Algoritmos dinámicos para el agrupamiento con traslape

Airel Pérez Suárez, José Fco. Martínez Trinidad,
José E. Medina Pagola, Jesús Ariel Carrasco Ochoa

Reporte Técnico No. CCC-10-001
25 de enero de 2010

Coordinación de Ciencias Computacionales
INAOE

Luis Enrique Erro 1
Sta. Ma. Tonantzintla,
72840, Puebla, México.

Algoritmos dinámicos para el agrupamiento con traslape

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

[1]Computer Science Department
National Institute of Astrophysics, Optics and Electronics
Luis Enrique Erro # 1, Santa María Tonantzintla, Puebla, 72840, México
[2]Data Mining Departament
Advanced Technologies Application Center
21812 e 218 y 222, Rpto. Siboney, Playa, La Habana, 12200, Cuba
E-mail: {airel,fmartine,ariel}@inaoep.mx,jmedina@cenatav.co.cu

Abstract

El desarrollo de nuevos algoritmos de agrupamiento, hoy en día, continúa siendo objeto de interés debido a su amplia variedad de aplicaciones. No obstante a los avances logrados, la mayoía de los algoritmos de agrupamiento propuestos hasta el momento son estáticos, es decir, suponen que antes de comenzar el agrupamiento se cuenta con todos los objetos a agrupar; luego, cuando ocurre algún cambio en la colección, estos algoritmos tienen que re-procesar toda la colección para poder agruparla. Adicionalmente, la mayoía de los algoritmos no permiten solapamiento entre los grupos formados y tienen además algunas limitaciones que pueden reducir su utilidad y su aplicación. Esta propuesta de tesis doctoral aborda el problema del desarrollo de algoritmos de agrupamiento, jerárquicos y no jerárquicos, que sean dinámicos y que obtengan un conjunto de grupos que pueden ser solapados. Como resultado preliminar se presenta primeramente un nuevo algoritmo incremental llamado ICSD, que permite la obtención de grupos que pueden ser traslapados. Posteriormente, se presenta un conjunto de resultados experimentales que muestran el desempeño de ICSD en comparación con otros algoritmos reportados. Los resultados experimentales mostraron que el algoritmo propuesto obtiene mejores resultados de eficacia, de acuerdo a las medidas utilizadas, que los algoritmos reportados y que además es más rápido en el procesamiento incremental de colecciones que los algoritmos incrementales utilizados en los experimentos.

1 Introducción

Desde finales de los años noventa los avances tecnológicos en áreas como la comunicación y la informática han permitido un incremento en las posibilidades de generación, acceso y almacenamiento de información. Para realizar un mejor y más fácil análisis de esta información (imágenes, texto, video, etc.) se han desarrollado varias aplicaciones, algunas de las cuales utilizan técnicas de Minería de Datos como el agrupamiento.

El desarrollo de nuevos algoritmos de agrupamiento, hoy en día, continúa siendo objeto de interés debido a su amplia variedad de aplicaciones. Algunas de estas aplicaciones incluyen por ejemplo: (a) el estudio de secuencias de genes [2] o de enfermedades como el cáncer de mama [3], (b) el procesamiento de imágenes [4] y datos espaciales [5], (c) la detección y seguimiento de flujos de noticias y correos [6, 7], (d) la determinación del comportamiento de usuarios en sitios web y la personalización de estos sitios de acuerdo a los intereses de los usuarios [8, 9], etc.

No obstante a los avances logrados, muchos de los algoritmos de agrupamiento propuestos hasta el momento tienen algunas limitaciones que les impiden satisfacer eficientemente requerimientos prácticos. Estos requerimientos están relacionados fundamentalmente con: (a) el tipo de grupo que se obtiene como resultado y (b) la capacidad de los algoritmos para procesar cambios que puedan ocurrir en las colecciones.

Los grupos obtenidos por un algoritmo de agrupamiento pueden ser disjuntos o solapados e incluso pueden ser difusos (fuzzy). La mayoría de los algoritmos reportados permiten obtener sólo grupos disjuntos, pese a que existen varias aplicaciones en las que un objeto puede pertenecer a más de un grupo.

Considérese por ejemplo una aplicación para un centro de salud donde, existen registros que contienen información relacionada con los pacientes. Es posible que algunos pacientes tengan diferentes enfermedades y sin embargo presenten síntomas en común; luego, si se supone que los síntomas determinan las enfermedades entonces existirán pacientes que pueden pertenecer a más de un grupo, cada uno representando una enfermedad. Este ejemplo no es un caso aislado, sino que puede encontrarse también en diferentes aplicaciones como por ejemplo: el análisis de flujo de noticias, en bibliotecas digitales, en la clasificación de zonas geográficas, en la prospección geológica, etc.

Los algoritmos de agrupamiento pueden clasificarse atendiendo a varios criterios [10]. Uno de los criterios utilizados, clasifica a los algoritmos de acuerdo a la capacidad que éstos tienen para procesar cambios en la colección a agrupar. Los *algoritmos incrementales* son aquellos capaces de procesar nuevos objetos a medida que éstos son adicionados a la colección y que, en consecuencia con los cambios, actualizan el conjunto de grupos utilizando el agrupamiento anterior. Los *algoritmos dinámicos*, por otra parte, permiten actualizar el conjunto de grupos cuando se adicionan nuevos objetos, cuando se eliminan objetos e incluso cuando se modifican objetos; esta última operación puede verse como una eliminación seguida de una adición y como tal será tratada en el presente trabajo.

Los *algoritmos estáticos* son aquellos en los que se supone que antes de comenzar el agrupamiento se cuenta con todos los objetos a agrupar. Luego, cuando ocurre algún cambio en la colección, estos algoritmos tienen que re-procesar toda la colección para poder agruparla; i.e., no utilizan el agrupamiento previo para actualizar los grupos. La mayoría de los algoritmos de agrupamiento reportados en la literatura son del tipo estático; sin embargo, este tipo de algoritmos resulta poco eficiente en aquellos sistemas que procesan colecciones que sufren cambios con cierta frecuencia, por ejemplo: sistemas de análisis de flujos de noticias, sistemas de compras online, sistemas para el análisis de las facturas de una empresa, etc.

En la literatura se encuentran reportados varios algoritmos incrementales [20, 11, 17] y dinámicos [25, 31, 34]. De forma general, estos algoritmos poseen algunas de las limitaciones siguientes: (a) realizan una asignación irrevocable de los objetos a los grupos, (b) los grupos obtenidos presentan *encadenamiento*; i.e., baja cohesión interna, (c) son ineficientes para objetos descritos por un gran número de rasgos (atributos), (d) imponen restricciones en cuanto al tipo de objeto a agrupar, e.g., sólo agrupan documentos u objetos ordenados cronológicamente, (e) necesitan optimizar varios parámetros cuyos valores son dependientes de la colección o (f) obtienen un número elevado de grupos, generalmente con muy pocos objetos.

Otro aspecto, que resulta de interés en varias aplicaciones en la actualidad, es brindar información acerca de cómo están relacionados los grupos obtenidos; es decir, qué los diferencia, qué los asemeja, etc. Para solucionar este problema se ha trabajado en el desarrollo de *algoritmos jerárquicos* [38, 45, 46, 41].

La mayoría de los algoritmos jerárquicos desarrollados hasta el momento son estáticos, no permiten obtener grupos con traslape y/o presentan además algunas de las siguientes limitaciones: (a) imponen restricciones en cuanto al tipo de objeto a agrupar; e.g., objetos descritos en espacios métricos o sólo documentos, (b) necesitan optimizar varios parámetros cuyos valores son dependientes de la colección, (c) sólo son eficientes para objetos descritos por un número peque˜no de rasgos, (d) forman grupos con encadenamiento, (e) poco estables; i.e., peque˜nos cambios provocan una re-estructuración considerable o (f) poco eficientes para colecciones de grandes volúmenes de objetos.

1.1 Definición del problema

Sea $C = \{O_1, O_2, \ldots, O_n\}$ una colección de objetos, cada uno descrito a través de un conjunto de rasgos $R = \{r_1, r_2, \ldots, r_l\}$; un algoritmo de agrupamiento es aquel que organiza la colección antes mencionada en clases o grupos de forma tal que la semejanza entre objetos pertenecientes a un mismo grupo sea alta, en contraposición a la semejanza entre objetos de grupos diferentes. Se dice que el conjunto de grupos, obtenido por un algoritmo de agrupamiento, es *solapado* si en dicho conjunto existen objetos que pertenezcan a más de un grupo a la vez.

Sea C una colección de objetos como la descrita anteriormente; un algoritmo de agrupamiento es *jerárquico* si el mismo construye una estructura compuesta de k niveles, siendo el nivel 1 el *más general o supremo* y el nivel k el *más específico o ínfimo*, que cumple las siguientes condiciones:

a) Cada nivel $N_i = \{g_{i_1}, g_{i_2}, \ldots, g_{i_{M_i}}\}$, $i = 1..k-1$ satisface que:
   - $|N_i| = M_i$
   - $\forall g_{i_j}, j = 1..M_i$ se cumple que $g_{i_j} \subseteq C$
   - $|N_i| < |N_{i+1}|$

b) $N_k = \{\{O_1\}, \{O_2\}, \ldots, \{O_n\}\}$; $|N_k| = |C| = n$

c) $\forall g_{i_j}$, $i = 1..k-1$, $j = 1..M_i$ existe una relación *padre-hijo* con al menos un elemento $g_{(i+1)_q}$, $q = 1..M_{i+1}$. Adicionalmente, si un grupo A tiene una relación *padre-hijo* con un grupo B entonces se cumple que $B \subseteq A$ y se puede decir también que B tiene una relación *es-hijo-de* con A.

d) $\forall g_{i_j}$, $i = 1..k-1$, $j = 1..M_i$ $g_{i_j} = \bigcup\{g_{(i+1)_q} \mid q = 1..M_{i+1} \wedge g_{(i+1)_q}$ *es-hijo-de* $g_{i_j}\}$

La condición a) establece cómo está formado cada nivel de la estructura, qué características presentan los grupos que conforman dichos niveles y qué relación existe entre el número de grupos de un nivel y el de su nivel inmediato inferior. La condición b) plantea que el último nivel (el nivel k) de la jerarquía es aquel en el cual cada elemento de C forma un conjunto unitario. La condición c), por otra parte, establece la relación que existe entre cada uno de los elementos de un nivel y los elementos del nivel inmediato inferior. Finalmente, la condición d) define que cualquier grupo g de un nivel se obtiene a través de la unión de los grupos del nivel inmediato inferior con los cuales g tienen una relación *padre-hijo*. Un ejemplo de una estructura de agrupamiento jerárquico se muestra en la figura 1.

Un algoritmo de agrupamiento jerárquico construye una jerarquía de grupos *solapados* si en dicha jerarquía se permite que el conjunto de grupos, obtenidos en cualquier nivel $N_i, i = 1..k-1$, pueda ser solapado.

El problema que se plantea en esta propuesta de tesis doctoral es el desarrollo de algoritmos de agrupamiento, jerárquicos y no jerárquicos, que sean dinámicos y que obtengan un conjunto de grupos que pueden ser solapados.

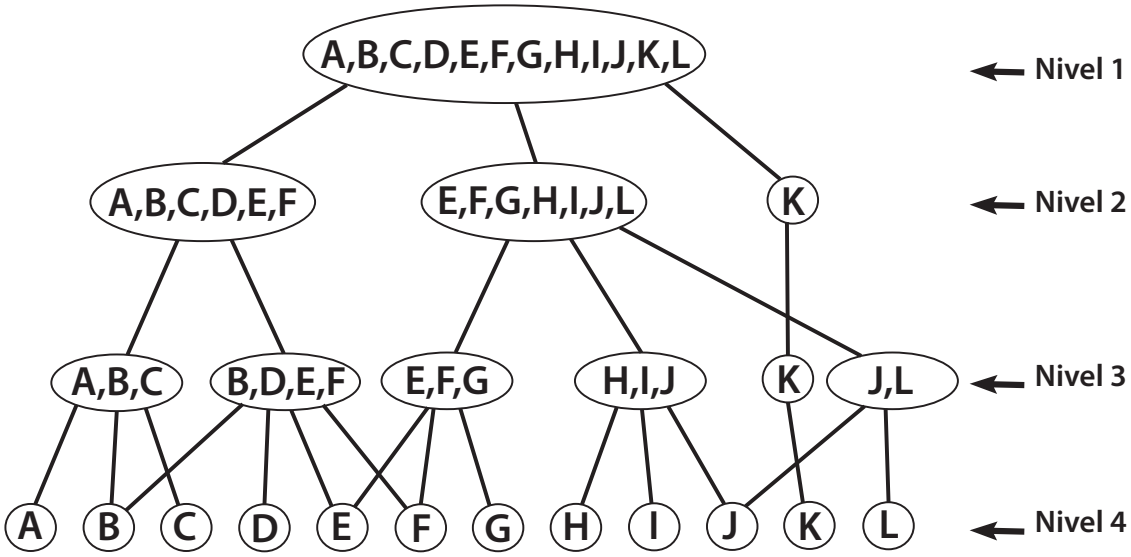


Figure 1. Ejemplo de estructura de agrupamiento jerárquico

1.2 Organización del documento

Esta propuesta de tesis doctoral está organizada como sigue: En la sección 2 se describe el trabajo relacionado y en la sección 3 se presenta la motivación para el desarrollo de esta investigación. La propuesta de investigación se presenta en la sección 4, incluyéndose además en esta sección las preguntas de investigación, los objetivos generales y específicos, la metodología, las contribuciones esperadas y el cronograma de actividades. Los resultados preliminares se presentan en la sección 5 y finalmente, en la sección 6 se exponen las conclusiones.

2 Trabajo relacionado

En la literatura se han reportados varios algoritmos de agrupamiento, jerárquicos y no jerárquicos, tanto del tipo incremental como del tipo dinámico. En esta sección se describen los principales trabajos relacionados con esta investigación.

2.1 Algoritmos no jerárquicos

2.1.1 Algoritmo Single-Pass

Single-Pass [11] es un algoritmo del tipo *"de una sola pasada"* que ha sido utilizado en sistemas para la detección de tópicos [12]. Este algoritmo, para agrupar cada nuevo objeto adicionado a la colección, calcula la semejanza máxima entre el objeto adicionado y los grupos existentes. Luego, si este valor máximo supera un umbral pre-establecido entonces el objeto es adicionado al grupo con el cual se alcanzó la semejanza máxima; si existe más de un grupo que cumpla esta condición entonces se selecciona uno de ellos aleatoriamente. En otro caso, si el valor máximo de semejanza calculado no supera el umbral entonces el objeto conforma un grupo independiente.

El Single-Pass es un algoritmo incremental que obtiene un conjunto de grupos disjuntos. Entre las limitaciones de este algoritmo están: (a) es dependiente del orden de análisis de los objetos, (b) realiza asignación

irrevocable de los objetos a los grupos, (c) no permite el procesamiento de adiciones múltiples[1] de objetos y (d) independientemente del orden de análisis de los objetos, puede obtener un agrupamiento diferente cada vez que se ejecuta sobre una misma colección.

2.1.2 Algoritmo IncrementalDBSCAN

IncrementalDBSCAN [13] es un algoritmo dinámico desarrollado para el procesamiento de un *Data Warehousing* (Almacén de datos). La estrategia de agrupamiento que sigue IncrementalDBSCAN se basa en el algoritmo DBSCAN [14]. Tanto DBSCAN como IncrementalDBSCAN son algoritmos *basados en densidad*.

Los algoritmos basados en densidad consideran como grupos a las regiones *densamente* pobladas del espacio de representación de los objetos y como ruido a aquellos objetos que se encuentran fuera de estas regiones. Una región está densamente poblada si el número de objetos dentro de la misma supera un umbral pre-establecido.

IncrementalDBSCAN ha sido utilizado para el agrupamiento de datos espaciales y records de una base de datos de logs de la WWW. Este algoritmo construye un conjunto de grupos disjuntos que es independiente del orden de análisis de los objetos y puede utilizar estructuras de datos como R*-tree [15] o M-tree [16] para almacenar los objetos.

Entre las limitaciones de IncrementalDBSCAN se encuentra que: (a) sólo es aplicable a datos que estén descritos en espacios métricos de baja dimensionalidad, (b) puede formar grupos con encadenamiento; i.e., baja cohesion interna, (c) se necesita optimizar varios parámetros cuyos valores son dependientes de la colección a procesar y (d) no permite el procesamiento de adiciones o eliminaciones múltiples de objetos.

2.1.3 Algoritmo STC

El algoritmo STC (*Suffix Tree Clustering*) [17] es un algoritmo incremental basado en árboles que fue creado para el agrupamiento de *snippets*[2]. STC parte de formar un *árbol de sufijos* (Suffix Tree) [18] que contiene todos los sufijos contenidos en el conjunto de snippets a agrupar; cada nodo de este árbol, que almacene dos o más snippets, es llamado *grupo base*. A continuación, se construye el conjunto final de grupos siguiendo una estrategia de agrupamiento que une iterativamente los grupos base que sean semejantes; la semejanza entre dos grupos base se define a partir del traslape de éstos.

Cuando se adiciona un nuevo snippet a la colección, se actualiza el árbol de sufijos y se determina el conjunto de grupos base que fueron modificados. A continuación se actualiza la semejanza de cada grupo base a los k grupos de mayor ranking y se re-calcula desde cero el conjunto final de grupos; el ranking de un grupo base se define en función del número de snippets contenidos en el mismo y de las palabras que componen la frase formada desde el nodo raíz hasta el grupo base.

El conjunto de grupos que obtiene el algoritmo STC puede ser solapado. Este algoritmo es capaz de procesar adiciones múltiples, pues puede incorporar todos los objetos al árbol de sufijos y luego entonces actualizar el agrupamiento. Entre las limitaciones de este algoritmo se encuentra que: (a) sólo es aplicable a documentos, (b) la construcción del árbol de sufijos puede ser extremadamente costosa para documentos de mayores dimensiones que los snippets debido a la alta redundancia que existe en dicho árbol, (c) el conjunto final de grupos puede tener encadenamiento y ser dependiente del orden de análisis de los grupos base, (d)

[1] Más de un elemento a la vez

[2] Los snippets son los pequeños textos utilizados por los sistemas de búsqueda como Google para describir los resultados de las búsquedas

se necesita optimizar varios parámetros cuyos valores son dependientes de la colección a procesar y (e) no es del todo incremental, pues una parte de su proceso es estático; luego, puede ser muy ineficiente debido a la construcción desde cero de los grupos cada vez que ocurren cambios en la colección.

2.1.4 Algoritmos GLC y GLC+

Los algoritmos GLC [20] y GLC+ [21] son algoritmos incrementales basados en grafos. Estos algoritmos representan la colección de objetos a través de su *grafo de β-semejanza*.

Sea $\beta \in [0, 1]$ un umbral de semejanza, $C = \{O_1, O_2, \ldots, O_n\}$ una colección de objetos y S una función de semejanza simétrica entre los objetos de C. Un *grafo de β-semejanza* se denota por $G_\beta = \langle V, E_\beta \rangle$ y es el grafo no dirigido en el cual los vértices son los objetos de C y existe una arista $(O_i, O_j) \in E_\beta$ entre los objetos O_i y O_j si y sólo si $S(O_i, O_j) \geq \beta$.

Un conjunto $B = \{s_1, s_2, \ldots, s_k\}$ es *β-conexo* si para todo par de elementos $s_i, s_j \in S$ existe un conjunto $P = \{p_1, p_2, \ldots, p_m\}$ tal que $P \subseteq B$ y se cumple que: $a)$ $s_i = p_1$ y $s_j = p_m$ o $s_j = p_1$ y $s_i = p_m$ y $b)$ $S(p_x, p_{x+1}) \geq \beta$ para todo $x = 1..m - 1$.

Un conjunto β-conexo $B = \{s_1, s_2, \ldots, s_k\}$ es una *componente β-conexa* si es máximo; i.e., si se adiciona un objeto más a S entonces éste pierde la propiedad de ser β-conexo.

El algoritmo GLC calcula las componentes β-conexas de G_β y éstas determinan el conjunto final de grupos. El algoritmo GLC+ utiliza al algoritmo GLC en su estrategia de agrupamiento y construye una partición de G_β en conjuntos β-conexos.

Ambos algoritmos no imponen restricciones al espacio de representación de los objetos ni a la función de semejanza y obtienen un conjunto de grupos disjuntos que es independiente del orden de análisis de los objetos. Estos algoritmos pueden modificarse para ser capaces de procesar adiciones múltiples, pero si se hiciera tales modificaciones entonces los algoritmos resultantes serían menos eficientes que los actuales pues se tendría que calcular obligatoriamente todas las semejanzas y luego construir las componentes conexas. La principal limitación de estos algoritmos es que construyen grupos con un alto encadenamiento. Adicionalmente, GLC+ necesita optimizar varios parámetros cuyos valores son dependientes de la colección a procesar y usualmente produce más grupos que el algoritmo GLC.

2.1.5 Algoritmo Compacto Incremental

El algoritmo Compacto Incremental (CI) [22] es un algoritmo incremental basado en grafos que representa la colección de objetos a través de su *grafo de máxima β-semejanza*.

Sea $\beta \in [0, 1]$ un umbral de semejanza, $C = \{O_1, O_2, \ldots, O_n\}$ una colección de objetos y S una función de semejanza simétrica entre los objetos de C. Un *grafo de máxima β-semejanza* se denota por $G_{\beta max} = \langle V, E_{\beta max} \rangle$ y es el grafo dirigido tal que $V = C$ y existe una arista dirigida $\langle O_i, O_j \rangle \in E_{\beta max}$ entre los objetos O_i y O_j si y sólo si $S(O_i, O_j) = \max \{S(O_i, O_k) \mid O_k \in V \wedge O_k \neq O_i \wedge S(O_i, O_k) \geq \beta\}$.

El algoritmo CI no impone restricciones al espacio de representación de los objetos ni a la función de semejanza. Este algoritmo construye un conjunto de grupos disjuntos en el cual cada grupo es un *conjunto compacto* de $G_{\beta max}$; los conjuntos compactos coinciden con las componentes conexas del grafo no dirigido asociado a $G_{\beta max}$. El conjunto de grupos que se obtiene es independiente del orden de análisis de los objetos. Este algoritmo es capaz de procesar adiciones múltiples de objetos, pero podría resultar poco eficiente en el procesamiento de dichas adiciones.

Entre las limitaciones de este algoritmo se encuentran: (a) obtiene una gran cantidad de grupos, cada uno formado por pocos elementos, (b) los grupos construidos pueden tener encadenamiento aunque en menor magnitud que los grupos generados por los algoritmos GLC o GLC+.

2.1.6 Algoritmo Fuertemente Compacto Incremental

El algoritmo Fuertemente Compacto Incremental (FCI) [23] es un algoritmo incremental basado en grafos que representa a la colección a través de su *grafo de máxima β-semejanza*.

El algoritmo FCI construye un conjunto de grupos solapados e independientes del orden de análisis de los objetos, en el cual, cada grupo es un *conjunto fuertemente compacto* en $G_{\beta max}$. Los conjuntos fuertemente compactos se determinan a partir de los conjuntos compactos de $G_{\beta max}$ basándose en la relación que establece que *"Todo conjunto compacto es la unión finita de conjuntos fuertemente compactos"* [24].

FCI obtiene un conjunto de grupos altamente cohesionados e independientes del orden de análisis de los objetos. Este algoritmo, de forma similar al algoritmo CI, es capaz de procesar adiciones múltiples pero podría resultar poco eficiente en tal procesamiento. La principal limitación de este algoritmo es que obtiene muchos grupos los cuales, generalmente, contienen pocos elementos; lo anterior puede ser poco útil en varias aplicaciones prácticas.

2.1.7 Algoritmo Star

El algoritmo Star [25] es un algoritmo dinámico basado en grafos el cual ha sido utilizado en aplicaciones relacionadas con el filtrado [26] y la organización de información [27]. Este algoritmo, de forma similar al algoritmo GLC, representa la colección de objetos a agrupar a través de su *grafo de β-semejanza G_β*.

La estrategia que sigue Star para obtener un agrupamiento, construye un conjunto de grupos que pueden ser solapados, a partir de un cubrimiento de G_β utilizando *sub-grafos en forma de estrella*.

Un *sub-grafo en forma de estrella* es un sub-grafo de $m+1$ vértices que tiene un vértice especial llamado *centro* y m vértices llamados *satélites*, en el cual existe una arista entre el centro y cada satélite. En este contexto, cada sub-grafo determina un grupo del agrupamiento final.

Este algoritmo no necesita conocer la cantidad de grupos a formar, no impone restricción ni al espacio de representación de los objetos ni a la función de semejanza entre éstos. El algoritmo en su estado actual no es capaz de procesar adiciones múltiples, aunque pudiese modificarse con tal propósito. Entre las limitaciones de este algoritmo está que: (a) obtiene un conjunto de grupos dependiente del orden de análisis de los objetos, (b) no permite el procesamiento de adiciones o eliminaciones múltiples de objetos y (c) obtiene muchos grupos los cuales, en promedio, contienen pocos elementos.

2.1.8 Novelty-based Incremental Document Clustering (NIDC)

NIDC [28] es un algoritmo incremental desarrollado para el agrupamiento de noticias. Este algoritmo utiliza, para definir la función que establece la semejanza entre dos noticias, el *document forgetting model* (dfm) introducido por el algoritmo F^{2}ICM [29].

El modelo dfm refleja intuitivamente el hecho de que las noticias pierden gradualmente su valor luego de insertarse en la colección y por tanto, mientras más pase el tiempo y una noticia se haga "vieja", su semejanza con otras noticias será menor.

La estrategia de agrupamiento que sigue NIDC es una extensión de la del algoritmo k-means [30]. En esta estrategia, a diferencia del k-means original, las noticias son adicionadas al grupo que logre el mayor incremento en su *semejanza interna*[3]; las noticias que no logren incremento alguno son adicionadas a una lista de *outliers* y procesadas en la próxima iteración. Una vez que se adicionan noticias a la colección se ejecuta sobre toda la colección la estrategia de agrupamiento anteriormente descrita, utilizando como centroides iniciales los calculados en la última iteración del agrupamiento previo.

[3]Esta semejanza se define en función de la semejanza existente de todas las noticias del grupo

NIDC es un algoritmo que construye un conjunto de grupos disjuntos cuyo número necesita conocer a priori. Entre las limitaciones de este algoritmo está que: (a) sólo es aplicable a documentos ordenados cronológicamente, (b) puede caer en mínimos locales de la función objetivo, (c) necesita optimizar varios parámetros cuyos valores son dependientes de la colección a procesar y (d) independientemente del orden de análisis de los objetos, puede obtener un agrupamiento diferente cada vez que se ejecuta sobre una misma colección.

2.1.9 Algoritmo Ant-cluster

Ant-Cluster [31] es un algoritmo dinámico basado en colonias de hormigas. Este algoritmo representa la colección de objetos $C = \{O_1, O_2, \ldots, O_n\}$ a través de un grafo dirigido y pesado $G = \langle V, E, w\rangle$ en el cual: a) $V = C$, b) existe una arista dirigida $\langle O_i$,$O_j\rangle \in E$ si el *índice de aceptación*[4] de O_i respecto a O_j es superior a cero y c) w es la función que asigna los pesos a las aristas; inicialmente, el peso de cada arista (su valor de feromona) es el índice de aceptación entre sus objetos.

Un conjunto de objetos $S = \{s_1, s_2, \ldots, s_k\}$, $S \subseteq V$, es una *componente fuertemente conexa* de G si: a) para todo par de elementos $s_i, s_j \in S$, existe un conjunto $P = \{p_1, p_2, \ldots, p_m\}$ tal que $P \subseteq S$, $s_i = p_1$, $s_j = p_m$ y existe una arista dirigida $\langle p_x$,$p_{x+1}\rangle$ en G para todo $x = 1..m-1$, b) al adicionar un objeto a S no se cumple la condición a).

Ant-Cluster actualiza el valor de feromona de cada arista a partir del recorrido por G de un número pre-establecido de "hormigas". Los vértices que cada hormiga visita en su recorrido por el grafo son determinados a través de un número aleatorio y una función que utiliza la feromona de los vértices. El proceso descrito anteriormente termina al alcanzar una cantidad máxima de iteraciones; adicionalmente en este proceso, cada cierta cantidad de iteraciones, se eliminan de G las aristas cuyo valor de feromona no sobrepasen un umbral pre-establecido. El agrupamiento final está determinado por las componentes fuertemente conexas de G.

Al adicionar un objeto a C, éste se inserta en el grupo más cercano. Cuando se elimina un objeto de la colección, éste se elimina del grupo al que pertenece conjuntamente con las aristas relacionadas. Cuando el número de cambios que ocurren en la colección superan un umbral pre-establecido ésta es re-agrupada desde cero.

Este algoritmo obtiene un conjunto de grupos disjuntos y tiene como limitaciones que: (a) independientemente del orden de análisis de los objetos, puede obtener un agrupamiento diferente cada vez que se ejecuta sobre una misma colección, (b) necesita optimizar varios parámetros cuyos valores son dependientes de la colección a procesar, (c) no permite el procesamiento de adiciones o eliminaciones múltiples de objetos y (d) la estrategia de actualización de los grupos no garantiza que cada grupo sea una componente fuertemente conexa de G.

2.1.10 Algoritmo XCLS

XCLS [32] es un algoritmo incremental diseñado para el agrupamiento de documentos XML. Este algoritmo introduce una medida de semejanza llamada *Level Similarity* (*LevelSim*) [33] para definir la semejanza entre dos documentos XML.

Inicialmente se considera que el primer documento de la colección forma un grupo. Posteriormente, cada vez que se adiciona un documento a la colección, XCLS actualiza el conjunto de grupos a través de un proceso compuesto de dos fases. En la primera fase el documento adicionado es agrupado siguiendo una estrategia de agrupamiento similar a la del algoritmo Single-Pass [11]. En la segunda fase, cada documento

[4] El índice de aceptación un objeto respecto a otro se calcula a partir de la semejanza entre éstos

de la colección es procesado en orden aleatorio y re-ubicado en el grupo con el cual alcance un valor de LevelSim que sea máximo (respecto al alcanzado con el resto de los grupos) y a la vez superior a un umbral pre-establecido.

Este algoritmo obtiene un conjunto de grupos disjuntos y tiene entre sus limitaciones que: (a) puede formar grupos con encadenamiento, (b) no permite el procesamiento de adiciones múltiples de objetos y (c) independientemente del orden de análisis de los objetos, puede no obtener siempre el mismo agrupamiento cada vez que se ejecuta con una colección.

2.1.11 Algoritmo DB-ColC

DB-ColC [34] es un algoritmo dinámico que se basa en el algoritmo de agrupamiento *b-coloring* (B-ColC) [35]. El algoritmo B-ColC representa la colección de objetos por un grafo de β*-dissimilaridad*; este grafo se define de forma similar al grafo de β*-semejanza* pero utilizando funciones de disimilaridad.

La estrategia de agrupamiento de B-ColC se basa en el *b-coloreado* del grafo de β*-dissimilaridad*. Un b-coloreado de un grafo consiste en asignar colores a los vértices de modo que se cumpla que: (i) dos vértices adyacentes tienen diferentes colores y (ii) cada color tiene al menos un vértice *dominante*; i.e., un vértice que es adyacente a cada uno de los otros colores. En este contexto, cada conjunto de vértices de un mismo color es interpretado como un grupo.

B-ColC es ejecutado sobre una misma colección varias veces, cada una con un valor diferente de β; el agrupamiento final es aquél que optimice el índice de Dunn [36]. Partiendo de este agrupamiento óptimo, el algoritmo DB-ColC ejecuta una estrategia de actualización para mantener el b-coloreado del grafo de dissimilaridad cada vez que se adiciona o elimina un objeto. Como resultado de esta actualización pueden surgir nuevos grupos o eliminarse grupos existentes.

El algoritmo DB-ColC obtiene un conjunto de grupos disjuntos y tiene como limitaciones que no permite procesar eliminaciones o adiciones múltiples y que puede resultar costoso para colecciones con un gran número de objetos debido a la estrategia de mantenimiento del b-coloreado del grafo que implementa.

2.1.12 Algoritmo SHC

SHC [37] es un algoritmo incremental que basa su funcionamiento en el *Cluster Similarity Histogram* (CSHr). El CSHr de un grupo es una representación, concisa y estadística, de las semejanzas existentes entre los objetos pertenecientes al grupo.

Este algoritmo utiliza además el concepto de Histogram Ratio (HR). El HR de un grupo C_i se define como:

$$HR(C_i) = \frac{CS_\beta}{CS},$$

donde CS es el número de pares de objetos del grupo C_i y CS_β es el número de pares de objetos del grupo C_i cuyo valor de semejanza supera un umbral β.

Cada vez que se adiciona un objeto a la colección, la estrategia de agrupamiento de SHC simula, para cada grupo existente, la adición del objeto al grupo y calcula el HR del grupo antes (HR_{old}) y después de adicionar el objeto (HR_{new}). Luego, si se cumple que: i) $HR_{new} \geq HR_{old}$ o ii) HR_{old} - $HR_{new} < \epsilon$ y HR_{new} es mayor que un umbral HR_{min}, entonces el objeto es adicionado al grupo; en otro caso el objeto forma un grupo nuevo. Adicionalmente, cada cierto tiempo SHC ejecuta un proceso de re-asignación de objetos a los grupos.

El algoritmo SHC construye un conjunto de grupos solapados y tiene las siguientes limitaciones: (a) es dependiente del orden de análisis de los objetos, (b) necesita optimizar varios parámetros cuyos valores

son dependientes de la colección a procesar, (c) no permite el procesamiento de adiciones múltiples de objetos y (d) el proceso de re-asignación ejecutado puede ser costoso dependiendo del número de objetos re-asignados.

2.1.13 Algoritmo INDC-NS

INDC-NS [38] es un algoritmo incremental diseñado para el agrupamiento de noticias que utiliza *búsquedas en vecindades* [39].

Sea $C = \{O_1, O_2, \ldots, O_n\}$ una colección de objetos, $\epsilon \in [0, 1]$ un umbral de semejanza, y S una función de semejanza simétrica entre los objetos de C. Dos objetos O_i,$O_j \in C$ son semejantes si $S(O_i, O_j) \geq \epsilon$. Consecuentemente, la ϵ-*vecindad* de un objeto $O_i \in C$ se define como el conjunto de todos los objetos similares a O_i.

La estrategia de agrupamiento de INDC-NS supone que inicialmente existe un sólo objeto y que éste forma un grupo. Posteriormente, cada vez que se adiciona un objeto O' a la colección se ejecuta un proceso formado de 3 etapas.

En la primera etapa se utiliza un *índice invertido* [40] para buscar eficientemente los objetos pertenecientes a la ϵ-*vecindad* de O'. En la segunda etapa se determina, dentro del conjunto de grupos a los que pertenecen los objetos de la ϵ-*vecindad* de O', cuál grupo es el más indicado para contener a O'; el grupo seleccionado es aquel cuya semejanza con O' sea máxima y además sobrepase un umbral pre--establecido. Por último, se realiza un proceso de re-agrupamiento en el cual se determinan cuáles de los grupos analizados en la segunda etapa pueden unirse al grupo que contiene a O'.

INDC-NS es un algoritmo que obtiene un conjunto de grupos disjuntos y que tiene como limitaciones que: (a) independientemente del orden de análisis de los objetos, puede obtener un agrupamiento diferente cada vez que se ejecuta sobre una misma colección, (b) la estrategia de re-agrupamiento puede formar grupos con encadenamiento, (c) no permite el procesamiento de adiciones múltiples de objetos y (d) necesita optimizar varios parámetros cuyos valores son dependientes de la colección a procesar.

2.2 Algoritmos jerárquicos

Los algoritmos jerárquicos se clasifican en *divisivos* o *aglomerativos*. Los algoritmos divisivos son aquellos que, partiendo de un grupo que contiene a todos los objetos, construyen una jerarquía dividiendo recursivamente cada nodo (usualmente en dos nodos) hasta que cada nodo tenga sólo un objeto. Los algoritmos aglomerativos, por otra parte, son aquellos que parten considerando a cada objeto como un grupo y construyen una jerarquía uniendo recursivamente el par de grupos más semejantes hasta formar un grupo que contiene a todos los objetos. A continuación se describen los algoritmos jerárquicos, incrementales y dinámicos reportados en la literatura.

2.2.1 Algoritmo DC-Tree

DC-Tree [41] es un algoritmo dinámico que está basado en árboles, creado para el agrupamiento de páginas web. La estrategia de agrupamiento de este algoritmo construye una jerarquía a través de la inserción de los objetos en una estructura de datos llamada DC-Tree (*Document Cluster Tree*); cada nodo de este árbol es un grupo de documentos. Esta estructura de datos constituye una representación compacta de la colección de documentos agrupados, en la cual: (a) existe un límite inferior (M) y superior (B) para la cantidad de hijos que puede tener un nodo y (b) un nodo puede tener como hijo a otro nodo del árbol, a un sólo documento e incluso a un grupo de documentos.

Cada vez que se adiciona un nuevo documento d a la colección el árbol se recorre desde la raíz, seleccionando en cada nivel el nodo hijo más semejante a d y cuya semejanza sobrepase un umbral $S1$; en caso de que dicho nodo no exista, d es insertado en el nodo actual. Si se alcanza un *nodo hoja* del árbol entonces d es adicionado como hijo en dicho nodo. Si al insertar d en un nodo A, la cantidad de hijos de A sobrepasa el umbral B permitido entonces se divide el nodo A en dos nodos. Para realizar esta division se seleccionan entre los hijos de A a aquellos dos que tengan la menor semejanza, cada uno de estos nodos es utilizado como *semilla* para la formación de dos nuevos grupos A_1 y A_2. A continuación, cada uno de los restantes hijos de A es colocado en el grupo que tenga la semilla más semejante al nodo. El proceso anteriormente mencionado, para el ajuste del número de hijos de un nodo, puede propagarse hasta la raíz del árbol.

Cuando se elimina un documento del árbol, éste se elimina del nodo en el que se encuentra almacenado. Si luego de la eliminación, dicho nodo tiene una cantidad de hijos menor que M entonces, de forma similar a como se hace en el B$^+$-Tree [42], se une dicho nodo con algún otro nodo hermano y se reduce en uno la cantidad de hijos de su nodo padre; el proceso anterior puede propagarse hasta la raíz del árbol.

El algoritmo DC-Tree construye una jerarquía de grupos disjuntos y tiene entre sus limitaciones que: (a) realiza asignación irrevocable de los documentos a los grupos, (b) puede formar grupos con encadenamiento, (c) es dependiente del orden de análisis de los objetos, (d) necesita optimizar varios parámetros cuyos valores son dependientes de la colección a procesar y (e) no permite el procesamiento de adiciones o eliminaciones múltiples de objetos.

2.2.2 Algoritmo IHC

IHC [43] es un algoritmo incremental que construye una jerarquía de grupos disjuntos, siguiendo una estrategia de agrupamiento aglomerativa y utilizando las propiedades de *Homogeneidad* y *Monotonía*.

Una jerarquía satisface la propiedad de Monotonía si cumple que la *densidad* de todo nodo es mayor que la de su nodo padre. Sea $N = \{e_1, e_2, \ldots, e_k\}$ un nodo de la jerarquía, la densidad de N es el promedio de las distancias de cada elemento $e_i \in N, i = 1..k$ a su vecino más cercano.

Sea $N = \{e_1, e_2, \ldots, e_k\}$ un nodo de la jerarquía y $D = \{d_1, d_2, \ldots, d_k\}$ el conjunto de distancias de cada elemento $e_i \in N, i = 1..k$ a su vecino más cercano. El nodo N satisface la propiedad de Homogeneidad si cumple que cada distancia d_i pertenece al intervalo [α,β], siendo α y β umbrales establecidos en función del promedio y la desviación estándar de las distancias de D; una jerarquía satisface la propiedad de Homogeneidad si cada uno de sus nodos la satisface.

Cada vez que se adiciona un objeto O' a la colección se determina cuál es su nodo hoja más cercano (H'). Posteriormente, se realiza un proceso de búsqueda, a partir del nodo padre de H' hasta el nivel superior de la jerarquía para encontrar qué nodo puede almacenar a O'. El nodo seleccionado para almacenar a O' sera aquél que: (a) tenga la menor variación en su densidad y (b) tenga la menor cantidad de elementos cuyas distancias a su vecino más cercano estén fuera del intervalo [α,β]. Una vez insertado el objeto en un nodo, se realiza un proceso de re-estructuración de la jerarquía para mantener las propiedades de Monotonía y Homogeneidad.

Entre las limitaciones de este algoritmo está que: (a) es dependiente del orden de análisis de los objetos, (b) independientemente del orden de análisis de los objetos, puede obtener un agrupamiento diferente cada vez que se ejecuta sobre una misma colección, (c) necesita optimizar varios parámetros cuyos valores son dependientes de la colección a procesar y (d) no permite el procesamiento de adiciones múltiples de objetos.

2.2.3 Algoritmo IHDC-NS

IHDC-NS [38] es un algoritmo jerárquico incremental aglomerativo que utiliza el algoritmo INDC-NS [38] y que fue diseñado para el agrupamiento jerárquico de noticias.

IHDC-NS construye una jerarquía de grupos disjuntos que consta sólo de tres niveles. El nivel 1 es aquél en el cual cada objeto de la colección forma un grupo y el nivel 2 está formado por los grupos que se obtienen al aplicar el algoritmo INDC-NS sobre los elementos del nivel 1. Para la construcción del nivel 3, se representa cada grupo del nivel 2 a través del vector centroide del grupo y se ejecuta entonces el algoritmo INDC-NS sobre dichos centroides.

Cada vez que se adiciona un objeto a la colección éste se adiciona al nivel 1 y posteriormente se ejecuta INDC-NS para actualizar los grupos del nivel 2. Cuando transcurre un tiempo pre-establecido y la colección no cambia, entonces se actualiza el centroide de cada grupo modificado hasta el momento y se re-construye desde cero el nivel 3 aplicando el algoritmo INDC-NS sobre los centroides de los grupos.

Entre las limitaciones de IHDC-NS se encuentran las del algoritmo INDC-NS y además que: (a) la estrategia de actualización de la jerarquía es muy costosa al reconstruir el nivel 3 de la misma siempre desde cero y (b) el nivel 3 de la jerarquía no se actualiza cada vez que se modifica la colección.

2.2.4 Algoritmo IHCA

IHCA [6] es un algoritmo incremental, jerárquico y aglomerativo, que fue diseñado para el descubrimiento de tópicos en colecciones de noticias. Este algoritmo construye una jerarquía de grupos aplicando el algoritmo CI [22] en cada nivel de la misma.

El nivel base de la jerarquía que construye IHCA, como en la mayoría de los algoritmos aglomerativos, es aquél en el que cada objeto de la colección forma un grupo. A partir del nivel base, cada nivel se construye aplicando el algoritmo CI sobre el conjunto de los *representantes* de los grupos del nivel inmediato inferior. El nivel superior de la jerarquía es aquél en el cual el grafo de máxima β-semejanza $G_{\beta max} = \langle V, E_{\beta max} \rangle$ cumple que $|E_{\beta max}| = 0$; es decir, todos los vértices están aislados. En este trabajo, el *representante* de un grupo es la unión de los documentos contenidos en dicho grupo y se obtiene sumando los vectores de términos de dichos documentos.

Cada vez que se adiciona un objeto a la colección éste es adicionado como un grupo al nivel base y se realiza un proceso para actualizar la jerarquía en cada uno de los niveles superiores. Como primer paso, se aplica el algoritmo CI en el nivel 2 de la jerarquía. Al construir este agrupamiento puede ocurrir que: (a) se creen grupos nuevos; luego se debe adicionar un representante al nivel inmediato superior y (b) se eliminen grupos existentes; luego se debe eliminar su correspondiente representante del nivel inmediato superior. Cada cambio que ocurre en un nivel provoca el re-procesamiento del nivel inmediato superior y por tanto la ejecución del algoritmo CI para actualizar el conjunto de grupos de dicho nivel. El proceso de actualización de la jerarquía termina cuando se alcanza el nivel superior de ésta.

IHCA obtiene una jerarquía de grupos disjuntos y el mismo tiene como limitaciones que: (a) en cada nivel se pueden obtener muchos grupos, cada uno formado por pocos objetos, (b) los grupos construidos pueden tener encadenamiento aunque en menor magnitud que los grupos generados por los algoritmos GLC o GLC+ y (c) no permite el procesamiento de adiciones múltiples de objetos.

2.2.5 Algoritmos DHCA, DHS y sp-DHCA

DHCA [45] y DHS [46] son algoritmos dinámicos y jerárquicos, derivados de una metodología propuesta por Gil-García en el 2005 para la creación de algoritmos jerárquicos estáticos y dinámicos [45]. En esta

metodología se obtiene una jerarquía de grupos siguiendo una estrategia de agrupamiento aglomerativa. Esta estrategia aglomerativa construye un conjunto de grupos en cada nivel mediante la aplicación de un algoritmo de agrupamiento basado en grafos sobre los objetos del nivel.

El proceso de creación de la jerarquía considera que el nivel base de la misma es aquél en el que cada objeto de la colección forma un grupo. A continuación, para construir cada nivel siguiente, se consideran como objetos del mismo a los grupos del nivel anterior y se forma el grafo de β-semejanza $G_\beta = \langle V, E_\beta \rangle$ asociado dichos objetos. Si G_β cumple que $|E_\beta| > 0$ entonces se ejecuta un algoritmo de agrupamiento basado en grafos sobre un sub-grafo de G_β; en otro caso, si $|E_\beta| = 0$ entonces se detiene el proceso.

Cuando se adiciona o se elimina un elemento de la colección, éste se adiciona o se elimina del nivel base. Los cambios realizados en el nivel base provocan que se actualice el agrupamiento del segundo nivel. En este momento, debe de ejecutarse, para actualizar el conjunto de grupos de dicho nivel, el algoritmo de agrupamiento basado en grafos utilizado. Al ejecutar el proceso anterior, de forma similar a como sucede con el algoritmo IHCA [6], pueden eliminarse y crearse grupos por lo que habrá que eliminar y adicionar elementos en el nivel inmediato superior y posteriormente actualizar el agrupamiento en dicho nivel. El proceso de actualización recorre cada uno de los niveles hasta alcanzar un nivel N' en el cual el grafo $G_\beta = \langle V, E_\beta \rangle$ asociado cumpla que $|E_\beta| = 0$.

En la construcción del grafo de β-semejanza de cada nivel, ambos algoritmos utilizan la medida *group--average* [49] para determinar la semejanza entre dos grupos. Esta medida define la semejanza entre dos grupos G_1 y G_2 como el promedio de las semejanzas entre todos los pares de objetos (g_1, g_2), donde $g_1 \in G_1$ y $g_2 \in G_2$. Por otra parte, para formar los grupos de cada nivel, DHCA utiliza el algoritmo CI [22] aplicado sobre el grafo de máxima β-semejanza $G_{\beta max} = \langle V, E_{\beta max} \rangle$ y DHS utiliza una versión del algoritmo Star [25] aplicado sobre el grafo no dirigido asociado a $G_{\beta max}$.

El algoritmo sp-DHCA [47] es una modificación del algoritmo DHCA que reduce el número de comparaciones a realizar para encontrar en un nivel de la jerarquía los objetos $\beta-$semejantes de un objeto dado; para lograr tal propósito se hace uso de la jerarquía ya formada.

Al adicionar un objeto en algún nivel n_i, no se calcula la semejanza de dicho objeto con el resto sino que se realiza una búsqueda *top-down* desde el nivel superior hasta n_i para encontrar un conjunto aproximado de objetos semejantes al adicionado. En esta búsqueda sólo se recorren las ramas del árbol cuyos grupos tengan con el objeto adicionado una semejanza mayor o igual que un umbral pre-establecido μ. El conjunto de objetos β-semejantes determinado, para cada objeto en cada nivel, es aproximado; i.e., el conjunto calculado puede no contener todos los objetos β-semejantes del objeto adicionado.

Los algoritmos DHCA y sp-DHCA construyen una jerarquía de grupos disjuntos mientras que el algoritmo DHS construye una jerarquía que puede ser solapada. Entre las limitaciones de estos algoritmos está que pueden ser poco eficientes cuando se aplican en colecciones con una gran cantidad de objetos. Otra limitación es que estos algoritmos obtienen muchos grupos, cada uno generalmente formado por pocos elementos. Adicionalmente, los algoritmos DHCA y sp-DHCA pueden obtener grupos con encadenamiento y el algoritmo sp-DHCA es dependiente del orden de análisis de los objetos.

2.2.6 Algoritmo SIHC

SIHC [48] es un algoritmo jerárquico incremental aglomerativo que sigue una estrategia de agrupamiento similar a la de los algoritmos aglomerativos clásicos: SLINK, CLINK y ALINK [49]. Aunque SIHC puede utilizarse por sí sólo para agrupar incrementalmente una colección de objetos, fue diseñado para actualizar la jerarquía construida por SLINK, CLINK o ALINK.

Cada vez que se adiciona un nuevo objeto O a la colección, se realiza un proceso top-down con el objetivo

de colocar a O dentro de la misma. El proceso anteriormente mencionado, parte de determinar la distancia entre O y el nodo raíz; luego, si esta distancia es mayor que la existente entre los nodos hijos del nodo raíz, se crea un nuevo nodo raíz que tendrá como hijos al anterior nodo raíz y a O. En otro caso, si la distancia es menor se repite el proceso anterior sobre el nodo hijo más cercano a O y se actualiza la distancia entre los hijos de dicho nodo. La forma de actualizar la distancia depende de la función de semejanza que se esté utilizando; i.e., la función del SLINK, la del CLINK o la del ALINK.

SIHC construye una jerarquía de grupos disjuntos y al utilizar una estrategia de agrupamiento como la del SLINK, CLINK o ALINK tiene como limitaciones que: (a) es dependiente del orden, (b) sólo se permite unir dos grupos en cada nivel; luego, no se permite que los niveles de la jerarquía representen diferentes grados de abstracción de la colección de objetos, (c) los grupos pueden tener encadenamiento y (d) no se permite el procesamiento de adiciones múltiples de objetos.

2.3 Recapitulación

Como se pudo apreciar en 2.1 y 2.2, se ha invertido un importante esfuerzo en el desarrollo de algoritmos de agrupamiento, jerárquicos y no jerárquicos, tanto del tipo incremental como del tipo dinámico. No obstante el trabajo realizado, sólo 5 de los algoritmos reportados permiten obtener grupos que pueden ser solapados, de ellos sólo uno es jerárquico y tiene las siguientes limitaciones: (a) puede ser poco eficiente cuando se aplica en colecciones con una gran cantidad de objetos y (b) obtiene muchos grupos, cada uno generalmente formado por pocos elementos. En la figura 2 se muestra una taxonomía de los algoritmos descritos anteriormente; los algoritmos que permiten obtener grupos con traslapes fueron resaltados en negro.

De forma general, los algoritmos reportados presentan un conjunto de limitaciones que pueden reducir la utilidad de éstos o afectar la calidad de los grupos obtenidos. Las principales limitaciones detectadas son: (a) asignación irrevocable de objetos a grupos, (b) obtención de grupos con encadenamiento, (c) dependencia del orden de análisis de los objetos, (d) imposición de restricciones a los objetos a agrupar, (e) obtienen muchos grupos, generalmente con pocos objetos, (f) necesitan optimizar varios parámetros cuyos valores son dependientes de la colección a procesar, (g) independientemente del orden de análisis de los objetos, pueden obtener un agrupamiento diferente cada vez que se ejecutan sobre una misma colección, (h) son poco eficientes procesando objetos descritos por un gran número de rasgos y/o colecciones de grandes volúmenes de objetos, (i) no actualizan el agrupamiento cada vez que se adicionan objetos a la colección y (j) no permiten procesar adiciones o eliminaciones múltiples.

3 Motivación

Hoy en día, la mayoría de las aplicaciones que utilizan técnicas de minería de datos trabajan con colecciones que varían con el tiempo [50]; luego, es fundamental el desarrollo de algoritmos dinámicos que sean capaces de lidiar con estas colecciones.

Algunas de las aplicaciones en las que el agrupamiento dinámico ha sido utilizado incluyen por ejemplo: la meteorología (en la detección y seguimiento de ciclones), en el análisis del tráfico en las ciudades, en el estudio de las migraciones de animales o incluso, en estudios médicos en los cuales se desea identificar los cambios que puedan disparar problemas mentales o de comportamiento. Como se aprecia, estas aplicaciones son muy útiles para el desarrollo social y/o permiten ampliar el conocimiento en otras ramas de la ciencia; luego, es necesario el desarrollo de algoritmos de agrupamiento dinámicos que puedan en todo momento reflejar el estado actual de los datos [51, 52].

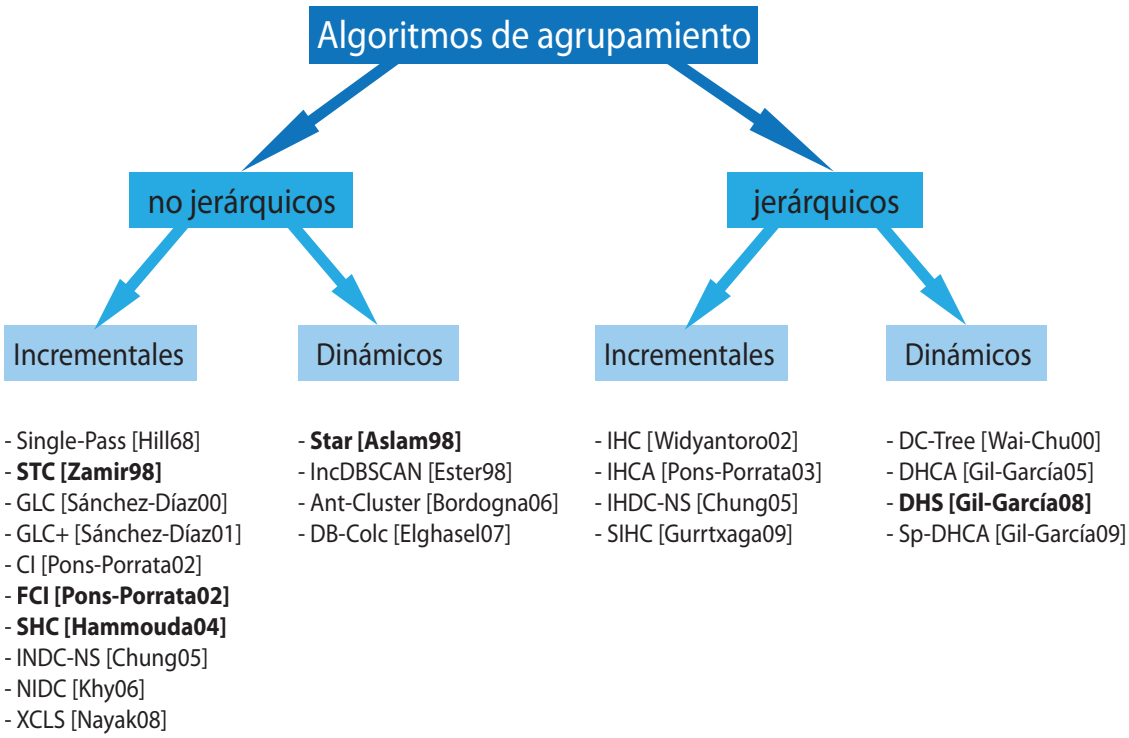


Figure 2. Taxonomía de los algoritmos de agrupamiento incrementales y dinámicos reportados

Existen otras aplicaciones en las cuales obtener sólo un conjunto de grupos no es suficiente, sino que se necesita o resulta de mucha utilidad establecer relaciones entre dichos grupos. Considérese por ejemplo un sitio web en el cual se desea brindar sugerencias a sus usuarios teniendo en cuenta las preferencias de éstos [53]. Si se supone que las preferencias de los usuarios se pueden determinar a partir de las páginas que el mismo visita cada vez que entra al sistema, entonces se puede pensar en agrupar esta información y brindar aquellas páginas relacionadas con cada grupo. El problema de este tipo de agrupamiento es que no permite reflejar las sub-categorías o sub-grupos específicos dentro de los grupos. Lo anterior dificulta que los usuarios puedan, dentro de un grupo, seleccionar con diferentes grados de abstracción la información que necesitan. La solución a este problema se puede obtener a través de los algoritmos de agrupamiento jerárquicos [54].

Otras aplicaciones, de los algoritmos jerárquicos, se encuentran en sistemas de filtrado de información, en aplicaciones desarrolladas para la detección de usuarios maliciosos y en la construcción de ontologías. Es importante notar que todas estas aplicaciones pudiesen trabajar también con colecciones que varíen; luego, sería deseable que los algoritmos jerárquicos que utilicen dichas aplicaciones sean a la vez dinámicos.

Haciendo una revision en la sección 2 se puede notar que muchos de los algoritmos propuestos en la literatura son *basados en grafos*. Los algoritmos basados en grafos constituyen una clase muy importante dentro de los algoritmos de agrupamiento. Estos algoritmos representan la colección de objetos a través de un grafo en el cual, los vértices son los objetos de la colección y las aristas representan relaciones entre dichos objetos.

Una característica importante de los algoritmos basados en grafos es que no imponen restricciones al es-

pacio de representación de los objetos de la colección, así como tampoco restringen la medida de semejanza utilizada para la formación de las aristas; estas características aumentan el campo de aplicación de dichos algoritmos. Adicionalmente, en los últimos años se han reportado varios algoritmos basados en grafos que han mostrado buenos resultados en la obtención de grupos solapados [27, 56, 55].

Siguiendo un análisis similar al anterior, se puede notar que la mayoría de los algoritmos jerárquicos incrementales o dinámicos reportados son del tipo aglomerativo. Los algoritmos aglomerativos son, dentro de los algoritmos jerárquicos, los que han sido más estudiados y desarrollados [57]. Adicionalmente, existen trabajos que han mostrado que los algoritmos jerárquicos aglomerativos generalmente obtienen mejores resultados que los divisivos [45, 58].

Adicionalmente, consideramos que existen algunas limitaciones de los algoritmos reportados, que pueden eliminarse con el objetivo de aumentar la utilidad práctica de estos algoritmos de agrupamiento. Las limitaciones que se propone resolver en esta investigación son: (a) la asignación irrevocable de objetos a grupos, (b) la obtención de grupos con encadenamiento, (c) la imposición de restricciones a los objetos a agrupar, (d) la obtención de muchos grupos, generalmente con pocos objetos, (e) la no actualización del agrupamiento cada vez que se adicionan objetos a la colección y (f) el no procesamiento de adiciones o eliminaciones múltiples.

Con base en lo anteriormente expuesto, consideramos importante continuar investigando en el desarrollo de algoritmos dinámicos, jerárquicos y no jerárquicos. Los algoritmos de agrupamiento que se propondrán en esta investigación serán basados en grafos; adicionalmente, los algoritmos jerárquicos serán aglomerativos.

4 Propuesta

En esta sección se presentan las preguntas de investigación, los objetivos generales y específicos, la metodología a seguir, las contribuciones esperadas y el cronograma de actividades

4.1 Preguntas de investigación

Teniendo en cuenta los problemas detectados en la sección anterior, surgen las siguientes preguntas de investigación:

a) ¿Es posible desarrollar un algoritmo de agrupamiento dinámico basado en grafos que permita obtener un conjunto de grupos que pueden ser solapados y que alcance mejores resultados de eficacia que los algoritmos reportados?

b) ¿Es posible determinar una representación que sea útil o efectiva para los grupos de los niveles de una jerarquía de grupos?

c) ¿Es posible desarrollar un algoritmo jerárquico aglomerativo, que sea dinámico, que permita obtener una jerarquía que puede ser traslapada y que obtenga mejores resultados de eficacia que los algoritmos reportados?

4.2 Objetivo general

El objetivo general de esta tesis de investigación doctoral consiste en desarrollar un algoritmo de agrupamiento jerárquico aglomerativo, que sea dinámico y que permita además construir una jerarquía de grupos que puede ser traslapada.

El algoritmo desarrollado debe de eliminar las limitaciones planteadas en la sección de motivación. Adicionalmente, el algoritmo propuesto debe alcanzar, respecto a los algoritmos reportados en la literatura, un rendimiento superior en cuanto a medidas de eficacia y un rendimiento similar o superior respecto a la eficiencia.

4.3 Objetivos particulares

1. Diseñar e implementar un algoritmo de agrupamiento incremental que permita la obtención de grupos con traslape.

2. Diseñar e implementar un algoritmo de agrupamiento dinámico que permita la obtención de grupos con traslape.

3. Diseñar e implementar un algoritmo de agrupamiento jerárquico aglomerativo que sea incremental, utilice el algoritmo desarrollado en el objetivo particular 1 y que construya una jerarquía de grupos que puede ser traslapada.

4. Diseñar e implementar un algoritmo de agrupamiento jerárquico aglomerativo, que sea dinámico, utilice el algoritmo desarrollado en el objetivo particular 2 y que construya una jerarquía de grupos que puede ser traslapada.

4.4 Metodología propuesta

Para alcanzar los objetivos específicos propuestos se definió la siguiente metodología:

1. Recopilar colecciones reportadas en la literatura en las cuales exista traslape entre las clases etiquetadas manualmente. Consideraremos inicialmente las colecciones TDT2, TREC-5 y *Reuters*-21578.

2. Seleccionar medidas de evaluación de algoritmos de agrupamiento, que estén reportadas en la literatura y permitan evaluar la calidad de los algoritmos de agrupamiento que obtienen grupos con traslapes. Consideraremos inicialmente las medidas Jaccard-index [59] y Fmeasure [60].

3. Desarrollar un algoritmo de agrupamiento incremental.

   (a) Desarrollar algoritmo de agrupamiento estático basado en grafos que permita obtener grupos con traslapes.

      i) Utilizar para representar la colección de objetos el grafo de β-semejanza G_β. Este grafo es simple de construir y además en él, la adición o eliminación de un vértice sólo provoca adiciones o eliminaciones de las aristas relativas a dicho vértice.

      ii) Utilizar para obtener un cubrimiento de G_β los *sub-grafos de tipo estrella* [25]. Este tipo de sub-grafo permite obtener grupos con traslape en los cuales el encadenamiento es reducido.

      iii) Definir una propiedad sobre los vértices de G_β que permita filtrar el número de vértices candidatos a centro y que además permita definir un orden sobre los elementos de dicho conjunto. Dado que interesa obtener grupos densos la propiedad definida debe de utilizar de cierta forma el grado de los vértices.

iv) Diseñar una estrategia de agrupamiento que utilice la propiedad definida en iii) para seleccionar un conjunto de vértices centros que cubran completamente a G_β. La estrategia diseñada debe impedir la selección de vértices que *a priori* forman grupos en los que todos los vértices pertenecen al menos a otro grupo.

v) Definir un criterio que permita filtrar el conjunto de centros seleccionados de forma que el conjunto resultante cumpla que contiene a los centros más densos del conjunto anterior que todavía pueden cubrir completamente a G_β. Este criterio debe de utilizar el grado de los vértices centro.

(b) Determinar condiciones que permitan actualizar el agrupamiento, formado por el algoritmo desarrollado en el punto 3.a, cuando se adiciona uno o más objetos a la colección. Analizar inicialmente:

i) ¿Cómo varía el valor de la propiedad definida en el punto 3.a.iii para cada uno de los vértices de G?

ii) ¿Qué grupos formados son los que necesitan ser actualizados?

iii) ¿Qué condiciones debe cumplir un vértice para ser considerado como candidato a centro?

(c) Diseñar e implementar, utilizando las condiciones determinadas en el paso 3.b, un algoritmo incremental.

(d) Comparar el desempeño del algoritmo desarrollado, utilizando las colecciones recopiladas en el paso 1, respecto a los algoritmos reportados en la literatura. Estas comparaciones deben de considerar:

i) La calidad de los grupos respecto a las medidas seleccionadas en el paso 2.

ii) La eficiencia de los algoritmos al agrupar una colección de forma incremental.

(e) Analizar los resultados experimentales y determinar limitaciones que puedan afectar el funcionamiento del algoritmo. En caso de existir dichas limitaciones, modificar el algoritmo para solucionarlas.

(f) Analizar la complejidad computacional del algoritmo desarrollado.

4. Desarrollar un algoritmo de agrupamiento dinámico.

(a) Determinar condiciones que permitan actualizar el agrupamiento, formado por el algoritmo desarrollado en el punto 3.a, cuando se elimina uno o más objetos de la colección. Analizar inicialmente:

i) ¿Cómo varía el valor de la propiedad definida en el punto 3.a.iii para cada uno de los vértices de G?

ii) ¿Qué grupos formados son los que necesitan ser actualizados?

iii) ¿Qué condiciones debe cumplir un vértice para ser considerado como candidato a centro?

iv) ¿Cuáles son los vértices que pueden quedar no cubiertos?

(b) Diseñar e implementar, utilizando las condiciones determinadas en el paso 3.b y 4.a, un algoritmo dinámico.

(c) Comparar el desempeño del algoritmo desarrollado, utilizando las colecciones recopiladas en el paso 1, respecto a los algoritmos reportados en la literatura. Estas comparaciones deben de considerar:

i) La eficiencia de los algoritmos al actualizar el agrupamiento cuando se eliminan uno o varios objetos de la colección.

ii) La eficiencia de los algoritmos al actualizar el agrupamiento cuando se modifican uno o varios objetos de la colección.

(d) Analizar los resultados experimentales y determinar limitaciones que puedan afectar el funcionamiento del algoritmo. En caso de existir dichas limitaciones, modificar el algoritmo para solucionarlas.

(e) Analizar la complejidad computacional del algoritmo desarrollado.

5. Desarrollar un algoritmo de agrupamiento jerárquico aglomerativo incremental.

(a) Diseñar un algoritmo jerárquico aglomerativo estático que:

i) Utilice para agrupar los objetos de cada nivel el algoritmo estático desarrollado en el paso 3.a.

ii) Considere que los objetos de todo nivel $N_i > 1$ son los grupos del nivel anterior.

iii) Detenga la construcción de la jerarquía cuando el grafo de β-semejanza, asociado a la colección de objetos del nivel, no tenga aristas.

iv) Pueda usar cualquier medida para el cálculo de la semejanza entre los objetos de los niveles $N_i > 1$. Consideraremos inicialmente la medida *group-average* que ha sido usada en varios algoritmos jerárquicos aglomerativos.

(b) Seleccionar un criterio para representar los objetos de los niveles $N_i > 1$.

i) Estudiar los criterios, alternativos al utilizado en el paso 5.a.ii, que estén reportados en la literatura. Consideraremos inicialmente para representar a un grupo: (I) al objeto que pertenece al grupo y que es el más cercano del centroide del mismo y (II) al vértice *centro* del sub-grafo que determina el grupo.

ii) Evaluar cada criterio sobre el algoritmo desarrollado en el paso 5.a.

iii) Análisis de los resultados y selección del criterio.

(c) Seleccionar una medida para determinar la semejanza entre los objetos de los niveles $N_i > 1$.

i) Estudiar las medidas, alternativas a la utilizada en el paso 5.a.iv, que estén reportadas en la literatura. Consideraremos inicialmente: (I) la semejanza entre los representantes de los grupos y (II) la semejanza entre un subconjunto de vértices de los sub-grafos que determinan a cada grupo.

ii) Evaluar cada medida sobre el algoritmo desarrollado en el paso 5.a.

iii) Análisis de los resultados y selección de la medida.

(d) Determinar condiciones que permitan actualizar los grupos de la jerarquía formada por el algoritmo desarrollado en el paso 5.a cuando se adicionan uno o más objetos a la colección. Analizar inicialmente:

i) ¿Cómo afecta la actualización de los grupos en un nivel a los objetos del nivel siguiente?

ii) Cuando en un nivel se eliminan, se modifican o se forman nuevos grupos, ¿Qué objetos del nivel siguiente necesitan ser eliminados y/o adicionados?

iii) ¿Cómo representar la relación entre los grupos de un nivel y los objetos del nivel siguiente de una forma eficiente que permita a la vez reflejar rápidamente los cambios que puedan ocurrir en un nivel?

(e) Diseñar un algoritmo jerárquico aglomerativo incremental que:

i) Utilice para agrupar los objetos de cada nivel el algoritmo dinámico desarrollado en el paso 4.

ii) Utilice las condiciones determinadas en el paso 5.d.

ii) Represente los objetos de todo nivel $N_i > 1$ utilizando el criterio seleccionado en el paso 5.b.

iii) Detenga la construcción de la jerarquía cuando el grafo de β-semejanza, asociado a la colección de objetos del nivel, no tenga aristas.

iv) Utilice para medir la semejanza entre dos objetos, en todo nivel $N_i > 1$, la medida seleccionada en el paso 5.c.

(f) Comparar el desempeño del algoritmo desarrollado, utilizando las colecciones recopiladas en el paso 1, respecto a los algoritmos reportados en la literatura. Estas comparaciones deben de considerar:

i) La calidad de los grupos respecto a las medidas seleccionadas en el paso 2.

ii) La eficiencia de los algoritmos al agrupar una colección de forma incremental.

(g) Analizar los resultados experimentales y determinar limitaciones que puedan afectar el funcionamiento del algoritmo. En caso de existir dichas limitaciones, modificar el algoritmo para solucionarlas.

(h) Analizar la complejidad computacional del algoritmo desarrollado.

6. Desarrollar un algoritmo de agrupamiento jerárquico dinámico.

(a) Determinar condiciones que permitan actualizar los grupos de la jerarquía formada por el algoritmo desarrollado en el paso 5.a cuando se eliminan uno o más objetos de la colección. Analizar inicialmente:

i) ¿Cómo afecta la actualización de los grupos en un nivel a los objetos del nivel siguiente?

ii) Cuando en un nivel se eliminan, se modifican o se forman nuevos grupos, ¿Qué objetos del nivel siguiente necesitan ser eliminados y/o adicionados?

iii) ¿Cómo representar la relación entre los grupos de un nivel y los objetos del nivel siguiente de una forma eficiente que permita a la vez reflejar rápidamente los cambios que puedan ocurrir en un nivel?

(b) Diseñar un algoritmo jerárquico aglomerativo dinámico que:

i) Utilice para agrupar los objetos de cada nivel el algoritmo dinámico desarrollado en el paso 4.

ii) Utilice las condiciones determinadas en el paso 6.a.

ii) Represente los objetos de todo nivel $N_i > 1$ utilizando el criterio seleccionado en el paso 5.b.

iii) Detenga la construcción de la jerarquía cuando el grafo de β-semejanza, asociado a la colección de objetos del nivel, no tenga aristas.

iv) Utilice para medir la semejanza entre dos objetos, en todo nivel $N_i > 1$, la medida seleccionada en el paso 5.c.

(c) Comparar el desempeño del algoritmo desarrollado, utilizando las colecciones recopiladas en el paso 1, respecto a los algoritmos reportados en la literatura. Estas comparaciones deben de considerar:

i) La eficiencia de los algoritmos al actualizar el agrupamiento cuando se eliminan uno o varios objetos de la colección.

ii) La eficiencia de los algoritmos al actualizar el agrupamiento cuando se modifican uno o varios objetos de la colección.

(d) Analizar los resultados experimentales y determinar limitaciones que puedan afectar el funcionamiento del algoritmo. En caso de existir dichas limitaciones, modificar el algoritmo para solucionarlas.

(e) Analizar la complejidad computacional del algoritmo desarrollado.

4.5 Contribuciones esperadas

Los principales contribuciones esperadas al término de esta investigación doctoral son las siguientes:

1. Un algoritmo de agrupamiento incremental que permita construir grupos con traslape y que obtenga mejores resultados de eficacia que los algoritmos reportados.

2. Un algoritmo de agrupamiento dinámico que permita construir grupos con traslape y que obtenga mejores resultados de eficacia que los algoritmos reportados.

3. Un algoritmo de agrupamiento jerárquico aglomerativo incremental, que permita construir una jerarquía que puede ser solapada y que obtenga mejores resultados de eficacia que los algoritmos reportados.

4. Un algoritmo de agrupamiento jerárquico aglomerativo dinámico, que permita construir una jerarquía que puede ser solapada y que obtenga mejores resultados de eficacia que los algoritmos reportados.

4.6 Calendario de actividades

El calendario de actividades para los 8 trimestres (24 meses) de esta investigación doctoral se muestra en la figura 3.

5 Resultados preliminares

En esta sección se presentan los resultados preliminares que se han alcanzado durante el desarrollo de esta investigación. Primeramente se presenta un nuevo algoritmo incremental llamado ICSD [61], que permite la obtención de grupos que pueden ser traslapados. Finalmente, se presenta un conjunto de resultados experimentales que muestran el desempeño de ICSD en comparación con otros algoritmos reportados.

5.1 Agrupamiento basado en *strength*

ICSD [61] es un algoritmo incremental que permite agrupar una colección de objetos sin suponer un espacio de representación o medida de semejanza específicos para los objetos de dicha colección. Este algoritmo representa la colección de objetos por su grafo de β-semejanza $G_\beta = \langle V, E_\beta \rangle$ y obtiene un

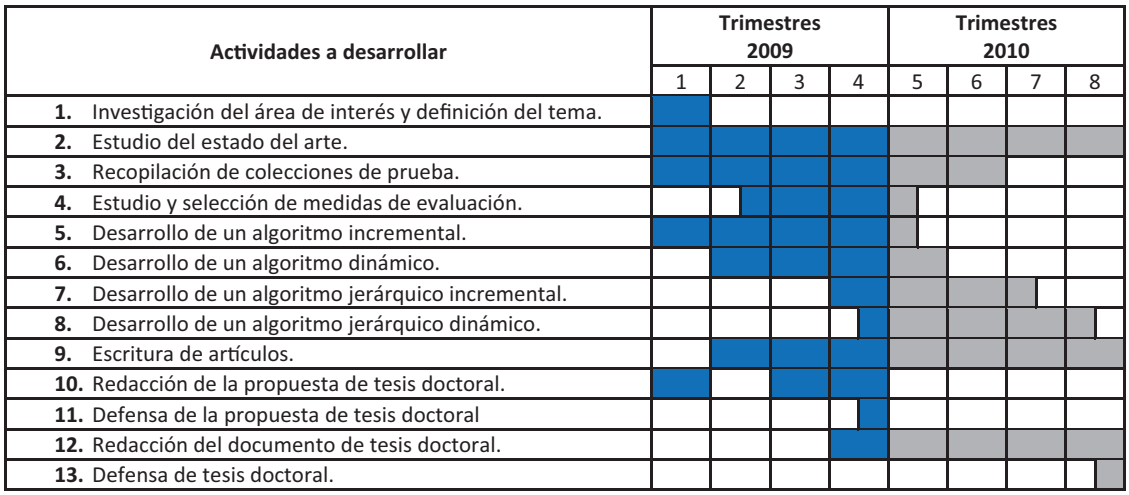


Figure 3. Calendario de actividades

conjunto de grupos solapados a partir del cubrimiento de G_β utilizando *sub-grafos en forma de estrella* [25].

Un *sub-grafo en forma de estrella* (FE) es un sub-grafo de $m+1$ vértices que tiene un vértice especial llamado *centro* y m vértices llamados *satélites*, en el cual existe una arista entre el centro y cada satélite. En este contexto, cada sub-grafo determina un grupo del agrupamiento final.

Dado que cada sub-grafo FE está determinado por su vértice centro, el problema de obtener un conjunto $S = \{Sg_1, Sg_2, \ldots, Sg_k\}$ de sub-grafos FE que cubra a G_β puede transformarse en el problema de determinar el conjunto de vértices $C = \{c_1, c_2, \ldots, c_k\}$ tal que $C \subseteq V$ y $\forall i = 1..k, c_i$ es el centro de Sg_i. Como cada vértice de G_β puede formar un sub-grafo FE, es necesario definir un criterio que permita reducir el número de vértices candidatos a centro y/o establecer un orden de selección entre dichos candidatos; el criterio utilizado por el algoritmo ICSD está basado en el concepto de *strength*.

Para el cálculo del *strength* de un vértice $v \in V$ se tiene en cuenta al conjunto $v.Adj$ y al conjunto de adyacentes de cada vértice $w \in v.Adj$; i.e., $w.Adj$. Luego, el *strength* de un vértice $v \in V$ se denota por $v.strength$ y se calcula como:

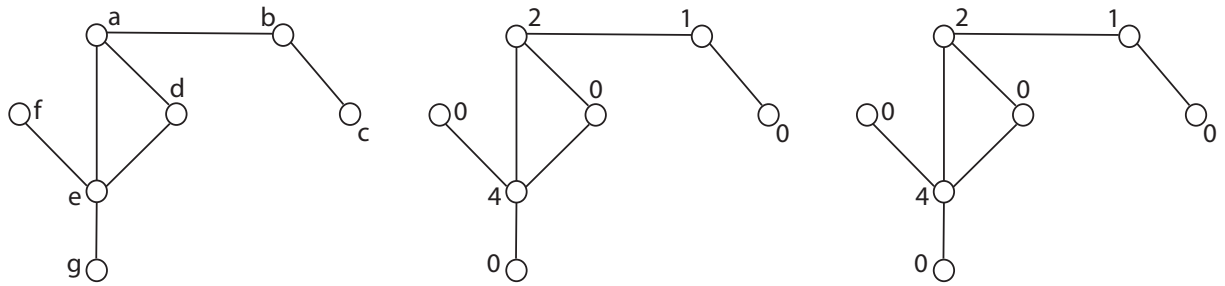
$$v.strength = |\{w \in v.Adj \mid v.strength_{pre} \geq w.strength_{pre}\}|,$$

donde $v.strength_{pre}$ es el número de vértices que son adyacentes a v y cuya densidad es menor o igual que la de v; $w.strength_{pre}$ se define de forma similar a $v.strength_{pre}$.

Intuitivamente, el *strength* de un vértice v es el número de satélites que puede cubrir v como centro de un sub-grafo FE, si se supone que todos los vértices que cubren más vértices que v ya fueron seleccionados como centros; luego, mientras mayor sea el *strength* de los vértices que se seleccionen como centros, más rápidamente será cubierto G_β y mayor será la densidad de los sub-grafos FE seleccionados.

En la Figura 4, a manera de ejemplo, se muestra un grafo de β-semejanza G_β con el valor de strength calculado para cada uno de sus vértices. En la Figura 4(a) se muestran los vértices de G_β etiquetados con letras, en la Figura 4(b) los vértices están etiquetados con su valor de strength$_{pre}$ y finalmente, en la Figura 4(c) están etiquetados con su valor de strength.

Como se puede observar en la Figura 4(c), el strength del vértice e es 4 porque su valor de strength$_{pre}$ supera al de sus 4 satélites adyacentes (a, f, d y g). El strength del vértice a es 2 dado que sólo uno de sus



(a) Grafo G_β con vértices etiquetados con letras.

(b) Grafo G_β con vértices etiquetados con su valor de $strength_{pre}$.

(c) Grafo G_β con vértices etiquetados con su valor de *strength*.

Figure 4. Ejemplo del cálculo del *strength* en un grafo de β-semejanza G_β.

3 satélites adyacentes, el vértice e, lo supera en strength$_{pre}$. El vértice b tiene un strength de 1 porque tiene un strength$_{pre}$ mayor que uno de sus dos adyacentes (a y c). Finalmente, el strength del resto de los vértices es 0 dado que no superan en strength$_{pre}$ a ninguno de sus adyacentes.

El algoritmo ICSD sólo considera como candidato para formar un sub-grafo FE a los vértices con un strength mayor que cero, puesto que son éstos los vértices que pueden cubrir al menos a otro vértice. Tomando como ejemplo el grafo de la Figura 4, sólo los vértices a, b y e son considerados como candidatos a centro al tener un *strength* mayor que cero.

El conjunto Q de vértices candidatos es ordenado descendentemente de acuerdo al strength y es procesado en ese orden; de esta forma se favorece la formación de grupos, sub-grafos FE, más densos. Cada vértice $v \in Q$ es seleccionado como centro si cumple algunas de las siguientes condiciones:

1. v no está cubierto.

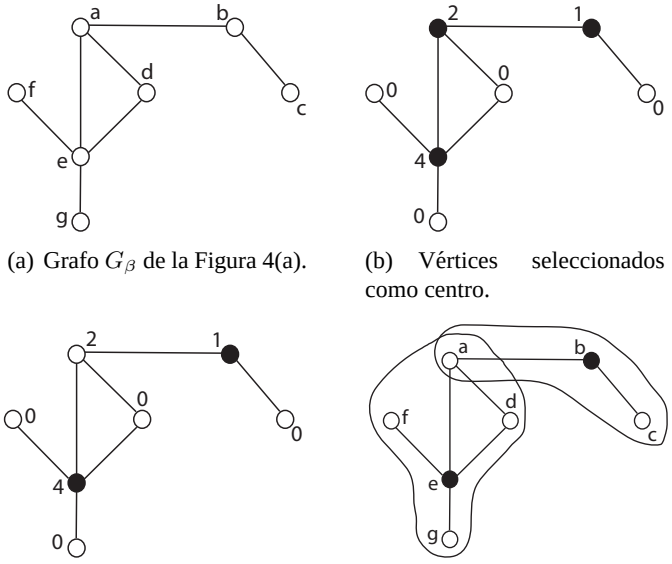
2. v está cubierto pero tiene al menos un vértice adyacente no cubierto. Esta condición evita la formación de sub-grafos que al tener todos sus satélites cubiertos por otros sub-grafos, no "ayudan" al cubrimiento de G_β; i.e., no permiten cubrir vértices no cubiertos.

Luego de construirse el conjunto de centros $C = \{c_1, c_2, \ldots, c_k\}$, éste es ordenado ascendentemente de acuerdo al grado de los centros y es filtrado, permitiendo así eliminar grupos *redundantes*.

Un centro $c \in C$ es *redundante* si es adyacente al menos a un centro más denso que él y si todos sus satélites son centros o están cubiertos por al menos otro centro. Usando el grado para la ordenación de los centros y en el criterio de redundancia se garantiza que: (a) se eliminen los centros redundantes menos densos del conjunto C y (b) que un centro sólo sea eliminado si éste ya está contenido en un grupo más denso.

En la Figura 5 se muestra, considerando el grafo de la Figura 4, cuál es el conjunto C de vértices seleccionados como centro, qué centros no son redundantes de acuerdo al criterio definido anteriormente y finalmente, cuáles son los grupos que conforman el agrupamiento final. En esta figura, los centros están resaltados en negro.

Como se puede observar en la Figura 5(c), del conjunto $C = \{a, b, e\}$ de vértices centro seleccionados inicialmente, sólo los vértices b y e no son redundantes. El centro b no es redundante porque, aunque es adyacente al centro a que es más denso que b, si se elimina b como centro entonces quedaría el vértice c no cubierto. Por otra parte, el centro a es redundante porque es adyacente al centro e que es más denso que a y porque si a se elimina como centro ninguno de sus satélites, los vértices e, d y b, queda sin cubrir. El centro



(a) Grafo G_β de la Figura 4(a).

(b) Vértices seleccionados como centro.

(c) Vértices centro no redundantes.

(d) Conjunto de grupos final.

Figure 5. Resultado del cubrimiento sobre el grafo G_β de la Figura 4(a).

e no es adyacente a ningún otro centro más denso y por tanto no es redundante. Finalmente, el conjunto final de grupos se muestra en la Figura 5(d).

5.1.1 Algoritmo ICSD

Para actualizar el conjunto de grupos construido por la estrategia de agrupamiento anteriormente mencionada, luego de la adición de uno o más objetos, es fundamental conocer cómo dichas adiciones afectan el agrupamiento actual.

Luego de adicionar uno o más vértices a G_β pueden ocurrir alguna o ambas situaciones siguientes:

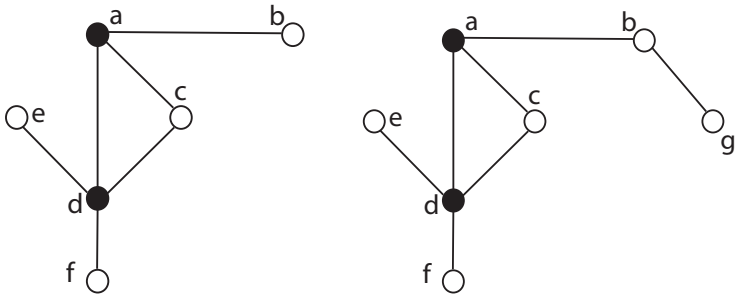
a) Aparecen vértices no cubiertos; luego, podría necesitarse seleccionar nuevos sub-grafos FE.

b) Cambia el strength de algunos vértices de G_β y aparece algún satélite v con un valor de strength mayor que algún centro adyacente o que algún centro que cubre al menos a un satélite adyacente a v. Los satélites como v representan una mejor opción para aumentar la densidad del agrupamiento y por tanto podrían ser seleccionados como nuevos sub-grafos FE.

En la Figura 6 se muestran situaciones en las que, al adicionar uno o más vértices a G_β, aparecen uno o más vértices no cubiertos.

En la Figura 7 se muestran situaciones en las que, luego de la adición de uno o más vértices a G_β, se modifica el strength de los vértices y puede aparecer algún satélite w que tienen un strength mayor que: (i) algún centro adyacente (en la Figura 7(c) el satélite h tiene un strength mayor que el centro b); (ii) algún centro que cubre al menos a un adyacente de w (en la Figura 7(f) el satélite g tiene un strength mayor que el centro a).

Finalmente, en la Figura 8 se presenta una combinación de las situaciones a) y b).



(a) Grafo G_β con los centros resaltados en negro.

(b) Adicionando el vértice no aislado g.

(c) Adicionando el vértice aislado g.

(d) Adicionando los vértices g y h.

Figure 6. Ejemplo de adiciones de vértices a G_β en las que ocurre la situación a).

Evidentemente, al sólo considerar la adición de vértices, no todos los grupos necesitan ser actualizados. Un vértice de G_β sólo puede variar su strength si en la componente conexa a la que éste pertenece existen vértices cuyo grado cambió. Luego, se puede concluir que los grupos que necesitan ser actualizados son aquellos contenidos en las componentes conexas de los vértices adicionados.

Para actualizar los grupos presentes en una componente conexa $G' = \langle V', E' \rangle$ primeramente se actualiza el valor de strength de cada vértice en V' y a continuación se construye la lista de candidatos Q'. Para formar la lista de candidatos Q' se procesan el conjunto S^+ de satélites con un strength mayor que cero y la lista C' de centros existentes.

Cada satélite $s \in S^+$ es procesado de acuerdo a las siguientes condiciones:

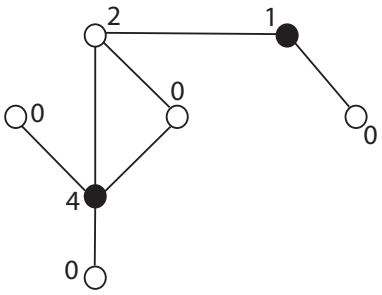
a) Si s no está cubierto o tiene algún adyacente no cubierto entonces, insertar s en Q'.

b) Si s tiene al menos un adyacente v, cuyo centro adyacente de mayor strength (w) cumple que $s.strength > w.strength$ entonces, insertar s en Q y marcar v como *activado*; los vértices *activados* son útiles en el procesamiento de C'.

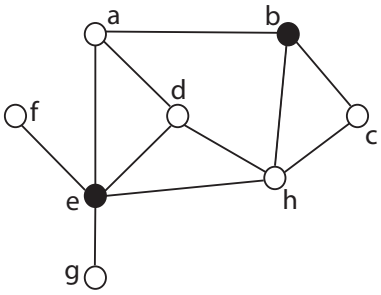
Cada centro $c \in C'$ es procesado de acuerdo a las siguientes condiciones:

a) Insertar en Q' cada satélite $v \in c.Adj$ que cumpla que $v.strength > c.strength$; marcar c como *débil*.

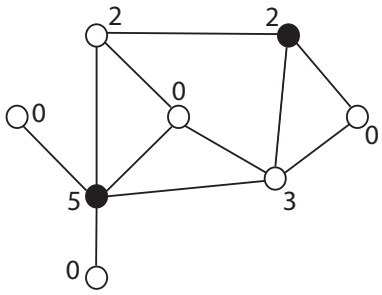
b) Si c es *débil* o tiene al menos un adyacente *activado* entonces, eliminar c de C', marcarlo como satélite e insertarlo en Q' sólo si $c.strength > 0$.



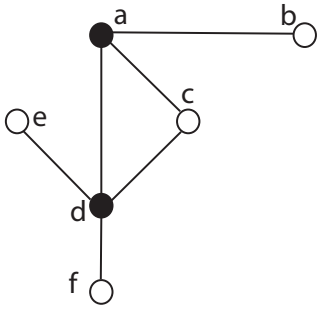
(a) Grafo G_β con los centros resaltados en negro.



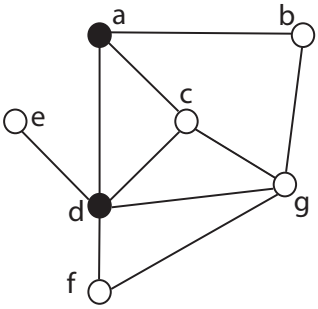
(b) Adicionando el vértice h.



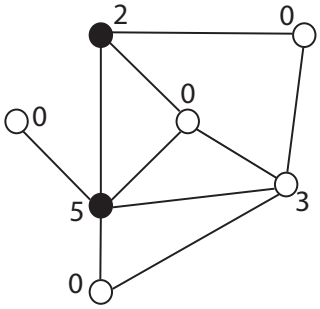
(c) Strength de los vértices en el grafo resultante de la adición.



(d) Grafo G_β con los centros resaltados en negro.

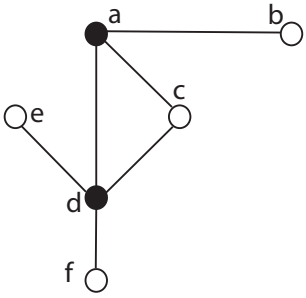


(e) Adicionando el vértice g.

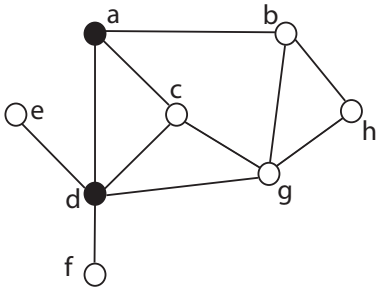


(f) Strength de los vértices en el grafo resultante de la adición.

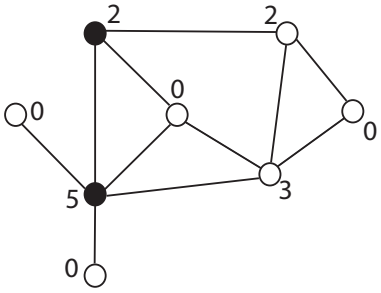
Figure 7. Ejemplo de adiciones de vértices a G_β en las que ocurre la situación b).



(a) Grafo G_β con los centros resaltados en negro.



(b) Adicionando los vértices g y h.



(c) Strength de los vértices en el grafo resultante de las adiciones.

Figure 8. Ejemplo de combinación de las situaciones a) y b).

Una vez que se construye Q', se actualiza el agrupamiento de G' utilizando la estrategia de agrupamiento propuesta anteriormente para obtener un conjunto de grupos con traslape. El pseudo-código del algoritmo ICSD se muestra en el Algoritmo 1.

El algoritmo ICSD construye un conjunto de grupos que pueden ser solapados y, a diferencia de los algoritmos Star, CI y FCI, permite adicionar a G_β, antes de comenzar el proceso de actualización del agrupamiento, el conjunto de vértices insertados a la colección; de esta forma, al procesar eficientemente las adiciones múltiples de objetos, ICSD ahorra tiempo de procesamiento.

Algoritmo 1: Algoritmo ICSD

Input: G_β - grafo de β-semejanza
L - conjunto de vértices a adicionar
β - umbral de semejanza
Output: S - conjunto de grupos

```
1  "adicionar vértices de L a G_β";
2  foreach vértice v ∈ L do
3      if v está marcado como no procesado then
4          "construir la componente conexa G' = ⟨V', E'⟩ de v";
5          if |V'| = 1 then "Marcar v como centro";
6          else
7              "actualizar strength en G'";
8              "construir C' y Q'";
9              "ordenar Q' descendentemente por strength";
10             foreach v ∈ Q' do if v cumple 1) o 2) then "C' := C' ∪ {v}";
11             "eliminar centros redundantes de C' ";
12             "marcar cada vértice de C' como centro";
13         end
14         "marcar cada vértice de G' como procesado";
15     end
16 end
17 "marcar cada vértice de G_β como no procesado";
18 "devolver conjunto S";
```

5.1.2 Resultados experimentales

A continuación se presentan los resultados de algunos experimentos en los que se compara el funcionamiento del algoritmo ICSD con a otros algoritmos basados en grafos que permiten obtener grupos solapados. Los algoritmos seleccionados para los experimentos fueron los algoritmos incrementales Star [25] y FCI [23] y el algoritmo estático Cstar [55], que ha reportado los mejores resultados dentro de los algoritmos basados en grafos que permiten obtener grupos solapados y que mejora también los resultados de los algoritmos SLINK y ALINK [49].

Los experimentos se realizaron sobre 6 colecciones de documentos, ya que en el agrupamiento de documentos es muy común que un documento pertenezca a más de una clase (tópico). Las características de las colecciones utilizadas en los experimentos se muestran en la Tabla 1.

En los experimentos, los documentos fueron representados utilizando el modelo vectorial de palabras (VSM) y se utilizó la medida del coseno para determinar la semejanza entre dos documentos. Para determinar la calidad de los grupos se utilizaron las medidas Fmeasure (Fme) [60] y Jaccard-index (Jindex) [59]; dos medidas utilizadas frecuentemente para evaluar algoritmos que permiten grupos solapados. Ambas medidas evalúan la calidad de un algoritmo basándose en cuánto el conjunto de grupos construido se asemeja a un conjunto de clases manualmente etiquetadas; a mayor valor de la medida mejor calidad del agrupamiento.

El primer experimento se enfocó en comparar la calidad de los algoritmos de acuerdo a las medidas anteriormente mencionadas. Para esto, los algoritmos se ejecutaron con valores de β en [0.15,0.75] y se

Tabla 1. Características de las colecciones de documentos

Colección	Documentos	Clases	Términos	Traslape
AFP	695	25	11785	1.02
Reu-Te	3587	100	15113	1.30
Reu-Tr	7780	115	21901	1.24
Reu-To	11367	120	27083	1.26
TDT2-v1	8603	176	51764	1.17
TDT2-v2	10258	174	53706	1.19

tomó, para cada algoritmo, los mejores resultados obtenidos de Fmeasure y Jaccard-index. En la Tabla 2 se muestra el resultado de este experimento; los mejores valores obtenidos en cada medida fueron resaltados en negro.

Tabla 2. Mejores valores de Fmeasure y Jaccard-index de cada algoritmo en las 6 colecciones.

		AFP				*Reu-Te*				*Reu-Tr*			
Measures		FCI	Star	Cstar	ICSD	FCI	Star	Cstar	ICSD	FCI	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		FCI	Star	Cstar	ICSD	FCI	Star	Cstar	ICSD	FCI	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

Como se puede observar en la Tabla 2, ICSD obtiene en todas las colecciones (a veces empatado con el algoritmo estático Cstar) los mejores valores de Fmeasure y Jaccard-index.

El segundo experimento se enfocó en comparar la cantidad y la densidad de los grupos construidos por cada algoritmo. Como el algoritmo FCI es el que construye la mayor cantidad de grupos, se tomó el valor de β para el cual FCI forma la menor cantidad de grupos y se comparó esta cantidad con el número de grupos formados por Star, Cstar y ICSD para el mismo valor de β. El resultado de este experimento se muestra en la Tabla 3; en esta tabla, las columnas “Grp” y “Dty” son el número de grupos y la densidad promedio de dichos grupos respectivamente.

Como se puede observar en la Tabla 3, ICSD obtiene un conjunto de grupos de menos cardinalidad y mayor densidad que los algoritmos Star y FCI en cada una de las colecciones. Por otra parte, ICSD obtiene resultados similares a los de Cstar en cuanto al número de grupos y mejores en cuanto a la densidad de éstos.

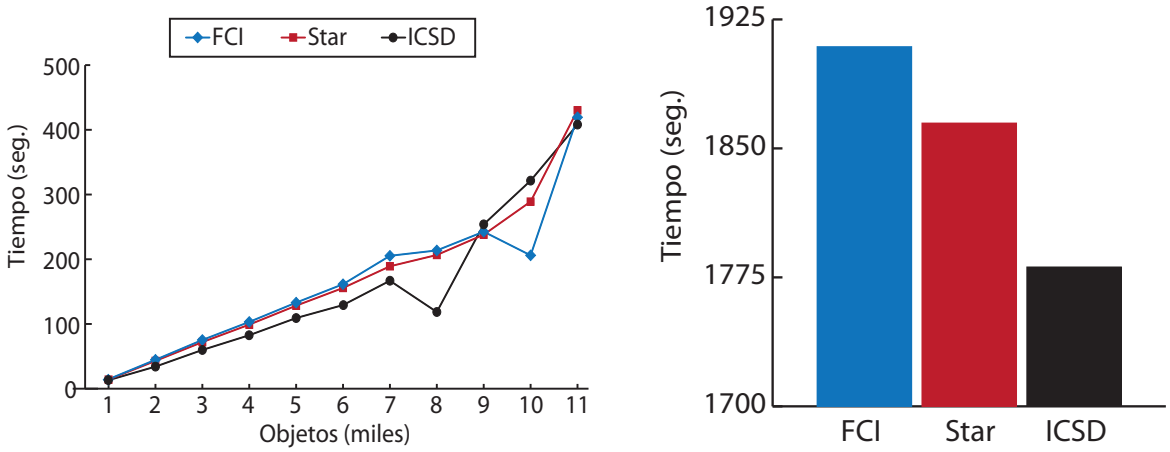
Finalmente, el tercer experimento se enfocó en comparar el tiempo empleado por cada uno de los algoritmos incrementales en la tarea de agrupar la colección de prueba más grande (*Reu-To*) de forma incremental. En la Figura 9(a) se muestra el tiempo empleado por los algoritmos ICSD, Star y FCI en la actualización del

Tabla 3. Cantidad y densidad de los grupos obtenidos por cada algoritmo.

	AFP		*Reu-Te*		*Reu-Tr*		*Reu-To*		*TDT2-v1*		*TDT2-v2*	
Algoritmo	Grp	Dty	Grp	Dty	Grp	Dty	Grp	Dty	Grp	Dty	Grp	Dty
FCI	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

conjunto de grupos cada vez que se adicionan a la colección 1000 documentos[5].

En la Figura 9(b) se muestra el tiempo total empleado por ICSD, Star y FCI en agrupar toda la colección *Reu-To* de forma incremental. En estas gráficas no se incluyó Cstar pues éste será analizado más adelante.



(a) Tiempo empleado en la actualización de los grupos durante cada adición de objetos.

(b) Tiempo total empleado por cada algoritmo para el agrupamiento de *Reu-To*.

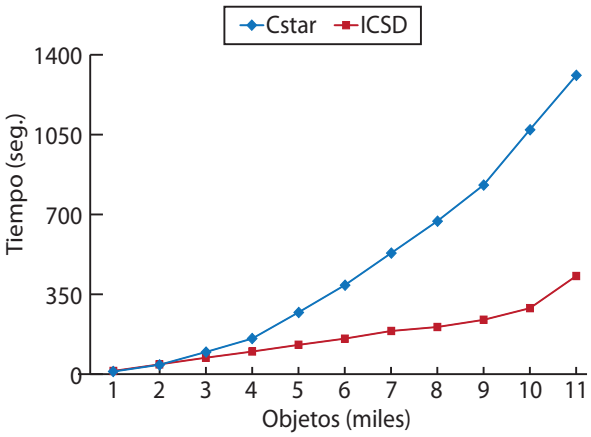
Figure 9. Gráfico con la comparación entre los algoritmos incrementales basados en grafos Star, FCI e ICSD.

Como se puede observar en la Figura 9(a) y Figura 9(b), ICSD es más rápido que los algoritmos Star y FCI; un comportamiento similar se apreció en las otras colecciones.

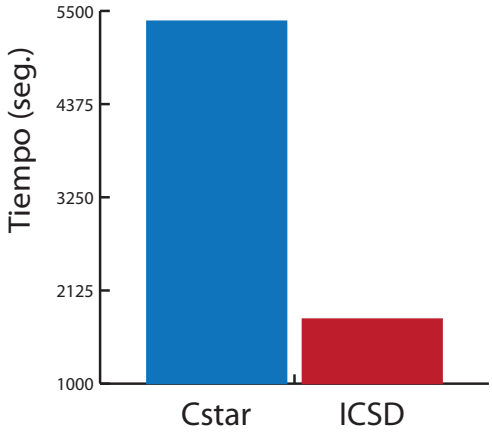
Adicionalmente, en la Figura 10 se compara el tiempo empleado por el algoritmo incremental ICSD y el algoritmo estático Cstar en la tarea de agrupar (*Reu-To*) de forma incremental.

Como se puede observar en la Figura 10, es grande la diferencia entre la actualización de los grupos y la construcción "desde cero" del agrupamiento cada vez que se adicionan objetos a la colección. Luego, el algoritmo ICSD es una mejor opción que los algoritmos Star, FCI y Cstar, para el procesamiento de colecciones en las que pueden haber cambios producto de adiciones.

[5]Se seleccionó 1000 por ser un valor ni muy pequeño ni muy grande en comparación con el tamaño de la colección



(a) Tiempo empleado en la actualización de los grupos durante cada adición de objetos.



(b) Tiempo total empleado por cada algoritmo para el agrupamiento de *Reu-To*.

Figure 10. Gráfico con la comparación entre el algoritmo incremental ICSD y el algoritmo estático Cstar.

6 Conclusiones

Los requerimientos y las disponibilidades de información existentes hoy en día, incrementan la necesidad de desarrollar nuevos métodos de agrupamiento que sean cada vez más útiles a las aplicaciones que procesan dicha información. De aquí la importancia de los métodos que se desarrollen en esta investigación.

Como parte de los resultados preliminares se presentó un nuevo algoritmo incremental para el agrupamiento de objetos llamado ICSD. El algoritmo propuesto no impone restricciones al tipo de objeto a agrupar, al espacio de representación de éste ni a la medida para determinar la semejanza entre dos objetos. ICSD permite obtener un conjunto de grupos solapados mediante el cubrimiento del grafo de β-semejanza a través de sub-grafos en forma de estrella; el uso de este tipo de sub-grafo le permite a ICSD reducir el encadenamiento en los grupos obtenidos.

ICSD, a diferencia de la mayoría de los algoritmos incrementales reportados en la literatura, permite el procesamiento eficiente de adiciones múltiples.

El algoritmo propuesto fue comparado contra otros tres algoritmos basados en grafos en seis colecciones de documentos. Los resultados experimentales mostraron que ICSD obtiene en las colecciones de prueba mejores resultados de calidad, considerando las medidas Fmeasure y Jaccard-index, que los otros métodos. Adicionalmente, en los experimentos se mostró que ICSD es más rápido en el procesamiento incremental de colecciones que los algoritmos incrementales basados en grafos usados en los experimentos.

Finalmente, con base en los resultados preliminares alcanzados, concluimos que siguiendo la metodología propuesta se pueden alcanzar los objetivos de esta investigación en el tiempo planteado.

References

[1] Vignes, M., Forbes, F.: Gene clustering via integrated Markov models combining individual and pairwise features. IEEE/ACM Transactions on Computational Biology and Bioinformatics. (2007) 260–270.

[2] Gupta, G., Liu, A., Ghosh, J.: Automated Hierarchical Density Shaving: A robust, automated clustering and visualization framework for large biological datasets. IEEE/ACM Transactions on Computational Biology and Bioinformatics. (2008).

[3] Mahata, P.: Exploratory Consensus of Hierarchical Clusterings for Melanoma and Breast Cancer. IEEE/ACM Transactions on Computational Biology and Bioinformatics. (2008).

[4] Bloy, G.J.: Blind Camera Fingerprinting and Image Clustering. IEEE Transactions on Pattern Analysis and Machine Intelligence. 30 (3) (2008) 532–534.

[5] Di Martino, F., Loia, V., Sessa, S.: Extended fuzzy C-means clustering algorithm for hotspot events in spatial analysis. International Journal of Hybrid Intelligent Systems. 5 (1) (2008) 31–44.

[6] Pons-Porrata, A., Berlanga-Llavorí, R., Ruiz-Shulcloper, J., Pérez-Martínez, J. M.: JERARTOP: A New Topic Detection System. In Proceedings of the 9th Iberoamerican Congress on Pattern Recognition, LNCS 3287, Springer Verlag. (2004) 446–453.

[7] Cselle, G., Albrecht, K., Wattenhofer, R.: BuzzTrack: Topic Detection and Tracking in Email. In Proceedings of the 12th international conference on Intelligent user interfaces (UI2007). (2007) 446–453.

[8] Lepouras, G.: Applying clustering algorithms to web-based adaptive virtual environments. Journal of Computational Methods in Science and Engineering. 7 (2) (2007) 93–103.

[9] Petridou, S. Koutsonikola, V.A., Vakali, A.I., Papadimitriou, G.I.: Time-Aware Web Users' Clustering. IEEE Transactions on Knowledge and Data Engineering. 20 (5) (2008) 653–667.

[10] Pons-Porrata, A., Berlanga-Llavorí, R.: Métodos y Algoritmos para la Agrupación Automática de Documentos. Universitat Jaume I, España. (2001).

[11] Hill, D.R.: A Vector clustering technique. In: Samuelson (eds.), Mechanized Information Storage, Retrieval and Dissemination, North-Holland, Amsterdam. (1968) 501–508.

[12] Spitters, M., Kraaij, W.: TNO at TDT2001: Language Model-Based Topic Detection. Topic Detection and Tracking (TDT) Workshop. (2001).

[13] Ester, M., Kriegel, H., Sander, J., Wimmer, M., Xu, X.: Incremental clustering for mining in a data warehousing environment. In Proceedings of the 24th Very Large Databases Conference. (1998) 323–333.

[14] Ester, M., Kriegel, H., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In Proceedings of the 2nd International Conference on Knowledge Discovery abd DAta Mining (KDD-96). (1996) 226–231.

[15] Beckmann, N., Kriegel, H., Schneider, R., Seeger, B.: The R*-tree: An efficient and Robust Access Method for Points and Rectangles. In Proceedings of the ACM SIGMD international Conference on Management of Data. (1990) 322–331.

[16] Ciaccia, P., Patella, M., Zezula, P.: M-tree: An Efficient Accsess Method for Similarity Search in Metric Spaces. In Proceedings of the 23 International Conference on Very Large DataBases. (1997) 426–435.

[17] Zamir, O., Etziony, O.: Web document clustering: A feasibility demonstration. In Proceedings of the 21st Annual International ACM SIGIR Conference. (1998) 46–54.

[18] D. Gusfield, Algorithms on strings, trees and sequences: computer science and computational biology. Chapter 6. Cambridge University Press. (1997).

[19] Sibson, R.: An optimally effcient algorithm for the single link cluster method. Computer Journal. 16 (1973) 30–34.

[20] Sánchez-Díaz, G., Ruiz-Shulcloper, J.: Mid Mining: a Logical Combinatorial Pattern Recognition Approach to clustering in Large Data Sets. In Proceedings of the 5th Iberoamerican Symposium on Pattern Recognition SIARP 2000. (2000) 475–483.

[21] Sánchez-Díaz, G., Ruiz-Shulcloper, J.: A Clustering Method for Very Large Mixed Data Sets. In Proceedings of the First IEEE International Conference on Data Mining (ICDM'01). (2001) 643.

[22] Pons-Porrata, A. Berlanga-Llavorí, R., Ruiz-Shulcloper, J.: On-line event and topic detection by using the compact sets clustering algorithm. Journal of Intelligent and Fuzzy Systems, 12 (3-4) (2002) 185–194.

[23] Pons-Porrata, A., Ruiz-Shulcloper, J., Berlanga-Llavorí, R., Santiesteban-Alganza, Y.: Un algoritmo incremental para la obtención de cubrimientos con datos mezclados. Reconocimiento de Patrones. Avances y Perspectivas. Research on Computing Science, CIARP2002. (2002) 405–416.

[24] Martínez-Trinidad, J.F., Ruiz-Shulcloper, J., Lazo-Cortés, H.: Structuralization of Universes. Fuzzy Sets and Systems. 112 (3) (2000) 485–500.

[25] Aslam, J., Pelekhov, K., Rus, D.: Static and Dynamic Information Organization with Star Clusters. In Proceedings of the seventh international conference on Information and knowledge management. (1998) 208–217.

[26] Aslam, J., Pelekhov, K., Rus, D.: Using Star Clusters for Filtering. In Proceedings of the Ninth International Conference on Information and Knowledge Management, USA. (2000) 306–313.

[27] Aslam, J., Pelekhov, E., Rus, D.: The Star Clustering Algorithm for Static and Dynamic Information Organization. Journal of Graph Algorithms and Applications. 8 (1) (2004) 95–129.

[28] Khy, S., Ishikawa, Y., Kitagawa, H.: Novelty-based Incremental Document Clustering for On-line Documents. In Proceedings of 22nd International Conferencet On Data Enggineering Workshops (ICDEW'06). (2006) 40–40.

[29] Ishikawa, Y., Chen, Y., Kitagawa, H.: An On-line Document Clustering Method Based on Forgetting Factors. In Proceedings of the Fifth European Conference on Research and Advanced Techonology for Digital Libraries. (2001) 325–539.

[30] Hartigan, J.: Clustering Algorithms. John Wiley & Sons, New York, NY. (1975).

[31] Chen, L., Tu, L., Chen, Y.: An Ant Clustering Method for a Dynamic Database. In Proceedings of the ICMLC 2006. (2006) 169–178.

[32] Nayak, R.: Fast and effective clustering of XML data using structural information. Knowledge and Information Systems. 14 (2) (2008) 197–215.

[33] Nayak, R., Xu, S.: XCLS: A Fast and Effective Clustering Algorithm fir Heterogenous XML Documents. In Proceedings of the PAKDD 2006, LNAI 3918. (2006) 292–302.

[34] Elghasel, H., Kheddouci, H., Deslandres, V., Dussauchoy, A.: A Partially Dynamic Clustering Algorithm for Data Insertion and Removal. In: V. Corruble, M. Takeda and E. Suzuki (eds.), DS 2007, LNAI 4755. (2007) 78–90.

[35] Elghasel, H., Deslandres, V., Hacid, M.S., Dussauchoy, A., Kheddouci, H.: A New Clustering Approach for Symbolic Data and Its Validation: Application to the Healthcare Data. In: F. Esposito et al (eds.), ISMIS 2006, LNAI 4203. (2006) 473–482.

[36] Kalyani, M., Sushmita, M.: Clustering and its validation in a symbolic framework. Pattern Recognition Letters. 24 (14) (2003) 2367–2376.

[37] Hammouda, K.M., Kamel, M.S.: Efficient Phrase-Based Document Indexing for Web Document Clustering. IEEE Transactions on Knowledge and Data Engineering. 16 (10) (2004) 1279–1296.

[38] Chung, S., McLeod, D.: Dynamic Pattern Mining: An Incremental Data Clustering Approach. Journal on Data Semantics II. 3360 (2005) 85–122.

[39] Chung, S., McLeod, D.: Dynamic Topic Mining from News Stream Data. On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE. 2888 (2003) 653–670.

[40] Salton, G., McGill, M.J.: Introduction to modern information retrieval. McGraw-Hill. (1983).

[41] Wai-chiu, W., Fu, A.W.: Incremental document clustering for web page classification. IEEE 2000 International Conference on Information Society in the 21st Century: Emerging technologies and new challenges. (2000).

[42] Clement, Y.T., Meng, W.: Principles of DAtabase Query Processing for Advanced Applications. Morgan Kaufmann Publisher, Inc.. 10 (1998) 359–385.

[43] Widyantoro, D.H., Ioerger, T.R., Yen, J.: An incremental approach to building a cluster hierarchy. In Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM2002). (2002) 705–708.

[44] Pons-Porrata, A., Berlanga-Llavorí, R., Ruiz-Shulcloper, J.: Building a Hierarchy of Events and Topics for Newspaper Digital Libraries. In Proceedings of the 25th European Conference on IR Research (ECIR 2003), LNCS 2633. (2003) 588–596.

[45] Gil-García, R.J., Badía-Contelles, J.M., Pons-Porrata, A.: Dynamic Hierarchical Compact Clustering Algorithm. In Proceedings of the X Iberoamerican Congress on Pattern Recognition (CIARP2005), LNCS 3773. (2005) 302–310.

[46] Gil-García, R.J., Pons-Porrata, A.: Hierarchical Star Clustering Algorithm for Dynamic Document Collections. In Proceedings of the XIII Iberoamerican Congress on Pattern Recognition (CIARP2008). (2008) 187–194.

[47] Gil-García, R.J., Pons-Porrata, A.: A speed-Up Hierarchical Compact Clustering Algorithm for Dynamic Document Collections. In Proceedings of the XIV Iberoamerican Congress on Pattern Recognition (CIARP2009), LNCS 5856. (2009) 497–504.

[48] Gurrutxaga, I., Arbelaitz, O., Martín, J.I., Muguerza, J., Pérez, J.M., Perona, I.: SIHC: A Stable Incremental Hierarchical Clustering Algorithm. In Proceedings of the 11th International Conference on Enterprise Information Systems (ICEIS2009). (2009) 300–304.

[49] Jain, A., Dubes, R.C.: Algorithms for Clustering Data. Prentice Hall. (1988).

[50] Sia, W., Lazarescu, M.: Clustering Large Dynamic Datasets Using Exemplar Points. In Proceedings of the 3th International Conference on Machine Learning and Data Mining in Pattern Recognition, MLDM 2005, LNAI 3587. (2005) 163–173.

[51] Fournier, D., Simon, G., Mermet, B.: A Dynamic Clustering Algorithm for Mobile Objects. In Proceedings of the PKDD 2007, LNAI 4702. (2007) 422–429.

[52] Liu, B., Shi, Y., Wang, Z., Wang, W., Shi, B.: Dynamic Incremental Data Summarization for Hierarchical Clustering. In Proceedings of the WAIM 2006, LNCS 4016. (2006) 410–421.

[53] Li, Y., Li, F.: Modeling Hierarchical User Interests Based on HowNet and Concept Mapping. In Proceedings of the Fourth International Conference on Semantics, Knowledge and Grid. (2008) 157–164.

[54] Bordogna, G., Pagani, M., Pasi, G.: A Dynamic Hierarchical Fuzzy Clustering Algorithm for Information Filtering. In Proceedings of the PKDD 2007, LNAI 4702. (2006) 3–23.

[55] Pérez-Suárez, A., Martínez-Trinidad, J.F., Carrasco-Ochoa, J.A., Medina-Pagola, J.E.: A New Graph-Based Algorithm for Clustering Documents. In Proceedings of the ICDM-Workshops 2008. (2008) 302–310.

[56] Gil-García, R.J., Badía-Contelles, J.M., Pons-Porrata, A.: Extended Star Clustering Algorithm. In Proceedings of the CIARP 2003, LNCS 2905. (2003) 480–487.

[57] Zhao, Y., Karipys, G., Fayyad, U.: Hierarchical Clustering Algorithms for Document Datasets. Data Mining and Knowledge Discovery. 10 (2) (2005) 141–168.

[58] Puzicha, J., Hofmann, T., Buhmann, J.: A theory of proximity based clustering: Structure detection by optimization. PATREC: Pattern Recognition. 23 (4) (2000) 617-634.

[59] Kuncheva, L., Hadjitodorov, S.: Using diversity in cluster ensembles. In Proceedings of the 2004 IEEE International Conference on Systems, Man and Cybernetics. (2004) 1214–1219.

[60] Banerjee, A., Krumpelman, C., Basu, S., Mooney, R., Ghosh, J.: A New Graph-Based Algorithm for Clustering Documents. In Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining (KDD2005). (2005) 532–537.

[61] Pérez-Suárez, A., Martínez-Trinidad, J.F., Carrasco-Ochoa, J.A., Medina-Pagola, J.E.: A New Incremental Algorithm for Overlapped Clustering. In Proceedings of the 14th Iberoamerican Congress on Pattern Recognition (CIARP2009), LNCS 5856. (2009) 497–504.