curve CSD represents the time spent to cluster all the prototypes from scratch. Fig. 1(B) shows the total time spent by ICSD, Star and SComp to cluster incrementally the entire dataset; CSD was not included in this figure to avoid scale problems.

As it can be observed, ICSD overcomes Star and SComp and also it scales well in comparison with a non incremental clustering. A similar behavior was observed in the other datasets.

Finally, as SComp is the algorithm which produces the large number of clusters, we selected the β value (for building G_β) for which SComp obtained the smallest number of clusters and we compared the number and density of those clusters with those obtained by Star, CStar and ICSD for the same β. Table 3 shows the aforementioned comparison; in this table, columns "Gpr" and "Dty" represent the number of clusters and the density of those clusters respectively.

As it can be noticed from Table 3, ICSD outperforms Star and SComp in all datasets getting a less number of clusters with a higher density and it also obtains similar results than those obtained by Cstar.

[2] We selected 1000 because it is a number neither big nor small considering the size of *Reu-To*.

5 Conclusions

In this paper, a new algorithm called Incremental Clustering by Strength Decision (ICSD) has been proposed. ICSD algorithm builds a set of dense and overlapped clusters applying a new heuristic for covering a thresholded similarity graph which allows its application for incremental environments.

The heuristic introduced by ICSD algorithm processes only the clusters actually affected by additions, which makes ICSD to save time, making it able to efficiently manage multiple insertions in incremental environments.

ICSD algorithm was compared against other graph-based algorithms on six document collections. The experimental results show that our proposal outperforms those methods. Moreover, ICSD achieves better time performance than previous incremental overlapped graph-based algorithms in incremental datasets.

As future work we will develop a version of ICSD that allows additions and deletions, in order to increase its applicability in other environments.

References

1. Gil-García, R.J., Badía-Contelles, J.M., Pons-Porrata, A.: Extended Star Clustering Algorithm. In: Sanfeliu, A., Ruiz-Shulcloper, J. (eds.) CIARP 2003. LNCS, vol. 2905, pp. 480–487. Springer, Heidelberg (2003)
2. Pérez-Suárez, A., Medina-Pagola, J.E.: A Clustering Algorithm Based on Generalized Stars. In: Perner, P. (ed.) MLDM 2007. LNCS (LNAI), vol. 4571, pp. 248–262. Springer, Heidelberg (2007)
3. Gago-Alonso, A., Pérez-Suárez, A., Medina-Pagola, J.E.: ACONS: A New Algorithm for Clustering Documents. In: Rueda, L., Mery, D., Kittler, J. (eds.) CIARP 2007. LNCS, vol. 4756, pp. 664–673. Springer, Heidelberg (2007)
4. Pérez-Suárez, A., Martínez-Trinidad, J.F., Carrasco-Ochoa, J.A., Medina-Pagola, J.E.: A New Graph-Based Algorithm for Clustering Documents. In: ICDM-Workshops 2008, pp. 710–719 (2008)
5. Aslam, J., Pelekhov, E., Rus, D.: The star clustering algorithm for static and dynamic information organization. Journal of Graph Algorithms and Applications 8(1), 95–129 (2004)
6. Pons-Porrata, A., Ruiz-Shulcloper, J., Berlanga-Llavori, R., Santiesteban-Alganza, Y.: Un algoritmo incremental para la obtención de cubrimientos con datos mezclados. In: CIARP 2002, pp. 405–416 (2002)
7. Pons-Porrata, A., Berlanga-Llavori, R., Ruiz-Shulcloper, J.: On-line event and topic detection by using the compact sets clustering algorithm. Journal of Intelligent and Fuzzy Systems 12(3), 185–194 (2002)
8. Ruiz-Shulcloper, J., Sanchez, G., Abidi, M.A.: Clustering in mixed incomplete data. Heuristics and Optimization for Knowledge Discovery, 88–106 (2002)
9. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification. John Wiley & Sons Inc., New York (2001)
10. Banerjee, A., Krumpelman, C., Basu, S., Mooney, R., Ghosh, J.: Model based overlapping clustering. In: KDD 2005, pp. 532–537 (2005)
11. Kuncheva, L., Hadjitodorov, S.: Using diversity in cluster ensembles. In: IEEE SMC 2004, The Netherlands, pp. 1214–1219 (2004)