ALGORITMO PARA AGRUPACIÓN DE DOCUMENTOS WEB GENERADO DESDE UN ENFOQUE HÍPER HEURÍSTICO

ANDREA DUQUE BOTERO

JAMITH BOLAÑOS VIDAL

Director: Ph.D. (c) MSc. CARLOS ALBERTO COBOS LOZADA

UNIVERSIDAD DEL CAUCA

FACULTAD DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES

DEPARTAMENTO DE SISTEMAS

GRUPO DE I+D EN TECNOLOGÍAS DE LA INFORMACIÓN

LÍNEA DE INTERÉS EN GESTIÓN DE LA INFORMACIÓN – OPTIMIZACIÓN

POPAYÁN, Mayo 2013

Tabla De Contenido

Lista De Tablas

Lista De Figuras

Capítulo 1

1 INTRODUCCIÓN

1.1 PROBLEMA

Hasta hace muy poco tiempo, la única manera en la que las personas podían obtener información y conocimientos nuevos era realizando consultas en los libros de las bibliotecas. Hoy en día además de las bibliotecas existe internet, un medio que se volvió muy popular alrededor del mundo y donde la gran mayoría de personas prefiere consultar información, debido a que se convirtió en una inmensa "biblioteca electrónica" [1].

Paralelamente a la gran acogida que ha tenido internet, la cantidad de documentos disponibles en este medio ha crecido exponencialmente y para acceder a estos documentos los usuarios deben usar buscadores web, directorios de documentos web o meta-buscadores, que aún hoy retornan resultados inconsistentes. Resultados que en general cumplen con los criterios de búsqueda del usuario del sistema, pero que no cumplen totalmente con los verdaderos intereses de información del usuario, razón por la cual, la investigación en técnicas de "recuperación de información" en la web que es lo que se conoce como "búsqueda web (web search)" toma día a día más importancia en la comunidad académica y científica, intentando hacer un tratamiento automático a los documentos y a las consultas de los usuarios para dar respuestas más precisas (relevantes) a las necesidades de información de dichos usuarios [2].

Con el paso del tiempo se ha intentado mejorar la forma de mostrar la información al usuario, de allí, aparece un grupo de investigaciones que apuntan a lo que se conoce como la "agrupación de documentos web", en la que se agrupan documentos por temas o tópicos, lo que permite al usuario obtener resultados más ordenados y una mejor aproximación al contexto de la búsqueda. Estos sistemas se denominan motores (normalmente meta buscadores) web que agrupan

documentos web o web clustering engines en inglés. En el marco de estos meta buscadores se ha propuesto gran variedad de algoritmos de agrupamiento que intentan mejorar las etiquetas de los grupos que se presentan a los usuarios, la organización de los documentos en los grupos y el número de grupos de documentos, entre otros. En estos meta-buscadores, en cada grupo o cluster, se busca agrupar documentos bastante similares entre sí (mismo tópico o tema) y que se diferencien bien de los otros grupos, ayudando de esta forma a que el usuario encuentre más rápidamente los documentos relacionados con su necesidad de información y descarte explícitamente los documentos que no son de su interés (logrando además, ampliar la cobertura de documentos revisados) [3].

Para agrupar los documentos se han usado diversos algoritmos, entre ellos, algoritmos particionales, algoritmos jerárquicos, algoritmos basados en densidad, algoritmos difusos y algoritmos basados en descomposición matricial [4]. Otra alternativa que se ha empezado a explorar, es el uso de heurísticas de optimización, heurísticas como los algoritmos genéticos, los algoritmos meméticos, la búsqueda armónica, entre otros, los cuales han mostrado ser una buena solución a este problema [5-7].

Aunque las heurísticas empiezan a mostrar su capacidad para resolver el problema de la agrupación de documentos web, en muchos casos, definir cuál de todas ellas y la afinación específica o diseño de cada proceso interno (selección, cruce, mutación, reemplazo [8]) es un proceso complejo. Como una alternativa para realizar este proceso, aparece el enfoque híper heurístico, en el cual se usan heurísticas de bajo nivel que resuelven el problema específico (en este caso, la agrupación de documentos web) y heurísticas de alto nivel que buscan cual heurística de bajo nivel es la más apropiada para resolver el problema específico (normalmente, un problema complejo) [9].

La principal motivación detrás de las híper heurísticas es el desarrollo de algoritmos que son de aplicación más general que muchas de las actuales implementaciones de algoritmos de búsqueda. Su característica principal es que opera en el espacio de búsqueda de la heurística y no directamente sobre el espacio de búsqueda de soluciones a un problema determinado. Al momento de utilizarla, se está tratando de encontrar el método adecuado o una secuencia de las heurísticas en una situación dada, en lugar de tratar de resolver el problema directamente. Por lo tanto, el objetivo es producir métodos genéricos, lo que debería producir soluciones de calidad aceptable, basado en un conjunto de funciones fáciles de implementar, de bajo nivel heurístico. Por lo que una híper heurística puede ser vista como una metodología que, cuando se le da una instancia particular de un problema o tipo de casos, y una serie de heurísticas de bajo nivel (o sus componentes), automáticamente produce una combinación adecuada de los componentes proporcionados para resolver eficazmente el problema dado [9].

Teniendo en cuenta que varias heurísticas de bajo nivel resuelven adecuadamente el problema de la agrupación de documentos web y que el enfoque híper heurístico es un mejor enfoque de diseño de soluciones heurísticas a un problema, al inicio de este proyecto se planteó la siguiente pregunta de investigación ¿Es posible mejorar el proceso de agrupación de documentos web mediante un algoritmo generado desde un enfoque híper heurístico buscando mejorar la calidad de las agrupaciones y facilitando con ello el proceso de búsqueda en la web?

Para dar respuesta a esta pregunta de investigación, se requirió la definición de un entorno híper heurístico compuesto de diversas heurísticas de alto y bajo nivel, luego la evaluación de las heurísticas con conjuntos de datos de prueba (como el disponible en http://www.unicauca.edu.co/~ccobos/wdc/wdc.htm) y medidas de calidad de la agrupación de los documentos.

1.2 APORTES

Los aportes de este trabajo de grado, se plantearon desde tres perspectivas:

1- Innovación: en la literatura revisada la mayoría de proyectos o investigaciones proponen enfoques híper heurísticos aplicados a procesos de optimización a problemas que hacen referencia a horarios de exámenes [10-13], problemas de variabilidad del tiempo de respuesta [14] y problemas de optimización para el lanzamiento aéreo de vehículos satelitales [15]. En esta investigación se aplicó al área de agrupación de documentos web, en el que se había aplicado un enfoque híper heurístico preliminar [16], con poco soporte experimental, un bajo soporte de heurísticas de alto y bajo nivel, así como de heurísticas de reemplazo.

2- Investigación: al aplicar el enfoque híper heurístico a un número de métodos heurísticos de bajo nivel, con el fin de obtener la heurística que mejora el proceso de agrupación de documentos web, se obtuvo nuevo conocimiento útil para la comunidad académica y científica en el área de recuperación de información, proponiendo un nuevo y mejor algoritmo especifico de agrupación de documentos web.

3- Desarrollo: se elaboró una aplicación en donde se implementó el algoritmo que usa los métodos heurísticos de bajo nivel, con el fin de encontrar la mejor solución para el problema de agrupación de documentos web.

Teniendo en cuenta limitaciones con el tiempo máximo para el desarrollo del trabajo de grado, se limitaron las heurísticas para el desarrollo del proyecto, como se presenta en los objetivos.

1.3 OBJETIVOS

A continuación se presentan el objetivo general y los objetivos específicos alcanzados con la presente investigación.

1.3.1 Objetivo general

Proponer un algoritmo para la agrupación de documentos web[1] generado desde un enfoque híper heurístico buscando mejorar la calidad de las agrupaciones y facilitando con ello el proceso de búsqueda web.

1.3.2 Objetivos específicos

- Definir un entorno híper heurístico de agrupación de documentos web con las siguientes características:
  - Un conjunto de heurísticas de alto nivel que permitan seleccionar y orientar el proceso de búsqueda en el espacio de las heurísticas de bajo nivel, teniendo en cuenta la selección: aleatoria, basada en rendimiento (ruleta), basada en probabilidad y por búsqueda tabú.
  - Un conjunto de heurísticas de bajo nivel que resuelven el problema específico de agrupación de documentos web, teniendo en cuenta las versiones más reconocidas de la búsqueda armónica, los esquemas de selección, cruce y mutación más ampliamente usados en los algoritmos genéticos, evolución diferencial, optimización basada en enjambres de partículas y optimización por colonias de abejas.
  - Un proceso de evolución que se centra en individuos y no en poblaciones y cada individuo es sujeto de un proceso de optimización local usando el algoritmo k-means (similar a un algoritmo memético).
  - Un conjunto de heurísticas de alto nivel que permitan definir cuales individuos entran a formar parte de la población (reemplazo), como las estrategias de reemplazar el peor individuo, competencia restringida, reemplazo estocástico y reemplazo probabilístico.

- Proponer un prototipo software basado en el entorno híper heurístico que permita:

[1] En el marco de un motor que agrupa documentos web (Web Clustering Engine) según la definición propuesta por Carpineto et al [3].

- La evaluación del mismo sobre un conjunto de datos basados en DMOZ (similares a los 50 data sets disponibles en www.unicauca.edu.co/~ccobos/wdc/wdc.htm).
- Medir la calidad del proceso de agrupación de documentos web del mejor algoritmo obtenido en el entorno, basado en precisión ponderada, recuerdo ponderado y medida F ponderada.
- Comparar los resultados del ítem anterior contra los algoritmos de Suffix Tree Clustering STC , Bisecting k-means y Lingo.

- Definir la calidad del proceso de agrupación de documentos web del mejor algoritmo generado en el entorno híper heurístico, a través de un conjunto de pruebas con usuarios, quienes evaluaran la claridad y utilidad de las etiquetas, la pertinencia de los documentos a los grupos y el orden de los mismos en los grupos y comparar los resultados contra Carrort2 (lingo) [8].

1.4 RESULTADOS

Como resultado de la investigación desarrollada se obtuvo lo siguiente:

- Diseño e implementación de un entorno híper heurístico con un proceso de evolución centrado en individuos que hace uso de: 1) cuatro heurísticas de alto nivel: Tabú, Rank, Aleatorio y Ruleta; 2) veinticinco (25) heurísticas híbridas de bajo nivel, que son el resultado de la combinación de las siguientes meta heurísticas: Colonia de abejas (CA), evolución diferencial (ED), búsqueda armónica (HS), mejor búsqueda armónica global (BH), búsqueda armónica mejorada (IH), nueva búsqueda armónica global (NH), optimización por enjambre de partículas (PS) y; diversas heurísticas (18) basadas en algoritmos genéticos (AG); Las dieciocho heurísticas de algoritmos genéticos se basaron en los siguientes métodos de selección: (rank (RK), selección por ruleta (RW) y selección por competencia restringida (RM)); cruce: (uniforme (CU), en un punto (UP) y multi-punto (CM)); mutación (de un bit (MO) y multi-bit (MM)); 3) cuatro métodos de reemplazo que se encargan de determinar si un individuo ingresa a la

población: remplazo del peor (WR), Rank (RR), competencia restringida (RC), y remplazo estocástico (SR). El entorno es ejecutado sobre los datasets DMOZ, AMBIENT, MORESQUE y ODP239, aplicado al problema de agrupación de documentos web.

- Producto de la ejecución del entorno híper heurístico y un análisis detallado de los resultados se obtiene que el algoritmo que mejor desempeño presenta en el proceso de agrupación de documentos web es la heurística de bajo nivel denominada: nueva mejor búsqueda armónica global GBHS usando como método de remplazo Rank.
- Articulo a presentar en revista internacional con el Algoritmo para agrupación de documentos web generado desde un enfoque híper heurístico, el cual presenta los principales resultados de la investigación (ver Anexo).

Capítulo 2

2 CONTEXTO TEÓRICO

2.1 RECUPERACIÓN DE INFORMACIÓN (RI)

La evolución de las tecnologías computacionales ha generado un aumento en la cantidad de información disponible a ser consultada por los usuarios, fenómeno conocido como explosión documental, cuyo principal inconveniente es tener acceso preciso y rápido a la información [17], con el fin de buscar soluciones a estos inconvenientes surge la recuperación de la información la cual se define como las técnicas empleadas para almacenar y buscar grandes cantidades de datos y ponerlos a disposición de los usuarios, es importante resaltar que el objetivo principal de la recuperación de la información es proporcionar información relevante al usuario para satisfacer su necesidad de información [18].

Chang [18] define tres formas de buscar información en la web: motores de búsqueda, directorios y meta buscadores. Los motores de búsqueda son grandes aplicaciones que manejan bases de datos que contienen referencias a páginas web, que son recopiladas de una manera automática por agentes de búsqueda también conocidos como robots que recorren la web recopilando nuevas direcciones, generando etiquetas que permiten su indexación y almacenamiento en la base de datos, entre estos se destaca Google, Altavista, Lycos entre otros; Los directorios son aplicaciones donde los humanos son los encargado de almacenar manualmente en las bases de datos la información relevante de las páginas web, y donde la clasificación se realiza en subdirectorios con categorías temáticas organizadas de forma jerárquica, esto permite el acceso a los recursos desde lo general a lo específico, la mayoría de los índices dan la posibilidad de realizar las consultas de dos formas: por medio de la navegación en la estructura de las categorías temáticas o a través de consultas con palabras clave, el directorio más conocido es Yahoo!

Los meta buscadores son aplicaciones desarrolladas para evitar el problema de tener que ingresar a diferentes motores de búsqueda con el fin de obtener resultados más completos sobre una consulta, la tarea principal de un meta buscador es trasladar la consulta hecha por el usuario a diferentes motores de búsqueda, colecciona las respuestas recibidas y las unifica para ser mostradas. Estos sistemas no contienen en sus bases de datos información sobre las páginas, pero cuentan con registros de motores de búsqueda e información sobre ellos; con el fin de mantener un buen desempeño en cuanto al tiempo de respuesta los meta buscadores no extraen la totalidad de los documentos presentados por los motores de búsqueda. Dentro de este grupo se encuentra copernic, Inquirus, All4One, entre otros [18].

2.2 AGRUPACIÓN DE DOCUMENTOS WEB (WDC)

Los sistemas de recuperación de información convencionales retornan largas listas de documentos jerarquizados que los usuarios están forzados a revisar para encontrar los documentos relevantes. Paradigma seguido por la mayoría de los motores de búsqueda web, los cuales se caracterizan por la baja precisión que presentan que junto con la presentación en lista jerarquizada hace difícil a los usuarios encontrar la información que están buscando.

Por lo anterior la agrupación de documentos web va tomando fuerza, proceso que recibe los resultados de los motores de búsqueda como entrada, crea las agrupaciones y las presenta al usuario. Este proceso se lleva a cabo en cuatro grandes etapas, de la siguiente manera [19]:

1- Adquisición del resumen

Recolección de los resúmenes que son retornados por un motor de búsqueda en respuesta a la consulta del usuario.

2- Pre-procesamiento

Consiste en transformar el texto sin procesar de los resúmenes a una forma adecuada para los algoritmos de agrupación. Esta fase normalmente involucra las siguientes técnicas de procesamiento:

- **Filtración:** Se eliminan las secuencias de caracteres que podrían introducir ruido (por ejemplo, %, #, o $) y así afectar la calidad de la agrupación.

- **Tokenización:** Proceso de identificación de palabras y límites de la frase de un texto. El tokenizer mas simple puede usar caracteres de espacio en blanco como delimitadores de palabras y seleccionar signos de puntuación como: ".", ",", "?" y "!" como delimitadores de sentencias.
  Sin embargo, en aplicaciones practicas, este enfoque simplista no suele ser lo suficiente robusto para manejar contenido "difícil" como resúmenes de documentos. Técnicas más elaboradas de tokenización toman en cuenta algunas características de texto adicionales como guiones o etiquetas HTML.

- **Stemming o División:** Todas las palabras de un texto son reemplazadas por su respectiva raíz. Una raíz es una parte de una palabra que queda después de la eliminación de sus afijos (es decir, sufijos y prefijos). Así, con el uso de un stemmer (Abreviatura de un algoritmo de división) diferentes formas gramaticales de una palabra pueden ser reducidos a una forma base.

- **Remoción de palabras vacías:** Proceso de identificación y eliminación de las palabras vacías de un texto. Las palabras vacías también conocidas como palabras de función (entre ellas los artículos, adjetivos y adverbios), son palabras que por si mismas no tienen ningún significado identificable y por lo tanto son de poca utilidad en algunas tareas de procesamiento de textos.

- **Reconocimiento del lenguaje:** Es importante para realizar el proceso de remoción de palabras vacías, el stemming y la corrección ortográfica. Algunas

de las técnicas que permiten realizar esta tarea son la gestión de trigramas y el análisis general de frecuencias en n-gramas del texto.

3- Selección de características

El objetivo de esta fase es identificar las palabras en el texto que no son informativas y que se pueden omitir durante el proceso de agrupación de documentos web. Las razones para usar la selección de características son: primero, en la mayoría de los casos limitando el número de características se incrementa considerablemente la eficiencia en cuanto al tiempo del algoritmo de agrupamiento. Segundo, puede ayudar a remover el ruido del texto, lo que puede mejorar la precisión del proceso de agrupación.

Dentro de las estrategias de selección de características se cuenta con:

- **Frecuencia del documento (DF):** Elige sólo las palabras presentes en más de un determinado número de documentos.
- **Fuerza del término (TS):** TS es computada por pares de documentos para los cuales la medida de similitud excede un umbral predefinido. Para un término en particular, TS es la probabilidad condicional de que éste término ocurra en un documento, dado que el mismo ocurra en el otro documento. Debido a que las probabilidades deben calcularse por todos los pares de documentos posibles, la complejidad computacional de TS es cuadrática con respecto al número de documentos.
- **Contribución del término (TC):** Una desventaja de DF es que favorece a términos que tienen una gran frecuencia de ocurrencia, sin considerar la distribución de los términos entre diferentes clases. TC mejora este problema agregando la contribución del término para documentar similitud.

4- Clustering

Existen varios modelos de representación del documento, entre ellos: El modelo espacio vectorial, frases frecuentes y n-gramas. El algoritmo propuesto en esta

investigación se baso en el modelo de espacio vectorial, en donde todo documento en la colección es representado por una matriz de N-documentos (filas) por M-términos (columnas). Cada componente horizontal de la matriz (vector) refleja una palabra o término clave particular conectado con el documento actual. El valor de cada componente depende del grado de relación entre el término asociado y el documento respectivo.

Cada documento se representa como un vector fila d en el espacio de términos tal que d= {w_1, w_2, …, w_M}, donde w_i es igual a la frecuencia del término (tf_{ij}) normalizado en la colección multiplicado por la inversa de la frecuencia del documento para ese término, en lo que se conoce como el valor TF-IDF que se resume en la fórmula (1) o una variación de la misma.

$$w_i = \frac{tf_{ij}}{\max(tf_{ij})} \times \log\left(\frac{N}{n_i}\right) \quad (1)$$

De donde, tf_{ij} denota cuantas veces el término i aparece en el documento j, N representa el total de documentos en la colección y n_i denota el número de documentos donde el término i aparece.

Adicionalmente en este modelo se usa la distancia de cósenos para medir el grado de similitud entre dos documentos o entre un documento y la consulta del usuario, calculado por la formula (2).

$$Sim\ (d, q) = \frac{\sum_{i=1}^{M} W_{i,d} \times W_{i,q}}{\sqrt{\sum_{i=1}^{M} W^{2}_{i,d}} \sqrt{\sum_{i=1}^{M} W^{2}_{i,q}}} \quad (2)$$

De donde, $W_{i,d}$ representa el grado de relación entre el término i y el documento d, $W_{i,q}$ representa el grado de relación entre el término i y la consulta del usuario q y M representa el número de términos.

Los valores de similitud de coseno van de 0.0, en caso de que no haya similitud entre d y q a 1 en caso de una estricta similitud entre ellos.

Para obtener buenos resultados de agrupación de documentos web se identifican algunos requerimientos claves que los algoritmos deben tener en cuenta [20]:

- El método debe producir agrupaciones que contengan los documentos relevantes separados de los irrelevantes a partir de la consulta del usuario.
- Proporcionar descripciones concisas y precisas de los grupos, que le permitan al usuario una inspección rápida.
- Un mismo documento puede estar presente en varios grupos puesto que los documentos tienen varios temas.
- Producir agrupaciones de alta calidad a pesar de que solo se tenga acceso a los resúmenes retornados de los motores de búsqueda.
- Ser capaz de agrupar una gran cantidad de resúmenes en pocos segundos.
- Para ahorrar tiempo, el método debe empezar a procesar cada resumen tan pronto como lo recibe desde la web.

2.2.1 Trabajos relacionados con Agrupación de documentos web

En [3] se estudian los meta buscadores que realizan agrupación de documentos web, conocidos como Motores de Agrupación Web (Web Clustering Engines), los cuales combinan las mejores características de la búsqueda basados en consultas y en categorías, el usuario realiza una consulta, el meta buscador envía la consulta a diferentes buscadores, generalmente motores de búsqueda, y los resultados obtenidos los organiza en grupos con etiquetas. Dentro de este grupo se encuentran Clusty (http://clusty.com), Carrot (www.carrot2.org) y SnakeT (http://snaket.di.unipi.it).

Los algoritmos más usados para la agrupación de documentos web han sido los jerárquicos y particionales, los algoritmos jerárquicos generan un dendograma o árbol de grupos, el cual parte de medidas de similitud (single link, complete link y average link), el algoritmo jerárquico que reporta mejor precisión se llama UPGMA, propuesto en 1990, se basa en el modelo de espacio vectorial y usa el enlace

promedio basado en la distancia de cosenos, dentro de sus desventajas esta su complejidad $O(n^3)$ y ser estático en la asignación de documentos en las agrupaciones [21].

Los algoritmos particionales generan una división inicial de los datos en agrupaciones y luego mueven los objetos de un grupo a otro basado en la optimización de un criterio predefinido. Dentro de los algoritmos más representativos de este grupo están: k-means, k-medois y Expectation Maximization. K-means presenta complejidad O(n), donde n es el número de registros, sus desventajas son: sensibilidad a valores atípicos, sensibilidad a selección de centroides iniciales, definición previa del número de grupos a obtener y sólo contempla formas de agrupación aproximadamente esféricas. En el algoritmo K-medois, cada agrupación queda representada en uno de los objetos que lo componen, al que se le llama “centroide real”, las agrupaciones serán subconjuntos de objetos que rodean este centroide. Dentro de las ventajas de K-medois esta que no presenta limitaciones en los tipos de datos y que es menos sensible a la presencia de valores atípicos. En la investigación desarrollada se implementa el algoritmo k-means debido a que éste se emplea en el proceso de optimización de las soluciones existentes y no en la construcción de una nueva solución, para dicho proceso de optimización k-medois no representa una buena opción porque en el proceso desarrollado por dicho algoritmo está presente en gran medida el componente de aleatoriedad. La deficiencia de los valores atípicos a los cuales es susceptible k-means se afronta mediante el pre-procesamiento aplicado a los documentos recuperados, en donde se eliminan términos que podrían representar un valor atípico. El algoritmo Expectation Maximization (EM), usa un refinamiento iterativo que asigna cada objeto a un grupo, según un peso que representa la probabilidad de admisión del objeto en el grupo [21].

Suffix Tree Clustering (STC), es un algoritmo que se basa en la identificación de frases comunes en los grupos de documentos, define una agrupación base para cada uno de los documentos que comparten frases comunes. STC consta de tres

pasos lógicos: limpieza de documento, identificación de grupos base mediante un árbol de sufijos y combinación de grupos base en grupos finales [20]. Dentro de sus desventajas esta: la poda de frases de alta calidad que tiende a eliminar, dejando las menos informativas y cortas; solo incluye documentos que contengan las frases de la colección, dejando de lado algún documento pertinente [21].

Bisecting K-means es un algoritmo que combina los métodos jerárquicos y particionales, reporta mejores resultados en cuanto a exactitud y eficiencia que UPGMA y K-means, los datos inicialmente forman parte de un solo grupo, luego basado en una regla selecciona la agrupación y la divide en dos usando K-means, el proceso se repite hasta obtener los grupos deseados, dentro de sus desventajas esta: no asigna etiquetas adecuadas a los grupos, se debe definir previamente el numero de agrupaciones y no maneja adecuadamente la alta dimensionalidad de las colecciones de documentos [21].

SHOC es un algoritmo que incluye los conceptos de: frases completas y la definición de agrupaciones continuas, por lo cual los documentos pueden pertenecer a varias agrupaciones, utiliza una estructura de datos llamada arreglo de sufijos (en lugar del árbol de sufijos de STC) para identificar frases completas, su frecuencia en tiempo es O(n), donde n es el tamaño de la colección, dentro de sus desventajas es que existen pocos detalles sobre los valores de los umbrales del algoritmo y poca claridad en las etiquetas de los grupos resultantes [21].

Lingo es un algoritmo usado por el buscador carrot2 y se basa en frases completas y LSI (Indexado Semántico Latente), Lingo es una mejora de SHOC y STC, y a diferencia del resto de los algoritmos primero intenta descubrir nombres descriptivos para las agrupaciones y solo después organiza los documentos dentro de los grupos, Lingo intenta descubrir cualquier relación implícita existente en la colección y define el número de grupos (k) que se deben formar. Una desventaja de este algoritmo es que la fase de separación de temas normalmente

requiere transformaciones algebraicas que demoran mucho tiempo de cómputo [21].

En 2009 [22] fue propuesto un algoritmo que usa la estructura hiperlink para encontrar unidades densas mejorando el proceso de unión para la creación de agrupaciones jerárquicas de documentos web, esta propuesta tiene la ventaja de crear varias agrupaciones de diferentes formas, con gran precisión, y remueve el ruido en los datos, para el proceso de unión usa una medida determinada que permite definir automáticamente la cantidad de agrupaciones. Los resultados experimentales muestran mejor calidad de las agrupaciones que otros algoritmos de agrupación basados en densidad. En este mismo año es propuesto un algoritmo basado en k-means y redes neuronales [23], hace uso de análisis de competencias principales (PCA por sus siglas en Inglés), para reducir la dimensionalidad de la matriz de documentos, SVD para encontrar la medida de similitud y redes neuronales multicapas para reducir el tiempo del proceso de agrupación, el algoritmo es probado con diferentes conjuntos de páginas web y sus resultados son satisfactorios.

ArteCM es un algoritmo propuesto en 2009 [24] hace uso de un enfoque incremental para la agrupación de documentos, permite el aumento de numero de agrupaciones de forma adaptativa, es comparado con K-means y SOM y muestra resultados satisfactorios en la calidad y rapidez del proceso de agrupación, una de sus limitantes es que el éxito del proceso depende en gran medida de la fijación de los parámetros. WDC Grc, es un método basado en computación granular [25] que transforma la matriz de términos por documentos (TF-IDF) a matriz binaria granular por documento, posteriormente hace uso del algoritmo de reglas de asociación para obtener conjuntos de palabras frecuentes entre los documentos, la investigación arroja buenos resultados en la calidad de las agrupaciones. Sin embargo no hace uso de los estándares de datos de referencia y no se realizan comparaciones con otros algoritmos del estado del arte.

KEYSRC es propuesto en 2009 [26] se basa en frases claves que se extraen de un árbol de sufijos generalizado construido a partir de los resultados de la búsqueda, posteriormente hace uso de un algoritmos jerárquico para el proceso de agrupación, también se propone una nueva medida para la evaluación de la recuperación de los subtemas llamada SSLk actualmente es una medida clave para evaluar los motores de agrupación web..

En el año 2010 es propuesto el algoritmo de agrupación de documentos relacional (RED Clustering) [4] que tiene en cuenta el contenido de los datos y la estructura de los híper vínculos de las páginas web, los resultados experimentales muestran que supera los algoritmos K-mean y Expectation Maximization en cuanto a eficiencia y pureza de los resultados. En este mismo año también se propone un nuevo enfoque basado en la detección automática del sentido de las palabras [27] que busca solucionar el problema de ambigüedad en la consulta por medio de los directorios web y la recuperación de información semántica (SIR), hace uso de un algoritmo basado en grafos donde los resultados de la búsqueda web se agrupan según la similitud semántica del sentido de las palabras, las pruebas se realizan con los datasets Ambient y Moresque, mostrando mejores resultados que STC, Lingo y KEYSRC.

Un estudio de los problemas de agrupación de documentos web [28] muestra que un algoritmo estocástico de optimización discreta puede aproximar a una solución óptima para el problema de búsqueda de resultados de agrupación SRC, se propone el algoritmo OPTIMSRC obteniendo mejores resultados que lingo, lingo 3G y KEYSRC, para el proceso de etiquetas usa las utilizadas por el algoritmo STC o Lingo.

2.3 HÍPER HEURÍSTICAS

La palabra heurística proviene de “heuriskein” que traduce encontrar o descubrir, generalmente estos métodos se usan cuando no se garantiza una buena solución de los métodos algorítmicos, las heurísticas son aplicadas a problemas de

optimización, pero no garantizan un grado de certeza predeterminado en cuanto a la calidad de las soluciones o el uso de recursos, estos métodos han permitido encontrar soluciones a problemas específicos pero difícilmente un método heurístico se puede aplicar a otro tipo de problema, razón por la cual surgen las meta heurísticas que se pueden describir como una estructura algorítmica general que puede ser empleada para resolver diferentes problemas de optimización con un número pequeño de modificaciones para ser aplicado a un problema específico, dentro de estas se encuentran: recocido simulado, búsqueda tabú, algoritmos genéticos entre otros [29, 30].
Los métodos heurísticos se clasifican en dos grupos:

Métodos constructivos: se caracteriza por construir una solución a partir de un problema dado, la forma de construir la solución depende de la estrategia empleada. Las estrategias más comunes son

- Voraz: se parte de una semilla, se construye paso a paso la solución factible, en cada paso se añade el elemento que representa el mejor avance para la consecución de la solución final.

- Descomposición: se divide el problema en problemas más pequeños, este proceso se aplica de forma recursiva hasta obtener un problema “fácil” de resolver, al final se combinan las soluciones obtenidas para generar la solución general

- Reducción: se identifican características comunes en las buenas soluciones conocidas y se asume que la optima solución también posee dicha característica, de ésta forma se reduce el espacio de búsqueda

Métodos de búsqueda: parten de una solución inicial y tratan de mejorarla

- Búsqueda local 1: examina la vecindad de la solución actual y selecciona el primer movimiento que permite una mejora en la solución.

- Búsqueda local 2: examina la vecindad de la solución actual y selecciona el mejor movimiento de todos los posibles movimientos a realizar, dicho movimiento es el que más le aporta a la solución final.

- Estrategia aleatoria: selecciona aleatoriamente soluciones vecinas a su vecindad [31]

El término híper heurístico puede ser observado como cualquier proceso heurístico donde se administren diversas heurísticas con el fin de resolver un problema de optimización concreto [29].

En el estudio de híper heurísticas de Burke, se lleva a cabo un recuento histórico en torno a los trabajos realizados antes de que el término “híper heurística” fuera concebido como tal a principios del año 2000 para referirse a la idea de “heurísticas que eligen heurísticas”. Al mismo tiempo el autor propone una clasificación general de las híper heurísticas de acuerdo a la naturaleza del espacio de búsqueda heurístico y a la fuente de retroalimentación durante el aprendizaje. Finalmente, se presentan los siguientes enfoques híper heurísticos: heurísticas que eligen heurísticas basadas en heurísticas constructivas, heurísticas que eligen heurísticas basadas en heurísticas de perturbación y población basada en híper heurísticas de perturbación [9].

Las híper heurísticas abarcan un conjunto de enfoques con el objetivo de automatizar el diseño y refinar los métodos heurísticos para resolver problemas de búsqueda computacional complejos. La motivación principal detrás de las híper heurísticas es desarrollar algoritmos que son aplicables más genéricamente que muchas de las actuales implementaciones de algoritmos de búsqueda. La característica más importante de las híper heurísticas es que éstas operan en un espacio de búsqueda de heurísticas en vez de hacerlo directamente en el espacio de búsqueda de soluciones de un problema dado. En consecuencia, cuando se

usan híper heurísticas se está intentando encontrar un método o secuencia de heurísticas en una situación dada en vez de tratar resolver el problema directamente. El objetivo de las híper heurísticas es producir métodos genéricos, que puedan producir soluciones de aceptable calidad, basadas en un conjunto de heurísticas de bajo nivel fáciles de implementar [9].

2.3.1 Trabajos relacionados con Híper Heurísticas

En la literatura se encontraron varios trabajos desarrollados que utilizan el enfoque híper heurístico para dar solución a diferentes problemas de optimización, sin embargo, sólo se encontró un trabajo aplicado al problema de agrupación de documentos web [16], donde se plantea un enfoque híper heurístico preliminar para el agrupamiento de documentos web, pero como se menciona posteriormente tiene varias deficiencias que se superaron con la presente investigación.

En [10] se presenta un enfoque híper heurístico con un conjunto de heurísticas (heurísticas de coloración de grafos), que emplea el método de búsqueda tabú para buscar la solución al problema de horarios de examen. Se usan cinco heurísticas de bajo nivel las cuales son introducidas por el método de ordenamiento aleatorio, las heurísticas son: SD (Least saturation degree first), CD (Largest color degree first), LD (Largest degree first), LE (Largest enrollment first), LWD (Largest weighted degree first), RO (Random ordering).La investigación muestra buenos resultados empleando como heurística de alto nivel a Tabú search, también es de resaltar que una de las conclusiones arrojada por esta investigación muestra que el desempeño de la híper heurística es mejor cuando se utilizan un número mayor de heurísticas de bajo nivel, una de las falencias de la investigación es que solo usan una heurística de alto nivel (Tabú Search), además se plantea que la investigación debe ser probada en un rango más amplio de problemas de horarios.

En [11] se presenta la investigación de una iteración aleatoria basado en híper heurísticas de grafos (GHH) que produce una colección de secuencia de heurísticas para la construcción de soluciones de diferente calidad, esta secuencia de heurísticas puede ser vista como una hibridación de diferentes heurísticas de coloración de grafos (graph colouring heuristics) que construyen soluciones paso a paso. Basada en esta secuencia se analiza estadísticamente la forma en que las heurísticas de coloración de grafos se hibridan automáticamente. Lo cual representa una nueva dirección en la investigación de híper heurísticas. Los resultados experimentales de la investigación se hacen sobre problemas que hacen referencia a horarios de examen y coloración de grafos. En el enfoque propuesto usan cinco secuencias de heurísticas basadas en grafos las cuales son LD (largest Degree), LWD (Largest Weighted Degree), LE (largest Enrolment), SD (Saturation Degree), CD (Colour Degree) y como heurística de alto nivel se usa Tabú Search.

En [32] se plantea que en los enfoques híper heurísticos las combinaciones heurísticas suelen tomar la forma de una lista de heurísticas de bajo nivel que se aplican de forma secuencial, por lo tanto, proponen una forma de presentación jerárquica para la combinación de heurísticas, lo cual es evaluado en problemas de programación de horarios de examen. Las heurísticas empleadas son las siguientes: Largest degree LD (mayor grado), Largest enrolment LE (mayor alistamiento), Largest weighted degree LWD (mayor grado ponderado), Saturation degree SD (grado de saturación), Highest cost HC (costo más alto). Los resultados de la investigación arrojan una combinación de heurísticas organizadas jerárquicamente, con el fin de aplicarlas al problema planteado.

En [33] se presenta la minería de datos desde un enfoque híper heurístico utilizando clasificación asociativa. En esta investigación se busca determinar la técnica de minería de datos que se debe usar para una situación específica basada en el desempeño de una heurística de bajo nivel. En particular, se busca dar respuesta a preguntas como: ¿qué algoritmo de aprendizaje puede derivar el

conocimiento que podría dirigir la búsqueda con el fin de producir buenas soluciones? Para lograr el objetivo, se comparan tres técnicas asociativas de clasificación: CBA (classification based on association), MCAR (multi-class classification based on association rule), MMAC (Multi-class, multi-label associative classification) y dos técnicas populares de clasificación tradicional, PART y Ripper (Repeated incremental pruning to produce error reduction algorithm) en data sets generados usando una híper heurística híbrida llamada Peckish que combina greedy y enfoques aleatorios, para el problema del entrenador de programación. Como resultado de las pruebas se muestra un mejor desempeño en las técnicas de clasificación asociativa MCAR, MMAC, CBA sobre los árboles de decisión C4.5, la regla de inducción RIPPER y el algoritmo PART, con referencia a la exactitud de predecir el conjunto apropiado de heurísticas de bajo nivel.

En [14] se proponen dos tipos de implementaciones de híper heurísticas, la primera se basa en construcción de heurísticas y la segunda usa métodos de mejora (Heurísticas Greedy), las cuales evalúan cada heurística y eligen la que tiene el mejor desempeño dentro del conjunto, dentro de esta segunda clase se diseña un marco general para el uso de procedimientos de búsqueda local y meta heurísticas como heurísticas de bajo nivel. También se concibe un esquema dinámico para guiar el uso de estos métodos, estas ideas se ponen a prueba en un problema de programación conocido como el problema de la variabilidad del tiempo de respuesta. La investigación muestra que se obtienen mejores resultados en un enfoque híper heurístico cuando se usan algoritmos sofisticados de bajo nivel como las meta heurísticas, con el inconveniente de que aumenta el tiempo de cálculo que estas requieren, para dar solución a esto se plantea usar estrategias de selección basadas en desempeño.

En [15], se plantea un enfoque híper heurístico (HHA) basado en meta heurísticas aplicado a la optimización del lanzamiento aéreo de un vehículo satelital (ASLV), para esto se propone una función aleatoria sin aprendizaje que controla las meta

heurísticas de bajo nivel. Las heurísticas de bajo nivel son algoritmos genéticos (AG), Optimización de Enjambre de Partículas (PSO) y Recocido Simulado (SA). La metodología que se propone muestra un mejor rendimiento en cuanto a diversidad y convergencia, y puede ser aplicada a diferentes problemas de optimización. Uno de los inconvenientes de la investigación es que solo usan como heurística de alto nivel selección aleatoria.

En [13] se plantea sustituir las tareas de selección de heurísticas de bajo nivel y la evaluación de la función objetivo por la aplicación de técnicas de minería de datos, específicamente con redes neuronales y regresión logística binaria. Con este cambio se propone una híper heurística más eficiente en cuanto a tiempo de procesamiento. Además, las técnicas anteriormente mencionadas son capaces de encontrar patrones globales escondidos en grandes conjuntos de datos con el fin de clasificar adecuadamente los datos. Finalmente, con el entrenamiento de las reglas de clasificación o los parámetros estimados, el rendimiento de la solución que resulta de la búsqueda híper heurística se puede predecir sin la necesidad de emprender las tareas computacionales implícitas en los enfoques híper heurísticos tradicionales. La evaluación del enfoque se realiza por medio de la generación de un gráfico basado en la híper heurística propuesta para los problemas de horarios de examen, los resultados muestran que la red neuronal y la regresión logística pueden acelerar el proceso de búsqueda de manera significativa.

En [16] se plantea el desarrollo del algoritmo HHWDC, el cual aplica un enfoque híper heurístico que usa siete meta heurísticas de bajo nivel las cuales son búsqueda armónica, búsqueda armónica mejorada, nueva búsqueda armónica, mejor búsqueda armónica global, algoritmo genético con emparejamiento restrictivo, algoritmo genético con selección por ruleta y optimización por enjambre de partículas. HHWDC hace uso del algoritmo k-means como estrategia para buscar la mejor solución local y se basa en el criterio de información bayesiana para definir automáticamente el número de agrupaciones, hace uso de dos estrategias para aceptación y reemplazo las cuales son reemplazo del peor y

reemplazo por competición restringida. Este trabajo es el primero en el campo de la agrupación de documentos web, muestra la viabilidad del enfoque pero por ser exploratorio tiene varias desventajas, entre ellas, el limitado uso de heurísticas de: alto nivel, bajo nivel y reemplazo. Además los experimentos se realizaron sobre un reducido conjunto de datos de prueba y no se realizaron experimentos con usuarios. Adicionalmente a lo propuesto en HHWDC, en la presente investigación se hizo uso de una mayor cantidad de heurísticas de alto y bajo nivel y reemplazo, todo lo anterior, con el objetivo de armar un algoritmo específico que encuentra los mejores resultados en la agrupación de documentos web, que finalmente fue validado en un conjunto más amplio de datos de prueba y por usuarios.

Capítulo 3

3 ENTORNO HÍPERHEURÍSTICO

El framework Híper heurístico es un algoritmo con aprendizaje en línea que combina heurísticas pre-existentes de una manera iterativa para buscar las mejores soluciones. El framework hace uso del criterio de información bayesiano balanceado (BBIC) [34] expresado por (3) para comparar dos soluciones y decidir cuál de ellas es mejor (es la función de aptitud o fitness) [35, 36].

$$BBIC = n * Ln\left(\frac{SSE}{n \times ADBC}\right) + k * Ln(n) \quad (3)$$

Donde n es el número total de documentos, k es el número de grupos, SSE es la suma del error al cuadrado expresado por la fórmula (5), y ADBC es la distancia promedio entre centroides expresada por la fórmula (4).

$$ADBC = \frac{2}{n \times (n-1)} \sum_{i=1}^{k-1} \sum_{j=i+1}^{K} \left(1 - SimCos(C_i, C_j)\right) \quad (4)$$

Donde SimCos es la similitud de cosenos definida por la formula (2).

El framework hace uso de heurísticas de alto nivel para controlar inteligentemente el uso de un número de heurísticas de bajo nivel sobre un problema real de optimización, en este caso, agrupación de documentos web. En el framework el algoritmo ejecuta varias islas (cada una de ellas puede evolucionar independientemente) y selecciona la mejor solución. En cada isla, primero una población o conjunto de soluciones es generada aleatoriamente (RG), luego, cada una de las soluciones es optimizada por K-means, después, trata de seleccionar la heurística apropiada en cada una de las iteraciones para generar un nuevo vector solución (cromosoma, solución armónica, vector, partícula, fuente de alimento o agente), se optimiza la solución actual con K-means y finalmente, se decide si la solución actual debe remplazar a un vector solución en la población. La **Figura 1** y

Figura 2 muestran los diferentes componentes del framework y explican el flujo de información entre esos componentes. En el framework se utilizan veinticinco heurísticas de bajo nivel, denominadas: Búsqueda Armónica (HS), Búsqueda Armónica Mejorada (IH), Nueva Búsqueda Armónica Global (NH), Mejor Búsqueda Armónica Global (BH), Optimización por Enjambre de Partículas (PS), Evolución Diferencial (ED), Colonia Artificial de Abejas (CA), y 18 heurísticas basadas en algoritmos genéticos resultantes de las combinaciones de los métodos de 1) selección: Emparejamiento Restringido (RM), Selección por Ruleta (RW) y selección Rank (RK); 2) cruce: en un punto (UP), uniforme (CU) y multi-punto (CM); 3) mutación: en un bit (MO) y multi-bit (MM). Al mismo tiempo cada una de las heurísticas excepto CA son combinadas con los cuatro métodos de reemplazo: Rank (RR), reemplazo del peor (WR), reemplazo por competencia restringida (RC), reemplazo estocástico (SR), obteniendo un total de 97 heurísticas de bajo nivel.

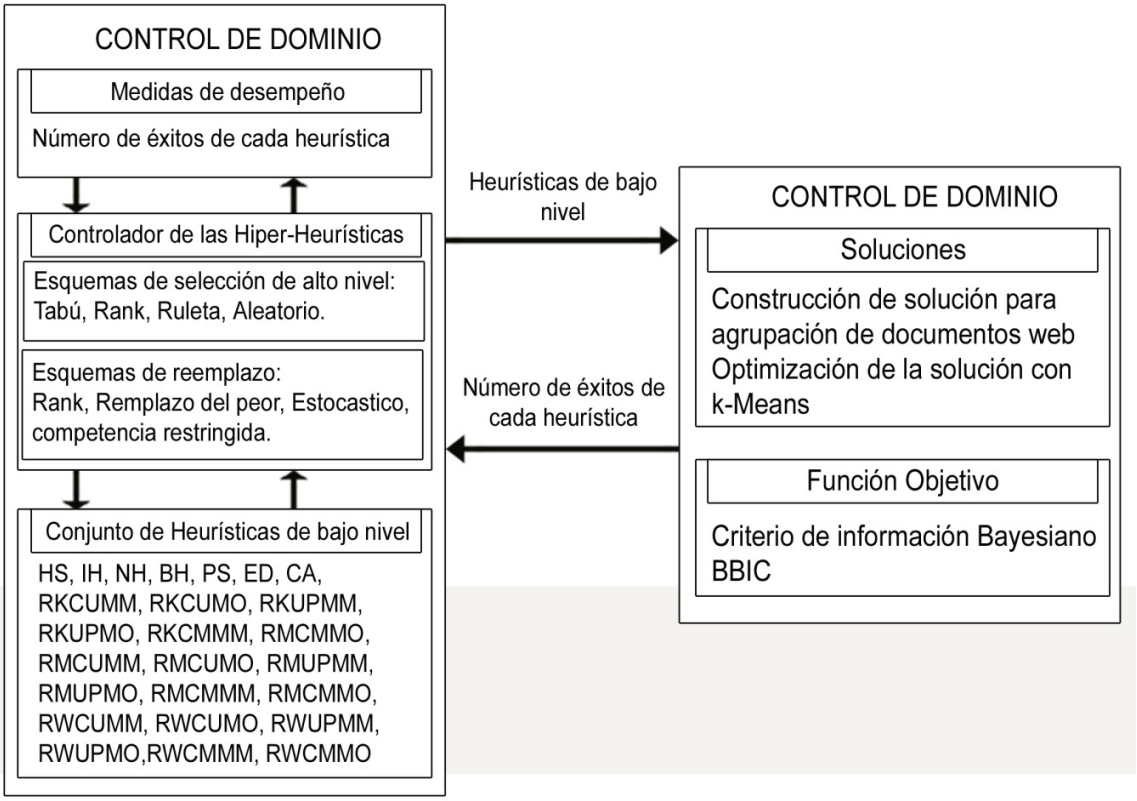

Figura 1. Esquema del framework híper heurístico

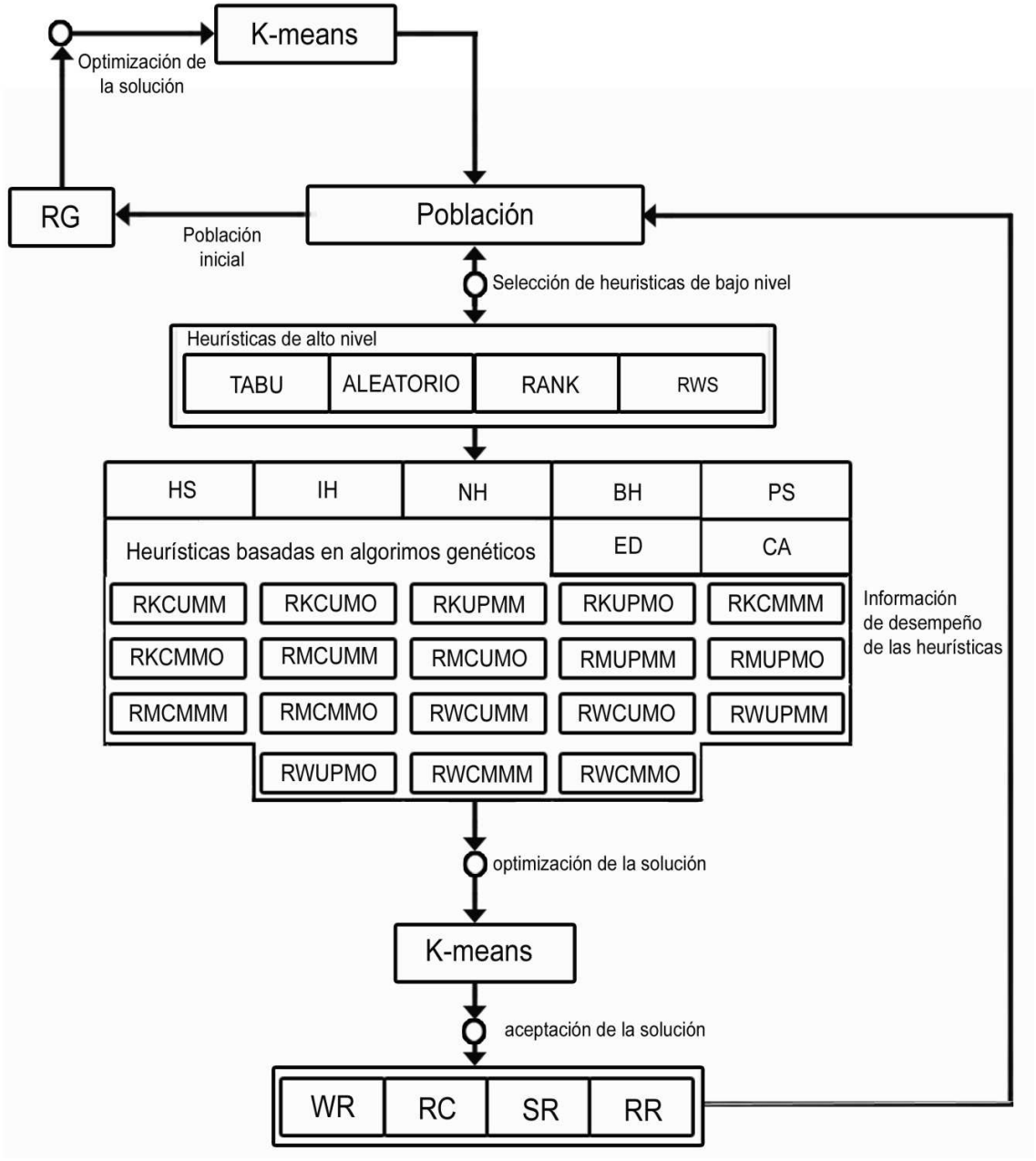

Figura 2. Diagrama general de pasos ejecutados por el framework

3.1 ALGORITMO K-MEANS

El algoritmo k-means es el más simple y más comúnmente usado para agrupación, empleando el criterio de la suma del error al cuadrado basado en (5).

$$SSE = \sum_{j=1}^{k} \sum_{i=1}^{n} P_{i,j} \left\| x_i - c_j \right\|^2 \quad (5)$$

Donde x_i es un documento (punto, tupla, objeto o registro) de la colección de datos (o documentos), c_j es el centroide del cluster j y $P_{i,j}$ es 1 si el objeto xi pertenece al cluster j o cero en otro caso.

Este algoritmo es popular porque encuentra el mínimo (o máximo) local en un espacio de búsqueda, es fácil de implementar y su tiempo de complejidad es O(n). Desafortunadamente, la calidad de los resultados depende de los puntos iniciales y puede converger a un mínimo local si la partición inicial no es elegida adecuadamente [4, 37-39].

Las entradas de K-means son: El número de grupos (K) y un conjunto (tabla, vector o colección) que contiene n objetos (o registros) en un espacio de características de D dimensiones, formalmente definido por $X = \{x_1, x_2, \ldots, x_n\}$(En nuestro caso, x_i es un vector fila, por razones de implementación). Las salidas de K-means son un conjunto que contiene K centroides (denotado por c_j). Los pasos del procedimiento de k-means pueden ser resumidos como se muestra en la **Figura 3**.

```
01  Selección de la partición inicial (k centroides)
02  Repeat
03          Re calcular membrecía
04          Actualizar centroides
05  Until (Criterio de parada)
06  Return Solución
```

Figura 3. Algoritmo k-means

En el paso 01, existen muchos enfoques para seleccionar los centroides iniciales K [40], por ejemplo Forgy (nuestra estrategia) que sugiere seleccionar aleatoriamente instancias de K para el conjunto de datos y McQueen que sugiere

seleccionar los primeros puntos K en el conjunto de datos como semillas preliminares y luego usando una estrategia incremental para actualizar y seleccionar los centroides reales K de la solución inicial [40].

En el paso 02, es necesario recalcular membrecías de acuerdo a la solución actual. Se pueden usar varias similitudes o mediciones de distancia. En esta investigación se hace uso de la distancia de cosenos definida por (2) tomado de [41].

A continuación en la **Figura 4** se hace una descripción algorítmica general del framework. Luego se detallan los pasos más importantes de dicha figura.

Framework híper heurístico:

01 Inicialización de parámetros
02 Pre-procesamiento de los documentos: Remoción de palabras vacías, Algoritmo Porter (stemming), eliminar documentos con contenido vacío, construcción de los resúmenes y creación de la matriz de términos por documento (TDM).
03 Inicialización de la BMR (memoria de los mejores resultados) y llamar a la rutina HHK.
For each i Є [1, BMRS] do
BMR[i] = HHK (TDM)
Next-for
04 Seleccionar el mejor resultado
05 Asignar las etiquetas a los grupos
06 Solapamiento de grupos

Rutina HHK:

01 Inicializar población: Definir centroides (estrategia Forgy), ejecutar k-means (pasos 02-06, mirar Figura 5) y calcular el fitness (BBIC) para cada vector solución generado en la población.

02 Repeat

03 Generar un Nuevo vector solución: usando selección aleatoria, selección tabú, selección Rank o selección basada en desempeño, para seleccionar la heurística de bajo nivel (HS, IH, NH, BH, PS, ED, CA y las heurísticas basadas en algoritmos genéticos resultantes de las combinaciones de los métodos de: 1) selección: emparejamiento restringido (RM), ruleta (RW) y Rank (RK) ; 2) cruce: en un punto (UP), uniforme (CU) y multi-punto (CM); 3) mutación: de un bit (MO) y multi-bit (MM)), estas 25 heurísticas excepto CA son combinadas con los cuatro métodos de reemplazo: Reemplazo por competencia restringida (RC), Reemplazo estocástico (SR), Reemplazo rank (RR) and reemplazo del peor (WR)) para generar una nueva solución.

04 Ejecuta k-means (pasos 02-06, ver Figura 5) y calcula el fitness (BBIC) para el nuevo vector solución.

05 Actualizar población: La solución compite contra otra solución para entrar en la población. Existen cuatro estrategias de reemplazo: Reemplazar el peor (WR), Reemplazo estocástico (SR), reemplazo Rank (RR) y reemplazo por competencia restringida (RC).

06 Hasta que la condición de parada se cumpla: por ejemplo, el máximo número de generaciones (NG) o el máximo tiempo de ejecuciones es alcanzado.

07 Selecciona la mejor solución de la población y lo retorna al framework híper heurístico.

Figura 4. Framework híper heurístico y rutina HHK

01: **Inicializar parámetros del algoritmo:** en esta investigación, el problema de optimización consiste en minimizar el criterio BBIC, llamado función Fitness. El framework necesita algunos parámetros específicos, el tamaño de la memoria de los mejores resultados (BMRS), tamaños de la población (PS), número de

iteraciones (NI) o máximo tiempo de ejecución (MET), y otros parámetros de las heurísticas de bajo nivel (los valores comunes para cada parámetro son mostrados en paréntesis): Tasa de consideración de la memoria armónica (HCMR, 0.95), tasa de ajuste de tono (PAR, entre 0.3 y 0.99), tamaño del grupo de selección (SG, 25% del tamaño de la población), para competencia restringida: tamaños del grupo de remplazo (RGS, 25% del tamaño de la población), para reemplazo por competencia restringida: tasa de mutación (MR, entre 0.2% y 0.5%), para operadores de mutación: mínimo de ancho de banda (MinB, 0.0005) y máximo de ancho de banda (MaxB, 0.005). Para la heurística ABC: Probabilidad de abeja empleada (PEB, 10%) y probabilidad de exploración aleatoria (EPR, 40 %). Para evolución diferencial: Factor de mutación (F, 0.50%) y probabilidad de recombinación (RC, 20%). Para la heurística PSO: Parámetros de escala social (C2, 1.49445) y partícula de inercia mínima y máxima (Wmax y Wmin) [42-45].

02: **Pre-procesamiento de los documentos:** Inicialmente, lucene (http://lucene.apache.org) hace uso de una etapa de pre-procesamiento de los documentos, que incluye: eliminar palabras vacías, stemming del texto usando el algoritmo de Porter [41], eliminar documentos con contenido vacío (resumen procesado) y la creación de la matriz de términos por documentos (TDM). TDM es la estructura más ampliamente utilizada para representación de documentos en RI, y es basada en modelo de espacio vectorial [41, 46]. En esta modelo, los documentos son diseñados como bolsa de palabras, la colección de documentos es representado por una matriz de N-documentos por D-términos, cada documento es representado por un vector de frecuencia de términos normalizado (tfi) por el inverso de la frecuencia del documento, que se conoce como el TF-IDF (expresado por la ecuación (1)), y la distancia de cosenos (mirar ecuación (2)) es usada para medir el grado de similitud entre los documentos o entre el documento y la consulta del usuario

03: **Inicializar la memoria de los mejores resultados y llamar a la rutina HHK:** La memoria de los mejores resultados (BMR) es una estructura de memoria donde

se almacenan los mejores vectores solución (ver **Figura 5**). Cada fila en la BMR almacena los resultados de una llamada a la rutina de k-means híper heurística (HHK, ver sección 3.2), en un ciclo básico. Cada fila en la BMR tiene dos partes: Los centroides y el valor de fitness de ese vector. El número de centroides en cada vector fila en la BMR puede ser diferente.

$$BMR = \begin{bmatrix} Centroids_1 & Fitness_1 \\ Centroids_2 & Fitness_2 \\ \text{M} & \text{M} \\ Centroids_{BMRS-1} & Fitness_{BMRS-1} \\ Centroids_{BMRS} & Fitness_{BMRS} \end{bmatrix}$$

Figura 5. Memoria de los mejores resultados

04: **Seleccionar el mejor resultado:** Encontrar y seleccionar el mejor resultado de la memoria de los mejores resultados (BMR). El mejor resultado es la fila con el valor de fitness más bajo (minimización de f(x)). Luego retornar esta fila como la mejor solución para el problema de agrupación de documentos web (centroides y fitness).

05: **Asignación de etiquetas a los grupos:** El framework híper heurístico contempla de asignación de etiquetas para cada grupo basados en frases frecuentes (FPH) similar a Lingo [47]. Ver sección 3.6.

06: **Solapamiento de grupos:** Finalmente, cada grupo incluye documentos que pueden estar también en otros grupos, si estos documentos están a una distancia menor o igual a la distancia media del grupo.

3.2 RUTINA HHK

En la rutina HHK, cada vector utilizado tiene un diferente número de grupos (centroides), y la función objetivo (BBIC) depende de la posición de los centroides en cada vector solución y del número de centroides (valor de K). La rutina HHK funciona de la siguiente manera:

01: **Inicializar población:** La población es una estructura de memoria donde todos los vectores solución son almacenados. Cada vector solución es creado con un número aleatorio de centroides (k < Kmax), Posiciones iniciales de centroides (K centroides con estrategia Forgy y valores en todas las dimensiones) y fitness para esta solución. Los centroides iniciales son seleccionados aleatoriamente del dataset original. Luego, el algoritmo k-means (**Figura 3** pasos 02 a 06) es ejecutado y son calculados los valores de fitness para esta solución. La estructura general de la población es similar a BMR.

En resumen, PS vector de soluciones (centroides) son generados y luego se calcula el valor de fitness para cada vector. Inicialmente, cada centroide corresponde a un documento diferente de la matriz TDM seleccionado aleatoriamente (estrategia Forgy en el algoritmo k-means). El número inicial de grupos k_i, es calculado aleatoriamente desde 2 hasta kmax (incluido), donde K es un número natural y kmax es el límite superior del número de grupos y se toma como $\sqrt{N} + 1$, (donde N es el número total de documentos en la matriz TDM, pero este valor no puede ser demasiado grande ni demasiado pequeño), lo que significa que es una regla de oro usada en muchas investigaciones sobre el problema de agrupación de documentos web en la literatura.

03: **Generar un Nuevo vector solución:** Un nuevo vector solución (centroides) es generado basado en una de las heurísticas de alto nivel: selección Rank, selección tabú, selección aleatoria o selección basada en desempeño, usada para seleccionar a la heurística de bajo nivel. Ver sección 3.3 y 3.4.

05: **Actualizar población:** La solución compite contra otra solución en la población para entrar a la población. Hay cuatro estrategias de reemplazo: reemplazo del peor (WR), reemplazo por competencia restringida (RC), reemplazo estocástico (SR) y reemplazo Rank (RR). Ver sección 3.5.

3.3 HEURÍSTICAS DE ALTO NIVEL

Encargadas de seleccionar la heurística de bajo nivel que se debe ejecutar. Dentro del entorno híper heurístico se encuentran:

3.3.1 Selección por ruleta

Estrategia inspirada en los muestreos de "Rueda de la ruleta" y "Stochastic universal" [48]. Esta estrategia selecciona una nueva heurística de bajo nivel basada en su número de éxitos previos. Esto es llevado a cabo con el operador de selección basado en la rueda de la ruleta donde las heurísticas con mayor cantidad de éxitos previamente tienen una mayor probabilidad de ser seleccionada de nuevo. La probabilidad de selección $P_{i,j}$ de la heurística de bajo nivel i para la creación de una nueva solución en la iteración j se puede calcular sobre la base de la estimación de Laplace [49], como (8).

$$P_{i,j} = \frac{NSH_i + 1}{TE + NH} \quad (6)$$

Donde NSH_i es el número de éxitos de la heuristica i, TE es el acumulado total de éxitos de las heurísticas y NH es el número de heurísticas.

3.3.2 Selección Rank

La selección Rank fue inicialmente propuesta por Baker para eliminar la alta convergencia que presentan los métodos de selección proporcional (por ejemplo, la selección por ruleta). En este método de selección, las heurísticas son organizadas con base en el número de éxitos en orden descendente , luego se crea la tabla de rank que contiene distintos valores calculados con la formula (7) [8, 48, 49]**.**

$$\frac{vNegativo - vRango * (i * 1.0/(n-1))}{n} \quad (7)$$

Donde vNegativo = 0.25, vPositivo = 1.75, vRango = vPositivo-vNegativo, n = Número de heurísticas, i = 0…n-1.

Posteriormente se organiza la tabla rank de forma descendente, se genera un número aleatorio entre 0 y 1 y se van sumando de uno en uno los valores asignados en la tabla, cuando la suma sea mayor o igual al número aleatorio generado, se selecciona la heurística de esa posición.

3.3.3 Selección aleatoria

Aleatoriamente se selecciona una heurística, sin tener en cuenta el desempeño ni cualquier otro factor [50].

3.3.4 Selección por búsqueda Tabú

Búsqueda tabú es una meta heurística que busca encontrar la solución más óptima de un problema, se caracteriza por el uso de memoria, donde se trata de minimizar en el proceso de búsqueda de la solución el componente aleatorio, dentro de sus características principales se tiene: permite movimientos de empeoramiento, con el fin de escapar a óptimos locales, para evitar la exploración en zonas visitadas anteriormente hace uso de la lista tabú y posee dos tipos de memoria, a corto plazo basado en lo reciente y a largo plazo basado en lo frecuente. Para afrontar el problema de esta investigación la memoria a corto plazo se implementa permitiendo un máximo de heurísticas que pueden ser visitadas, la memoria a largo plazo se implementa por medio de la lista tabú la cual tiene un tamaño equivalente al 18% [51] del total de heurísticas de bajo nivel, cuando una heurística de bajo nivel sobrepasa el máximo de visitas posibles se agrega a la cola de la lista tabú [52]. Ver **Figura 6**.

```
01  Inicializar parámetros tabu: Tamaño_maximo_lista_tabu, Máximo_visitas_permitidas
02  do //Selecciona una de las heurísticas que no este en la lista tabú
03      indice ~ U (1…total de heurísticas);
04  While lista_tabú no contenga heurística[indice]
05      Heurisica[indice].Numero_de_visitas ++
06  //Si la lista tabú está llena
07  If (lista_tabu.tamaño > tamaño_maximo_lista_tabú) then
08      Eliminar primer elemento de la lista tabú
09  End-if
10  //Si se sobrepasa el número máximo de visitas (Memoria a corto plazo)
11  If (Heurisitca[indice].Numero_de_visitas > Maximo_visitas_permitidas) then
12      lista_tabu.Adicionar(Heurisica[index]) //Adicionar a la lista tabu (memoria a largo plazo)
13      Heurística[indice].Numero_de_visitas = 0
14  End-if
15  Algoritmo seleccionado = heurística[indice]
```

Figura 6. Selección Tabú

3.4 HEURÍSTICAS DE BAJO NIVEL

Encargadas de generar un nuevo individuo a partir de la población inicial y con base en diferentes criterios determinar si el nuevo individuo entra o no a reemplazar a otro individuo de la población.

3.4.1 Búsqueda armónica (HS)

Búsqueda armónica es una meta heurística creada para resolver problemas de optimización que está inspirada en el proceso de improvisación musical, esto se debe a que en la música el objetivo es encontrar un estado perfecto de armonía, en este sentido se encuentran similitudes en el esfuerzo requerido para encontrar la armonía en una nueva pieza musical y el proceso de encontrar una buena solución en un problema de optimización, al crear una nueva melodía la calidad de ésta depende en gran medida de la experiencia del músico y el conocimiento previo de armonías existentes, cuando un músico experto desea crear una nueva melodía tiene tres posibles opciones a escoger: 1) tocar una melodía conocida 2) tocar algo similar a la melodía mencionada, ajustando levemente el tono y 3)

realizar una nueva composición con notas aleatorias, estas tres opciones en el proceso de optimización se convierten en: uso de la memoria armónica, realizar ajuste de tono y uso de la aleatoriedad; El uso de la memoria armónica es importante porque esto asegura que las buenas soluciones son consideradas dentro del nuevo vector solución, para tal fin búsqueda armónica adopta el parámetro HMCR que es la tasa de consideración de la memoria armónica, que toma valores entre 0 y 1, si su valor es demasiado bajo pocas armonías son seleccionadas de la memoria armónica y su convergencia sería demasiado lenta, por otra parte si su valor es demasiado alto, cercano a 1, los tonos de la memoria armónica serían los mayormente usados, esto conlleva a que no haya una buena exploración en el espacio de búsqueda por tanto no se genera una buena solución, los valores recomendados en la literatura se encuentran entre [0.7 – 0.95], para esta investigación se toma el valor 0.95 [53, 54].

El ajuste de tono se lleva a cabo por medio de los parámetros: tasa de ajuste de tono PAR y el ancho de banda del tono BW, en términos musicales el ajuste de tono genera un cambio de frecuencia, en el algoritmo esta acción produce un nuevo tono producto de la modificación de un valor aleatorio en un tono, esta acción se muestra en la siguiente ecuación (8):

$$X_{i+1} = X_i + r * bw \quad (8)$$

Donde, X_i es el tono almacenado en la memoria armónica, X_{i+1} es el nuevo tono después de realizar el ajuste y r es un número aleatorio que está entre [-1,1], ésta operación se asemeja a la mutación en los algoritmos genéticos y sólo se realiza si un valor aleatorio es menor a la tasa de ajuste de tono (PAR), si su valor es muy bajo y el ancho de banda (bw) muy estrecho retrasa la convergencia de HS debido a que se explora solo una parte del espacio de búsqueda, por otro lado un valor alto en los parámetros tasa de ajuste de tono (PAR) y ancho de banda (bw), genera dispersión en la búsqueda de la solución, acercándose a una búsqueda aleatoria. Los valores recomendados en la literatura para la tasa de ajuste de tono (PAR) están entre [0.1 – 0.5], para esta investigación se toma el valor 0.35

La aleatoriedad permite aumentar la diversidad de las soluciones, esto permite la exploración de soluciones diversas con el fin de encontrar el óptimo global. A continuación se presenta el algoritmo.

```
For i=1 to K (Total de centroides) do
   If U (0, 1) ≤ HMCR then
   Begin /*Consideración de la memoria*/
      j ~ U (1… PS);
      p ~ U (1… Población [j].K) //Selección de centroide
      New [i] = Población [j].Centroide[p]
      If U(0,1) ≤ PAR then
      Begin /*Ajuste de tono*/
         For j=1 to D (Número de dimensiones) do
            New [i] = New[i] ± BW
         Next-For
      End-if
   Else /*Selección aleatoria – estrategia Forgy*/
      j ~ U (1… N);
      New [i] = TDM[j]
   End-if
Next-for
```

Figura 7. Búsqueda armónica (HS)

3.4.2 Búsqueda Armónica Mejorada (IH)

Como su nombre lo indica, esta es una mejora del algoritmo búsqueda armónica (HS), en donde se observa que una buena solución está sujeta a una buena fijación de los parámetros tasa de ajuste de tono (PAR) y el ancho de banda (bw), es por esto que Mahdavi en 2007 propone el algoritmo búsqueda armónica mejorada (IHS) [44], su factor diferenciador de búsqueda armónica (HS) es que determina de forma dinámica el ajuste de los parámetros PAR y bw, que se determinan de con las formulas (9) y (10).

$$PAR(gn) = PAR_{min} + \frac{(PAR_{max} - PAR_{min})}{NI} \times gn \quad (9)$$

Donde PAR es la tasa de ajuste de tono para cada generación, PAR_{min} es el ajuste mínimo de la tasa de tono, PAR_{max} es el ajuste máximo de la tasa de tono, NI es el número de iteraciones, y gn es la iteración actual.

$$bw(gn) = bw_{max} \exp(c \times gn) \quad (10)$$

Donde $bw(gn)$ es el ancho de banda para cada generación, bw_{min} es el mínimo ancho de banda, y bw_{max} es el máximo ancho de banda.

$$c = \frac{Ln\left(\frac{bw_{min}}{bw_{max}}\right)}{NI} \quad (11)$$

Dónde bw es el ancho de banda para cada generación, bw_{min} es el ancho de banda mínimo y bw_{max} es el ancho de banda máximo.

Éstas modificaciones hacen que el algoritmo mejore en cuanto a precisión y convergencia [55].

3.4.3 Optimización mediante enjambre de partículas (PS)

Es una meta heurística desarrollada por Kennedy y Eberhart en 1995, es conocido como un algoritmo bio-inspirado debido a que se basa en el comportamiento social de las bandadas de peces y pájaros, donde las relaciones sociales de los individuos del grupo permite la existencia de un líder que es reconocido y seguido por el resto del grupo, dentro de las funciones del líder están orientar actividades como la búsqueda de alimento o desplazarse a otras zonas cuando el alimento se termina, si en la población surge un individuo con mejores características que el líder, este lo sustituye en el mando del grupo, en el algoritmo esto se realiza por medio de la evaluación de la función objetivo, todos los miembros del grupo en su recorrido tratan de buscar fuentes de alimento y comunicar al resto del grupo la ubicación que deben tomar, el individuo con mejores resultados en la función objetivo es escogido como líder, todos los individuos de la población tendrán que asemejar su solución (posición) hasta estar prácticamente en la misma localización [56]. Cada partícula actualiza su posición y su velocidad de la siguiente manera:

$$v_{t+1}^{i} = \omega_t * v_t^i + c_1 * R_1 * \left(p_t^i - x_t^i\right) + c_2 * R_2 * (p_t^g - x_t^i) \quad (12)$$

$$x_{t+1}^{i} = x_t^i + v_{t+1}^i \quad (13)$$

La ecuación (12) representa la actualización del vector velocidad de cada partícula *i* en cada iteración k; $c_1 * Rand_1 * \left(p_k^i - x_k^i\right)$ representa el componente cognitivo que cuantifica el desempeño de la partícula i con respecto a su desempeño pasado (memoria individual), en esta investigación c_1 toma el valor de cero, debido a que no hay una evolución del vector solución [16], por otra parte $c_2 * Rand_2 * (p_k^g - x_k^i)$ representa el componente social, su resultado muestra la distancia entre la posición actual y la mejor del grupo de individuos, es decir la decisión que tomará la partícula según la influencia que el resto de cúmulo ejerce sobre ella [57], las variables usadas son: v_k^i: Representa la velocidad de la partícula i en la iteración k, ω: Factor inercial, c_1, c_2: Ratios de aprendizaje (pesos), controlan los componentes cognitivo y social, $Rand_1$, $Rand_2$: Números aleatorios entre cero y uno, x_k^i: Posición de la partícula i en la iteración k, p_k^i: Mejor posición (solución) encontrada por la partícula i hasta el momento, p_k^g: Posición de la partícula con el mejor fitness en toda la población.

La ecuación (13) modela el movimiento de la partícula i en cada iteración k, a continuación se presenta el pseudocódigo del algoritmo implementado [16, 58] **Figura 8**.

```
01  Definir Parámetros: velocidad, C2, wt
02      i ~ U (1… PS)
03      New = población[i]
04  For (i = 1 to k (Número de centroides)) do
05      p ~ U (1… población[Mejor].k)
06      For (j = 1 to D, (Número de dimensiones)) do
07              r ~ U(1… PS)
08          Velocidad=wt*velocidad+C2*r*( población[mejor].Centroide[p][j]-New[i][j])
09          New[i][j] = New[i][j] + velocidad
10      End-for
11  End-for
```

Figura 8. Optimización mediante enjambre de partículas (PS)

3.4.4 Nueva Búsqueda Armónica Global (NH)

Este algoritmo combina los mejores componentes de la optimización mediante enjambre de partículas (PS) y búsqueda armónica (HS), incluye dos operaciones importantes: actualización de la posición y probabilidad de mutación genética, la primera operación permite que la peor armonía de la memoria armónica se mueva a la mejor armonía global en cada iteración, debido a que en PSO cada individuo está inclinado a imitar a su compañero exitoso, y la segunda característica evita caer en óptimos locales [59].

```
01  Definir Parámetros: HCMR, PAR
02  For (i = 1 hasta k (Número de centroides)) do
03      Mejor ~ U(1…población[Mejor Solución].k)
04      Peor ~ U(1…población[Peor Solución].k)
05      For (j = 1 to D (número de dimensiones)) do
06              Alto = población [Mejor Solución]. Centroides[Mejor][j]
07              Bajo = población [Peor Solución]. Centroides[Peor][j]
08              X = 2 * Alto - Bajo
09              If (X < 0) then
10                      X = 0
11              End-if
12              r = U(0…1)
13              New[i][j] = Bajo + r * (X - Bajo)
14              If U(0,1)<PM then
15                      P ~U(1… N)
16                      New [i][j] = TDM [j][p]
17              End-if
18      End-for
19  End-for
```

Figura 9. Nueva búsqueda armónica global (NH)

La variable Mejor representa un centroide aleatorio de la mejor solución en la memoria armónica, del mismo modo la variable Peor representa un cendroide aleatorio de la peor solución almacenada en la memoria armónica, la operación:

$$X = 2 * high - low \quad (14)$$

La ecuación (14) crea un área de confianza, lo cual permite realizar la actualización de la posición, en donde se toman valores más cercanos a la mejor solución almacenada en la memoria armónica [59], dicha región se presenta en la **Figura 10**.

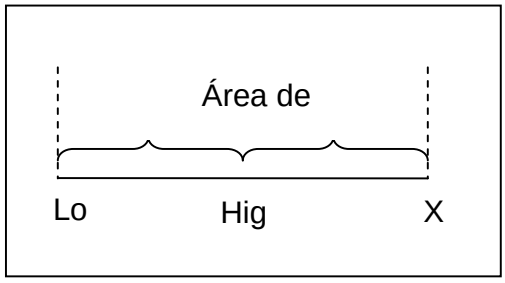

Figura 10. Actualización de posición

3.4.5 Mejor Búsqueda Armónica Global (BH)

El algoritmo mejor búsqueda armónica global, toma como base la meta heurística búsqueda armónica (HS) y modifica la forma en que se realiza el ajuste de tono, implementando la característica de optimización mediante enjambre de partículas (PS) en donde los individuos de la población siguen la solución más óptima almacenada en la memoria armónica. A continuación **Figura 11** se presenta el algoritmo.

```
01  Definir Parámetros: HCMR, PAR
02  For (i = 1 to k (Número de centroides)) do
03      If  U (0,1) <= HCMR then
04              j ~ U (1… PS)
05              p ~ U (1… población[j].k)
06              New [i] = población [j].Centroide[p]
07              If U(0,1) < = PAR then
08                      p ~ U(1… población [mejor][k])
09                      New [i] = población [mejor].Centroide[p]
10              End-if
11      Else
12              j = U (1…N)
13              New [i] = TDM [j]
14      End-if
15  End-if
```

Figura 11. Mejor Búsqueda Armónica Global (BH)

3.4.6 Evolución Diferencial (ED)

Inicialmente son seleccionados de forma aleatoria tres padres diferentes tomados de la población. Se calcula el tamaño del nuevo individuo con base en la fórmula (15) [60]:

$$|Xr1 + F(Xr2 - Xr3)| \quad (15)$$

Donde Xri con i = 1…3 es el número de centroides del padre i y F es un factor real y constante de mutación entre [0,2] que controla la amplificación en la variación diferencial (Xr2-Xr3), en este caso se toma como 0.5 [60].

Por cada centroide del nuevo individuo se genera un número aleatorio entre 0 y 1:

En caso de que el número sea menor a la probabilidad de reproducción o recombinación (CR) la cual equivale al 20%, se le asigna al nuevo individuo un centroide de otro individuo perteneciente a la población original y diferente a los tres padres seleccionados inicialmente. En caso contrario se selecciona un centroide aleatorio de cada uno de los padres y se calculan los atributos del centroide con base en la formula (15).

3.4.7 Colonia de abejas (CA)

Inspirado en [56], se genera un valor aleatorio entre 0 y 1: Cuando este valor es menor o igual a 0.1 se simula la labor de la abeja empleada generando un individuo nuevo creando sus centroides de forma aleatoria. Si el desempeño del nuevo individuo es mejor que el desempeño del peor individuo en la población, el nuevo entra a reemplazar al peor. Si está entre (0.1, 0.4], se elije aleatoriamente un individuo de la población el cual es perturbado creando los centroides en forma aleatoria, si el nuevo individuo presenta un desempeño mejor que el individuo original, el nuevo reemplaza al original. Si es mayor a 0,4, se realiza una explotación de los mejores individuos, aplicando el método de ruleta se elije a un individuo de la población el cual se perturba creando centroides aleatorios y si el desempeño del nuevo individuo es mejor al desempeño del individuo original, el nuevo entra en reemplazo del individuo original.

3.4.8 Heurísticas basadas en algoritmos genéticos

Las heurísticas basadas en algoritmos genéticos son el resultado de la combinación de varios métodos de selección, cruce y mutación como se muestra en la **Figura 12**.

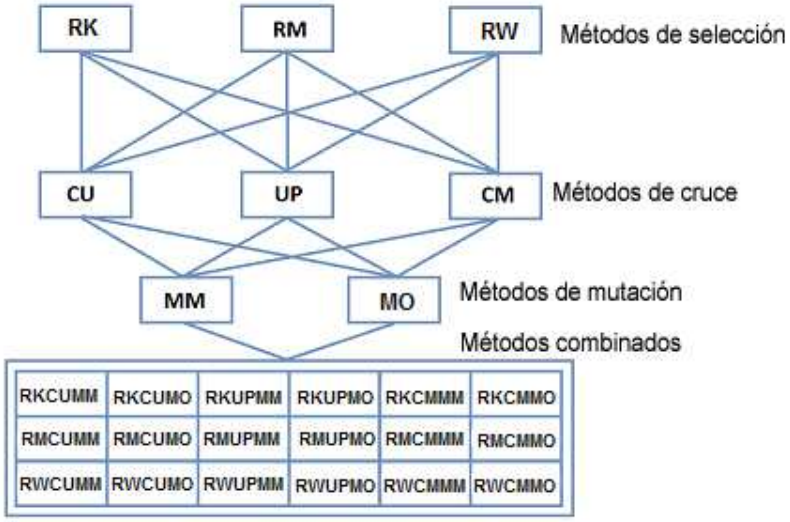

Figura 12. Heurísticas basadas en algoritmos genéticos

Ruleta (RW)

Aplica el mismo proceso visto en la sección 3.3.1 pero en vez de heurísticas se seleccionan individuos (padres) con base en su fitness.

Rank (RK)

Los individuos de la población son organizados desde el de menor desempeño hasta el de mayor desempeño, luego los dos padres son seleccionados de la siguiente manera:

Se elige un número aleatorio con base en el total de individuos de la población, por cada individuo de la población se va acumulando el valor obtenido de aplicar la fórmula (7) de la sección 3.3.2 [8, 48, 49]:

Cuando el acumulado sea igual o supere al número aleatorio se selecciona como padre al individuo que este en la posición i mientras no haya sido elegido con anterioridad.

Selección emparejamiento restrictivo (RM)

El primer padre es seleccionado aleatoriamente de la población. Su pareja es elegida del grupo de selección (SGS vector solución seleccionado aleatoriamente de la población) con el número más similar de grupos al primer padre. Si el resultado es un grupo con más de una solución candidata, la similitud de los centros de los grupos (basada en la distancia de coseno) es usada para seleccionar al más similar [16].

Una vez seleccionados los dos padres ya sea usando selección por ruleta, Rank o aleatoria, se lleva a cabo el método de cruce que puede ser:

Cruce uniforme (CU)

Se calcula el tamaño del nuevo individuo generando un valor aleatorio entre el tamaño del padre menor y el tamaño del padre mayor. Posteriormente para cada centroide del nuevo hijo se genera un número aleatorio entre 0 y 1, si es 0 se toma el centroide del padre 1 y si es 1 se toma el centroide del padre 2. Verificando en todo momento que los centroides no se repitan [8].

Cruce en un punto (UP)

Este método genera dos hijos a partir de dos padres, primero se elige un punto de corte para ambos padres, como se esta trabajando por individuos solo se necesita generar a un hijo, de manera que se genera uno de los dos con base en una probabilidad del 50% para cada hijo. El primer hijo estará conformado por los centroides a la izquierda del primer padre y los centroides a la derecha del segundo padre usando como referencia el punto de corte, y el segundo con los centroides a la izquierda del segundo padre y los centroides a la derecha del primer padre usando como referencia el punto de corte [8].

Cruce multipunto (CM)

Se define el total de puntos de cruce entre 1 y el menor tamaño de los dos padres. Luego se tiene una probabilidad de 50% de generar un hijo conformado por los

centroides a la izquierda de los puntos de cruce del primer padre y por los centroides a la derecha del segundo padre o un hijo conformado por los centroides a la derecha de los puntos de cruce del primer padre y por los centroides a la izquierda del segundo padre [61].

Finalmente se lleva a cabo la mutación del nuevo individuo, la cual se realiza con cualquiera de los siguientes métodos de mutación:

Mutación de un bit (MO)

Se selecciona aleatoriamente un centroide del nuevo individuo, y se le modifican algunos de sus atributos sumando o restando un valor que es calculado con la fórmula (1), ésta operación se lleva a cabo, sólo si un número aleatorio es menor a la probabilidad de mutación, que es fijada por el parámetro MR (para ésta investigación MR toma un valor del 0.5%).

$$(BWmax - BWmin) * NumeroAleatorioDouble + BWmin \quad (16)$$

Donde BWmax = 0.005 y BWmin = 0.0005.

Mutación en varios bits (MM)

Por cada uno de los centroides del nuevo individuo se lleva a cabo una modificación sobre los atributos sumando o restando un valor resultado de la fórmula (1), la probabilidad de que se modifique un atributo está dada por el parámetro MR (MR = 0.5% para ésta investigación).

3.5 HEURÍSTICAS DE REEMPLAZO

3.5.1 Rank (RR)

Para la heurística de reemplazo rank los individuos son organizados con base en su valor de desempeño, desde el mejor al peor, luego se crea la tabla de ranking que contiene distintos valores calculados con la formula (7) vista anteriormente [8, 48, 49]**.**

Posteriormente la tabla Rank es organizada de menor a mayor valor, luego se genera un número aleatorio entre 0 y 1 y se van sumando de uno en uno los valores asignados en la tabla, cuando la suma sea mayor o igual al número aleatorio, se selecciona el individuo que se desea remplazar, si el desempeño del nuevo individuo es mejor al desempeño del individuo seleccionado, el nuevo entra en su reemplazo.

3.5.2 Reemplazo del peor (WR)

Si el nuevo individuo generado presenta un mejor desempeño que el peor individuo de la población, el nuevo individuo entra en reemplazo del otro.

3.5.3 Reemplazo por competencia restringida (RC)

Se selecciona un individuo de la población que posea centroides similares a los del nuevo individuo utilizando la distancia de cosenos, luego los dos individuos se ponen a competir y el que tenga un mejor desempeño pasa a formar parte de la población [62].

3.5.4 Reemplazo Estocástico (SR)

Se selecciona al mejor individuo de la población con base en el desempeño y se compara con el nuevo individuo generado, si el nuevo individuo supera al mejor de la población entra a reemplazarlo, en caso contrario se aplica la fórmula (2) para decidir si entra o no a la población [50].

$$e^{-\frac{10*iteracionActual}{NI}} \quad (17)$$

Se genera un número aleatorio entre 0 y 1 si es menor al resultado de aplicar la fórmula el nuevo individuo entra a reemplazar al peor individuo de la población.

3.6 FRASES FRECUENTES (ETIQUETADO)

Este paso corresponde al paso 2 “Extracción de frases frecuentes” en lingo [75] pero en esta investigación este método es usado para cada agrupación generada en los pasos anteriores, su funcionamiento es de la siguiente manera:

1- **Conversión de la representación:** Cada documento en el grupo actual se convierte de representación basado en caracteres a presentación basado en palabras.
2- **Concatenación del documento:** Todos los documentos en el grupo actual son concatenados y un nuevo documento con la versión inversa de la concatenación es creada.
3- **Completar la frase descubierta:** Las frases completas a la derecha y las frases completas a la izquierda son descubiertas en la agrupación actual, luego las frases completas son almacenadas alfabéticamente, y luego las frases frecuentes a la derecha e izquierda son combinadas en un conjunto de frases completas.
4- **Selección final:** Los términos y frases cuya frecuencia excede el umbral de frecuencia de términos son seleccionadas para la agrupación actual.
5- **Construcción de las “Otras” etiquetas y grupos:** Si algunos documentos no llegan al umbral de frecuencia de términos, son enviados al grupo otros.
6- **Inducción de la etiqueta de la agrupación:** En la agrupación actual, una matriz de términos por documento es construida, luego usando similitud de cosenos, es seleccionada la mejor solución candidata por términos o frases para la agrupación (con optimización SSE).

3.7 IMPLEMENTACIÓN DEL FRAMEWORK

3.7.1 ARQUITECTURA GENERAL

El framework esta implementado con base en una arquitectura multi-capa entre las que están: 1) capa de presentación, 2) capa de lógica de negocio, 3) capa de lógica del servicio y 4) capa de datos, ver **Figura 13**. Cada capa tiene una función determinada, la capa de presentación denominada Laboratorio es la encargada de definir la conexión con las bases de datos que hacen parte de la capa de datos:

BDDmoz, BDMoresque, BDOdp239 o BDAmbient, solicitar información del test que se va a desarrollar y finalmente presentar los resultados de la ejecución de dichas evaluaciones en un formulario y guardarlos en archivos de Excel. Si la capa de presentación que está ejecutando el usuario es Minerva, esta aplicación permite ingresar la consulta del usuario en la web, definir las opciones de búsqueda y mostrar los resultados en grupos temáticos con la metáfora de carpetas.

La capa de lógica de negocio se encarga del proceso de agrupación de documentos web que abarca procesos como: generación del modelo espacio vectorial (matriz TDM), uso del algoritmo k-means para optimizar las soluciones, etiquetado de grupos, cálculo de las medidas de evaluación por ejemplo: número de grupos k, medidaF, precisión, recuerdo, medida SSLk, etc

En la capa de lógica de servicios se encuentran servicios como: google, bing y yahoo! utilizados para obtener los resultados de búsquedas en la web, babel que permite el reconocimiento del lenguaje de los documentos, Lucene.Net que facilita el proceso de creación de la matriz TDM con las tareas de tokenizacion, remoción de palabras vacías, stemming en inglés y en español, entre otras, y WordNet que provee una base de datos léxica del idioma inglés, entre otros.

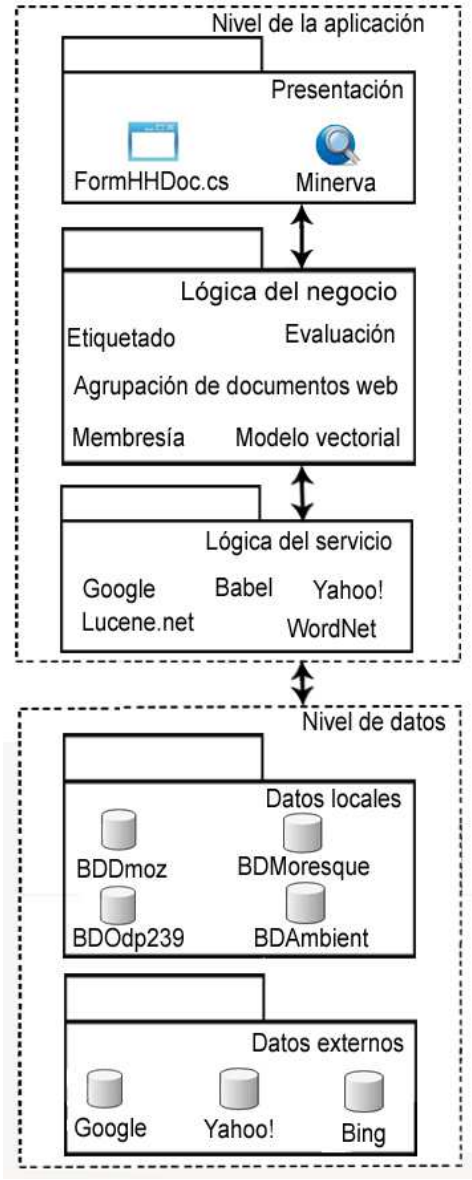

Figura 13. Arquitectura general del framework

3.7.2 DESCRIPCIÓN GENERAL DE CLASES

A continuación se realiza una descripción general de las clases más importantes de la capa de lógica de negocio del framework:

- **HHKmeansConcreto.cs:** Encargada de inicializar los parámetros del algoritmo, como lo son: el número de iteraciones, tamaño de la población. Genera la población aleatoria, hace el llamado a la clase HeuristicaAltoNivel.cs, si la solución generada es mejor a la peor solución de la

memoria de los mejores resultados entra a reemplazarla siempre y cuando no se encuentre una solución igual en el vector (para garantizar diversidad).

- **VectorSolucion.cs:** Vector que almacena los centroides y el fitness asociado a una solución.
- **VectorRG:** Genera una nuevo vector solución de manera aleatoria (creación de centroides de forma aleatoria).
- **HeuristicaAltoNivel.cs:** Aplica la heurística de alto nivel a ejecutar: Rank, Aleatorio, Tabú o RWS (Ruleta), la cual decide que heurística de bajo nivel ejecutar.
- **ControllerReplace.cs:** Selecciona una de las cuatro estrategias de reemplazo a ejecutar para la nueva solución: Reemplazo Rank, reemplazo del peor, reemplazo estocástico o reemplazo por competencia restringida. Cuando la heurística de bajo nivel que se esta ejecutando es ABC, dependiendo del caso realiza un reemplazo del peor o reemplazo por competencia restringida.
- La lógica de cada una de las heurísticas de bajo nivel están en las clases: VectorABC.cs, VectorED.cs, VectorGBHS.cs, vectorHS.cs, VectorIHS.cs, VectorNGHS.cs, VectorPSO.cs,
- Para las heurísticas de bajo nivel basadas en algoritmos genéticos se utilizan para la selección de los padres de nuevo vector csolución las clases: VectorRankS.cs, VectorRS.cs o vectorRWS.cs; para el cruce: CruceMultipunto.cs, CruceUniforme.cs o CruceUnPunto.cs; y para el proceso de mutación: MutacionMultibit.cs o MutacionOneBit.cs.

En la **Figura 14** se pueden observar todas las clases que heredan de la clase VectorSolucion, donde VectorSolución es representa la clase principal.

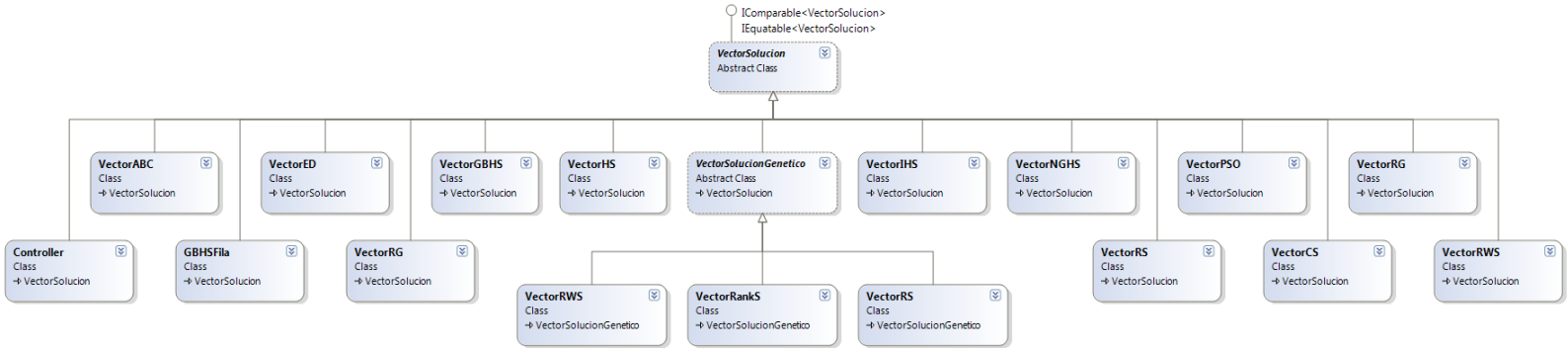

Figura 14. Diagrama de algunas de las clases de la capa de lógica de negocio

3.7.3 CASOS DE USO E INTERFAZ

En el paquete Laboratorio de la aplicación se encuentra la interfaz de usuario del framework híper heurístico que permite realizar las pruebas con cada uno de los datasets: ambient, dmoz, moresque u odp239 **Figura 16**, para esto primero se debe elegir el conjunto de datos sobre el cual se va a realizar la prueba, luego el sistema realiza el proceso y presenta los resultados, en la **Figura 15** se pueden ver los casos de uso.

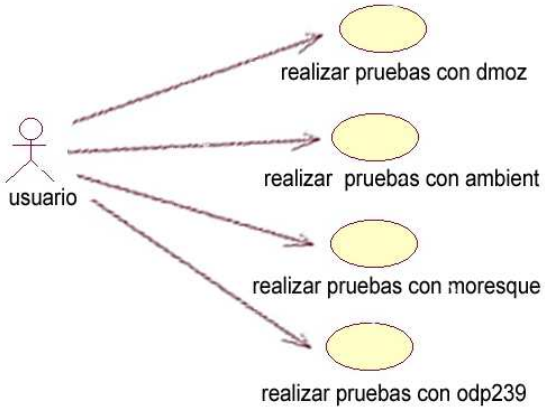

Figura 15. Casos de uso de la interacción del usuario con el sistema

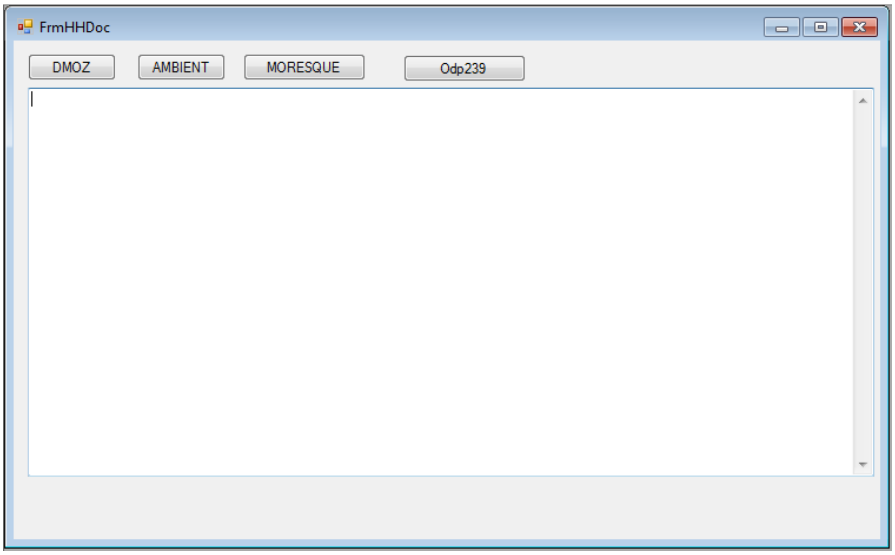

Figura 16. Ventana para procesar test en cada base de datos

En la **Figura 17** se presentan los casos de uso de la interacción del usuario con la interfaz web minerva, a través de la cual una vez el usuario se haya registrado **Figura 19** e iniciado sesión **Figura 18**, puede asignar los parámetros del algoritmo **Figura 20** y realizar la consulta sobre el meta buscador web introduciendo el texto a buscar para obtener los resultados (documentos arrojados por los motores de búsqueda: google, yahoo! y bing) agrupados y ordenados por tópicos **Figura 21**.

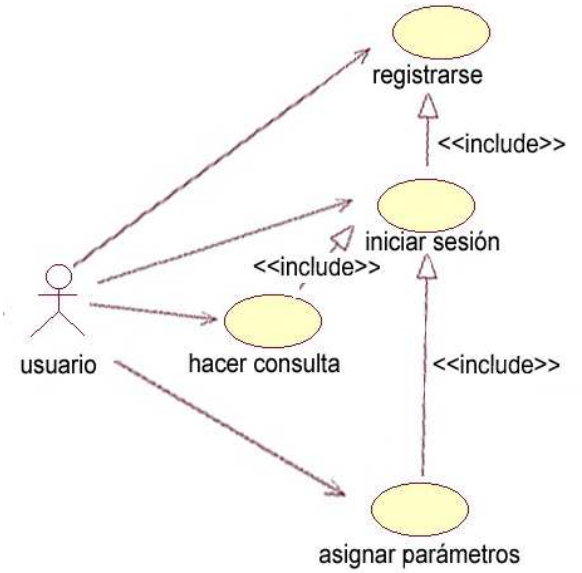

Figura 17. Casos de uso de la interacción de usuario con la aplicación minerva

Figura 18. Ventana de inicio de sesión

Figura 19. Ventana para registro de un nuevo usuario

Figura 20. Ventana de asignación de parámetros del algoritmo

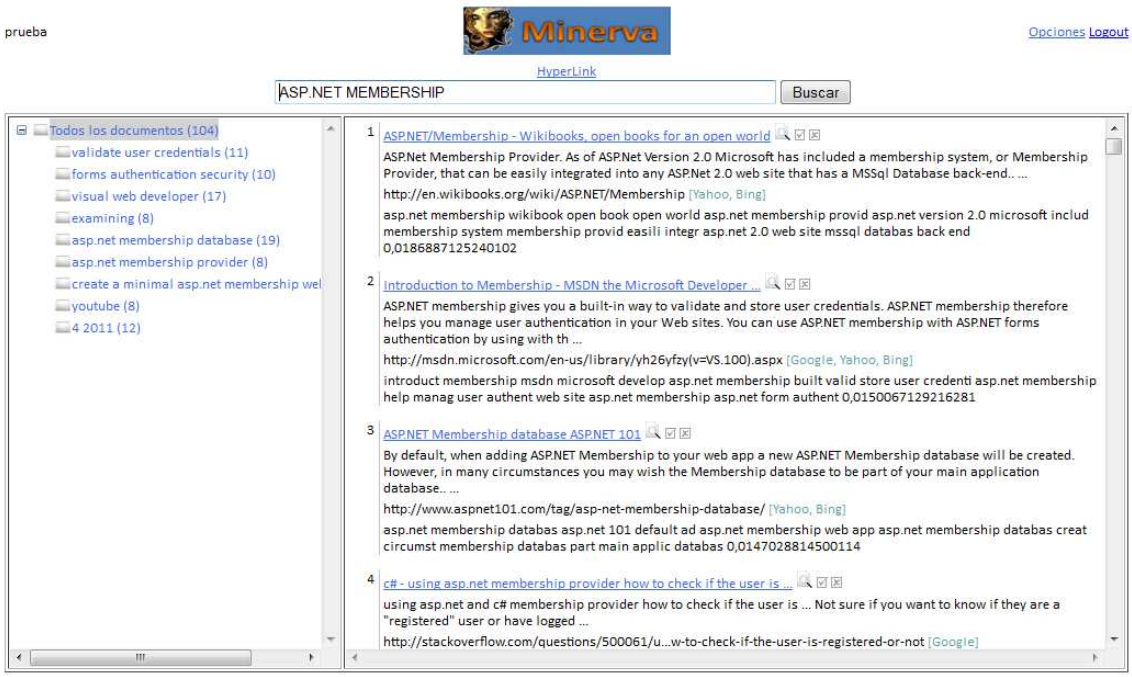

Figura 21. Ventana de resultados

Capítulo 4

4 EXPERIMENTACIÓN

4.1 DATASETS

El framework y la mejor heurística obtenida utilizaron para el agrupamiento de resultados web cuatro conjuntos de datos tradicionales de referencia, llamados: DMOZ-50, AMBIENT, MORESQUE and ODP-239. En estos datos se encuentran un total de 447 consultas con sus respuestas ideales.

DMOZ-50 conjunto de datos que consta de 50 preguntas derivadas de Open Directory Project (acrónimo de Directorio de Mozilla). Cada consulta tiene un promedio de 129.14 documentos, 6,02 subtemas (significados de muy distintas materias) y 22.62 los resultados pertinentes por subtema recuperado. Cada consulta es una gran colección de documentos, cada uno con un conjunto de clases relativamente pequeño y gran número de documentos por clase. En estos conjuntos de datos, palabras clave de consulta no están disponibles. La colección está disponible para su descarga en http://artemisa.unicauca.edu.co/~ccobos/wdc/wdc.htm.

AMBIENT (AMBIguous ENTries) conjunto de datos consta de 44 consultas extraídas de las entradas ambiguas de Wikipedia. Cada consulta tiene, en promedio, 50,55 resultados recogidos de búsqueda de Yahoo! clasificados (anotados manualmente con juicios relevancia de documento por cada subtema), 7,91 subtemas, y 7.72 resultados relevantes por cada subtema recuperado. La mayoría de las consultas en el conjunto de datos AMBIENT son de una sola palabra (1 palabra clave) y están todas disponibles. El conjunto de datos AMBIENT mide la capacidad de recuperar los subtemas que figuran en los resultados de la búsqueda (documentos recuperados por Yahoo!), no todos los posibles subtemas de una consulta. Este conjunto de datos se puede descargar en http://credo.fub.it/ambient.

MORESQUE (MORE Sense-tagged QUEry results) Es un Conjunto de datos que se compone de 114 preguntas ambiguas que se desarrollan como complemento del dataset AMBIENT. Permite probar el comportamiento de los algoritmos de búsqueda web en consultas de diferentes longitudes, que van de 1 a 4 palabras. MORESQUE ofrece 114 consultas de longitud 2, 3 y 4 (todos ellos disponibles), junto con un promedio del 53,54 de los primeros resultados (documentos) de Yahoo!, 3,82 subtemas y 19.43 resultados relevantes por cada subtema recuperado. Este conjunto de datos se puede descargar en http://lcl.uniroma1.it/moresque.

ODP-239 conjunto de datos que consta de 239 preguntas derivadas de Open Directory Project (http://www.dmoz.org). Cada consulta tiene en promedio 106.95 documentos (cada documento consiste en una URL, título y una breve descripción), 9,56 subtemas y 11,38 resultados relevantes por cada subtema recuperado. ODP-239 se compone de muchas colecciones pequeñas, cada una con un conjunto de clases comparativamente grandes, en lugar de tener una gran colección de documentos con un pequeño número de clases. Los temas, subtemas, y sus documentos asociados se seleccionaron de tal manera que la distribución de documentos a través de subtemas refleja la importancia relativa de subtemas. La colección está disponible para su descarga en http://credo.fub.it/odp239.

4.2 MEDIDAS DE EVALUACIÓN

La evaluación incluye dos puntos de vista: la contrastación contra un ideal (validación por Ground-truth) y la evaluación del comportamiento del usuario (SSL). La validación por Ground-truth está dirigido a evaluar que tan bueno es un método de agrupación con respecto a la recuperación de las agrupaciones conocidas (denominadas clases) de una partición ideal (conocida también como regla de oro). Varias medidas de evaluación están disponibles para esta tarea, incluyendo precisión, recuerdo, medida-F, Fall-out y exactitud (índice de Rand)

[63]. En esta investigación, la precisión ponderada, el recuerdo ponderado y la medida-F ponderada (las medias armónicas de precisión y recuerdo), el fall-out ponderado y la exactitud ponderada se utilizan para evaluar la calidad de la solución.

Dada una colección de agrupaciones, $\{C_1, C_2, \dots C_k\}$, para evaluar su precisión ponderada, recuerdo ponderado y medida-F ponderada con respecto a una colección ideal de agrupaciones $\{C_1^i, C_2^i, \dots C_h^i\}$, se siguen los siguientes pasos: (a) encontrar para cada agrupación ideal C_n^i una agrupación distinta C_m que se aproxime mejor en la colección que está siendo evaluada $P(C, C^i)$, $R(C, C^i)$, y $F(C, C^i)$ tal como se define en(1) y (2). (b) Calcular la precisión ponderada (P), recuerdo ponderado (R) y medida-F ponderada (F) basada en (3).

$$P(C, C^i) = \frac{|C \cap C^i|}{|C|} \; y \; R(C, C^i) = \frac{|C \cap C^i|}{|C^i|} \tag{1}$$

Donde C es una agrupación de documentos y la agrupación C^i es una agrupación ideal de documentos.

$$F(C, C^i) = \frac{2 * P(C, C^i) * R(C, C^i)}{P(C, C^i) + R(C, C^i)} \tag{2}$$

$$P = \frac{1}{T}\sum_{j=1}^{h}|C_j^i| * P(C_m, C_j^i), \; R = \frac{1}{T}\sum_{j=1}^{h}|C_j^i| * R(C_m, C_j^i), \text{ y } \; F = \frac{2*P*R}{P+R} \text{ donde } \; T = \sum_{j=1}^{h}|C_j^i| \tag{3}$$

Respecto de la evaluación del comportamiento de los usuarios, la métrica SSLk (Subtopic Search Length under k document) fue usada para evaluar la facilidad con la que los usuarios pueden utilizar los resultados de la agrupación, en resumen, es la evaluación del comportamiento del usuario respecto a los resultados [64-66]. Esta medida se define como el número promedio de elementos (etiquetas de las agrupaciones o resultados de búsqueda) que deben ser revisados antes de encontrar un número suficiente (k) de documentos relevantes a cualquiera de los subtemas de la consulta, suponiendo que tanto las etiquetas de la agrupación, como de los resultados de búsqueda se leen secuencialmente de arriba a abajo, y sólo las agrupación con las etiquetas correspondientes al sub-tema que interesa se abren. SSL_k permite una evaluación de todos los sub-temas recuperados (la recuperación de varios documentos relacionados con cualquier sub-tema) en lugar de centrarse en todos los sub-temas (es decir, la recuperación

de al menos un documento relevante para algunos subtemas). SSL_k también permite una modelización realista de la conducta de búsqueda de usuario debido a que el papel desempeñado por las etiquetas de los grupos se tiene en cuenta.

4.3 COMPARACIÓN

En la literatura revisada se destacan algunos algoritmos de agrupación de documentos web, para la evaluación de esta investigación se toman los resultados de dichos algoritmos y se comparan con los obtenidos, a continuación se presentan dichos métodos de agrupación:

STC, Suffix Tree Clustering, es un algoritmo propuesto en 1998, toma el enfoque de frases frecuentes compartidas por los documentos, cuenta con tres pasos lógicos: 1. Limpieza del documento 2. Identificación de los grupos base, mediante un árbol de sufijos 3. Combinación de los grupos base en los grupos finales, una de sus mayores ventajas es que usa frases que proporcionan descripciones concisas y significativas de los grupos [20].

Bisecting k-means, algoritmo propuesto en el año 2000, combina la fortaleza de los métodos jerárquicos y particionales, inicialmente los datos son tratados como una sola agrupación, basado en una regla se selecciona una agrupación y esta se dividen en dos agrupaciones con la ayuda del algoritmo k-means, este proceso se repite hasta obtener las agrupaciones deseadas [21].

Lingo, algoritmo propuesto en el año 2003, usado por el buscador carrot2, se basa en frases completas, una de sus características es que primero define nombres descriptivos para sus agrupaciones y posteriormente organiza los documentos en los grupos, su estrategia es extraer las frases frecuentes en los documentos como fuente informativa de descripción de temas en los grupos, luego intenta reducir la matriz de termino por documento, Lingo define el número de grupos k que se deben formar y relaciona las descripciones de los grupos con los documentos [67].

4.4 RESULTADOS Y DISCUSIÓN

Para el análisis de los resultados producto de la realización de las pruebas se emplearon los test de Friedman y Wilcoxon, el primero es una prueba no paramétrica desarrollada por el economista Milton Friedman, ésta prueba es aplicada a situaciones en las que se seleccionan k grupos con n elementos, su resultado demuestra el grado de similitud existente entre los elementos de cada grupo. El test de Wilcoxon igualmente es una prueba no paramétrica empleada para comprobar la media de dos muestras relacionadas y determinar el grado de diferencias entre ellas [68].

4.4.1 Etapa I de pruebas

En esta etapa se llevó a cabo el análisis de desempeño de las heurísticas de bajo nivel en un ambiente donde trabajan juntas para crear la mejor solución al problema de agrupación de documentos web sin tener en cuenta el tiempo de ejecución. Se hace uso de la base de datos Dmoz y la función BIC (19), se evaluó la precisión, medidaF y SSL de las heurísticas de bajo nivel, las cuales fueron probadas individualmente con 20 y 30 iteraciones, obteniendo los siguientes resultados (**Tabla 1** y **Tabla 2**) ordenados con base en la mayor medidaF, mayor precisión y menor SSL.

$$BIC = n * Ln\left(\frac{SSE}{n}\right) + k * Ln(n) \quad (19)$$

Donde n es el número total de documentos, k es el número de agrupaciones y SSE es la suma del error al cuadrado (5).

Heurística	K	Precisión	Recuerdo	MedidaF	SSL1	SSL2	SSL3	SSL4	SUMA SSL
DMOZ-BIC-RWCMMOWR	7,943333333	80,70349354	64,91569475	69,5911873	16,17942275	21,09468704	24,62925979	27,5848746	89,48824418
DMOZ-BIC-RWCMMMWR	8,010666667	80,89823802	64,69762205	69,52672324	16,13175847	20,96830159	24,5974246	27,56104947	89,25853413
DMOZ-BIC-RWCUMMWR	7,190666667	78,66238471	65,93135271	69,47416732	15,83469471	20,88630635	24,64610741	27,72744841	89,09455688
DMOZ-BIC-RWCUMOWR	7,160666667	78,3733953	65,9100329	69,34165725	15,89780899	21,05720529	24,80853492	27,83228677	89,59583598
DMOZ-BIC-RWCMMORK	7,842	80,13608557	64,68043073	69,2059393	16,1570828	21,13901455	24,73620714	27,72420476	89,75650926
DMOZ-BIC-BHWR	6,404666667	75,06739764	68,03377262	69,19890298	16,21581852	21,24920079	24,98802619	27,93176852	90,38481402
DMOZ-BIC-RMCMMOWR	7,835333333	80,08019527	64,72479389	69,19062779	16,34852196	21,31574074	24,94370979	27,87807063	90,48604312
DMOZ-BIC-HSWR	7,248	77,97754912	66,00124585	69,17957218	16,16582857	20,94064921	24,66666534	27,66694312	89,44008624
DMOZ-BIC-RWCMMMRK	7,849333333	80,11553198	64,69142777	69,17299848	16,26025291	21,22952196	24,72155476	27,71465979	89,92598942
DMOZ-BIC-RMCMMMWR	7,832	79,87781566	64,70390723	69,14257894	16,15817646	20,87421958	24,38662593	27,35263757	88,77165952
DMOZ-BIC-RWCUMMRK	7,121333333	77,99507494	65,83850966	69,11085215	16,15803254	21,22829286	24,83468201	27,82582804	90,04683545
DMOZ-BIC-BHRK	6,381333333	74,74464858	67,91196873	69,00174436	16,26345899	21,3846246	24,81531799	27,79385291	90,2572545
DMOZ-BIC-RWUPMMWR	8,642666667	82,25991034	62,9656988	68,94585918	15,96573598	20,77287037	24,49963148	27,46712751	88,70536534
DMOZ-BIC-HSRK	7,216666667	77,5568523	65,84729355	68,92874004	16,07743677	20,94996058	24,4771836	27,45607884	88,96065979
DMOZ-BIC-RWCMMMRC	7,441333333	78,47989828	65,25764305	68,89055693	16,12569392	21,29051455	24,88127566	27,91689233	90,21437646

Tabla 1. Mejores 15 heurísticas aplicando 20 iteraciones

Heurística	K	Precisión	Recuerdo	MedidaF	SSL1	SSL2	SSL3	SSL4	SUMA SSL
DMOZ-BIC-RMCMMMWR	8,211333333	83,2480823	65,74837348	71,16118516	16,61723677	21,44506667	24,97607566	27,85256772	90,89094683
DMOZ-BIC-RWCMMMWR	8,456	84,02555054	65,21260948	71,10132653	16,55789418	21,48348069	25,00310741	27,75758942	90,80207169
DMOZ-BIC-HSWR	7,341333333	80,17679542	67,52172408	71,08829395	16,70054762	21,53831667	24,99673069	27,93524286	91,17083783
DMOZ-BIC-RWCMMOWR	8,423333333	83,98178154	65,16520549	71,05067689	16,56284683	21,48556561	24,84680529	27,77595238	90,67117011
DMOZ-BIC-HSRK	7,300666667	80,04396734	67,54059871	71,03675183	16,57480952	21,29342354	24,80608175	27,59226323	90,26657804
DMOZ-BIC-BHWR	6,356666667	76,58199686	69,8634442	71,00434179	16,77098704	21,73205397	25,26467381	28,12697037	91,89468519
DMOZ-BIC-RWCMMORK	8,270666667	83,42051304	65,32466001	70,91469794	16,41997646	21,27065397	24,70730079	27,45205053	89,84998175
DMOZ-BIC-RWCMMMRK	8,271333333	83,21605863	65,26303356	70,82381258	16,62458069	21,37683598	24,88458783	27,63085185	90,51685635
DMOZ-BIC-RMCMMOWR	8,287333333	83,34155438	65,17767396	70,78221689	16,7528418	21,62358466	24,98691058	27,86594497	91,22928201
DMOZ-BIC-BHRK	6,361333333	76,20202099	69,62369751	70,71268434	16,9141209	21,82242593	25,36205423	28,26537328	92,36397434
DMOZ-BIC-RWCUMMRK	7,194	79,88442508	66,93072499	70,57802939	16,40058492	21,58703571	24,99831455	27,87543413	90,86136931
DMOZ-BIC-RWCUMMWR	7,216666667	79,89620775	66,83894866	70,5254429	16,35902222	21,26268386	24,93398915	27,8461455	90,40184074
DMOZ-BIC-RWCMMMRC	7,788	81,29590543	65,97468055	70,52498597	16,75905397	21,60394868	24,88752698	27,83508677	91,0856164
DMOZ-BIC-RMCMMORK	8,052666667	82,28867233	65,44399118	70,52337885	16,40262937	21,17109206	24,77118492	27,65798995	90,0028963
DMOZ-BIC-RWCUMOWR	7,204666667	79,77517278	66,86090006	70,50342264	16,21874921	21,40157275	25,02361587	27,90984127	90,5537791

Tabla 2. Mejores 15 heurísticas aplicando 30 iteraciones

Las 15 heurísticas de bajo nivel que presentan un mejor comportamiento en cuanto a precisión, medidaF y SSL fueron:

1- **RMCMMMWR** = Selección por emparejamiento restrictivo, cruce multipunto, mutación multibit y reemplazo del peor individuo.
2- **RWCMMMWR** = Selección por ruleta, cruce multipunto, mutación multibit y reemplazo del peor individuo.
3- **HSWR** = Búsqueda armónica y reemplazo del peor individuo
4- **RWCMMOWR** = Selección por ruleta, cruce multipunto, mutación en un bit y reemplazo del peor.
5- **HSRK** = Búsqueda armónica y reemplazo del peor.
6- **BHWR** = Mejor búsqueda armónica y reemplazo del peor.
7- **RWCMMORK** = Selección por ruleta, cruce multipunto, mutación de un bit y reemplazo Rank.
8- **RWCMMMRK** = Selección por ruleta, cruce multipunto, mutación multibit y reemplazo Rank.
9- **RMCMMOWR** = Selección por emparejamiento restrictivo, cruce multipunto, mutación en un bit y reemplazo del peor.
10- **BHRK** = Mejor búsqueda armónica y reemplazo Rank.
11- **RWCUMMRK** = Selección por ruleta, cruce uniforme, mutación multibit y reemplazo Rank.
12- **RWCUMMWR** = Selección por ruleta, cruce uniforme, mutación multibit y reemplazo del peor.
13- **RWCMMMRC** = Selección por ruleta, cruce multipunto, mutación multibit y reemplazo por competencia restringida.
14- **RMCMMORK** = Selección por emparejamiento restrictivo, cruce multipunto, mutación en un bit y reemplazo RANK.
15- **RWCUMOWR** = Selección por ruleta, cruce uniforme, mutación de un bit y reemplazo del peor.

Las 15 heurísticas de bajo nivel fueron probadas sobre los datasets de DMOZ, MORESQUE, AMBIENT y ODP-239 con 380 iteraciones, haciendo uso de las heurísticas de alto nivel: Tabu, Rank, RWS y Aleatorio junto con las funciones BIC y BBIC, en un ambiente donde las heurísticas de bajo nivel compiten entre ellas con base en las veces que ganan, obteniendo mejores resultados con la función BBIC y con las heurísticas de alto nivel: Rank y Tabu en cuanto a las medidaF, como se muestra en la **Tabla 3** con los resultados del test de Friedman considerando una reducción del desempeño (distribuida de acuerdo con el chi-cuadrado con 7 grados de libertad igual a: 462.53132) y valor P computado por el test de Friedman igual a: 1.6677059733183341E-10.

Algorithm	Ranking
BICB-RANK	3.5783
BICB-TABU	3.5078
BICB-ALEATORIO	3.7159
BICB-RWS	3.6868
BIC-RANK	5.2774
BIC-TABU	5.453
BIC-ALEATORIO	5.4016
BIC-RWS	5.3792

Tabla 3. Test de Friedman de las heurísticas de alto nivel y las funciones.

Al aplicar el test de Friedman utilizando el porcentaje del número de veces que ganó cada heurística del conjunto de 15 heurísticas de bajo nivel, junto con TABU como heurística de alto nivel con la función BBIC (**Tabla 3** anterior) y considerando una reducción del desempeño (distribuida de acuerdo con el chi-cuadrado con 14 gados de libertad de: 2,715.192058) y valor P computado por el test de Friedman de: 0.0 se obtuvieron los resultados mostrados en la **Tabla 4**. Con base en el porcentaje de las heurísticas que ganaron (**Tabla 7**) y al test de Friedman (**Tabla 4**), se seleccionaron las 11 heurísticas de bajo nivel: RMCMMMWR, RWCMMMWR, RWCMMOWR, RWCMMORK, RWCMMMRK, RMCMMOWR, RWCUMMRK, RWCUMMWR, RWCMMMRC, RMCMMORK y RWCUMOWR, con las que se repitió el proceso de pruebas junto con las heurísticas de alto nivel: Tabu y Rank.

Algorithm	Ranking
RMCMMMWR	6.9978
RWCMMMWR	6.9441
HSWR	12.755
RWCMMOWR	5.6275
HSRK	12.8501
BHWR	12.5034
RWCMMORK	5.6588
RWCMMMRK	6.9899
RMCMMOWR	5.566
BHRK	12.3859
RWCUMMRK	6.6074
RWCUMMWR	6.8803
RWCMMMRC	6.7852
RMCMMORK	5.1611
RWCUMOWR	6.2875

Tabla 4. Test de Friedman sobre las 15 heurísticas de bajo nivel

Se aplicó el test de Friedman sobre las heurísticas de alto nivel con base en la medidaF obtenidas con 15 y 11 heurísticas de bajo nivel considerando una reducción del desempeño (distribuida de acuerdo con el chi-cuadrado con 3 gados de libertad de: 31.238926) y valor P computado por el test de Friedman de: 7.570819300362075E-7 con el conjunto de 11 heurísticas se puede ver en la **Tabla 5** que se obtuvieron mejores resultados con la heurística de alto nivel RANK.

Algorithm	Ranking
BICB-TABU-11H	2.7517
BICB-RANK-11H	2.5615

Tabla 5. Test de Friedman sobre las heurísticas de alto nivel TABU y RANK

Al aplicar el test de Friedman utilizando el porcentaje del número de veces que ganó cada heurística del conjunto de las 11 heurísticas de bajo nivel, haciendo uso de RANK como heurística de alto nivel con la función BBIC (**Tabla 5**) y considerando una reducción del desempeño (distribuida de acuerdo con el chi-cuadrado con 10 grados de libertad de: 225.340655) y valor P computado por el test de Friedman de: 9.819822732737293E-11 se obtuvieron los resultados

mostrados en la **Tabla 6**. Con base en el porcentaje de las heurísticas que ganaron (**Tabla 7**) y al test de Friedman (**Tabla 6**), se seleccionaron las 5 heurísticas de bajo nivel: RWCMMOWR, RWCMMORK, RMCMMOWR, RMCMMORK y RWCUMOWR y se calculan de nuevo las medidas usando las heurísticas de alto nivel: Tabu y Rank.

Algorithm	Ranking
RMCMMMWR	6.7841
RWCMMMWR	6.3691
RWCMMOWR	5.1275
RWCMMORK	5.2696
RWCMMMRK	7.0414
RMCMMOWR	5.33
RWCUMMRK	6.3926
RWCUMMWR	6.2897
RWCMMMRC	6.821
RMCMMORK	5.1488
RWCUMOWR	5.4262

Tabla 6. Test de Friedman sobre las 11 heurísticas de bajo nivel

En la **Tabla 7** se puede ver el promedio ponderado de los datasets DMOZ, AMBIENT, MORESQUE y ODP-239 de los resultados obtenidos con cada conjunto de 15, 11 y 5 heurísticas, utilizando las heurísticas de alto nivel Rank y Tabu junto con la función BBIC. En cuanto a medidaF se obtuvieron buenos resultados con 15 heurísticas utilizando Rank o Tabu; con 5 heurísticas con Tabu o Rank se lograron mejores resultados que con 11 heurísticas. En cuanto a medida SSL se obtuvieron mejores resultados cuando se utilizaban las 5 heurísticas, seguido del grupo de 15 heurísticas, pero con muy poca diferencia entre las seis.

Algoritmo	15H-TABU-BBIC	15H-RANK-BBIC	5H-RANK-BBIC	5H-TABU-BBIC	11H-RANK-BBIC	11H-TABU-BBIC
K	10,01	10,02	10,35	10,40	10,38	10,42
Tiempo	38,94	34,25	21,54	21,55	19,25	22,72
ICC-Purity	47,96	47,96	47,70	47,66	47,56	47,40
Precision	79,46	79,49	79,67	79,72	79,66	79,77
Recuerdo	47,96	47,96	47,70	47,66	47,56	47,40
MedidaF	**53,92**	**53,92**	53,72	53,67	53,59	53,50
Fall-Out	0,03	0,03	0,03	0,03	0,03	0,03
RI – Accuracy	0,77	0,77	0,77	0,77	0,77	0,77
RMCMMMWR	**7,4%**	**7,5%**			7,6%	7,6%
RWCMMMWR	**7,6%**	**8,0%**			8,3%	7,9%
HSWR	1,0%	0,8%				
RWCMMOWR	**9,4%**	**10,2%**	21,4%	20,7%	**10,6%**	**10,6%**
HSRK	0,7%	0,8%				
BHWR	1,0%	0,9%				
RWCMMORK	**9,8%**	**9,5%**	19,4%	20,7%	**10,6%**	**11,0%**
RWCMMMRK	8,0%	7,4%			7,2%	7,9%
RMCMMOWR	**10,1%**	**10,0%**	19,9%	20,0%	**10,2%**	**10,6%**
BHRK	1,1%	0,8%				
RWCUMMRK	**8,8%**	**8,5%**			8,2%	8,0%
RWCUMMWR	**8,3%**	**8,8%**			8,3%	8,0%
RWCMMMRC	**7,7%**	**7,1%**			7,7%	8,1%
RMCMMORK	**9,7%**	**9,9%**	19,2%	20,5%	**10,6%**	**10,8%**
RWCUMOWR	**9,3%**	**10,0%**	19,7%	17,6%	**10,3%**	**9,3%**
SSL1	18,81	18,77	18,77	18,83	18,90	18,94
SSL2	27,16	27,17	27,08	27,14	27,35	27,29
SSL3	33,93	33,87	33,78	33,84	34,07	34,05
SSL4	41,23	41,20	41,04	41,05	41,21	41,21
SUMA SSL	121,12	121,02	120,66	120,86	121,53	121,49

Tabla 7. Promedio ponderado de los resultados

4.4.2 Etapa II de pruebas

Se llevaron a cabo pruebas sobre los datasets DMOZ, AMBIENT, MORESQUE y ODP-239, para un tiempo aproximado de ejecución de 1 segundo (un escenario real de agrupación de documentos web) para cada una de las heurísticas de bajo nivel, los conjuntos de 15, 11 y 5 heurísticas identificados en la etapa I de pruebas y con base en el desempeño de las heurísticas individuales, la simplicidad del método de reemplazo y la menor diferencia entre el K obtenido y el K ideal (7.54),

se escogió un segundo grupo de 5 heurísticas (BHRK BHWR HSWR HSRK RWCUMMRK), las cuales se probaron con todas las posibles combinaciones: 10 parejas, 10 tríos, 5 cuartetos y 1 quinteto junto con las heurísticas de alto nivel: Rank y Tabu, obteniendo el promedio ponderado de la **Tabla 8**, en donde se muestran los 24 mejores resultados en cuanto a medidaF, de igual manera en la **Tabla 9** se pueden ver los porcentajes de las veces que ganan las heurísticas individuales y en la **Tabla 10** se encuentran los porcentajes de las veces que ganan las heurísticas en un ambiente donde las heurísticas compiten entre ellas (pares, tríos y cuartetos).

Los resultados de las heurísticas combinadas abarcan un mayor número de posiciones que las individuales, dentro de las 24 mejores heurísticas. Sin embargo, la heurística de bajo nivel BH (Global Best Harmony Search) probada con los cuatro métodos de reemplazo (Rank, Reemplazo del peor, Reemplazo estocástico y Competencia restringida) hace parte del ranking de las 24 mejores, siendo la mejor Global Best Harmony Search con reemplazo Rank (BHRK).

Heurística	K	ICC-Purity	Precisión	Recuerdo	Medida F	Fall-Out	RI – Accuracy
BHRK	7,37	49,18	69,27	49,18	52,40	0,05	0,78
Tabu: BHRK-BHWR	7,39	49,16	69,28	49,16	52,38	0,05	0,78
BHWR	7,36	49,16	69,26	49,16	52,37	0,05	0,78
Rank: BHRK BHWR	7,39	49,10	69,29	49,10	52,33	0,05	0,78
Tabu: BHRK BHWR HSRK	7,69	48,66	70,09	48,66	52,27	0,05	0,78
Rank: BHRK BHWR HSRK	7,71	48,66	70,15	48,66	52,24	0,05	0,78
Rank: BHRK BHWR HSWR	7,74	48,59	70,26	48,59	52,23	0,05	0,78
Tabu: BHRK-BHWR-HSWR	7,70	48,63	70,12	48,63	52,22	0,05	0,78
BHSR	7,50	48,81	69,55	48,81	52,19	0,05	0,78
Tabu: BHWR-HSRK	7,83	48,44	70,46	48,44	52,19	0,05	0,77
Tabu: BHRK-HSRK	7,83	48,42	70,48	48,42	52,17	0,05	0,77
Tabu: BHWR-HSWR	7,86	48,40	70,50	48,40	52,16	0,05	0,77
Tabu: BHRK-HSWR	7,85	48,36	70,48	48,36	52,15	0,05	0,77
Tabu: BHRK BHWR HSWR HSRK	7,85	48,39	70,43	48,39	52,13	0,05	0,77
BHRC	7,39	48,89	69,07	48,89	52,13	0,05	0,78
Rank: BHRK BHWR HSWR HSRK	7,87	48,32	70,52	48,32	52,10	0,05	0,77
Tabu: BHWR HSWR HSRK	8,02	48,15	70,95	48,15	52,09	0,05	0,77

Tabu: BHRK BHWR RWCUMMRK	7,84	48,32	70,65	48,32	52,08	0,05	0,77
Tabu: BHRK BHWR HSWR RWCUMMRK	7,94	48,20	70,89	48,20	52,08	0,05	0,77
Rank: BHRK BHWR HSRK RWCUMMRK	8,00	48,07	71,11	48,07	52,05	0,05	0,77
Tabu: BHRK BHWR HSRK RWCUMMRK	7,93	48,16	70,86	48,16	52,04	0,05	0,77
Rank: BHWR HSWR	8,11	48,01	71,08	48,01	52,04	0,05	0,77
Rank: BHWR HSWR HSRK	8,07	48,07	71,00	48,07	52,03	0,05	0,77
Rank: BHRK BHWR HSWR RWCUMMRK	8,00	48,04	71,07	48,04	52,01	0,05	0,77

Tabla 8. Promedio ponderado de los mejores resultados

Heurística	% de veces ganadas
BHRK	92,0%
BHWR	93,0%
BHSR	91,5%
BHRC	91,3%

Tabla 9. Porcentaje de veces ganado por cada una de las heurísticas

Heurística	HSWR	HSRK	BHWR	BHRK	RWCUMMRK
Tabu: BHRK-BHWR			46,2%	46,1%	
Rank: BHRK BHWR			67,6%	25,0%	
Tabu: BHRK BHWR HSRK		32,2%	29,5%	31,1%	
Rank: BHRK BHWR HSRK		34,5%	31,7%	26,1%	
Rank: BHRK BHWR HSWR	36,3%		31,0%	25,5%	
Tabu: BHRK-BHWR-HSWR	33,4%		29,3%	30,5%	
Tabu: BHWR-HSRK		47,9%	44,4%		
Tabu: BHRK-HSRK		48,0%		44,8%	
Tabu: BHWR-HSWR	48,4%		44,2%		
Tabu: BHRK-HSWR	48,5%			44,2%	
Tabu: BHRK BHWR HSWR HSRK	24,2%	24,5%	21,4%	22,5%	
Rank: BHRK BHWR HSWR HSRK	25,6%	23,1%	20,9%	22,7%	
Tabu: BHWR HSWR HSRK	31,5%	31,0%	29,9%		
Tabu: BHRK BHWR RWCUMMRK			16,7%	17,4%	61,1%
Tabu: BHRK BHWR HSWR RWCUMMRK	15,9%		14,4%	14,8%	50,0%
Rank: BHRK BHWR HSRK RWCUMMRK		15,4%	13,3%	13,4%	53,2%
Tabu: BHRK BHWR HSRK RWCUMMRK		16,3%	14,4%	14,6%	49,5%
Rank: BHWR HSWR			22,9%	69,7%	
Rank: BHWR HSWR HSRK	32,6%	32,7%	27,3%		
Rank: BHRK BHWR HSWR RWCUMMRK	15,6%		12,8%	13,7%	53,1%

Tabla 10. Porcentaje de veces que gano cada heurística combinada

Con base en la medida SSL las heurísticas individuales están dentro de las mejores 24 posiciones, siendo la heurística Global Best Harmony Search la mejor con los cuatro métodos de reemplazo: Rank, Competencia restringida, Estocástico y Reemplazo del peor, manteniéndose en la primera posición BHRK (**Tabla 11**).

Algoritmo	SSL1	SSL2	SSL3	SSL4	SUMA SSL
BHRK	16,55	25,41	32,92	40,89	**115,78**
BHRC	16,55	25,49	33,00	40,97	116,02
BHSR	16,63	25,58	33,13	41,11	116,44
BHWR	16,71	25,64	33,09	41,00	116,44
IHRC	16,12	25,35	33,38	41,69	116,55
IHSR	16,21	25,47	33,36	41,63	116,66
IHRK	16,21	25,45	33,39	41,68	116,73
IHWR	16,22	25,46	33,37	41,68	116,73
EDWR	16,33	25,46	33,34	41,74	116,88
NHWR	16,23	25,46	33,40	41,81	116,90
HSRK	16,85	25,71	33,20	41,16	116,92
RMCUMMRC	16,77	25,70	33,25	41,25	116,98
HSRC	16,76	25,65	33,29	41,30	117,01
RMCUMORK	16,75	25,72	33,24	41,34	117,05
RMCUMORC	16,74	25,70	33,28	41,35	117,06
NHRK	16,25	25,45	33,49	41,91	117,09
RMCUMOWR	16,83	25,71	33,29	41,28	117,10
HSSR	16,76	25,69	33,33	41,35	117,12
RWCUMMWR	16,87	25,76	33,27	41,22	117,13
RMCUMMSR	16,75	25,72	33,27	41,39	117,13
HSWR	16,98	25,77	33,24	41,15	117,14
PSRC	16,11	25,49	33,58	42,00	117,17
RWCUMMRC	16,85	25,76	33,28	41,28	117,18
RWCUMMRK	16,88	25,81	33,27	41,23	117,19

Tabla 11. Medida SSL

Comparando a la mejor heurística de bajo nivel contra otros algoritmos (**Tabla 12**) tenemos que en el dataset DMOZ-50, WDC-HH-BHRK es mejor en todas las medidas de evaluación excepto en precisión a los algoritmo Bisecting Kmeans, Lingo y STC. También la medida Fall-out es competitiva en éste dataset. En el dataset AMBIENT WDC-HH-BHRK es mejor en cuanto a recuerdo, medidaF y

accuracy que los algoritmos Bisecting Kmeans, Lingo y STC. Los resultados sobre el dataset de MORESQUE son favorables para STC. En el dataset ODP-239, WDC-HH-BHRK es mejor en cuanto recuerdo, medidaF, fall-out y accuracy con respecto a los otros algoritmos. WDC-HH-BHRK obtiene mejor número de agrupaciones en todos los datasets, y con respecto a los algoritmos Lingo y STC la diferencia es bastante significativa. En promedio la diferencia entre número de agrupaciones de WDC-HH-BHRK y el número ideal de agrupaciones (7.54) esta alrededor de 0.17 agrupaciones, Mientras que en Bisecting Kmeans es de 3.64, Lingo de 20.28 y STC de 6.73. WDC-HH-BHRK también es mejor en recuerdo, medidaF que Bisecting Kmeans, Lingo y STC, en accuracy es muy similar a STC.

En promedio, WDC-HH-BHRK reporta mejores resultados (**Tabla 13**) en recuerdo, fall-out y medidaF respecto a los otros algoritmos. En accuracy es mejor que los algoritmos Bisecting Kmeans y Lingo. La clasificación promedio de precisión usando el test de Friedman muestra que Lingo es el mejor algoritmo con las estadísticas de Friedman (distribuidas de acuerdo con el chi-square con 3 grados de libertad) igual a 274.65906 y valor P computado por el test de Friedman igual a 1.0856937571190883E-10 (**Tabla 13**). La clasificación promedio de recuerdo usando el test de Friedman muestra que WDC-HH-BHRK es el mejor algoritmo con las estadísticas de Friedman (distribuidas de acuerdo con el chi-square con 3 grados de libertad) igual a 639.971141 y valor P computado por el test de Friedman igual a 2.0178947401916503E-10 (**Tabla 13**). La clasificación promedio de medidaF usando el test de Friedman muestra que WDC-HH-BHRK es el mejor algoritmo con las estadísticas de Friedman (distribuidas de acuerdo con el chi-square con 3 grados de libertad) igual a 537.924832 y valor P computado por el test de Friedman igual a 1.7906720550797672E-10 (Tabla 13). La clasificación promedio de accuracy usando el test de Friedman muestra que WDC-HH-BHRK es el mejor algoritmo con las estadísticas de Friedman (distribuidas de acuerdo con el chi-square con 3 grados de libertad) igual a 417.380537 y valor P computado por el test de Friedman igual a 1.44349310282621E-10 (**Tabla 13**). La clasificación promedio de fall-out usando el test de Friedman muestra que Lingo es

el mejor algoritmo con las estadísticas de Friedman (distribuidas de acuerdo con el chi-square con 3 grados de libertad) igual a 55.872483 y valor P computado por el test de Friedman igual a 3.1869618055679894E-11 (**Tabla 13**).

Dataset	Algoritmo	K estimado	Diferencia con el k ideal	Precisión	Recuerdo	medidaF	Accuracy	Fall-out
DMOZ-50	WDC-HH-BHRK	**8,19**	**2,17**	**84,03**	**70,00**	**74,25**	**91,65**	**0,03**
	Bisecting Kmeans	10,98	4,96	70,21	42,77	49,57	84,17	0,05
	Lingo	34,29	28,27	83,85	37,88	48,23	83,41	0,05
	STC	16,00	9,98	**84,82**	57,85	65,12	88,81	**0,03**
AMBIENT	WDC-HH-BHRK	**5,82**	**2,09**	74,11	**62,36**	**63,21**	**84,30**	0,04
	Bisecting Kmeans	11,23	3,32	76,02	40,75	46,00	77,09	0,04
	Lingo	20,86	12,95	**86,75**	50,21	58,68	80,43	**0,03**
	STC	11,00	3,09	72,40	53,14	55,38	81,89	0,06
MORESQUE	WDC-HH-BHRK	**6,09**	**2,27**	86,81	43,30	52,43	60,22	0,05
	Bisecting Kmeans	10,36	6,54	87,12	30,05	38,58	53,38	**0,04**
	Lingo	20,16	16,34	**90,50**	39,35	50,55	59,18	0,06
	STC	11,17	7,35	82,83	**49,96**	**57,18**	**65,45**	0,13
ODP-239	WDC-HH-BHRK	**8,09**	**1,47**	56,93	**45,21**	**45,83**	**81,92**	**0,06**
	Bisecting Kmeans	11,60	2,04	55,08	32,09	34,70	78,09	**0,06**
	Lingo	31,39	21,83	**71,56**	32,93	41,01	79,15	0,07
	STC	15,98	6,42	57,33	39,74	41,80	80,65	0,10
Average	WDC-HH-BHRK	**7,37**	**0.17**	69,27	**49,18**	**52,40**	**77,71**	**0,05**
	Bisecting Kmeans	11,18	3,64	67,01	33,62	38,47	72,37	**0,05**
	Lingo	27,81	20,28	**79,26**	36,82	45,99	74,66	0,06
	STC	14,27	6,73	68,39	45,69	49,67	**77,81**	0,09

Tabla 12. Resultados WDC-HH-BHRK

	Precision		Recuerdo		MedidaF		Accuracy		Fall-out	
Algoritmo	Ranking	Posición	Ranking	Posición	Ranking	Posición	Ranking	Posición	Ranking	Posición
WDC-HH-BHRK	2,7494	3	**1,4362**	**1**	**1,6790**	**1**	**1,6667**	**1**	2,4877	2
Bisecting Kmeans	3,0515	4	3,4329	4	3,6275	4	3,3993	4	2,5235	3
Lingo	**1,6890**	**1**	2,9989	3	2,4284	3	2,6264	3	2,1723	1
STC	2,5101	2	2,1320	2	2,2651	2	2,3076	2	2,8166	4

Tabla 13. Ranking y Posición de algoritmos según test de Friedman

Los algoritmos se probaron con todas las consultas de los conjuntos de datos y el rendimiento de la salida correspondiente se evaluó usando SSL_k, con k = 1, 2, 3, 4. Los resultados de la medida SSL, promediados sobre los conjuntos de prueba, se reportan en la **Tabla 14**.

Dataset	Algoritmo	SSL_1	SSL_2	SLL_3	SSL_4	Suma de SSL_k
DMOZ-50	WDC-HH-BHRK	15,1	19,1	22,1	24,6	80,9
	Lingo	14,2	16,6	18,5	21,9	**71,2**
	STC	12,1	16,4	18,6	21,3	68,4
AMBIENT	WDC-HH-BHRK	14,6	26,0	32,5	37,1	**110,2**
	Lingo	22,4	36,5	47,2	54,3	160,4
	STC	27,2	44,9	54,8	60,4	187,3
	Best combination*	21,7	29,3	33,2	37,3	**121,5**
	OPTIMSRC*	20,6	28,9	34,1	38,9	122,5
	Lingo*	24,4	30,6	36,6	40,7	132,3
	KeySRC*	24,1	32,4	38,2	42,1	136,8
	Lingo3G*	24,0	32,4	39,6	43,0	139,0
	Yahoo!*	21,6	35,5	42,0	47,6	146,7
MORESQUE	WDC-HH-BHRK	11,1	18,6	24,1	27,8	**81,6**
	Lingo	16,5	26,4	33,9	39,2	116,0
	STC	19,6	32,3	40,2	45,2	137,3
ODP-239	WDC-HH-BHRK	19,8	29,9	39,5	51,2	**140,4**
	Lingo	25,6	38,1	51,4	66,4	181,5
	STC	26,3	43,1	60,7	78,4	208,5
	Lingo**	22,0	35,0	48,3	63,8	169,1
	Lingo3G**	21,5	34,4	48,2	63,3	167,4
	KeySRC**	22,8	40,1	57,3	75	195,2

* Tomado de [28]

** Tomado de [69]

Tabla 14. Evaluación del comportamiento del usuario

En los datasets de AMBIENT, MORESQUE y ODP-239, WDC-HH-BHRK supera en SSL_1, SSL_2, SSL_3, SSL_4 y la suma de SSL_k a Lingo y STC. En el dataset de DMOZ WDC-HH-BHRK tiene pobres resultados porque este dataset no tiene consultas, por su parte MORESQUE tiene 2, 3 o 4 palabras clave para describir la consulta y por lo tanto WDC-HH-BHRK mejora los resultados de SSL_k en gran medida. Las palabras clave de las consultas son muy importantes para el proceso de etiquetado en WDC-HH-BHRK. El algoritmo Bisecting k-means no se tiene en cuenta en los resultados con SSL debido a que usan términos estadísticamente representativos y no frases frecuentes. En el estado del arte ya está definido que la presentación con términos no es la adecuada para sistemas que agrupan documentos web.

4.4.3 Etapa III de pruebas con usuarios

Después de completado el proceso de definición, creación y evaluación en laboratorio del framework híper heurístico, se realizaron dos experimentos a ciegas con estudiantes (usuarios) de últimos semestres (VIII, IV y X) del programa ingeniería de sistemas de la Universidad del Cauca. Primero se llevaron a cabo las pruebas del algoritmo Lingo con un grupo de trece estudiantes, seguidamente de las pruebas del mejor algoritmo obtenido por el framework híper heurístico aplicadas a los mismos trece estudiantes. En ambos experimentos se evaluaron los resultados de una consulta para medir la calidad de los resultados en cuanto a claridad y utilidad de las etiquetas, la pertinencia de los documentos a los grupos y el orden de los mismos en los grupos, a través de una encuesta (**Tabla 15**), que cuenta con 3 secciones de: 1) preguntas específicas aplicadas a los 10 primeros grupos generados por cada algoritmo; 2) preguntas generales y 3) observaciones donde los estudiantes pueden dar sugerencias y apreciaciones sobre el comportamiento del algoritmo evaluado.

Preguntas específicas

Grupo #

¿La etiqueta del grupo es representativa con respecto a los documentos del grupo?

☐ TA A ☐ PA ☐ PD ☐ TD ☐

¿La etiqueta es útil para seleccionar el sub-tema (tópico) específico de la consulta?

☐ TA A ☐ PA ☐ PD ☐ TD ☐

¿Los documentos del grupo están relacionados con la etiqueta del grupo al que pertenecen?

☐ TA A ☐ PA ☐ PD ☐ TD ☐

¿La relevancia (posición u orden) de los documentos en el grupo es la más adecuada?

☐ TA A ☐ PA ☐ PD ☐ TD ☐

Preguntas generales

1. ¿La cantidad de grupos es apropiada?

☐ TA ☐ A ☐ PA ☐ PD ☐ TD

2. ¿La calidad de las etiquetas en general es adecuada?

☐ TA ☐ A ☐ PA ☐ PD ☐ TD

Tabla 15. Formato encuesta pruebas con usuarios

En donde las respuestas posibles son: TA = Totalmente de acuerdo, A= De acuerdo, PA = Parcialmente de acuerdo, PD = Parcialmente en desacuerdo y TD = Totalmente en desacuerdo cada una de las cuales tiene un peso de 1 a 5 donde 1 es la calificación más baja y 5 la más alta.

De las trece encuestas se eliminaron 3 de ellas porque los resultados evidentemente eran sesgados, por ejemplo a todas las repuestas asignaban como repuesta PA (3), lo cual no es viable estadísticamente hablando. Basados en los resultados de los algoritmos evaluados, se calculo el promedio de cada una de las preguntas específicas por cada grupo evaluado, posteriormente se realizó el promedio general por cada pregunta (mismo proceso aplicado a los resultados de

las preguntas generales). Obteniendo como resultados los presentados en **Tabla 16** y **Tabla 17**.

	Preguntas específicas				Preguntas generales	
	Pregunta 1	**Pregunta 2**	**Pregunta 3**	**Pregunta 4**	**Pregunta 1**	**Pregunta 2**
Estudiante 1	4,5	4,3	4,4	4,4	4	4
Estudiante 2	3,3	2,2	3,6	3,1	1	2
Estudiante 3	3	3,6	3,6	2,9	2	2
Estudiante 4	3,5	3,2	3,3	2,9	3	3
Estudiante 5	3,1	3,3	3,2	3	3	2
Estudiante 6	3,5	3,7	3,5	3,1	2	3
Estudiante 7	3,9	3,9	3,7	3	3	3
Estudiante 8	3,1	2,7	3,8	3,4	3	4
Estudiante 9	3,6	3,7	3,7	3,7	4	4
Estudiante 10	2,2	2,5	2,5	2,1	2	1
Promedio General	3,37	3,31	3,53	3,16	2,7	2,8

Tabla 16. Resultados sobre el algoritmo Lingo

	Preguntas específicas				Preguntas generales	
	Pregunta 1	**Pregunta 2**	**Pregunta 3**	**Pregunta 4**	**Pregunta 1**	**Pregunta 2**
Estudiante 1	4,5	4,4	4,3	4,3	5	5
Estudiante 2	3,3	3	3,8	3	4	4
Estudiante 3	2,9	3,5	3,3	3	3	4
Estudiante 4	2,9	3,2	3,1	3	3	2
Estudiante 5	2,6	1,8	2,4	2,2	4	2
Estudiante 6	3,4	3,3	3,2	3,1	4	4
Estudiante 7	3,8	4	3,8	3,7	2	2
Estudiante 8	4,5	4,2	4,3	4,5	5	4
Estudiante 9	3,1	3,3	3,5	3,5	5	3
Estudiante 10	3,9	3,9	4	3,9	4	4
Promedio General	**3,49**	**3,46**	**3,57**	**3,42**	**3,9**	**3,4**

Tabla 17. Resultados sobre el algoritmo BHRK

La **Figura 22** muestra los resultados obtenidos a las preguntas realizadas para cada uno de los algoritmos, en todas las preguntas el algoritmo BHRK es superior, pero en las preguntas generales (5 y 6) existe una diferencia más marcada entre Lingo y BHRK, esto quiere decir que la cantidad de grupos y la calidad de las etiquetas de BHRK son más apropiadas que las de Lingo.

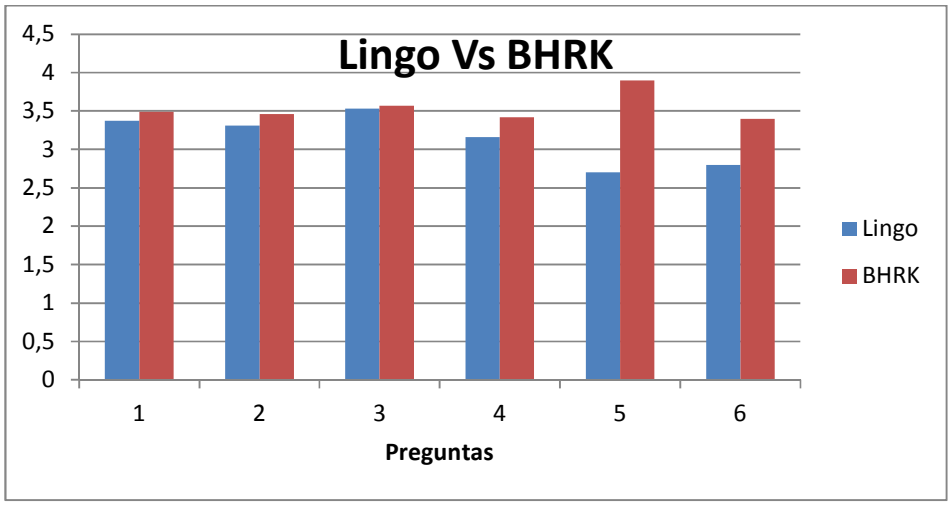

Figura 22. Comparativo de los resultados de las pruebas con usuarios

A continuación en la **Figura 23** y la **Figura 24** se puede apreciar un resultado global de la encuesta, donde Lingo tiene un porcentaje mayor de apreciaciones negativas (totalmente y parcialmente en desacuerdo), mientras que BHRK tiene un porcentaje mayor de apreciaciones positivas (acuerdo y totalmente de acuerdo). Las apreciaciones intermedias (parcialmente de acuerdo) son casi similares entre los dos sistemas, la diferencia es apenas del 1%.

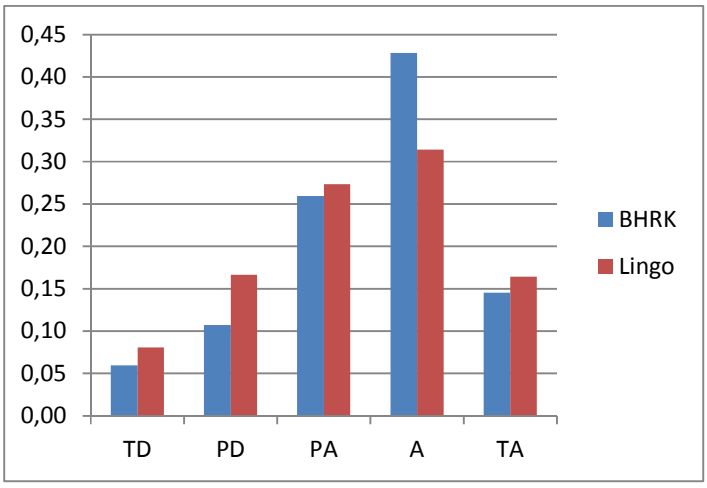

Figura 23. Comparativo general de los resultados para cada algoritmo

Figura 24. Comparativo agrupado de resultados para cada algoritmo

Basados en los resultados de los algoritmos evaluados **Tabla 18** y **Tabla 20**, se llevo a cabo el test de Kappa de Fleiss para medir el nivel de concordancia entre las respuestas de los diferentes estudiantes, dando como resultado una pobre y leve concordancia para la evaluación de los algoritmos (ver **Tabla 19** y **Tabla 21**), lo que quiere decir que los resultados de la prueba no son contundentes y que los estudiantes (jueces) expresan opiniones diferentes.

TOPICO	PREGUNTA	E2	E3	E5	E6	E8	E9	E10	E11	E12	E13
1	1	4	3	3	4	2	4	4	5	4	2
	2	5	4	4	3	5	4	4	5	5	2
	3	5	3	2	3	3	4	4	5	5	3
	4	5	3	2	2	4	3	3	4	4	2
2	1	4	3	3	2	5	3	5	5	4	4
	2	4	2	3	3	5	4	5	5	3	4
	3	4	3	4	3	5	3	5	5	4	4
	4	4	1	2	2	2	3	3	5	3	3
3	1	5	2	4	3	1	3	5	1	3	1
	2	5	1	3	2	1	3	5	1	3	2
	3	5	2	4	4	2	3	5	1	4	2
	4	5	1	3	2	3	2	5	3	4	2
4	1	4	3	2	3	5	4	5	5	3	4
	2	3	2	4	4	5	4	5	3	4	4
	3	4	4	4	3	5	4	4	4	3	3
	4	4	4	3	2	5	4	3	4	4	3
5	1	5	4	5	3	2	5	4	3	4	1
	2	4	2	5	4	2	4	4	1	3	2
	3	4	5	5	3	3	5	4	4	3	1
	4	4	4	3	2	3	4	3	3	4	2
6	1	4	4	4	5	5	3	1	5	5	3
	2	4	4	5	4	5	4	1	5	5	3
	3	4	4	4	5	4	3	1	5	5	3
	4	4	4	5	4	3	2	1	4	4	2
7	1	5	2	1	4	2	2	2	2	4	3
	2	5	1	3	3	4	2	2	2	4	3
	3	5	3	1	2	2	2	2	3	3	3
	4	5	3	3	4	1	2	2	3	4	2
8	1	4	3	2	4	3	4	4	1	3	2
	2	4	1	2	4	1	4	4	1	3	1
	3	4	3	4	4	2	4	4	3	3	1
	4	4	3	2	3	2	4	4	1	4	1
9	1	5	5	3	3	2	4	4	1	3	1
	2	4	2	5	2	2	4	4	1	4	2
	3	4	5	4	3	3	4	3	5	3	3
	4	4	4	4	4	3	4	3	3	3	2
10	1	5	4	3	4	4	3	5	3	3	1
	2	5	3	2	3	3	4	5	3	3	2
	3	5	4	4	3	3	3	5	3	4	2
	4	5	4	2	4	4	3	3	4	3	2
Preguntas generales	1	4	1	2	3	3	2	3	3	4	2
	2	4	2	2	3	2	3	3	4	4	1

Tabla 18. Respuestas de los usuarios, algoritmo Lingo

TOPICO	PREGUNTA	TD (1)	PD (2)	PA (3)	A (4)	TA (5)	TOTAL	Pi	PROMEDIO	MODA
1	1	0	2	2	5	1	10	0,267	3,5	4
	2	0	1	1	4	4	10	0,267	4,1	5
	3	0	1	4	2	3	10	0,222	3,7	3
	4	0	3	3	3	1	10	0,200	3,2	3
2	1	0	1	3	3	3	10	0,200	3,8	4
	2	0	1	3	3	3	10	0,200	3,8	4
	3	0	0	3	4	3	10	0,267	4,0	4
	4	1	3	4	1	1	10	0,200	2,8	3
3	1	3	1	3	1	2	10	0,156	2,8	3
	2	3	2	3	0	2	10	0,178	2,6	1
	3	1	3	1	3	2	10	0,156	3,2	2
	4	1	3	3	1	2	10	0,156	3,0	3
4	1	0	1	3	3	3	10	0,200	3,8	4
	2	0	1	2	5	2	10	0,267	3,8	4
	3	0	0	3	6	1	10	0,400	3,8	4
	4	0	1	3	5	1	10	0,289	3,6	4
5	1	1	1	2	3	3	10	0,156	3,6	5
	2	1	3	1	4	1	10	0,200	3,1	4
	3	1	0	3	3	3	10	0,200	3,7	4
	4	0	2	4	4	0	10	0,289	3,2	4
6	1	1	0	2	3	4	10	0,222	3,9	5
	2	1	0	1	4	4	10	0,267	4,0	4
	3	1	0	2	4	3	10	0,222	3,8	4
	4	1	2	1	5	1	10	0,244	3,3	4
7	1	1	5	1	2	1	10	0,244	2,7	2
	2	1	3	3	2	1	10	0,156	2,9	3
	3	1	4	4	0	1	10	0,267	2,6	3
	4	1	3	3	2	1	10	0,156	2,9	3
8	1	1	2	3	4	0	10	0,222	3,0	4
	2	4	1	1	4	0	10	0,267	2,5	4
	3	1	1	3	5	0	10	0,289	3,2	4
	4	2	2	2	4	0	10	0,200	2,8	4
9	1	2	1	3	2	2	10	0,133	3,1	3
	2	1	4	0	4	1	10	0,267	3,0	4
	3	0	0	5	3	2	10	0,311	3,7	3
	4	0	1	4	5	0	10	0,356	3,4	4
10	1	1	0	4	3	2	10	0,222	3,5	3
	2	0	2	5	1	2	10	0,267	3,3	3
	3	0	1	4	3	2	10	0,222	3,6	3
	4	0	2	3	4	1	10	0,222	3,4	4
Preguntas generales	1	1	3	4	2	0	10	0,222	2,7	3
	2	1	3	3	3	0	10	0,200	2,8	4
	TOTAL	34	70	115	132	69	420	9,64	3,3	4
	Pr	0,08	0,17	0,27	0,31	0,16		0,23	P-	
	Pr^2	0,01	0,03	0,07	0,10	0,03	0,24	Pe-		
			0,25	0,27	0,48					

Kappa de Fleiss: -0,01 Poor Concordance

Tabla 19. Resultados test Kappa de Fleiss del algoritmo Lingo

TOPICO	PREGUNTA	E2	E3	E5	E6	E8	E9	E10	E11	E12	E13
1	1	5	3	2	3	3	2	3	4	3	4
	2	5	4	4	3	4	2	4	5	1	4
	3	5	3	2	4	3	2	2	5	3	4
	4	5	2	3	3	3	2	5	4	4	4
2	1	5	4	3	2	3	3	3	5	4	4
	2	5	4	3	3	3	3	5	4	4	4
	3	5	5	4	3	3	3	3	4	4	4
	4	5	4	3	3	2	2	4	5	4	4
3	1	4	4	5	4	3	4	3	5	3	4
	2	4	5	4	3	1	4	4	5	4	4
	3	4	5	5	3	2	4	4	4	3	4
	4	4	4	4	4	2	4	4	5	4	4
4	1	4	3	3	3	5	4	3	5	5	4
	2	4	3	4	4	1	4	4	5	4	4
	3	4	4	4	3	4	4	4	5	4	4
	4	4	2	4	3	3	4	3	4	4	4
5	1	5	4	3	2	2	5	5	5	1	4
	2	5	2	4	3	2	4	5	4	2	4
	3	4	5	3	3	3	4	5	5	3	4
	4	4	4	3	2	2	4	5	5	2	4
6	1	4	3	3	3	2	4	3	5	3	3
	2	4	4	4	4	1	4	3	4	4	3
	3	4	3	3	2	2	4	3	4	3	4
	4	4	2	3	3	1	3	3	5	4	3
7	1	4	5	4	3	3	4	5	5	3	4
	2	4	4	3	4	3	4	4	4	3	4
	3	4	5	4	3	3	3	4	4	3	4
	4	4	4	2	3	3	4	4	5	2	4
8	1	5	4	3	3	3	4	5	5	1	4
	2	5	1	2	2	1	4	4	5	3	4
	3	5	4	3	4	2	4	5	5	4	4
	4	5	4	2	3	2	4	5	5	4	4
9	1	5	1	1	3	1	1	4	3	4	4
	2	4	1	4	2	1	1	4	3	4	4
	3	4	1	2	3	1	1	4	3	4	4
	4	4	1	3	4	2	1	1	3	4	4
10	1	4	2	2	3	1	3	4	3	4	4
	2	4	2	3	4	1	3	3	3	4	4
	3	4	3	3	3	1	3	4	4	4	4
	4	4	3	3	2	2	3	3	4	3	4
Preguntas generales	1	5	4	3	3	4	4	2	5	5	4
	2	5	4	4	2	2	4	2	4	3	4

Tabla 20. Respuestas de los usuarios, algoritmo BHRK

TOPICO	PREGUNTA	(TD) 1	(PD) 2	(PA) 3	(A) 4	(TA) 5	TOTAL	Pi	PROMEDIO	MODA
1	1	0	2	5	2	1	10	0,267	3,2	3
	2	1	1	1	5	2	10	0,244	3,6	4
	3	0	3	3	2	2	10	0,178	3,3	3
	4	0	2	3	3	2	10	0,178	3,5	3
2	1	0	1	4	3	2	10	0,222	3,6	3
	2	0	0	4	4	2	10	0,289	3,8	4
	3	0	0	4	4	2	10	0,289	3,8	4
	4	0	2	2	4	2	10	0,200	3,6	4
3	1	0	0	3	5	2	10	0,311	3,9	4
	2	1	0	1	6	2	10	0,356	3,8	4
	3	0	1	2	5	2	10	0,267	3,8	4
	4	0	1	0	8	1	10	0,622	3,9	4
4	1	0	0	4	3	3	10	0,267	3,9	3
	2	1	0	1	7	1	10	0,467	3,7	4
	3	0	0	1	8	1	10	0,622	4,0	4
	4	0	1	3	6	0	10	0,400	3,5	4
5	1	1	2	1	2	4	10	0,178	3,6	5
	2	0	3	1	4	2	10	0,222	3,5	4
	3	0	0	4	3	3	10	0,267	3,9	3
	4	0	3	1	4	2	10	0,222	3,5	4
6	1	0	1	6	2	1	10	0,356	3,3	3
	2	1	0	2	7	0	10	0,489	3,5	4
	3	0	2	4	4	0	10	0,289	3,2	4
	4	1	1	5	2	1	10	0,244	3,1	3
7	1	0	0	3	4	3	10	0,267	4,0	4
	2	0	0	3	7	0	10	0,533	3,7	4
	3	0	0	4	5	1	10	0,356	3,7	4
	4	0	2	2	5	1	10	0,267	3,5	4
8	1	1	0	3	3	3	10	0,200	3,7	5
	2	2	2	1	3	2	10	0,133	3,1	4
	3	0	1	1	5	3	10	0,289	4,0	4
	4	0	2	1	4	3	10	0,222	3,8	4
9	1	4	0	2	3	1	10	0,222	2,7	1
	2	3	1	1	5	0	10	0,289	2,8	4
	3	3	1	2	4	0	10	0,222	2,7	4
	4	3	1	2	4	0	10	0,222	2,7	4
10	1	1	2	3	4	0	10	0,222	3,0	4
	2	1	1	4	4	0	10	0,267	3,1	4
	3	1	0	4	5	0	10	0,356	3,3	4
	4	0	2	5	3	0	10	0,311	3,1	3
Preguntas generales	1	0	1	2	4	3	10	0,222	3,9	4
	2	0	3	1	5	1	10	0,289	3,4	4
	TOTAL	25	45	109	180	61	420	12,33	3,5	4
	Pr	0,06	0,11	0,26	0,43	0,15		0,29	P-	
	Pr^2	0,00	0,01	0,07	0,18	0,02	0,29	Pe-		
			0,17	0,26	0,57					

Kappa de Fleiss: 0,01 Slight Concordance

Tabla 21. Resultados test Kappa de Fleiss del algoritmo BHRK

Capítulo 5

5 CONCLUSIONES, RECOMENDACIONES Y TRABAJO FUTURO

5.1 CONCLUSIONES

Se logro modelar, implementar y evaluar exitosamente un framework híper heurístico para el problema de la agrupación de documentos web. El framework se puede ejecutar directamente para la agrupación de documentos web y funciona como un algoritmo centrado en la descripción. El framework utiliza cuatro estrategias de selección de alto nivel: selección aleatoria, selección tabu, selección rank y selección por ruleta basada en el desempeño de la heurística de bajo nivel. Al mismo tiempo utiliza diversas heurísticas de bajo nivel: búsqueda armónica (HS), búsqueda armónica mejorada (IH), nueva búsqueda armónica global (NH), mejor búsqueda armónica global (BH), optimización por enjambre de partículas (PS), colonia artificial de abejas (CA), evolución diferencial (ED) y heurísticas basadas en algoritmos genéticos producto de la mezcla de los métodos de selección: emparejamiento restrictivo (RM), selección por ruleta (RW) y Rank (RK); los métodos de cruce: en un punto (UP), Uniforme (CU) y multi-punto (CM) y; los métodos de mutación: multi-bit (MM) y de un bit (MO). También utiliza el algoritmo k-means como estrategia de mejora de la solución local estrategia local, y con base en el criterio de información bayesiano balanceado es capaz de definir automáticamente el número de grupos. Finalmente usa cuatro estrategias de reemplazo: reemplazar el peor (WR), Reemplazo por competencia restringida (RC), reemplazo estocástico (SR) y reemplazo Rank (RR).

A partir de los resultados obtenidos de las pruebas aplicadas a los dataset: dmoz, ambient, moresque y odp239, se logro encontrar a la mejor heurística de bajo nivel BHRK, compuesta por mejor búsqueda armónica global y la estrategia de reemplazo Rank, la cual presento un mayor desempeño con base en las medidas de: precisión, recuerdo y medidaF, contra los algoritmos Lingo, STC y Bisecting Kmeans.

Como se pudo observar en los resultados organizados en cuanto a medidaF de las heurísticas individuales y las combinaciones con las heurísticas seleccionadas (**Tabla 8**), hacer uso de heurísticas que trabajan juntas para obtener la mejor solución da buenos resultados (posición 2 de la tabla) por lo que es posible que si se prueban todas las posibles combinaciones de las heurísticas se logre mejorar los resultados del problema de agrupación de documentos web y también sobrepasar a las heurísticas individuales, en especial a la heurística BHRK seleccionada como la mejor hasta el momento.

Las pruebas con usuarios no cuentan con mucho prestigió dentro de la comunidad de investigadores, debido a que suelen ser muy subjetivas y están ligadas muchas veces al estado emocional del usuario, lo cual se logro comprobar con los resultados de las pruebas presentadas en esta investigación, a partir de los cuales no se puede obtener una conclusión certera acerca de la calidad y utilidad de las etiquetas, la cantidad de los grupos y el orden de los documentos en los grupos, en este tipo de situaciones. El cálculo de la medida SSL en forma automática por el contrario, es de mucha utilidad y sirve para comparar más objetivamente el comportamiento de los usuarios.

El framework híper heurístico hace uso del algortimo K-means como estrategia de optimización, el cual es aplicado a cada individuo de la población para buscar la mejor solución local, lo cual permite mejorar la calidad del proceso de agrupación de documentos web, debido a que K-means es un algoritmo eficiente, presenta un orden de complejidad de orden O(n) por lo tanto su tiempo de procesamiento es adecuado y además es un algoritmo fácil de implementar.

Usar centroides como unidad mínima de procesamiento en el vector solución evita el problema de las unidades muertas (dead unit) en el proceso de clustering, haciendo más eficiente el proceso evolutivo de las heurísticas.

5.2 RECOMENDACIONES Y TRABAJO FUTURO

Para poder realizar una prueba exhaustiva del framework híper heurístico es necesario llevar a cabo un total aproximado de 9,62E+151 pruebas (97!) donde se analice el comportamiento de todas las posibles combinaciones con las 97 heurísticas de bajo nivel, motivo por el cual se recomienda el uso de arreglos de cobertura (covering arrays) por medio de los cuales se reduce significativamente el número de pruebas y así obtener resultados más determinantes. Además, realizar convenios con universidades y centros de investigación del país o del exterior con el objetivo de realizar evaluaciones más exhaustivas usando super-computadoras.

Para trabajo futuro también se recomienda extender el framework híper heurístico con la inclusión de más heurísticas de alto nivel, bajo nivel y heurísticas de reemplazo, así como de micro heurísticas para la generación de heurísticas hibridas compuestas por nuevos métodos de selección, cruce y mutación, comparando los resultados obtenidos con otros algoritmos del estado del arte.

Uso de técnicas que permitan reducir la ambigüedad en las consultas que digitan los usuarios y así poder mejorar la calidad de los resultados de las agrupaciones.

Hacer uso de WordNet u otra herramienta semántica que trabaje con conceptos en lugar de términos, con el fin de observar el comportamiento del proceso de agrupación de documentos web, comparando los resultados con los reportados por otros algoritmos encontrados en el estado del arte.

Capítulo 6

6 REFERENCIAS

[1] **T. Lozano, "La gestión del cambio en las bibliotecas electrónicas," 2002.**

[2] **R. Baeza-Yates, C. Castillo, and B. Keith, "Web Searching," in *Encyclopedia of Language & Linguistics* Oxford: Elsevier, 2006, pp. 527-538.**

[3] **C. Carpineto, S. Osiński, G. Romano, and D. Weiss, "A survey of Web clustering engines," *ACM Comput. Surv.,* vol. 41, pp. 1-38, 2009.**

[4] **A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM Comput. Surv.,* vol. 31, pp. 264-323, 1999.**

[5] **S. Weiguo, L. Xiaohui, and M. Fairhurst, "A Niching Memetic Algorithm for Simultaneous Clustering and Feature Selection," *Knowledge and Data Engineering, IEEE Transactions on,* vol. 20, pp. 868-879, 2008.**

[6] **W. Song and S. C. Park, "Genetic algorithm for text clustering based on latent semantic indexing," *Computers & Mathematics with Applications,* vol. 57, pp. 1901-1907, 2009.**

[7] **G. Zong Woo, "Recent Advances in Harmony Search Algorithm," p. 176, april 22, 2010 2010.**

[8] **A. Kuri and J. Galaviz, *Algoritmos Genéticos*: Fondo de Cultura Económica/UNAM/IPM, 2002.**

[9] **E. K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Ozcan, and R. Qu, "A Survey of Hyper-heuristics," University of Nottingham, Nottingham, UK 2009.**

[10] **E. K. Burke, B. McCollum, A. Meisels, S. Petrovic, and R. Qu, "A graph-based hyper-heuristic for educational timetabling problems," 2005.**

[11] **K. Inna Gelfer and K. Oren, "Cluster-based query expansion," in *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval* Boston, MA, USA: ACM, 2009.**

[12] **O. Gospodnetic, E. Hatcher, and D. Cutting, *Lucene in action*: Mannaging, 2005.**

[13] **J. Li, E. K. Burke, and R. Qu, "Integrating neural networks and logistic regression to underpin hyper-heuristic search," 2010.**

[14] **A. Villoria G, S. Salhi, A. Corominas, and R. Parstor, "Hyper-heuristic approaches for the response time variability problem," 2010.**

[15] **A. Rafique F, H. Linshu, A. Kamran, and Q. Zeeshan, "Hyper Heuristic Approach for Design and Optimization of Satellite Launch Vehicle," 2010.**

[16] **C. Cobos, M. Mendoza, and E. Leon, "A hyper-heuristic approach to design and tuning heuristic methods for web document clustering," January 28,2011 2011.**

[17] **K. Rodríguez and R. Ronda, "Organización y recuperación de la información: un enfoque desde la perspectiva de la automatización," *ACIMED* vol. 14, 2006.**

[18] **f. Martinez, *Recuperación de información: modelos sistemas y evaluación*. Murcia, 2004.**

[19] **S. Osiński, "DIMENSIONALITY REDUCTION TECHNIQUES FOR SEARCH RESULTS CLUSTERING," 2004.**

[20] **Z. Oren and E. Oren, "Web document clustering: a feasibility demonstration," in *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval* Melbourne, Australia: ACM, 1998.**

[21] **C. Cobos and E. Leon, "State of the Art in Web Document Clustering Engines," Popayán, 2011, p. 16.**

[22] **O. Zarei, M. Fesanghary, B. Farshi, R. J. Saffar, and M. R. Razfar, "Optimization of multi-pass face-milling via harmony search algorithm," *Journal of Materials Processing Technology,* vol. 209, pp. 2386-2392, 2009.**

[23] **M. Hemalatha and D. Sathyasrinivas, "Hybrid neural network model for web document clustering," in *Applications of Digital Information and Web Technologies, 2009. ICADIWT '09. Second International Conference on the*, 2009, pp. 531-538.**

[24] **M. Carullo, E. Binaghi, and I. Gallo, "An online document clustering technique for short web contents," *Pattern Recognition Letters,* vol. 30, pp. 870-876, 2009.**

[25] **S. Zheng, X. Zhao, B. Zhang, and H. Bu, "Web Document Clustering Research Based on Granular Computing," in *Electronic Commerce and Security, 2009. ISECS '09. Second International Symposium on*, 2009, pp. 446-450.**

[26] **A. Bernardini, C. Carpineto, and M. D'Amico, "Full-Subtopic Retrieval with Keyphrase-Based Search Results Clustering," in *WI-IAT '09: IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technologies*, 2009, pp. 206-213.**

[27] **R. Navigli and G. Crisafulli, "Inducing word senses to improve web search result clustering," in *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing* Cambridge, Massachusetts: Association for Computational Linguistics, 2010, pp. 116-126.**

[28] **C. Carpineto and G. Romano, "Optimal meta search results clustering," in *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval* Geneva, Switzerland: ACM, 2010, pp. 170-177.**

[29] **I. Ribeiro, "Um Estudo Empírico de Hiper-Heurísticas," 2007.**

[30] **I. Ribeiro and F. Soares, "MÉTODOS HEURÍSTICOS GENÉRICOS: METAHEURÍSTICAS E HIPER-HEURÍSTICAS," 2004.**

[31] **C. Rodríguez, A. Duarte, and J. J. Pantrigo, "Algoritmos heurísticos y meta heurísticos para el problema de localización de generadores ".**

[32] **N. Pillay and W. Banzhaf, "A study of heuristic combinations for hyper-heuristic systems for the uncapacitated examination timetabling problem," 2007.**

[33] **F. Thabtah and P. Cowling, "Mining the data from a hyperheuristic approach using associative classification," 2007.**

[34] **C. Cobos, L. Muñoz, M. Mendoza, E. León, and E. Herrera-Viedma, "Fitness Function Obtained from a Genetic Programming Approach for Web Document Clustering Using Evolutionary Algorithms," in *Advances in Artificial Intelligence – IBERAMIA 2012*. vol. 7637, J. Pavón, N. Duque-Méndez, and R. Fuentes-Fernández, Eds.: Springer Berlin Heidelberg, 2012, pp. 179-188.**

[35] **A. K. Jain and R. C. Dubes, *Algorithms for clustering data*: Prentice-Hall, Inc., 1988.**

[36] **A. Webb, *Statistical Pattern Recognition, 2nd Edition*: {John Wiley & Sons}, 2002.**

[37] **P. Berkhin, "Survey Of Clustering Data Mining Techniques," Accrue Software, Inc. 2002.**

[38] **J. Han, M. Kamber, and A. K. H. Tung, "Spatial Clustering Methods in Data Mining: A Survey," in *Geographic Data Mining and Knowledge Discovery*: Taylor and Francis, 2001.**

[39] **G. H. O. Mahamed, P. E. Andries, and S. Ayed, "An overview of clustering methods," *Intell. Data Anal.,* vol. 11, pp. 583-605, 2007.**

[40] **S. J. Redmond and C. Heneghan, "A method for initialising the K-means clustering algorithm using kd-trees," *Pattern Recognition Letters,* vol. 28, pp. 965-973, 2007.**

[41] **R. Baeza-Yates, A. and B. Ribeiro-Neto, *Modern Information Retrieval*: Addison-Wesley Longman Publishing Co., Inc., 1999.**

[42] **M. G. H. Omran and M. Mahdavi, "Global-best harmony search," *Applied Mathematics and Computation,* vol. 198, pp. 643-656, 2008.**

[43] **M. Mahdavi, M. H. Chehreghani, H. Abolhassani, and R. Forsati, "Novel meta-heuristic algorithms for clustering web documents," *Applied Mathematics and Computation,* vol. 201, pp. 441-451, 2008.**

[44] **M. Mahdavi, M. Fesanghary, and E. Damangir, "An improved harmony search algorithm for solving optimization problems," *Applied Mathematics and Computation,* vol. 188, pp. 1567-1579, 2007.**

[45] **Z. Geem, J. Kim, and G. V. Loganathan, "A New Heuristic Optimization Algorithm: Harmony Search," *SIMULATION,* vol. 76, pp. 60-68, 2001.**

[46] **K. Hammouda, "Web Mining: Clustering Web Documents A Preliminary Review," 2001.**

[47] **S. Osiński, "An Algorithm for clustering of web search results." vol. Master Poland: Poznań University of Technology, 2003, p. 91.**

[48] **M. Mitchell, *An Introduction to Genetic Algorithms*: MIT Press 1999.**

[49] **T. T. Bickle, L., "A comparison of selection schemes used in genetics algorithms," 1995.**

[50] **J. Grobler, A. P. Engelbrecht, G. Kendall, and V. S. S. Yadavalli, "Alternative hyper-heuristic strategies for multi-method global optimization," in *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pp. 1-8.**

[51] **S. Tsubakitani and R. J. Evans, "Optimizing Tabu list size for the traveling for the traveling salesman problem," 1997.**

[52] **L. Bianchi, M. Dorigo, L. M. Gambardella, and W. J. Gutjahr, "A survey on metaheuristics for stochastic combinatorial optimization," *Natural Computing: an international journal,* vol. 8, pp. 239-287, 2009.**

[53] **Z. W. Geem, *Recent Advances In Harmony Search Algorithm* vol. 270. Annandale,Virginia: Springer 2010.**

[54] **Z. W. Geem, *Music-Inspired Harmony Search Algorithm: Theory and Applications* vol. 191. Rockville, Maryland: Springer Publishing Company, Incorporated, 2009.**

[55] **C. cobos, J. pérez, and D. estupiñan, "a survey of harmony search," 2010.**

[56] **P. Lira, "Adaptación del algortimo de colonia artificial de abejas para resolver problemas de diseño en ingeniería," 2009.**

[57] **A. Villagra and G. Leguizamon, "Metaheurísticas aplicadas a Clustering."**

[58] **J. M. Garcia, E. Torres Alba, and G. J. Luque, "Algoritmos Basados en Cúmulos de Partículas Para la Resolución de Problemas Complejos," 2006.**

[59] **D. Zou, L. Gao, J. Wu, S. Li, and Y. Li, "A novel global harmony search algorithm for reliability problems," *Computers & Industrial Engineering,* vol. 58, pp. 307-316, 2010.**

[60] **M. G. H. Omran, A. P. Engelbrecht, and A. Salman, "Bare bones differential evolution," *European Journal of Operational Research,* vol. 196, pp. 128-139, 2009.**

[61] **D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*: Addison-Wesley Longman Publishing Co., Inc., 1989.**

[62] **C. Cobos, C. Montealegre, M. Mejía, M. Mendoza, and E. León, "Web Document Clustering based on a New Niching Memetic Algorithm, Term-Document Matrix and Bayesian Information Criterion," in *IEEE Congress on Evolutionary Computation (IEEE CEC)*, Barcelona, Spain, 2010, pp. 4629-4636.**

[63] **M. Mahdavi and H. Abolhassani, "Harmony K-means algorithm for document clustering," *Data Mining and Knowledge Discovery,* vol. 18, pp. 370-391, 2009.**

[64] **J. Fourie, S. Mills, and R. Green, "Visual Tracking Using Harmony Search," in *Recent Advances In Harmony Search Algorithm*. vol. 270: Springer Berlin / Heidelberg, 2010, pp. 37-50.**

[65] **N. Farhad, K. Ahamad Tajudin, and A.-B. Mohammed Azmi, "Adaptive genetic algorithm using harmony search," in *Proceedings of the 12th annual conference on Genetic and evolutionary computation* Portland, Oregon, USA: ACM, 2010.**

[66] **L. dos Santos Coelho and D. de A. Bernert, "A Harmony Search Approach Using Exponential Probability Distribution Applied to Fuzzy Logic Control Optimization," in *Recent Advances In Harmony Search Algorithm*. vol. 270: Springer Berlin / Heidelberg, 2010, pp. 77-88.**

[67] **C. A. C. Lozada, "Modelo de un Meta Buscador que Realiza Agrupación de Documentos Web, Enriquecido con una Taxonomía, Ontologías e Información del Usuario," in *Departamento de Ingeniería de Sistemas e Industrial* Bogotá: Universidad Nacional de Colombia, 2009, p. 39.**

[68] **D. M. S. García, M. Lozano, and F. Herrera, "Un estudio experimental sobre el uso de test no paramétricos para analizar el comportamiento de los algoritmos evolutivos en problemas de optimización."**

[69] **C. Carpineto, M. D'Amico, and G. Romano, "Evaluating subtopic retrieval methods: Clustering versus diversification of search results," *Information Processing & Management,* vol. 48, pp. 358-373, 2012.**