Algoritmo de agrupamiento compacto jerárquico dinámico paralelo

R. J. Gil-García[1], J. M. Badía-Contelles[2], A. Pons-Porrata[1]

1. Centro de Reconocimiento de Patrones y Minería de Datos
Universidad de Oriente
Avda. Patricio Lumumba S/N. 90500 Santiago de Cuba, Cuba
{gil,aurora}@csd.uo.edu.cu

2. Departamento de Ingeniería y Ciencia de la Computación
Universitat Jaume I
Avda. Sos Baynat S/N. 12071 Castellón, España
badia@icc.uji.es

Resumen

En este trabajo exponemos la paralelización del algoritmo Compacto Jerárquico Dinámico sobre una arquitectura de memoria distribuida. Para paralelizar este algoritmo se utilizó el paradigma de particionado de datos y se identificaron los procesos susceptibles de ejecutar concurrentemente. En los experimentos realizados utilizando colecciones estándares de documentos, el algoritmo paralelo obtenido reduce claramente los tiempos de cómputo. Sin embargo, las aceleraciones tienen un comportamiento sublineal y no alcanzan valores significativos debido a la pequeña granularidad de los procesos de actualización de los niveles superiores de la jerarquía. No obstante, el algoritmo paralelo propuesto es útil para procesar grandes volúmenes de datos de forma eficiente e imprescindible cuando la memoria disponible en un solo procesador no es suficiente para el almacenamiento de los datos.

1 Introducción

En muchos problemas prácticos se necesita crear jerarquías de objetos como la construcción de taxonomías biológicas y la detección en un flujo continuo de noticias de los tópicos y de los sucesos en los que éstos se descomponen. Nótese que en algunos de estos ejemplos, se necesita además que el algoritmo jerárquico sea dinámico por la naturaleza de las colecciones a agrupar.

Se han desarrollado muchos algoritmos de agrupamiento jerárquico, entre ellos podemos citar el *Single-link*, *Complete-link*, *Average-link*, *Bisecting K-Means* [1],

ICT [2] y *UMASS* [3]. Estos algoritmos requieren la colección completa de objetos para realizar el agrupamiento y es ineficiente utilizarlos en problemas donde las colecciones sean dinámicas. Además, los pocos algoritmos jerárquicos incrementales que existen, tales como *GALOIS* [4], el algoritmo de Charikar [5] y *DC-tree* [6], presentan varios de los siguientes inconvenientes:

1. tienen una complejidad temporal exponencial con respecto a la dimensión de los objetos,
2. necesitan fijar a priori la cantidad de grupos a obtener,
3. requieren muchos parámetros, o
4. los grupos obtenidos tienen una elevada dependencia del orden de presentación de los objetos.

En [7] propusimos el algoritmo Compacto Jerárquico Dinámico. Este algoritmo es capaz de mantener actualizada una jerarquía de grupos y no presenta ninguna de las limitaciones anteriores. Su complejidad computacional es cuadrática. Para posibilitar su uso en grandes colecciones de objetos, en este trabajo proponemos su versión paralela. El algoritmo Compacto Jerárquico Dinámico Paralelo aprovecha el poder de cómputo disponible en una computadora de memoria distribuida. En el trabajo mostramos los resultados obtenidos en el agrupamiento de cuatro colecciones de documentos en un cluster de PCs. Este algoritmo logra reducir los tiempos de cómputo hasta cierta cantidad de procesadores.

2 Compacto Jerárquico Dinámico

Dos objetos cuya semejanza es mayor o igual que un cierto umbral β (definido por el usuario) se denominan β-semejantes. Si un objeto no es β-semejante con ningún otro objeto se denomina β-aislado. Se llama grafo de β-semejanzas al grafo donde los vértices son los objetos a agrupar y existen aristas entre los objetos β-semejantes, etiquetadas con el valor de su semejanza. Llamaremos grafo de máxima β-semejanza (*max-S*), al grafo orientado donde los vértices son los objetos a agrupar y existe un arco del vértice o_i al vértice o_j si se cumple que o_j es el objeto más β-semejante a o_i. Por ultimo, denotaremos como *U-max-S* al grafo que se obtiene del grafo *max-S* ignorando la orientación de sus arcos (ver Figura 1).

El algoritmo Compacto Jerárquico Dinámico realiza un agrupamiento multi-capas para obtener la jerarquía, donde se buscan sucesivamente los conjuntos compactos [8]. Los conjuntos compactos coinciden con las componentes conexas del grafo *U-max-S*. Este algoritmo, además de la medida de semejanza entre objetos, requiere de una medida de semejanza entre grupos. En este caso, se utiliza el promedio de las semejanzas entre los objetos de los grupos que se comparan.

Dada una jerarquía de grupos, previamente construida por el algoritmo, cada vez que se agrega o se elimina un objeto es necesario actualizar los grupos existentes en todos los niveles de la jerarquía. Ante la llegada de un nuevo objeto, se crea un grupo unitario y se a añade al nivel inferior de la jerarquía. Luego, se actualizan los grafos de β-semejanza y *max-S* y, además, se actualizan los conjuntos compactos que constituyen los grupos del siguiente nivel de la jerarquía. Cuando se crea o se elimina un grupo de un nivel de la jerarquía, el grafo de β-semejanza del próximo nivel debe actualizarse. Este proceso se repite hasta que el grafo de β-semejanza sea completamente inconexo, es decir, todos sus vértices sean β-aislados. Es posible que el grafo obtenido sea completamente inconexo antes de haber alcanzado el nivel tope de la jerarquía. En este caso, los siguientes niveles de la jerarquía se eliminan (ver algoritmo 1). Note que en los grafos de β-semejanza y *max-S* de un cierto nivel de la jerarquía, los vértices se corresponden con los grupos del nivel anterior.

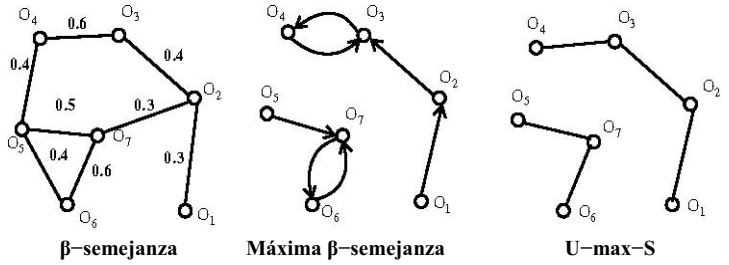

Figura 1: Grafos basados en la β-semejanza, β = 0.3.

Algoritmo 1 Algoritmo Compacto Jerárquico Dinámico

1. Llegada del objeto a agregar (o eliminar).
2. Crear un grupo unitario con el objeto (o eliminar el grupo unitario al cual pertenece el objeto).
3. *nivel* =0.
4. Actualizar el grafo de β-semejanza, G_{nivel}.
5. Mientras G_{nivel} no sea completamente inconexo:
   a. Actualizar el grafo *max*-S_{nivel}.
   b. Actualizar las componentes conexas en el subgrafo *U-max-* S_{nivel}.
   c. *nivel* = *nivel* + 1.
   d. Actualizar el grafo de β-semejanza, G_{nivel}.
6. Si existen niveles mayores que *nivel* en la jerarquía, eliminarlos.

Como podemos observar, el algoritmo dinámico en cada nivel de la jerarquía

comprende tres etapas: la actualización del grafo de β-semejanza, del grafo *max-S*, y la actualización de las componentes conexas en el subgrafo *U-max-S.* La actualización del grafo de β-semejanza es trivial, pues ante la llegada de un nuevo objeto simplemente deben calcularse sus semejanzas con los restantes, añadirse un nuevo vértice al grafo y aquellas aristas cuya semejanza entre el nuevo objeto y uno existente supere el umbral β. Del mismo modo, al eliminar un objeto se elimina el vértice correspondiente junto con todas sus aristas. En todos los niveles de la jerarquía el proceso es exactamente el mismo, con la única diferencia de que en lugar de añadir o eliminar objetos, estamos añadiendo o eliminando grupos.

Veamos entonces cómo se realiza la actualización del grafo *max-S* (ver algoritmo 2). Cuando se presenta un nuevo objeto, se crea un nuevo vértice y se pueden crear y eliminar arcos. Por el contrario, cuando se elimina un objeto, se elimina el vértice correspondiente y sus arcos, y se buscan los objetos más semejantes a aquellos objetos que tenían al objeto eliminado como su máximo semejante.

Algoritmo 2 Actualización del grafo *max-S*.

1. Sea N el conjunto de vértices a agregar y R el conjunto de vértices a eliminar en el grafo *max-S*.
2. Sea M el conjunto de vértices cuyos vértices más β-semejantes pertenecen a R.
3. Eliminar del grafo *max-S* los vértices de R y agregar los de N .
4. Buscar los vértices más β-semejantes a cada vértice de $M \cup N$ y agregar los arcos correspondientes.
5. Buscar los vértices para los cuales un vértice de N es su más β-semejante y agregar los arcos correspondientes.

Dado que se tiene actualizado el grafo *max-S*, es trivial actualizar su correspondiente grafo *U-max-S*. Cada vez que se elimina una arista del subgrafo *U-max-S*, el grupo (conjunto compacto) al que pertenecen los vértices de esta arista puede perder la propiedad de conectividad. Por tanto, este grupo debe ser reconstruido. Por el contrario, cada vez que se agrega una arista al subgrafo *U-max-S*, los grupos de sus vértices se unen si son diferentes. La actualización de las componentes conexas provoca que aparezcan nuevos grupos y que se eliminen otros (ver algoritmo 3).

El algoritmo Compacto Jerárquico Dinámico tiene una complejidad temporal $O(n^2)$. El paso más complejo es el cálculo de las semejanzas entre los nuevos grupos y los restantes en cada nivel de la jerarquía. En este cálculo se utilizan las fórmulas de actualización de Lance-Williams [9] que permiten calcular la semejanza entre los grupos de un nivel a partir de las semejanzas del nivel inferior. La complejidad espacial es también $O(n^2)$, pues deben almacenarse todas las semejanzas entre grupos.

3 Compacto Jerárquico Paralelo

En el algoritmo Compacto Jerárquico Paralelo los objetos se distribuyen cíclicamente entre los procesadores, de forma tal que el procesador i almacena la descripción del objeto *j* si *j mod p = i*, donde *p* es la cantidad de procesadores. Cada procesador almacena, además, la porción del grafo de β-semejanza y la porción del grafo *max-S* correspondientes a sus objetos. Los grupos de cada nivel de la jerarquía también están distribuidos entre todos los procesadores de forma cíclica y cada procesador almacena las semejanzas entre sus grupos y los restantes. En general, no existe relación entre los objetos y los grupos almacenados en cada procesador. Por tanto, por cada nivel de la jerarquía cada procesador almacena, además, las porciones del grafo de β-semejanza y del grafo *max-S* correspondientes a sus grupos de ese nivel.

Algoritmo 3 Actualización de las componentes conexas

1. Sea *N* el conjunto de vértices añadidos al subgrafo *U-max-S* y *R* el conjunto de vértices eliminados. Sean, además, *N E* el conjunto de aristas añadidas al subgrafo *U-max-S* y *RE* el conjunto de aristas eliminadas.
2. Sea *Q* la lista de vértices a procesar. Inicialmente, $Q = \varnothing$.
3. Eliminar todos los vértices de *R* de los grupos (componentes conexas) a los que pertenecían. Añadir a *Q* los restantes vértices de esos grupos. Agregar, además, a *Q* todos los vértices de los grupos donde al menos una arista en *RE* es incidente a un vértice del grupo. Eliminar todos estos grupos de la lista de grupos existentes.
4. Agregar a *Q* todos los vértices de *N*.
5. Construir las componentes conexas existentes entre los vértices de *Q* y agregarlas a la lista de grupos existentes.
6. Para cada arista en *N E*, unir los grupos a los que pertenecen sus vértices si son diferentes.

Cada vez que se presenta un nuevo objeto se difunde su descripción. Si, por el contrario, se desea eliminar un objeto del agrupamiento, se difunde su índice. A continuación, en el procesador dueño del objeto se crea (o se elimina) el grupo unitario correspondiente. Entre todos los procesadores se actualiza el grafo de β-semejanza y se realiza una operación de procesamiento centralizado para determinar si este grafo es completamente inconexo. Para ello, cada procesador de forma independiente determina si su porción del grafo de β-semejanza es completamente inconexa. Esta información se recoge en un procesador y si todas las porciones fueron completamente inconexas, se difunde que el grafo de β-semejanza también lo es. Si esto se cumple, se termina la actualización de la jerarquía y se eliminan los niveles superiores si existen. Por el contrario, si el grafo

de β-semejanza no es completamente inconexo, se actualizan el grafo max-*S* y las componentes conexas sobre el subgrafo U-max-*S* del nivel actual. Estas actualizaciones provocan que surjan y se eliminen grupos en este nivel, por lo que se procede a actualizar el próximo nivel de la jerarquía. Las actualizaciones del grafo de β-semejanza, del grafo max-*S* y de las componentes conexas del subgrafo U-max-*S* en cada nivel de la jerarquía se realizan en paralelo, pero hasta que no finalice cada una de estas etapas en todos los procesadores, no pueden iniciar la siguiente. Las comunicaciones requeridas en cada etapa funcionan como puntos de sincronización (ver algoritmo 4).

Algoritmo 4 Compacto Jerárquico Dinámico Paralelo.

1. Difusión del objeto a agregar (o a eliminar).
2. Si es el procesador dueño del objeto, crear un grupo unitario con el objeto (o eliminar el grupo unitario al cual pertenece el objeto).
3. *nivel* = 0.
4. Sea G_{nivel} el grafo de β-semejanza global correspondiente al nivel de la jerarquía *nivel*. Sea, además, S_{nivel} la porción de este grafo correspondiente a este procesador.
5. Actualizar S_{nivel}, aplicando el algoritmo 5.
6. Operación de procesamiento centralizado para determinar si G_{nivel} es completamente inconexo.
7. Mientras G_{nivel} no sea completamente inconexo:
   a. Actualizar *max-*S_{nivel}.
   b. Actualizar las componentes conexas sobre *U-max-Snivel*.
   c. *nivel* =*nivel* + 1.
   d. Actualizar S_{nivel}, aplicando el algoritmo 5.
   e. Operación de procesamiento centralizado para determinar si G_{nivel} es completamente inconexo.
8. Si existen niveles mayores que *nivel* en la jerarquía, eliminarlos.

Como mencionamos anteriormente, la actualización del grafo de β-semejanza se realiza en paralelo (pasos 5 y 7d). En cada procesador, para cada uno de los grupos a eliminar se remueven los vértices correspondientes de su porción del grafo de β-semejanza. Siempre que se elimina un vértice del grafo se eliminan también todas las aristas incidentes en él. Por el contrario, cada vez que se presentan grupos a agregar, se calculan las semejanzas entre ellos y los grupos que pertenecen al procesador. Finalmente, se actualiza su porción del grafo de β-semejanza, agregando las aristas correspondientes a las β-semejanzas (semejanzas mayores o iguales que el umbral β). Los pasos básicos del método se muestran en el algoritmo 5.

En el primer nivel de la jerarquía, el paso 5 del algoritmo 5 se reduce a calcular las semejanzas entre objetos, pues los grupos son unitarios. El algoritmo 6 muestra los pasos básicos que deben realizarse. En este proceso, se calculan las semejanzas entre los objetos de este procesador y todos los nuevos objetos. Las β-semejanzas de sus objetos con los nuevos objetos que no le pertenecen se almacenan en una estructura de datos independiente por cada procesador, de forma tal que cada una de ellas contenga las semejanzas correspondientes a sus objetos. Al finalizar esta etapa, los procesadores realizan una comunicación todos a todos para obtener las β-semejanzas de los nuevos objetos que les pertenecen.

En los siguientes niveles de la jerarquía, el paso 5 se reduce a calcular las semejanzas entre grupos, en nuestro caso, la semejanza promedio. Este cálculo lo explicaremos más adelante para facilitar su comprensión. Veamos primeramente cómo se lleva a cabo la actualización del grafo *max-S* y de las componentes conexas.

Algoritmo 5 Actualización del grafo de β-semejanza en paralelo.

1. Sea *N* el conjunto de grupos a agregar y *R* el conjunto de grupos a eliminar.
2. Sea *S* la porción del grafo de β-semejanza correspondiente a los grupos que pertenecen al procesador.
3. Eliminar de *S* los vértices correspondientes a los grupos de *R*.
4. Agregar a *S* los vértices correspondientes a los grupos de *N* que pertenecen a este procesador.
5. Calcular las semejanzas entre los grupos de *N* y los grupos que pertenecen a este procesador.
6. Agregar a *S* las aristas correspondientes a las semejanzas calculadas que sean mayores o iguales que el umbral β.

Algoritmo 6 Cálculo de las semejanzas entre los objetos.

1. Calcular las semejanzas entre los objetos que pertenecen a este procesador y los objetos de *N*. Crear una estructura de datos por procesador con las β-semejanzas entre los objetos de este procesador y los objetos de *N* que pertenecen a cada procesador.
2. Comunicación todos a todos para obtener en cada procesador las semejanzas de los objetos de *N* que les pertenecen.

Expliquemos, primero, el proceso de actualización del grafo *max-S* (ver algoritmo 7). Al igual que el grafo de β-semejanza, el grafo *max-S* se encuentra distribuido entre todos los procesadores. Así, cada procesador almacena la porción del grafo *max-S* correspondiente a sus grupos, es decir, de cada grupo G se tiene los índices

de sus grupos más β-semejantes (*DeEl(G)*), los índices de los grupos para los cuales él es su más β-semejante (*AEl(G)*) y el valor de la máxima semejanza (*MaxSem*). Los vértices que se agregaron y eliminaron en el grafo de β-semejanza, deben ser actualizados en el grafo *max-S*. Los vértices eliminados deben ser removidos del grafo max-S, pero éstos pueden ser los más β-semejantes de otros grupos. Por tanto, es necesario determinar cuáles son ahora sus grupos más β-semejantes y agregar en el grafo los arcos correspondientes, lo que implica actualizar los conjuntos *AEl*, *DeEl* y *MaxSem* de los grupos. Si estos arcos involucran a grupos de otro procesador, entonces se crean los conjuntos *ArcosAgregados* por cada procesador. El conjunto *ArcosAgregados* del procesador *k* contiene los arcos (*Gi* ,*Gj*) que hay que agregar en su porción del grafo *max-S*, donde *Gi* es un grupo del procesador que crea el conjunto y *Gj* es un grupo que pertenece al procesador k. La búsqueda de los grupos más β-semejantes a un grupo puede hacerse, pues se tiene el grafo de β-semejanza actualizado en cada nivel de la jerarquía.

Algoritmo 7 Actualización del grafo de máxima β-semejanza en paralelo.

1. Sea N el conjunto de vértices agregados y R el conjunto de vértices eliminados del grafo de β-semejanza.
2. Sea S la porción del grafo max-S correspondiente a este procesador.
3. Para cada vértice v en R:
   a. Buscar los grupos más β-semejantes de los grupos G de este procesador que tienen a v como su más β-semejante, esto es, v β DeEl(G). Agregar a S los arcos correspondientes. En este proceso se crea el conjunto ArcosAgregados por cada procesador.
   b. Eliminar v de S.
4. Para cada vértice v de N que pertenece al procesador:
   a. Buscar los grupos más β-semejantes a v y los grupos de los que v es su más β-semejante. En este proceso se crea el conjunto ArcosP erdidos por cada procesador.
5. Comunicación todos a todos de ArcosAgregados y ArcosP erdidos.
6. Actualizar el grafo S con los arcos recibidos.

Por otra parte, cada vértice agregado en el grafo de β-semejanza debe ser incorporado a su porción del grafo *max-S* si es un vértice correspondiente a un grupo perteneciente al procesador. Para cada uno de ellos es necesario construir los conjuntos *AEl*, *DeEl* y *MaxSem*. En este proceso puede suceder que el grupo agregado sea el más β-semejante a un grupo existente y los que eran sus más β-semejantes dejan de serlo, por lo que se rompen arcos. Si estos arcos involucran a grupos de otro procesador, entonces se crean los conjuntos *ArcosPerdidos* por cada

procesador. Al finalizar este proceso, en cada procesador se han creado y eliminado arcos que pertenecen a otros procesadores. Por ello se realiza una comunicación todos a todos que permite la actualización de esos arcos en los procesadores correspondientes, lo que implica actualizar los conjuntos *AEl* de sus grupos.

Algoritmo 8 Actualización de las componentes conexas en paralelo.

1. Sean *N* el conjunto de vértices añadidos y *R* el conjunto de vértices eliminados del subgrafo *U-max-S*. Sea, además, *RE* el conjunto de aristas eliminadas en la porción del subgrafo *U-max-S* correspondiente a este procesador.
2. Sea *L* la lista de las componentes conexas de este procesador.
3. Crear el conjunto de vértices a procesar *C*. Este conjunto estará formado por todos los vértices de *N* que pertenecen al procesador y todos los vértices pertenecientes a este procesador de las componentes conexas de L que tengan al menos una arista en *RE*.
4. Buscar las componentes conexas existentes entre los vértices de *C* y agregarlas a *L*.
5. Eliminar de *L* las componentes conexas que tengan al menos una arista en *RE* y las componentes conexas unitarias que contengan algún vértice de *R*.
6. Crear una lista con los grupos globales a los que pertenecen las componentes conexas eliminadas de *L* en el paso anterior.
7. Operación de procesamiento centralizado donde se unen todas las listas creadas en el paso anterior en todos los procesadores y se difunden los grupos eliminados a nivel global.
8. Crear una lista *LN* con las componentes conexas encontradas entre los vértices de *C* y las componentes conexas de *L* pertenecientes a los grupos globales eliminados.
9. Operación de procesamiento centralizado donde se recogen las listas *LN* creadas en todos los procesadores, se determinan los nuevos grupos globales y se difunden los nuevos grupos globales y a qué grupo global pertenece cada componente conexa.
10. Asignar a cada componente conexa de *LN* el grupo global al que pertenece.

Explicaremos ahora el proceso de actualización de las componentes conexas del subgrafo *U-max-S* (paso 7b del algoritmo 4). Una vez actualizado el grafo *max-S* es fácil ver que se cuenta con el subgrafo *U-max-S* actualizado, pues simplemente debemos ignorar la orientación de sus arcos. El proceso de actualización de las componentes conexas se realizará en cada procesador partiendo de los vértices eliminados y añadidos en todo el grafo *max-S* y de las aristas eliminadas y añadidas en la porción del grafo *max-S* correspondiente al procesador. La actualización de los conjuntos compactos se realiza también entre todos los procesadores. En cada procesador se almacena para cada nivel de la jerarquía la lista de las componentes conexas existentes en su porción del grafo *max-S* y a qué grupo global pertenece cada una de estas componentes conexas. Los grupos globales se formarán mediante la unión de las componentes conexas parciales obtenidas en cada procesador. En el

algoritmo 8, dados los vértices añadidos y eliminados y las aristas eliminadas durante la actualización del grafo *max-S* en un procesador, se determina el conjunto de vértices pertenecientes a las componentes conexas de este procesador que pueden perder la propiedad de conectividad y que, por tanto, son eliminadas. A continuación, se reconstruyen las componentes conexas existentes entre dichos vértices y se agregan a la lista de las componentes conexas del procesador. Cuando se elimina una componente conexa en un procesador, el grupo global al cual pertenece puede perder la propiedad de conectividad. Para reconstruir a nivel global estos grupos se realiza una operación de procesamiento centralizado donde se determina qué grupos a nivel global deben ser eliminados. Mediante otra operación de procesamiento centralizado se determinan los nuevos grupos globales que se forman a partir de las componentes conexas encontradas en cada procesador y de las componentes conexas que formaban parte de los grupos eliminados a nivel global. Finalmente, se actualiza la pertenencia de las componentes conexas de cada procesador a los grupos globales obtenidos.

Por ultimo, analicemos cómo colaboran todos los procesadores para calcular la semejanza promedio entre grupos durante la actualización del grafo de β-semejanza en los siguientes niveles de la jerarquía (ver paso 5 del algoritmo 5). Como explicamos anteriormente, en cada procesador se tiene un conjunto de componentes conexas y el grupo global al que pertenece cada una. Cada componente conexa representa la porción del grupo global que se encuentra en el procesador. Además cada procesador recibió los nuevos grupos globales creados. Esto implica que un procesador no dispone de toda la información necesaria para calcular las semejanzas entre sus grupos globales y los restantes, pues de cada uno de sus grupos sólo tiene una información parcial. Por tanto, es necesario que los procesadores cooperen entre sí para realizar este cálculo. Por otro lado, es muy costoso calcular la semejanza entre grupos directamente a partir de los objetos que los componen. Por tanto, se utilizan fórmulas similares a las de Lance-Williams que permiten calcular la semejanza entre los grupos de un nivel a partir de las semejanzas entre grupos del nivel inferior. En el caso paralelo, como las semejanzas entre grupos se calculan de forma parcial entre las distintas porciones de los grupos en cada procesador, necesitamos almacenar la suma de las semejanzas entre los objetos que forman los grupos de cada procesador y los restantes grupos. En particular, las sumas de las semejanzas entre grupos en el primer nivel coinciden con el grafo de β-semejanza entre los objetos. A partir de esta suma y los tamaños de los grupos puede obtenerse la semejanza promedio entre los grupos.

Para calcular dicha suma de semejanzas y los tamaños de los grupos, cada procesador calcula la suma de las semejanzas entre sus grupos del nivel inferior que conforman las componentes conexas existentes en este procesador y los grupos del nivel inferior que conforman los nuevos grupos globales. De esta forma, en cada procesador se tiene una

semejanza parcial entre cada grupo existente en el procesador y los nuevos grupos. En cada procesador se determina además el tamaño parcial de los nuevos grupos a partir del tamaño de los grupos del nivel inferior de este procesador que pertenecen al nuevo grupo. Luego, se realiza una comunicación todos a todos para informar a cada procesador de las semejanzas parciales de sus grupos con los nuevos. A cada procesador dueño de un nuevo grupo se le envía también los tamaños parciales de sus grupos. Con esta información, en cada procesador se calcula la suma de las semejanzas entre sus grupos y los nuevos grupos. En los dueños de los nuevos grupos se calcula a su vez su tamaño. Finalmente, se realiza otra comunicación todos a todos para que cada procesador dueño de los nuevos grupos tenga la suma de las semejanzas de sus grupos con los restantes y todos los procesadores tengan los tamaños de los nuevos grupos. Con estos datos, cada procesador puede calcular las semejanzas promedios entre sus grupos y los nuevos grupos y actualizar su porción del grafo de β-semejanza. Los pasos básicos del método se muestran en el algoritmo 9.

Algoritmo 9 Cálculo de las semejanzas entre grupos.

1. Calcular la suma de las semejanzas entre los grupos del nivel inferior que forman las componentes conexas existentes en este procesador y los grupos del nivel inferior que forman los grupos de N (nuevos grupos creados).
2. Calcular el tamaño parcial de los grupos de N a partir del tamaño de los grupos del nivel inferior que pertenecen a este procesador.
3. Comunicación todos a todos para obtener en cada procesador las sumas de semejanzas parciales de sus grupos con los grupos de N y los tamaños parciales de los nuevos grupos en sus procesadores.
4. Calcular las sumas de semejanzas entre los grupos que pertenecen a este procesador y los grupos de N.
5. Calcular los tamaños de los nuevos grupos.
6. Comunicación todos a todos para obtener en cada procesador las sumas de semejanzas de los grupos de N que les pertenecen y en todos los procesadores los tamaños de los nuevos grupos.
7. Calcular las semejanzas promedio entre sus grupos y los nuevos grupos y actualizar el grafo de β-semejanza que involucra a sus grupos.

4 Resultados experimentales

Para evaluar la eficiencia del algoritmo Compacto Jerárquico Dinámico Paralelo desarrollamos los experimentos en un cluster de PC. Esta computadora paralela está formada por 34 nodos y está interconectada mediante una red Myrinet. Cada nodo es un biprocesador Intel Xeon a 2,4 GHz con 1Gb de RAM. Como datos de prueba utilizamos cuatro colecciones de documentos escritos en castellano y en inglés provenientes de diversas fuentes. La descripción de las colecciones se muestra en la tabla 1. En los experimentos realizados fijamos el valor de β en 0.13,

para el cual se obtuvieron resultados de calidad satisfactorios en todas las colecciones.

Colección	Fuente	Cantidad de documentos	Dimensión	Promedio de términos
afp	TREC-5	694	12575	103
eln	TREC-4	5829	84344	174
tdt	TDT2	9824	55112	103
reu	Reuters-21578	10369	35297	42

Tabla 1: Descripción de las colecciones de prueba

4.1 Comportamiento del algoritmo paralelo

Para evaluar el comportamiento del algoritmo paralelo medimos los tiempos del mejor algoritmo secuencial y del paralelo utilizando la colección *eln*. Estos tiempos se tomaron incrementando la cantidad de objetos a agrupar en intervalos de 100 objetos. En el caso del algoritmo paralelo las ejecuciones se realizaron en 2, 4, 8, 16 y 32 procesadores.

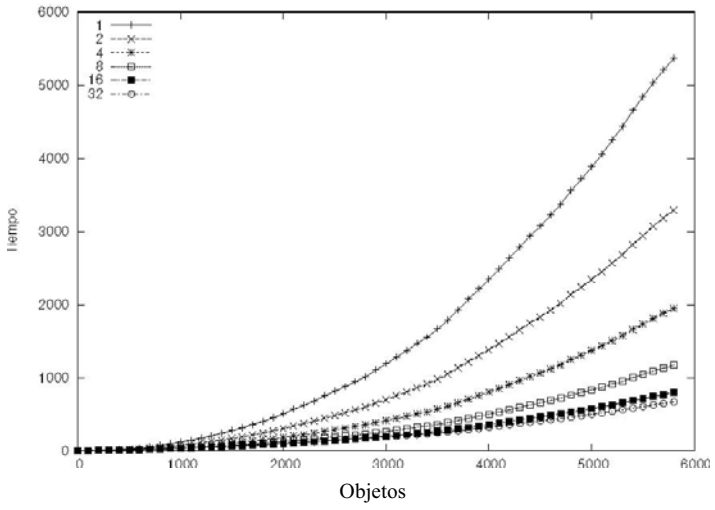

Figura 2: Comportamiento del Compacto Jerárquico Dinámico Paralelo

Como podemos observar en la figura 2, tanto el algoritmo secuencial como el paralelo, tienen un comportamiento cuadrático. Nuestro algoritmo paralelo es capaz de disminuir claramente los tiempos de cómputo a medida que aumenta la cantidad de procesadores. Como se esperaba, con el incremento de la cantidad de

procesadores, las reducciones en los tiempos son menores debido a que las comunicaciones tienen más peso en el coste total del algoritmo.

4.2 Aceleración

En la figura 3 se muestran las aceleraciones obtenidas en cada colección de documentos. En primer lugar, cabe decir que las aceleraciones obtenidas no son muy notables y su comportamiento es sublineal. Este comportamiento se explica fácilmente debido al elevado coste de comunicaciones que tiene este algoritmo, provocado por la pequeña granularidad que se obtiene en la actualización de los niveles superiores de la jerarquía. A medida que aumenta el número de procesadores la aceleración crece hasta una determinada cantidad de procesadores donde empieza a decrecer. Los valores de aceleración obtenidos, al igual que el punto de inflexión de la aceleración dependen de la longitud promedio de los documentos en cada colección y su tamaño. En particular, las aceleraciones de la colección *reu* son mayores en parte porque los datos que se necesitan no caben en la memoria primaria en un procesador y es necesario el uso de la memoria secundaria.

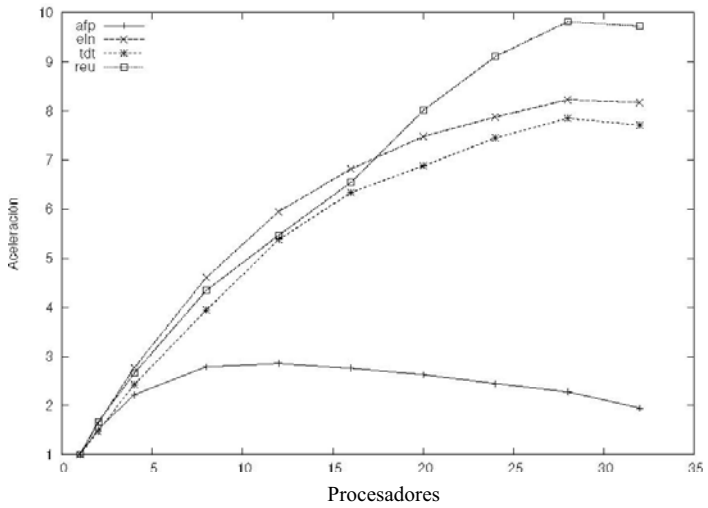

Figura 3: Aceleraciones del Compacto Jerárquico Dinámico Paralelo

5 Conclusiones

Para posibilitar la aplicación del algoritmo Compacto Jerárquico Dinámico a grandes volúmenes de datos en este artículo presentamos su versión paralela. El algoritmo paralelo obtenido, a pesar de la complejidad y las dependencias de las

operaciones del algoritmo secuencial, logra reducir apreciablemente los tiempos de cómputo hasta cierto número de procesadores. No obstante, debido a la gran cantidad de comunicaciones, las aceleraciones son sublineales y no alcanzan valores notables.

El algoritmo paralelo obtenido es útil, ya que además de permitir agrupar colecciones de gran tamaño en un tiempo razonable, es imprescindible cuando la memoria disponible en un solo procesador no es suficiente para el almacenamiento de los datos. En particular, el algoritmo puede aplicarse en problemas concretos donde se necesita obtener una jerarquía de grupos y se requiere procesar grandes volúmenes de datos. Ejemplos de estas aplicaciones son la obtención de directorios de la web y la detección en un flujo continuo de noticias de los tópicos y de los sucesos en los que éstos se descomponen.

Como trabajo futuro pensamos estudiar otras paralelizaciones de la actualización de los niveles de la jerarquía para aumentar la granularidad de este proceso y obtener mejores prestaciones.

Referencias

[1] M. Steinbach, G.Karypis, and V. Kumar. A comparison of document clustering techniques. In *KDD Workshop on Text Mining*, 2000.

[2] Man-Quan Yu, Wei-Hua Luo, Zhao-Tao Zhou, Shuo Bai. ICT's Approaches to HTD and Tracking at TDT2004. In *TDT2004 Workshop*, 2004.

[3] M. Connell and et al. UMASS at TDT 2004. In *TDT2004 Workshop*, 2004.

[4] Claudio Carpineto and Giovanni Romano. A lattice conceptual clustering system and its application to browsing retrieval. *Machine Learning 24*, (2):95–122, 1996.

[5] M. Charikar et al. Incremental clustering and dynamic information retrieval. In *29th Annual Symposium on Theory of Computing*, pages 626–635, 1997.

[6] Wong Wai-chiu and Ada Wai-chee Fu. Incremental Document Clustering for Web Page Classification. In *IEEE 2000 International Conference on Information Society in the 21st Century: Emerging technologies and new challenges*, 2000.

[7] R. J. Gil-García, J. M. Badía-Contelles, and A. Pons-Porrata. Dynamic Hierarchical Compact Clustering Algorithm. *Lecture Notes on Computer Sciences*, 3773:302–310, 2005.

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

[9] G.N. Lance and W.T. Williams. A general theory of classificatory sorting strategies. 1:Hierarchical systems. *Computer Journal*, 9:373–380, 1967.