Agrupamiento de Documentos Web Basados en la Búsqueda Cucú

HENRY MUÑOZ COLLAZOS

RICHAR ANDERSON URBANO MUÑOZ

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

GESTIÓN DE LA INFORMACIÓN – BÚSQUEDAS WEB

POPAYÁN, AGOSTO DE 2013

Agradecimientos

A Dios, por ser nuestro creador, por permitirnos seguir nuestro camino y darnos la fuerza para continuar cada día.

A nuestras familias, que sin esperar nada a cambio, han acompañado cada momento de nuestras vidas, con su esfuerzo y aliento para que lleguemos a nuestras metas propuestas.

Al Ph.D. (c). Carlos Alberto Cobos Lozada por su dedicación, tiempo, apoyo y enorme conocimiento para guiarnos en este reto.

A nuestros amigos y educadores, fieles compañeros en este proceso con su ánimo, colaboración y palabras de aliento.

Para finalizar, nuestros agradecimientos a la Universidad del Cauca, institución que nos forjó como personas, brindándonos la oportunidad a través del programa de Ingeniería de Sistemas de realizar nuestros estudios de pregrado.

Tabla de Contenido

LISTA DE TABLAS

LISTA DE FIGURAS

Presentación

En este documento se encuentran diferentes secciones que contienen la descripción de los conceptos teóricos, la metodología utilizada para el desarrollo del proyecto Agrupamiento de Documentos Web Basado en la Búsqueda Cucú. A continuación se describe de manera general el contenido de esta monografía y su organización.

Capítulo 1 – Introducción: Este capítulo presenta el planteamiento del problema que dio origen al proyecto, la justificación del mismo, objetivos y los resultados obtenidos durante su proceso de desarrollo.

Capítulo 2 – Marco Teórico: Este capítulo enmarca las bases teóricas utilizadas para el desarrollo del proyecto.

Capítulo 3 – Algoritmo CS+LEM y Estudio comparativo: En este capítulo se presenta en detalle los aspectos relacionados con una nueva versión del algoritmo Búsqueda Cucú propuesto en el proyecto y el estudio comparativo realizado entre diferentes versiones del algoritmo Búsqueda Cucú.

Capítulo 4 – Algoritmo WDC-CSK y Experimentación: En este capítulo se presenta en detalle los aspectos relacionados con el algoritmo propuesto para el problema de agrupamiento de documentos Web, el cual está basado en el algoritmo meta-heurístico de la Búsqueda Cucú, el algoritmo K-means y el Criterio de Información Bayesiano Balanceado.

Capítulo 5 – Conclusiones, Recomendaciones y Trabajo Futuro: En este capítulo se presentan las conclusiones generadas después de terminar el proyecto, además se describen posibles mejoras o elementos adicionales que se puedan incluir en un trabajo futuro para la continuidad del proyecto.

Finalmente, Glosario y Bibliografía, donde se presenta un catálogo de palabras con su respectiva definición y las referencias bibliográficas empleadas en la realización del proyecto.

Capítulo 1

1 INTRODUCCIÓN

1.1 PLANTEAMIENTO DEL PROBLEMA

En la actualidad, los motores de búsquedas Web como Google, Yahoo!, Bing y Ask son algunas de las herramientas más utilizadas por las personas para encontrar información en Internet [1]. Sin embargo, los resultados de las búsquedas realizadas en estos motores no son los más indicados, saturando al usuario de información y mostrando que la efectividad de estos buscadores no es completa [2].

Para facilitar este proceso, algunos buscadores se esfuerzan continuamente en mejorar su rendimiento y la forma en que presentan la información al usuario. Una de las técnicas de visualización y análisis que se ha empezado a usar desde hace un tiempo atrás es el agrupamiento de información por afinidad, también conocida como agrupamiento de documentos Web [3]. Estas técnicas de clustering o agrupamiento han sido ampliamente usadas sobre los resultados devueltos por un buscador Web, de tal forma que las páginas relacionadas temáticamente se ubican en un mismo grupo o tema, haciendo más eficientes las búsquedas de información por parte de los usuarios, ya que al estar agrupadas por temas, el usuario puede centrarse en revisar solamente las páginas del grupo o grupos que le interesan, invirtiendo de esta forma menos tiempo en realizar tareas de búsqueda y obteniendo más páginas relevantes [4, 5]. La aplicación de estas técnicas ha sido evaluada en diferentes trabajos, y se han obtenido significativos resultados tanto en entornos de laboratorio, utilizando métricas que evalúan la calidad de los grupos, como en entornos reales evaluando la eficiencia de la recuperación de información. A pesar de todos los avances, los resultados muestran que todavía existe mucho por hacer, por ejemplo, los valores de precisión[1] , recuerdo[2] y medida F[3] reportadas en los estudios más recientes oscilan entre 60% y 84% (dependiendo de la colección de datos) cuando la meta es 100%.

[1] La precisión corresponde al número de documentos relevantes recuperados divididos entre el total de documentos recuperados, P = |Ra| / |A|, donde A es el conjunto de documentos recuperados por el motor de búsqueda, |A| el número de documentos en A, |Ra| es el número de elementos en la intersección del conjunto R y el conjunto A, R es el conjunto de documentos relevantes, es decir, el número total de documentos relevantes recuperados.

[2] El recuerdo corresponde al número de documentos relevantes recuperados dividido entre el total de documentos relevantes, R = |Ra| / |R|, donde |R| es el número de documentos en el conjunto R

[3] La medida F, también conocida como F1 es la media armónica de la precisión y el recuerdo, F = 2 x P x R / (P + R).

Partiendo del hecho que en la bibliografía consultada actualmente, existen diversas investigaciones donde el agrupamiento de documentos Web es tratado como un problema de optimización, que el “Non-free lunch theorem” [6] plantea que ciertas heurísticas pueden ser mejores para resolver ciertos problemas y que el algoritmo meta-heurístico Búsqueda Cucú (Cuckoo Search, CS) ha obtenido mejores resultados en diversos problemas de optimización, en este proyecto se planteó la siguiente pregunta de investigación ¿Es posible proporcionar mejores resultados en cuanto a precisión, recuerdo y medida F en el proceso de agrupamiento de documentos Web (basado en snippets, resúmenes cortos de los documentos generados por los motores de búsqueda tradicionales) con un algoritmo basado en CS?.

Esta pregunta se soporta en que el algoritmo CS presenta mejores resultados en algunas funciones de prueba (tales como las funciones multimodales), como también en la optimización (proceso relacionado con la búsqueda de la mejor manera de utilizar los recursos a partir de unos escenarios dados por parámetros) de procesos de diseño en estructuras de ingeniería [7], vistos en tiempos de respuesta y efectividad de los resultados.

Es así como la presente propuesta de investigación muestra la efectividad de un algoritmo para resolver el problema de agrupamiento de documentos Web basado en el algoritmo meta-heurístico CS. Este algoritmo entrega a los usuarios resultados empleando un tiempo razonable, con una adecuada presentación (mostrando al usuario grupos de documentos y etiquetas relacionados), evitando así que éstos se saturen de información, ayudando a resolver sus necesidades de información y con una mayor cobertura de resultados.

1.2 APORTES

Desde la perspectiva de investigación las contribuciones de este proyecto fueron encaminadas a generar conocimiento que sirve para acercarse aún más a la solución del proceso de agrupamiento de documentos Web, para posteriormente ser utilizado por las comunidades de recuperación de la información en sus desarrollos e investigaciones, teniendo en cuenta que en la actualidad no se ha encontrado ningún reporte definitivo del tema en las bases de datos de IEEE, ACM, ScienceDirect y SpringerLink.

Este nuevo conocimiento de tipo exploratorio y descriptivo, respondió a las preguntas: ¿Es posible lograr una mejora porcentual en la calidad de los grupos y sus etiquetas en el agrupamiento de documentos Web con el uso de un algoritmo que toma como base la Búsqueda Cucú?, ¿Cuáles son las condiciones que hacen

que este algoritmo meta-heurístico mejore o empeore la precisión, recuerdo y medida F del agrupamiento de documentos Web?

Desde la perspectiva de innovación con este proyecto se propuso un nuevo algoritmo basado en la hibridación del algoritmo meta-heurístico Búsqueda Cucú y el algoritmo K-means para el agrupamiento de documentos Web, que refleja un mejor número de grupos de documentos, mayor precisión y recuerdo/exhaustividad que los reportados en la bibliografía y finalmente en cuanto a desarrollo se presenta un prototipo software que muestre el funcionamiento del algoritmo.

1.3 OBJETIVOS

A continuación se muestran los objetivos del proyecto, conforme fueron aprobados por el Consejo de la Facultad de Ingeniería Electrónica y Telecomunicaciones de la Universidad del Cauca en el documento de anteproyecto.

1.3.1 Objetivo general

Proponer un algoritmo para el agrupamiento de documentos Web[4] que hibride la Búsqueda Cucú y el algoritmo K-means, buscando proporcionar mejores resultados en cuanto a precisión, recuerdo y medida F que los reportados en la literatura por Bisecting K-means, STC y Lingo.

1.3.2 Objetivos específicos

1. Definir la mejor estrategia de Búsqueda Cucú, partiendo de las cinco versiones conocidas (a saber: la propuesta de Yang y Deb [8], la propuesta de Rajabioun [9], la propuesta de Valian, Mohanna y Tavakoli [10], la propuesta de Walton, Hassan, Morgan y Brown [11] y la propuesta de Tuba, Subotiv y Stanarevic [12]) sobre un conjunto de funciones de prueba basadas en IEEE CEC 2005 Competition[5] y las restricciones propias del problema de agrupamiento de documentos Web.
2. Modelar un algoritmo de agrupamiento de documentos Web basado en la hibridación[6] de la Búsqueda Cucú (como estrategia global de búsqueda) y el K-means (como estrategia local de búsqueda) de tal forma que se puedan

[4] Se trabajará con documentos en inglés, documentos de los conjuntos de datos previamente seleccionados.
[5] Funciones estándar de prueba, ver enlace: http://www.lri.fr/~hansen/Tech-Report-May-30-05.pdf
[6] En algoritmos evolutivos, la hibridación se refiere a la combinación de dos o más técnicas de optimización en un solo algoritmo, generalmente relacionado con la adición de conocimiento en el interior del algoritmo basado en el conocimiento del problema, ver enlace: http://www.multilingualarchive.com/ma/enwiki/es/Hybridisation

evaluar diferentes funciones de aptitud, esquemas de representación de documentos (basado en el modelo de representación vectorial) y esquemas de creación de etiquetas de los grupos.

3. Definir la calidad del proceso de agrupación de documentos Web con el algoritmo propuesto, basado en precisión y recuerdo ponderado sobre una colección de datos de prueba como la disponible en http://www.unicauca.edu.co/~ccobos/wdc/wdc.htm[7] y comparar los resultados frente a Bisecting K-means, STC y Lingo.

1.4 RESULTADOS OBTENIDOS

- Artículo: "Clustering of Web Search Results based on the Cuckoo Search Algorithm and Balanced Bayesian Information Criterion" en proceso de evaluación en la revista Information Sciences (revista internacional categoría A1 según el PUBLINDEX de COLCIENCIAS).

- Artículo: "A Comparative Study of Different Cuckoo Search Algorithms in Large Scale Continuous Problems" en proceso de evaluación en la revista Applied Mathematics and Computation (revista internacional categoría A1 según el PUBLINDEX de COLCIENCIAS).

- Aplicación Software de Comparación entre las versiones de CS. Utilizada en las pruebas de laboratorio correspondiente a la comparación de las versiones del Algoritmo Búsqueda Cucú, código fuente e instaladores.

- Aplicación Software Asistente. Utilizada para el procesamiento de los datos obtenido tras las comparaciones de las versiones de los algoritmos, código fuente e instaladores.

- Aplicación Software de Laboratorio. Utilizada en las pruebas de laboratorio de agrupamiento de documentos Web, código fuente e instaladores.

- Monografía del trabajo de grado. Corresponde al presente documento, donde se describe el proceso seguido en el desarrollo del proyecto, los aportes más sobresalientes, las conclusiones y las recomendaciones para el desarrollo de futuras investigaciones en el área.

[7] Conjunto de datos estándar de prueba reconocidos dentro de la comunidad académica e investigativa de recuperación de información.

Capítulo 2

2 CONTEXTO TEÓRICO

Teniendo en cuenta que la temática de este proyecto de investigación: "Búsqueda de Agrupamientos Web Basado en la Búsqueda Cucú", dentro de la comunidad académica de la Universidad del Cauca y en general en el país, es novedosa, en este capítulo se hace una introducción a los temas tratados dentro de la investigación y luego se muestra el estado del arte del tema central de investigación.

2.1 BÚSQUEDA CUCÚ

Los algoritmos meta-heurísticos son algoritmos aproximados de propósito general que consisten en procedimientos iterativos que guían una heurística y combinan de forma inteligente distintos conceptos para explorar y explotar adecuadamente el espacio de búsqueda [13]. En el marco de las meta-heurísticas[8] se incluyen los algoritmos genéticos (Genetic Algorithm, GA) [14], los algoritmos meméticos (Memetics Algorithm, MA) [14], la búsqueda Tabú (Tabu Search)[15], la optimización por colonia de hormigas (Ant Colony Optimization) [16], la optimización por enjambre de partículas (Particle Swarm Optimization, PSO) [17], el algoritmo de las abejas (Bee Algorithm) [16, 17], el algoritmo de la libélula (Firefly Algorithm) [17, 18], los algoritmos harmónicos (Harmony Search) [17, 19, 20], entre otros.

En los últimos años, los algoritmos de optimización se han convertido en uno de los campos más importantes de la ciencia y la ingeniería, especialmente cuando se trata de minimizar tiempo, costos, materiales y espacio. Es por eso que actualmente esta área es objeto de investigación y desarrollo en la ciencia, buscando mejorar aspectos como la productividad, el costo y el tiempo de procesamiento [21].

El algoritmo de la Búsqueda Cucú (Cuckoo Search, CS) es un algoritmo meta-heurístico basada en el interesante comportamiento reproductivo parasitario de cría de ciertas especies de cucús, en combinación con el comportamiento de los Vuelos de Lévy de algunas aves o moscas de la fruta [8]. El éxito de este algoritmo, en comparación con los algoritmos genéticos (GA) y los algoritmos de

[8] Un marco de trabajo o enfoque algorítmico de alto nivel que se especializa en resolver problemas de optimización. También se puede definir como una estrategia de alto nivel que guía otras heurísticas en la búsqueda de soluciones factibles (National Institute of Standards and Technology. Ver enlace: http://www.itl.nist.gov/div897/sqg/dads/HTML/metaheuristic.html).

optimización por enjambre de partículas (Particle Swarm Optimization, PSO), ha hecho que sea útil para resolver problemas de optimización de ingeniería, entre los que se destacan el diseño de resortes, el diseño de estructuras de vigas, la fusión de datos en redes de sensores inalámbricos, entre otros [7, 8]. La ventaja de CS frente a GA y PSO reside en el equilibrio de la aleatorización, la intensificación y el menor número de parámetros a controlar [8].

2.1.1 Diferentes versiones del algoritmo

CS es un algoritmo que ha sido objeto de varias modificaciones. Con estas modificaciones se ha logrado algunas mejoras en la precisión y el tiempo de convergencia, además de la disminución de la sensibilidad al ruido. A continuación se describe la versión original del algoritmo de la búsqueda cucú y cuatro modificaciones que serán consideradas en el estudio comparativo, a saber: el Algoritmo de Optimización Cucú basado en K-means (Cuckoo Optimization Algorithm, COA) [9], el Algoritmo de la Búsqueda Cucú Mejorada (Improved Cuckoo Search Algorithm, ICS) [10], la Búsqueda Cucú Modificada (Modified Cuckoo Search, MCS) [11] y el Algoritmo Modificado de la Búsqueda Cucú (Modified Cuckoo Search Algorithm, MCSA)[12].

2.1.1.1 Búsqueda Cucú original (CS)

El algoritmo de la Búsqueda Cucú, propuesto por Yang y Deb [22] en el 2009, proporciona un nuevo camino para la intensificación (propone buscar alrededor de las mejores soluciones actuales y seleccionar las mejores soluciones candidatas) y la diversificación (asegura que el algoritmo puede explorar el espacio de búsqueda de manera más eficiente) [17]. Simplificando el comportamiento de reproducción del cucú, se establece un conjunto de tres reglas idealizadas [8]:

1. Cada cucú pone un huevo a la vez, y lo hace en un nido escogido de forma aleatoria.
2. Los huevos (soluciones) de los mejores nidos (alta calidad) se trasladan a las próximas generaciones.
3. El número de nidos anfitriones disponibles es fijo, y un huevo puesto por un cucú se puede descubrir por el ave anfitrión con una probabilidad $p_a \in [0, 1]$ (p_a, porcentaje de abandonos). En este caso, el ave anfitrión puede ya sea deshacerse del huevo, o simplemente abandonar el nido y construir un nido nuevo.

La principal característica de CS reside en la habilidad para simultáneamente refinar la búsqueda local, mientras todavía busca globalmente. Esto se realiza debido al proceso de abandono de los nidos y la búsqueda a través de los vuelos de Lévy.

2.1.1.1.1 Vuelos de Lévy

En la naturaleza, la mayoría de animales (incluyendo algunas especies de aves cucú) buscan su alimento (o nidos anfitriones en relación con las aves cucú) en forma de un camino aleatorio o cuasi-aleatorio (debido a que el próximo paso se basa en la ubicación actual y la probabilidad de pasar a la siguiente ubicación) que puede ser modelada con una distribución de Lévy (distribución de probabilidad continua para una variable aleatoria no negativa) [17], conocida como vuelos de Lévy [16]. Muchos estudios han demostrado que el comportamiento de vuelo de muchos animales e insectos siguen las características típicas de los vuelos de Lévy. Por ejemplo, un estudio realizado por Reynolds y Frye [8] muestra que las moscas de la fruta o *Dorsophila Melanogaster* exploran su paisaje utilizando una serie de trayectorias de vuelo lineal marcada por un repentino giro de 90°, dando lugar a un patrón de búsqueda intermitente sin escalas similar a un vuelo de Lévy [7]. La **Figura 1** muestra un ejemplo de un vuelo de Lévy en 2 dimensiones.

Teniendo en cuenta los vuelos de Lévy, en la versión original del algoritmo CS el paso “Obtener un nuevo nido de forma aleatoria usando los vuelos de Lévy” se lleva a cabo usando la Ecuación (1).

$$X_i^{(t+1)} = X_i^t + \alpha \oplus \text{Lévy}(\lambda) \quad (1)$$

Donde α representa el tamaño del paso, α > 0; que debe estar relacionado con la escala del problema que el algoritmo intenta resolver. En la mayoría de los casos, α puede establecerse en el valor de 1.

$$\text{Lévy}(\lambda) \approx t^{-\lambda} \quad (2)$$

Donde λ es la longitud del paso, λ ϵ (0, 3]; λ es generado aleatoriamente usando la distribución de Lévy.

La ecuación (1) es en esencia una ecuación estocástica para un camino aleatorio la cual es una cadena de Markov, donde la próxima ubicación (estado) depende de dos parámetros: la ubicación actual (primer término de la ecuación) y la probabilidad de transición (segundo término). El símbolo $\oplus$ representa los multiplicadores “entry-wise”. Este producto es similar al visto en el algoritmo PSO, pero las caminatas aleatorias a través de los vuelos de Lévy son mucho más eficientes para explorar el espacio de búsqueda [10].

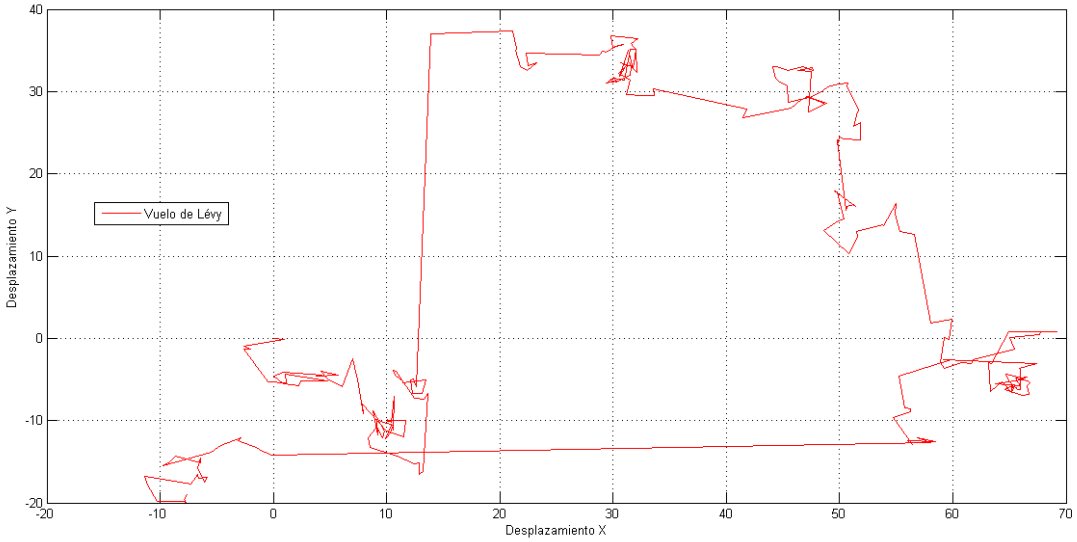

Figura 1. Ejemplo pasos de un vuelo de Lévy en 2-Dimensiones (Propia)

Con base en las tres reglas idealizadas presentadas anteriormente, los pasos básicos de la búsqueda de cucú se pueden resumir en la **Figura 2**.

Inicio
Función Objetivo f(x); $x = (x_1, x_2, \ldots, x_d)^T$
Generar la población inicial de los n nidos huésped x_i (i=1,2,…n)
Evaluar la calidad de todos los nidos en la población inicial
Mientras (t<MaximaGeneracion) **o** (CriteriosDeParada) **Hacer**
Obtener un nuevo nido de forma aleatoria usando los vuelos de Lévy
Evaluar la Calidad/Fitness F_i del nuevo nido
Elegir aleatoriamente un nido entre los n disponibles en la población (Por ejemplo j)
Si (F_i> F_J) **Entonces**
Reemplazar nido j por el nuevo nido
Fin si
Una fracción (p_a) de los peores nidos de la población se abandonan y se reemplazan por nuevos nidos generados aleatoriamente
Mantener las mejores soluciones (o los nidos con soluciones de calidad)
Ordenar las soluciones y encontrar el mejor nido actual
Fin mientras
Post-procesar y visualizar resultados
Fin

Figura 2. Pseudocódigo del algoritmo Búsqueda Cucú vía vuelos de Lévy (Traducción de [8])

2.1.1.2 Algoritmo de Optimización Cucú basado en K-means (COA)

El Algoritmo de Optimización Cucú basado en K-means (Cuckoo Optimization Algorithm, COA) [9], es una nueva versión del algoritmo CS y fue propuesto en el 2011 por Rajabioun. En COA se genera un hábitat (matriz de tamaño N_{pop} x N_{var}, donde N_{pop} representa el tamaño de la población y N_{var} la posición en el hábitat actual) con puntos al azar y para cada uno de ellos se calcula la función de

utilidad. A cada cucú se le asignan algunos huevos (5-20 huevos) y un radio de puesta de huevos (ELR, Egg Laying Radius) que define la distancia máxima donde los cucús pueden alojar sus huevos. En cada generación se van definiendo zonas (grupos) para el hábitat de cada uno de los cucús. Cada zona es evaluada y se le asigna un valor de utilidad; este valor representa la tasa de supervivencia para un individuo y sus huevos, definiéndose a través de la cantidad de individuos que han proliferado en la zona, la disponibilidad de alimentos y las similitudes entre las características de los huevos cucú y las aves huésped. En consecuencia, la zona con mayor valor de utilidad se establece como el punto objetivo, es decir, el mejor hábitat de emigración para los cucús maduros, quienes establecen una trayectoria de vuelo definida por un porcentaje de la distancia λ y una ángulo de desviación ϕ (-π/6, π/6 rad) hacia este hábitat. Generación tras generación los cucús se distribuyen irregularmente sobre el espacio de búsqueda y por esto se hace difícil identificar los cucús que pertenecen a cada grupo, por lo anterior, se usa el algoritmo de agrupamiento K-means (donde k está entre 3-5 grupos) para soportar la definición adecuada de los cucús en cada grupo. La principal ventaja de COA es la rápida convergencia al óptimo global en el menor número de iteraciones. La desventaja de éste algoritmo es el tiempo adicional de ejecución requerido para encontrar la solución óptima el cual puede ser hasta 5 veces mayor en relación con la versión de original de la búsqueda cucú. Los principales pasos de COA son representados en la **Figura 3**.

01	Inicializar los hábitats del cucú con algunos puntos al azar sobre la función de aptitud
02	Asignar algunos huevos a cada cucú
03	Definir ELR para cada cucú
04	Poner los huevos de cucú dentro de su correspondiente ELR
05	Matar los huevos descubiertos por el ave anfitriona
06	Dejar que los huevos Eclosionen y los polluelos crezcan
07	Evaluar el hábitat de cada cucú maduro
08	Limitar el número máximo de cucús en el entorno y matar los que viven en los peores hábitats
09	Agrupar los cucús, encontrar el mejor grupo y seleccionar el hábitat objetivo
10	Emigrar la nueva población de cucús hacia el hábitat objetivo
11	Si la condición de parada se cumple parar, sino ir al paso 02

Figura 3. Pseudocódigo del algoritmo de optimización cucú (Traducción de [9])

2.1.1.3 Búsqueda Cucú Mejorada (ICS)

La principal diferencia de la Búsqueda Cucú Mejorada (Improved Cuckoo Search, ICS) propuesto por Valian, Mohanna y Tavakoli y el algoritmo CS original se encuentra en el modo de ajuste de los parámetros P_a y α (porcentaje de abandonos y tamaño del paso en el vuelo de Lévy, respectivamente). En ICS los valores de P_a y α se cambian dinámicamente con el número de generaciones, usando las ecuaciones (3) y (5). En las primeras generaciones, los valores de P_a y α debe ser lo suficientemente grandes para hacer cumplir que el algoritmo

aumente la diversidad de vectores solución (diversificación). Sin embargo, estos valores se disminuyen en las generaciones finales para dar lugar a un mejor ajuste de los vectores solución (intensificación) [10]. Una clara ventaja de ICS frente a CS reside en el modo de ajuste dinámico de los parámetros del algoritmo que contribuye a mejorar la precisión y radio de convergencia de las soluciones obtenidas.

$$P_a(gn) = P_{a_{max}} - \frac{gn}{NI}(P_{a\,max} - P_{a_{min}}) \quad (3)$$

Donde NI es el número total de iteraciones y gn el número de la iteración actual.

$$c = \frac{1}{NI} Ln\left(\frac{\alpha_{min}}{\alpha_{max}}\right) \quad (4)$$

$$\alpha(gn) = \alpha_{max} \exp(c * gn) \quad (5)$$

2.1.1.4 Búsqueda Cucú Modificada (MCS)

La propuesta de Walton, Hassan, Morgan y Brown, Búsqueda Cucú Modificada (Modified Cuckoo Search, MCS), presenta dos modificaciones al algoritmo CS original. La primera modificación se hizo en el tamaño de los pasos de los vuelos de Lévy (α). El valor de α disminuye a medida que aumenta el número de generaciones, aumentando de esta forma la intensificación. La segunda modificación consiste en añadir un intercambio de información entre los huevos en un intento de acelerar la convergencia a la solución óptima. En el algoritmo CS original, no hay intercambio de información entre individuos y, fundamentalmente, las búsquedas se realizan de forma independiente. En esta versión, una fracción de los huevos con el mejor fitness se ubica en un grupo de huevos “top”. Para los cuales se escoge un segundo huevo de forma aleatoria y un nuevo huevo se genera sobre la línea de conexión de estos dos grupos. En el caso de que ambos huevos tengan el mismo valor de aptitud (fitness), el nuevo huevo se genera en el punto medio. Haciendo uso del método que utiliza la razón aurea (Golden ratio, constante matemática irracional, frecuentemente encontrada cuando se toman las proporciones de las distancias en figuras geométricas simples, como el pentágono, pentagrama (estrella pitagórica), decágono y el dodecaedro, definido por $\varphi = (1 + \sqrt{5})/2)$) se encontró que tiene un rendimiento significativamente mejor que el método original que usa una fracción al azar [11]. La ventaja de MCS más notable es el intercambio de información entre individuos que evita la convergencia prematura hacia óptimas locales controlando la tasa de cambio entre generaciones. Los pasos a seguir en MCS se muestran en detalle en la **Figura 4**.

```
Inicio
    A = MaxTamanioPasoDeLevy
    φ = RadioDeOro
    Inicializar una población de n nidos x_i (i = 1, 2, …, n)
    Para todo x_i Hacer
        Calcular fitness F_i = f (x_i)
    Fin para
    Número de generación G = 1
    Mientras (NumeroEvaluacionesFuncionObjetivo < MaxNumeroDeEvaluaciones) Hacer
        G = G + 1
        Ordenar nidos por orden de fitness
        Para todo nido a ser abandonado Hacer
            Posición actual x_i
            Calcular tamaño vuelo de Lévy α = A/√G
            Realizar un vuelo de Lévy de x_i para generar un nuevo nido x_k
            x_i = x_k
            Fi = f(xi)
        Fin para
        Para  todo nido top Hacer
            Posición actual xi
            Elegir al azar otro nido de los nidos top xj
            Si (xi == xj) Entonces
                Calcular el tamaño del paso de un vuelo de Lévy α = A/G2
                Realizar un vuelo de Lévy de xi para generar un nuevo nido xk
                Fk = f(xk )
                Seleccionar un nido aleatorio l de todos los nidos
                Si (Fk > Fl ) Entonces
                    xl  = xk
                    Fl  = Fk
                Fin si
            Sino
                dx= |xi- xj| / φ
                Mover una distancia dx desde el peor nido al mejor nido
                    para encontrar xk
                Fk  = f(xk )
                Seleccionar un nido aleatorio l de todos los nidos
                Si (Fk > Fl ) Entonces
                    xl  = xk
                    Fl  = Fk
                Fin si
            Fin si
        Fin para
    Fin mientras
Fin
```

Figura 4. Pseudocódigo de la búsqueda cucú mejorada (Traducción de [11])

2.1.1.5 Algoritmo Modificado de la Búsqueda Cucú (MCSA)

En la propuesta de Tuba, Subotiv y Stanarevic, se presenta una variante en la forma de calcular el tamaño de los pasos aleatorios. Esto se hace con la ecuación (6), donde se define una función que permite ordenar la matriz de soluciones candidatas (nidos) por el valor de aptitud (fitness) de las soluciones contenidas.

De esta manera, las soluciones con mayor fitness tienen una ligera ventaja sobre las soluciones con menor valor de aptitud. Este método mantiene la presión de selección (el grado en que las soluciones de alto nivel de fitness son seleccionadas) hacia las mejores soluciones, y por lo tanto le permite alcanzar mejores resultados [12].

r * nest[sorted[i]][j] – nests[permute[i]][j]

Donde r es un número aleatorio en el rango [0, 1], nests es la matriz que contiene las soluciones candidatas (nidos) junto con sus parámetros, permute es la función de permutación de diferentes filas aplicada sobre la matriz nests. (6)

2.2 AGRUPAMIENTO DE DOCUMENTOS WEB

En los últimos años, la agrupación de resultados Web se ha convertido en un área muy interesante de investigación entre las comunidades académicas y científicas involucradas en la recuperación de información (RI) y búsqueda Web [23]. Esta alternativa para representar los resultados se basa en la denominada hipótesis de clúster [24], por lo cual el agrupamiento de documentos podría traer muchas ventajas a los usuarios de un sistema de RI, debido a que es probable que los resultados que son relevantes para el usuario están cerca uno del otro en el espacio del documento y, por tanto, tiendan a quedar en un número relativamente reducido de grupos [25] y con eso se disminuya el tiempo de búsqueda en comparación con el uso de un buscador común, dado que es muy probable que los resultados relevantes para el usuario están cerca unos de otros en el espacio de documentos, y por lo tanto tienden a caer en un número relativamente pequeño de grupos [25], y con ello, lograr una reducción significativa del tiempo de búsqueda. En RI, estos sistemas de agrupamiento de resultados Web son llamados motores de agrupamiento Web (Web Clustering Engines) y los exponentes principales en el campo son Carrot^2 (www.carrot2.org), SnakeT (http://snaket.di.unipi.it), Yippy (http://yippy.com, originalmente llamado Vivisimo y luego Clusty), iBoogie (www.iboogie.com), y KeySrc (http://keysrc.fub.it) [26]. Estos sistemas normalmente consisten de cuatro componentes principales, a saber: adquisición de resultados de búsqueda, pre-procesamiento de entradas, construcción de agrupaciones y etiquetado, y visualización de las agrupaciones resultantes [23] (ver **Figura 5**).

En la **Adquisición de resultados de búsqueda** se parte de una ***consulta*** definida por el usuario, con ella se realiza la búsqueda de documentos en diversas fuentes de datos, en este caso, en motores de búsqueda Web tradicionales como Google, Yahoo! o Bing. En general, los motores de

agrupamiento Web se comportan como meta buscadores que recolectan entre 50 y 200 resultados de los buscadores tradicionales. Los resultados contienen como mínimo una dirección URL, un ***snippet*** (resumen corto del contenido del recurso) y un titulo [23].

A continuación se realiza el **Pre-procesamiento de los resultados de búsqueda**. Con este componente se convierten los resultados de búsqueda (snippets) en una secuencia de términos, frases, cadenas o en general ***atributos*** o características, que son posteriormente usadas por el algoritmo de agrupamiento. Hay una serie de tareas que se llevan a cabo sobre estos resultados de búsqueda, entre ellas: la remoción de caracteres especiales y acentos, la conversión de la cadena a minúsculas, la eliminación de palabras vacías (stop words), derivado de las palabras (proceso conocido como stemming) y el control de términos o conceptos permitidos por un vocabulario [23].

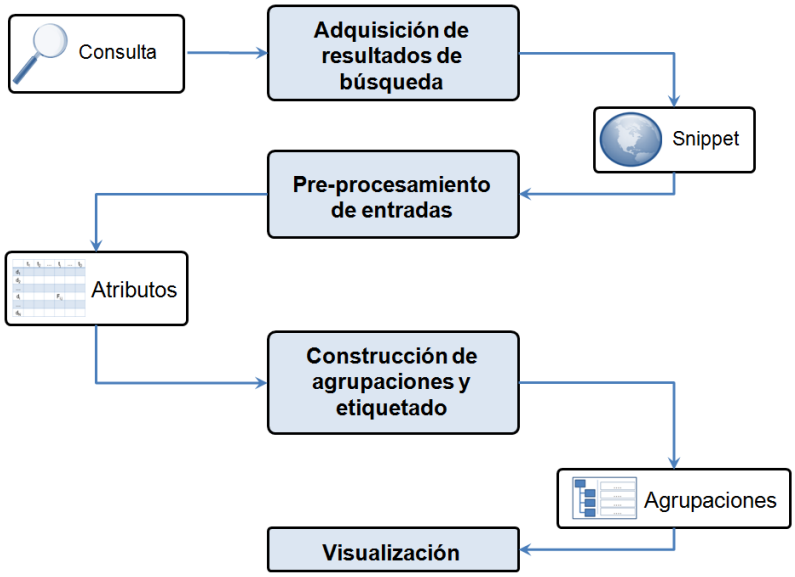

Figura 5. Componentes de un motor de agrupamiento Web (Adaptado de [23])

Terminado el pre-procesamiento se pasa a la **Construcción de agrupaciones y etiquetado de grupos**. Esta etapa hace uso de tres tipos de algoritmo [23]: centrados en datos, conscientes de la descripción y centrados en la descripción. Cada uno de ellos desarrolla grupos de documentos y asigna una etiqueta a los grupos.

Finalmente, en el proceso de **Visualización**, el sistema muestra los resultados al usuario en carpetas organizadas jerárquicamente. Cada carpeta busca tener una etiqueta o título que sea bien representativo de los documentos que contiene y fácil de identificar por parte del usuario; de esta manera, el usuario sólo explora

las carpetas que realmente están relacionadas con sus necesidades específicas. La presentación de árbol de carpetas ha sido adoptada por varios sistemas como Vivisimo, Carrot^2, SnakeT, entre otros, debido a que la metáfora de carpetas ya es familiar para los usuarios de las computadoras. Otros sistemas como Grokker y Kart004 utilizan un esquema de visualización diferente basada en grafos [23].

Para obtener buenos resultados en el agrupamiento de documentos Web los algoritmos deben cumplir con los siguientes requisitos específicos [23, 27]: Definir automáticamente el número de grupos que se van a crear; generar grupos relevantes para el usuario y asignar estos documentos a grupos apropiados; definir etiquetas o nombres de los grupos que sean fácilmente comprensibles para los usuarios del sistema; manejar grupos superpuestos (implica que los documentos pueden pertenecer a más de un grupo); reducir la alta dimensionalidad de las colecciones de documentos; manejar el tiempo de procesamiento lo que implica por ejemplo, que el algoritmo debe poder trabajar con snippets y no sólo con el texto completo del documento; y manejar el ruido con frecuencia en los documentos.

Otro aspecto que es importante al momento de estudiar o proponer un algoritmo que realice agrupación de documentos Web, es el modelo de representación del documento. Los modelos más utilizados son[28]: Modelo de espacio vectorial [29, 30], Indexado Semántico Latente (Latent Semantic Indexing, LSI) [30, 31], modelo basado en ontologías [32, 33], n-gramas [27], modelo basado en frases [27], y Modelo basado en conjuntos de palabras (términos) frecuentes [33, 34].

En el Modelo de Espacio Vectorial (Vector Space Model, VSM), los documentos se conciben como bolsas de palabras. La colección de documentos se representa por una matriz de M-términos por N-documentos. Esta matriz se denomina comúnmente Matriz de términos por documentos (Term by Document Matrix, TDM). En TDM, cada documento es representado por un vector de frecuencia del término 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. En VSM, se usa la similitud de cosenos para medir el grado de similitud entre dos documentos o entre un documento y la consulta del usuario. En VSM como en la mayoría de los modelos de representación, un proceso de remoción de palabras vacías y *stemming* [30] se debe hacer antes de volver a presentar el documento. La eliminación de palabras vacías se refiere a la eliminación de las palabras comunes (como artículos y preposiciones, lo que disminuye la dimensionalidad de la matriz TDM en más de un 40%), mientras stemming se refiere a la reducción de palabras a su lema o raíz común.

Los dos problemas predominantes con los motores de agrupación Web existentes son las inconsistencias en el contenido del grupo (el contenido de un grupo no siempre corresponde con su etiqueta) y las incoherencias en la descripción del grupo (la necesidad de descripciones más expresivas de los grupos; las etiquetas de los grupos son confusas) [23]. Estos dos problemas son la principal motivación de este trabajo, en el que se presenta un nuevo algoritmo que obtiene mejores resultados en cuanto a medida F, recuerdo, exactitud y fall-out para el proceso de agrupamiento de resultados Web que los reportados en el estado del arte. El algoritmo propuesto se basa en el algoritmo meta-heurístico de la búsqueda cucú, el algoritmo K-means, el criterio de información Bayesiano balanceado y los métodos Split y Merge sobre los grupos (usados en remplazo de los vuelos de Lévy del algoritmo búsqueda cucú original). Hasta donde llega el conocimiento de los autores, esta investigación es la primera en integrar sinérgicamente los componentes anteriormente mencionados.

Algoritmos para el Agrupamiento de Documentos Web

Uno de los temas de principal importancia en la agrupación de documentos Web, hace referencia con las etiquetas de los grupos, ya que si ellas son ambiguas o poco descriptivas, los usuarios pueden obviar (pasar por alto) ese grupo de documentos. En este sentido, los algoritmos que realizan agrupación de documentos Web se pueden clasificar en [23]: centrados en los datos, conscientes de la descripción y centrados en la descripción.

Algoritmos centrados en datos: Basados en algoritmos tradicionalmente usados para el agrupamiento (particionales, jerárquicos, difusos, basados en densidad, entre otros) de datos[23, 29, 35-38]. Los cuales buscan la mejor solución en el agrupamiento de los datos, pero no de la presentación de las etiquetas o la explicación de los grupos obtenidos. Tratan el problema de agrupamiento de documentos Web como otro problema más de agrupamiento de datos. Los algoritmos más utilizados para el agrupamiento de resultados Web han sido los jerárquicos y los particionales [29].

Los algoritmos jerárquicos generan un dendograma o árbol de grupos. Este árbol parte de una medida de similitud, entre las que están: enlace simple, enlace completo, enlace promedio. En cuanto al agrupamiento de documentos Web, el algoritmo jerárquico que reporta mejores resultados en exactitud se llama UPGMA (Unweighted Pair-Group Method using Arithmetic averages) [35]. UPGMA es un algoritmo propuesto en 1990[33, 36] que se basa en el modelo de espacio vectorial y usa un enlace promedio basado en la distancia de cosenos de las agrupaciones dividida por el tamaño de las dos agrupaciones que se están

evaluando. UPGMA tiene las desventajas de tener un tiempo de ejecución O (n^3) y de ser estático en el proceso de asignación de los documentos a las agrupaciones.

En agrupamiento particional, los algoritmos 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 o función objetivo[38]. Dentro de los algoritmos más representativos que emplean esta técnica están: K-means, K-medois y Expectation Maximization (Maximización de la Expectativa). El algoritmo K-means es muy popular porque es fácil de implementar y su tiempo de complejidad es O(n), donde n es el número de patrones o registros, pero tiene desventajas muy importantes: la sensibilidad a valores atípicos, la sensibilidad a la selección de los centroides iniciales, la necesidad de definir previamente el número de grupos y sólo contempla formas de agrupación aproximadamente esféricas [27]. En el 2000, se propone el algoritmo Bisecting K-means[29, 33]. Este algoritmo combina las fortalezas de los métodos jerárquicos y particionales, reportando mejores resultados en cuanto a exactitud y eficiencia que UPGMA y K-means

En agrupamiento particional desde un enfoque evolutivo, en el año 2007 se comparan tres métodos de hibridación entre la Búsqueda Armónica (Harmony Search, HS) [39] y el algoritmo K-means. Los métodos fueron: el método Sequential Hybridization (Hibridación Secuencial), el método de Interleaved Hybridization (Hibridación Intercalada) y el método Hybridizing K-means as one step of HS (Hibridación de K-means como un paso de HS). Como resultado general, el último método fue la mejor opción de los tres. Luego en 2008 [39, 40] [13] con base en teoría de cadenas de Markov los investigadores demuestran que el último algoritmo converge al óptimo global. Esta propuesta es un algoritmo centrado en los datos [23] porque entre otras cosas no define el número de grupos de forma automática y no muestra adecuadas etiquetas de los grupos.

En el 2009, se propone un algoritmo basado en enlaces [41]. Este algoritmo utiliza la estructura de hipervínculos de la Web para encontrar unidades densas y también para mejorar el proceso de unión para la creación de grupos jerárquicos de documentos Web. Esta propuesta tiene la ventaja de crear grupos de varias formas (con alta precisión) y de remover los datos anómalos. Para el proceso de unión, este algoritmo utiliza una medida específica que permite determinar de manera dinámica las fronteras del grupo. Los resultados experimentales muestran una calidad de agrupamiento superior a la de otros algoritmos basados en densidad, pero los conjuntos de datos de prueba y los algoritmos con los que se

compara no son los tradicionales (en el estado del arte) del área de investigación. Además, los autores no tienen en cuenta el proceso de etiquetado de los grupos.
También en el 2009 se propone un nuevo algoritmo de aprendizaje basado en K-means y redes neuronales [42]. Esta propuesta utiliza Principal Component Analysis (PCA) para reducir la dimensionalidad de la matriz de documentos (selección de características), Descomposición en Valores Singulares (Singular Value Decomposition, SVD) para encontrar la medida de similitud y una red neural multicapas para reducir el tiempo del proceso del agrupamiento de documentos. El algoritmo se probó con diferentes tipos de páginas Web y los resultados fueron satisfactorios. El sistema puede ser utilizado para agrupar y clasificar las páginas Web descargadas y otros documentos de texto electrónicos, pero el conjunto de datos de prueba y los algoritmos comparados no son los tradicionales del área de investigación del agrupamiento documento Web. Además, esta propuesta no presta atención al proceso de etiquetado de los grupos y requiere de un proceso de entrenamiento que no es "realmente" posible en la agrupación de resultados Web.

Finalmente, en el 2009 se propone un nuevo algoritmo llamado ArteCM [43]. Este algoritmo utiliza un enfoque incremental para el agrupamiento de documentos Web y se basa en dos principios fundamentales: ofrecer la capacidad de incrementar el número de grupos (*k*) de forma adaptativa, y emplear una medida de similitud específica para el dominio del problema (domain-tailored similarity measures). El uso de medidas de similitud implica la necesidad de ir más allá de las representaciones de texto restrictivas como el Modelo de Espacio Vectorial (Vector Space Model, VSM), y adoptar una representación directa de texto. Por consiguiente, se evita una definición explicita de centroides y se sustituye por una basada en conceptos de centroides. ArteCM se comparó con dos variantes de K-means y SOM con resultados satisfactorios en velocidades y calidad de agrupamiento. La solución propuesta incluye el requerimiento de un medida de similitud de domino especifico especializado, dos parámetros (mínima y máxima similitud aceptada) que pueden ser una limitación, ya que una definición eficaz y eficiente de estos componentes puede ser complicada de obtener.

En agrupamiento difuso, FTCA [44] usa un algoritmo de agrupamiento basada en transducción difusa (2010). Los resultados de FTCA son prometedores pero no son comparados sobre conjuntos de datos reconocidos, y tampoco utilizan las medidas apropiadas, que son necesarios para comparar correctamente los resultados del algoritmo con otros algoritmos del estado del arte.

También en el 2010 se propuso el algoritmo de agrupamiento de documentos relacional (RED-clustering) [45]. Este algoritmo tiene en cuenta el contenido de los

datos y la estructura de los hipervínculos de las páginas Web. Los resultados experimentales muestran que supera a K-means y Expectation Maximization, en cuanto a eficacia, pureza y concordancia entre clases y particiones, pero no utilizan conjuntos de datos de prueba tradicionales del área de investigación y tampoco se compara con otros algoritmos del estado del estado del arte.

En el 2011, se presentó un algoritmo que realiza operaciones de división y unión espectral (bisecting y merge) sobre documentos Web, llamado METIS [46]. Las operaciones de división y unión son optimizadas para trabajar con distribuciones de tamaños de grupos sesgadas. Los resultados muestran una mejora en el desempeño de aproximadamente 56% y 36% en comparación con división espectral y K-means respectivamente en términos de medida F, aunque en esta propuesta el número de grupos debe ser previamente definido y los conjuntos de datos usados para las pruebas no son los tradicionales del área de investigación. También, en este año se propuso otro método basado en agrupamiento espectral multi clase para la agrupación de documentos (incluyendo páginas Web en inglés y chino) [47]. El algoritmo comienza a partir de una tradicional matriz de términos por documento (TF-IDF) pero utiliza diferentes algoritmos de preprocesamiento basado en el idioma de la página Web. Para construir la matriz de similitud se utiliza la medida de similitud de cosenos. En general, los resultados son prometedores.

Algoritmos conscientes de la descripción: Estos algoritmos le dan mayor importancia a una característica específica del proceso de agrupamiento que a otras. Por ejemplo, le dan prioridad a la calidad de las etiquetas de los grupos y con ello consiguen resultados que son mejor interpretados por el usuario pero pierden calidad en el proceso de creación de las agrupaciones. Un ejemplo de este tipo de algoritmos es Suffix Tree Clustering (STC) [27, 33] propuesto en 1998, que en forma incremental crea etiquetas fáciles de entender por los usuarios, basado en las frases frecuentes que aparecen en los documentos. STC utiliza un modelo basado en frases para la representación de documentos.

Algoritmos centrados en la descripción: Son algoritmos [23, 25, 31, 33, 48-50] diseñados específicamente para el agrupamiento de documentos Web, buscando un balance entre la calidad de las agrupaciones y la descripción (etiquetado) de las mismas. En 2001, se introdujo el algoritmo SHOC (Semantic, Hierarchical, Online Clustering) [48]. SHOC incluye dos conceptos importantes: frases completas y la definición de agrupaciones continuas (los documentos pueden pertenecer a varias agrupaciones con diferente intensidad). SHOC usa un enfoque de agrupamiento semántico, jerárquico y en línea, que se basa en LSI y frases frecuentes, este mejora a STC. Después en 2003, se diseña el algoritmo

Lingo [31, 51-55]. Este algoritmo es usado por el buscador Carrot^2 y se basa en frases completas y LSI con SVD. Lingo es una mejora de SHOC y STC, y a diferencia de la mayoría de los algoritmos, primero intenta descubrir nombres descriptivos para las agrupaciones y solo después organiza los documentos en las agrupaciones apropiadas. NMF (2003) es otro ejemplo de estos algoritmos. Se basa en factorización de matrices no negativas (Non-Negative Matrix Factorization, NNMF) de la matriz de términos por documentos del cuerpo del documento determinado[56]. Este algoritmo supera el LSI y los métodos de agrupamiento espectral en la precisión del agrupamiento de documentos, pero no se preocupa por las etiquetas de los grupos. En el 2007 se propone el algoritmo Dynamic SVD Clustering (DSC) [25] el cual usa SVD y el árbol de recubrimiento mínimo (Minimum Spanning Tree, MST). Este algoritmo tiene un mejor desempeño que Lingo, puesto que no requiere calcular la SVD completa de la matriz original. Otro enfoque se propone por el algoritmo denominado Pairwise Constraints guided Non-negative Matrix Factorization (PCNMF) [57] (2007). Este algoritmo transforma el problema de agrupamiento de documentos desde un problema no supervisado a un problema semi supervisado usando relaciones debe-enlazarse (especifica que dos documentos deben pertenecer al mismo grupo) y no debe-enlazarse (especifica que dos documentos no deben pertenecer al mismo grupo) entre los documentos.

En el 2009, se presenta un método basado en computación granular (WDCGrc) [58]. Este algoritmo transforma la matriz de términos por documentos (TF-IDF) a una matriz granular binaria por documentos, posteriormente, utilizando un algoritmo de reglas de asociación se obtiene un conjuntos de palabras frecuentes entre documentos. Este conjunto de palabras frecuentes se reduce y finalmente se utiliza para crear grupos. WDCGrc toma el número de las palabras compartidas por documento como medida de similitud. Por último, este artículo muestra que WDCGrc es práctico y viable, con buena calidad de agrupamiento, pero no utilizan conjuntos de datos de prueba estándar y tampoco se compara con otros algoritmos del estado del arte.

En el 2009 también se propuso KeySRC (Recuperación completa de subtemas basado en agrupamiento de frases clave en resultados de búsqueda Web, Full-Subtopic Retrieval with Keyphrase-Based Search Results Clustering) [59], Este algoritmo se basa en frases clave que se extraen de un árbol de sufijos generalizado construido a partir de los resultados de búsqueda. Entonces los documentos se agrupan basados en un algoritmo de agrupamiento jerárquico aglomerativo. En esta propuesta también se presenta una nueva medida para evaluar el desempeño de recuperación completa de subtemas. La métrica se llama SSL_k (Longitud de Búsqueda que logran al menos k documentos de los

subtemas, Subtopic Search Length under k document sufficiency (SSL_k) y actualmente es una de las medidas del estado del arte para evaluar el rendimiento de los motores de agrupamiento Web. Usando la medida propuesta KeySRC supera los algoritmos STC y Lingo sobre el conjunto de datos de AMBIENT.

En el año 2010 se presenta un nuevo enfoque basado en la detección automático del sentido de las palabras, una tarea referida como WSI (Inducción del sentido de las palabras, Word Sense Induction) [60], que busca solucionar el problema de ambigüedad en la consulta por medio de los directorios Web y sistemas de recuperación semántica de información (Semantic Information Retrieval, SIR). En esta propuesta se 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. Los experimentos muestran mejores resultados que los algoritmos STC, Lingo y KeySRC sobre el conjunto de datos de AMBIENT y MORESQUE. En este mismo año un estudio de los problemas de agrupamiento de documentos Web [61] muestra que un algoritmo de optimización discreta estocástica puede proporcionar aproximaciones rápidas a la solución óptima para el problema de agrupamiento de resultados de búsqueda (SRC). El algoritmo propuesto se llama OPTIMSRC (OPTImal Meta Search Results Clustering) y supera los resultados presentados por KeySRC, Lingo, Lingo3G y el orden original de los resultados reportados por el motor de búsqueda de Yahoo!. Para el proceso de etiquetado usan etiquetas generadas por otros algoritmos (por ejemplo, STC o Lingo) y coinciden los grupos generados con la mayoría de las etiquetas apropiadas.

En los años 2010 y 2011, se presentan tres nuevos algoritmos basados en heurísticas, agrupación particional y diferentes estrategias para etiquetado. El primer algoritmo, llamado IGBHSK [62] se basa en la mejor búsqueda armónica global (global-best harmony search), K-means y conjuntos de términos frecuentes. El segundo, llamado WDC-NMA[62] se basa en algoritmos meméticos con técnicas de niching y frases frecuentes. El último, llamado HHWDC [63] se diseña desde un enfoque hiper-heurístico y permite definir el mejor algoritmo para el agrupamiento de documentos Web basado en varias heurísticas de bajo nivel y estrategias de sustitución. Estas investigaciones superan los resultados obtenidos con STC y Lingo, evalúan dos modelos diferentes de representaciones de documentos (matriz de términos por documentos y matriz de términos frecuentes por documentos) y usan el Criterio de Información Bayesiano (Bayesian Information Criterion, BIC) para evaluar la calidad de las soluciones.

En el 2012, se presenta un nuevo algoritmo llamado Topical [64]. El cual Modela el problema de agrupamiento de resultados Web como el problema de etiquetar nodos de grupos de un grafo de temas. Los temas son páginas de Wikipedia identificados por medio de anotadores de temas y los extremos del grafo indican la relación de estos temas. El nuevo grafo se basa en la anotación de Tagme que reemplaza el tradicional paradigma de bolsa de palabras. Este construye un buen agrupamiento de etiquetado en términos de diversificación y cobertura de los snippet de temas, la coherencia del contenido de los grupos, la pertinencia de las etiquetas de los grupos y el pequeño número de grupos balanceados. Finalmente, un estudio realizado a un gran número de usuarios en Amazon Mechanical Turk, el cual tenía por objeto determinar la calidad de las etiquetas de los grupos producidas por este enfoque frente a Clusty y Lingo3G muestra que el algoritmo supera otros enfoques mejorando la medida SSL_k alrededor de 20% en promedio para diferentes valores de k.

Capítulo 3

3 ALGORITMO CS+LEM Y ESTUDIO COMPARATIVO

Para dar cumplimiento al primer objetivo específico propuesto en este proyecto, este capítulo muestra un estudio comparativo de las diferentes versiones del algoritmo búsqueda cucú mediante una serie de pruebas experimentales sobre 61 funciones clásicas usadas en optimización continua, con la finalidad de seleccionar la mejor versión y ver su viabilidad en el problema del agrupamiento de documentos Web. Adicionalmente, se propone una nueva versión del algoritmo CS llamado Búsqueda Cucú usando Modelos Evolutivos que Aprenden y que en adelante se denominará CS+LEM por sus siglas en inglés (Cuckoo Search using Learnable Evolutionary Models). La definición del nuevo algoritmo y los resultados del experimento se detallan a continuación.

3.1 ALGORITMO PROPUESTO: CS+LEM

Inspirados en el concepto de Modelos Evolutivos que Aprenden (Learnable Evolution Model, LEM) propuesto por Michalsky [65], en este trabajo se propone una nueva versión del algoritmo CS. En LEM se emplean técnicas de aprendizaje de máquina para generar nuevas poblaciones adicionalmente al método Darwiniano, aplicado en la computación evolutiva, el cual se basa en mutación y selección natural. Este método puede determinar cuáles individuos de una población (o un conjunto de individuos de poblaciones anteriores) son mejores a otros en la realización de ciertas tareas. Estas razones, expresadas como hipótesis inductivas, se usan para la generación de nuevas poblaciones. Luego, cuando el algoritmo se ejecuta en modo de evolución Darwiniana, usa operaciones aleatorias o semi-aleatorias para la generación de nuevos individuos (usando técnicas de mutación y/o recombinación tradicionales).

En esta investigación, se lleva a cabo el proceso de aprendizaje de máquina mediante el algoritmo propuesto en [66], el cual es responsable del proceso de inferencia de reglas. El proceso de inferencia de reglas define un conjunto de reglas conjuntivas ($P \leftarrow R_1 \wedge R_2 \wedge ... \wedge R_n$), que delimitan las regiones alrededor de las cuales hay una mayor posibilidad de encontrar un mejor valor para cada dimensión x_i (por ejemplo, $LV_{x_i} \leq x_i \leq HV_{x_i}$, donde LV y HV son los límites inferior y superior de las reglas para el valor en la dimensión x_i). Con la conjunción de las reglas (R) para cada dimensión se limita el espacio de búsqueda a las regiones con más posibilidades de generar un óptimo global. El proceso de inferencia de reglas se ejecuta por primera vez inmediatamente después de creada la población

inicial de nidos. Los pasos del proceso de inferencia de reglas, se resumen a continuación:

1. Se escoge de la población actual de nidos el grupo de alto rendimiento y el grupo de bajo rendimiento siguiendo las ecuaciones (7) y (8):

$$\mathrm{Hgroup} = \mathrm{P}_{\mathrm{actual}}(1, \dots, \mathrm{i}) \quad (7)$$

$$\mathrm{Lgroup} = \mathrm{P}_{\mathrm{actual}}(\mathrm{n} - \mathrm{i}, \dots, \mathrm{n})$$

Donde $\mathrm{i} = \lfloor n/2 \rfloor$, n es el número de nidos huésped y $\lfloor n/2 \rfloor$ es el tamaño de los grupos de alto y bajo rendimiento. (8)

2. Se genera una matriz para cada clase i perteneciente a cada grupo. En esta matriz se almacena el valor del atributo en esa posición y el *fitness* correspondiente de la siguiente forma, si es *Hgroup* se asigna 1 y en caso contrario se asigna 0, después se ordena con respecto al valor de su atributo de menor a mayor.
3. Se calcula la probabilidad de la ocurrencia de cada atributo.
4. Se selecciona el atributo con la mayor probabilidad de ocurrencia y se agrega a la lista de reglas *P*.
5. Se repiten los pasos 2, 3 y 4 hasta que se evalúen todos los atributos de cada grupo. La regla resultante es de tipo $P \rightarrow Q$, donde *P* es una conjunción de las reglas que tienen la mayor probabilidad para cada atributo y Q corresponde a la clase 1 (grupo de alto rendimiento).

Si está activado el proceso LEM (cuando la variable LEM es *true*), cada vez que la población de nidos cambia se ejecuta el proceso de inferencia de reglas para actualizar las reglas de cada dimensión. Si el proceso LEM está activado el nuevo cucú generado aleatoriamente usando los vuelos de Lévy se muta en algunas dimensiones. El proceso muta una dimensión con una probabilidad definida por el parámetro denominado tasa de consideración de reglas (Rule Consideration Rate, RCR). Este parámetro (RCR) decide en que porcentaje de las veces se utilizará las reglas. Por defecto el parámetro RCR se establece en 0.5 lo que significa que se crea la mitad de las dimensiones a través de los vuelos de Lévy y la otra mitad basada en las reglas.

Cuando el algoritmo crea un número específico de cucús sin lograr una mejora (Maximum Number of Iterations Without Improvements, MNIWI) en la función de aptitud, el proceso LEM se desactiva. Entonces, si el algoritmo repite esta situación, no mejora el valor de aptitud para un específico MNIWI, el proceso LEM se activa y así sucesivamente.

El nuevo algoritmo propuesto se llama Búsqueda Cucú usando Modelos Evolutivos que Aprenden (CS+LEM) y los pasos del algoritmo se presentan en la **Figura 6**.

Inicio
Función Objetivo f(x); $x = (x_1, x_2, \ldots, x_d)^T$
Generar la población inicial de los n nidos huésped x_i (i=1,2,…n)
LEM = verdadero
IteracionSinMejora = 0
Ejecutar proceso de inferencia de reglas
Mientras (t<MaximaGeneracion) **o** (CriteriosDeParada) **Hacer**
Obtener un nuevo nido de forma aleatoria usando los vuelos de Lévy
Para cada dimensión d[i] del nuevo cucú **Hacer**
Si (LEM) **Entonces**
Si (U(0,1) < RCR) **Entonces**
d[i] = $U(LB_i^{best}, UB_i^{best})$, donde ***best*** es el mejor conjunto de reglas en el Límite Superior (*Lower Bound, LB*) y (*Upper Bound, UB*) Límite Inferior para la dimensión actual
Fin si
Fin si
Fin para cada
Evaluar la Calidad/Fitness de F_i
Elegir aleatoriamente un nido entre los n disponibles en la población (Por ejemplo j)
Si (F_i> F_j) **Entonces**
Reemplazar nido j por el nuevo nido
Si (LEM) **Entonces**
Ejecutar el proceso de inferencia de la regla
Fin si
Sino
IteracionesSinMejora ++
Si (IteracionesSinMejora > MNIWI) **Entonces**
IteracionesSinMejora = 0
LEM = no LEM
Fin si
Fin si
Una fracción (Pa) de los peores nidos de la población se abandonan y se reemplazan por nuevos nidos generados aleatoriamente
Mantener las mejores soluciones (o los nidos con soluciones de calidad)
Ordenar las soluciones y encontrar el mejor nido actual
Fin mientras
Post-procesar y visualizar resultados
Fin

Figura 6. Pseudocódigo de CS+LEM (Propia)

3.2 SISTEMA SOFTWARE COMPARATIVO (PCS)

Para el desarrollo de la experimentación, se realizó un prototipo software que permitió efectuar las comparaciones entre las diferentes versiones del algoritmo CS mediante una serie de pruebas sobre 61 funciones clásicas usadas en optimización continua, éste prototipo recibe el nombre de Project Cuckoo Search (PCS). Para más detalles sobre el proceso de desarrollo refiérase al Anexo A – Implementación del Sistema Comparativo (PCS).

3.3 EXPERIMENTACIÓN

Con el desarrollo del software de laboratorio PCS, se realizaron las evaluaciones de las versiones del algoritmo CS. En esta sección se muestra el desempeño de las versiones del algoritmo CS en tres diferentes evaluaciones, a saber: 1) mejor valor óptimo alcanzado en diferentes períodos de tiempo, 2) número de evaluaciones de la función objetivo requeridas para alcanzar el óptimo global, 3) mejor valor óptimo alcanzado en diferente número de evaluaciones de la función objetivo. Éste proceso tuvo como objetivo seleccionar la mejor versión del algoritmo CS y ver su viabilidad en el problema del agrupamiento de documentos Web. El proceso realizado y los resultados obtenidos se presentan a continuación.

3.3.1 Pregunta de investigación

A partir del primer objetivo específico del proyecto se evidenció la necesidad de identificar la mejor versión del algoritmo de la Búsqueda Cucú y su viabilidad en el problema de agrupamiento de documentos Web. Así la pregunta que éste estudio comparativo busca responder es:

¿Cuál es la mejor versión del algoritmo de la Búsqueda Cucú para el problema de agrupamiento de documentos Web?

3.3.2 Objetivo del estudio comparativo

El objetivo de este estudio comparativo es definir la mejor estrategia del algoritmo de la Búsqueda Cucú, partiendo de las seis versiones conocidas y ver su viabilidad en la solución del problema de agrupamiento de documentos Web.

3.3.3 Contexto del estudio comparativo

Para la realización del estudio comparativo se empleó el prototipo software PCS que permitió efectuar las comparaciones entre las diferentes versiones del algoritmo CS mediante 3 tipos diferentes de evaluación sobre una serie de pruebas sobre 61 funciones clásicas usadas en optimización continua. Cada evaluación fue ejecutada 30 veces para garantizar resultados confiables.

3.3.4 Diseño del estudio comparativo

De acuerdo al objetivo del estudio comparativo, se diseñaron los indicadores, métricas e instrumentos a emplear, la **Tabla 1** relaciona estos elementos para el estudio comparativo.

Tabla 1. Parámetros estudio comparativo.

Objetivo	Indicadores	Métricas	Instrumentos
Definir la mejor estrategia del algoritmo de la Búsqueda Cucú, partiendo de las seis versiones conocidas y ver su viabilidad en la solución del problema de agrupamiento de documentos Web.	Mejor valor óptimo de la función objetivo.	Mejor valor en diferentes periodos de tiempo.	Registros de resultados de tiempo.
		Mínimo número de evaluaciones de la función objetivo.	Registro de resultados de evaluaciones de la función objetivo.
		Mejor valor en diferentes números de evaluaciones de la función objetivo.	

3.3.5 Configuración de las pruebas

Cada versión del algoritmo CS tiene parámetros específicos definidos por sus autores para el proceso de optimización. En la **Tabla 2** se presenta los parámetros usados para ejecutar todos los algoritmos en cada experimento.

Tabla 2. Configuración general de las pruebas.

Variable	CS	COA	ICS	MCS	MCSA	CS+LEM
Número de Nidos (Number of nests, Nn)	30	30	30	30	30	30
Número de Dimensiones (Number of dimensions, Nd)	30	30	30	30	30	30
Porcentaje de Abandonos (Percentages of abandonments, Pa)	0.25	N.A.	0.25	0.75	0.25	0.25
Máximo Número de Cucús (Maximum number of cuckoos , Mc)	N.A.	50	N.A.	N.A.	N.A.	N.A.
Número de Grupos (Number of clusters , k)	N.A.	2	N.A.	N.A.	N.A.	N.A.
Máximo Número de Generaciones (Maximum number of Generations , Gn)	50.000	N.A.	50.000	N.A.	50.000	50.000
Máximo Número de Iteraciones (Maximum number of iterations , Mi)	N.A.	N.A.	50.000	N.A.	N.A.	N.A.
Máximo Número de Evaluaciones de la Función Objetivo (Maximum number of evaluaciones de la función objetivo , Me)	N.A.	N.A.	50.000	N.A.	N.A.	N.A.
Máximo Número de Pasos (Maximum number of Steps, Ms)	N.A.	N.A.	N.A.	100	N.A.	N.A.
Máximo Número de Pasos de Lévy (Maximum Lévy step size, Ml)	N.A.	N.A.	N.A.	0.01	N.A.	N.A.
Tasa de Consideración de reglas (Rule Consideration Rate, RCR)	N.A.	N.A.	N.A.	N.A.	N.A.	0.5
Máximo número de Iteraciones sin Mejora (Maximum Number of Iterations Without Improvements, MNIWI)	N.A.	N.A.	N.A.	N.A.	N.A.	120
Ancho de Banda (Bandwidth, Bw)	N.A.	N.A.	N.A.	N.A.	N.A.	0.001

Para el desarrollo de las pruebas de laboratorio las especificaciones del equipo de cómputo utilizado son las siguientes: Procesador Intel Xeon de 2.40 GHz, 16 GB de Memoria RAM y Sistema operativo Windows 7 de 64 Bits.

3.3.6 Funciones de prueba usadas en la evaluación

Para el estudio comparativo se usaron las funciones de prueba que se muestran en la **Tabla 3**, la cual está formada por 6 funciones unimodales separables, 22 funciones unimodales no separables, 7 funciones multimodales separables y 26 funciones multimodales no separables. Estas funciones se basan en las propuestas en los artículos “Benchmark Functions for the CEC’2010 Special Session and Competition on Large-Scale Global Optimization” [67-69] y “A comparative study of Artificial Bee Colony algorithm” [18] que proporcionan un balance adecuado de niveles de complejidad. Para cada una de las funciones se busca encontrar el mínimo global definido como:

Dado $f = \Re^{N_d} \rightarrow \Re$ encontrar $x^* \in \Re^{N_d}$ tal que $f(x^*) \leq f(x), \forall x \in \Re^{N_d}$, donde N_d, es el número de dimensiones.

Todas las funciones de prueba usadas fueron implementadas para 30 dimensiones, excepto Beale, Easom, Matyas, Schaffer, Six-Hump Camel-Back, Bohachevsky 2, Bohachevsky 3 y Goldstein & Price las cuales tienen 2 dimensiones, Colville, Perm y Power Sum tienen 4 dimensiones, StepInt, Langerman 5 y Shubert tienen 5 dimensiones, Hartman 6 tiene 6 dimensiones, Kowalik tiene 11 dimensiones y Shekel’s Foxholes la cual tiene 24 dimensiones.

Para cada evaluación, se reportaron la media y desviación estándar de 30 ejecuciones para cada función. En cada caso se especifica cuántas dimensiones se utilizaron en la prueba específica, además del número de iteraciones específico. La población inicial se genera aleatoriamente dentro de los rangos especificados para cada función.

Tabla 3. Funciones de referencia (D: Dimensión, C: Característica, U: Unimodal, M: Multimodal, S: Separable, N: No Separable)

No	Rango	D	C	Función	Fórmula
1	[-5.12, 5.12]	5	US	StepInt [18]	$f(x) = 25 + \sum_{i=1}^{N_d} \lfloor x_i \rfloor$
2	[-100, 100]	30	US	Step [12]	$f(x) = \sum_{i=1}^{N_d} (\lfloor x_i + 0.5 \rfloor)^2$
3	[-100,100]	30	US	Sphere (De Jong's First Function[10, 69]	$f(x) = \sum_{i=1}^{N_d} {x_i}^2$
4	[-100,100]	30	US	Sum of Squares [18]	$f(x) = \sum_{i=1}^{N_d} i{x_i}^2$
5	[-1.28, 1.28]	30	US	Quartic (De Jong's Forth Function) [18]	$f(x) = \sum_{i=1}^{N_d} i{x_i}^4 + \text{random}[0,1)$
6	[-1, 1]	30	US	Sum of Different Power [70]	$f(x) = \sum_{i=1}^{N_d} \lvert x_i \rvert^{i+1}$
7	[-4.5, 4.5]	2	UN	Beale [18]	$f(x) = (1.5 - x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1x_2^2)^2 + (2.625 - x_1 + x_1x_2^3)^2$
8	[-100, 100]	2	UN	Easom [69]	$f(x) = -\cos(x_1)\cos(x_2)\, e^{(-(x_1-\pi)^2-(x_2-\pi)^2)}$
9	[-10, 10]	2	UN	Matyas [18]	$f(x) = 0.26(x_1^2 + x_2^2) - 0.48x_1x_2$
10	[-10, 10]	4	UN	Colville [18]	$f(x) = 100(x_1^2 - x_2)^2 + (x_1 - 1)^2 + (x_3 - 1)^2 + 90(x_3^2 - x_4)^2 + 10.1((x_2 - 1)^2 + (x_4 - 1)^2) + 19.8(x_2 - 1)(x_4 - 1)$
11	[-5, 10]	30	UN	Zakharov [18]	$f(x) = \sum_{i=1}^{N_d} x_i^2 + \left(\sum_{i=1}^{N_d} 0.5 i x_i\right)^2 + \left(\sum_{i=1}^{N_d} 0.5 i x_i\right)^4$
12	[-10, 10]	30	UN	Schwefel's Problem 2.22 [18]	$f(x) = \sum_{i=1}^{N_d} \lvert x_i \rvert + \prod_{i=1}^{N_d} \lvert x_i \rvert$
13	[-10, 10]	30	UN	Schwefel's Problem 1.2 [18, 69]	$f(x) = \sum_{i=1}^{N_d} \left(\sum_{j=1}^{i} x_j\right)^2$
14	[-10, 10]	30	UN	Rosenbrock [69]	$f(x) = \sum_{i=1}^{N_d - 1} [100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2]$
15	[-100, 100]	30	UN	Dixon-Price [12]	$f(x) = (x_1 - 1)^2 + \sum_{i=2}^{N_d} i(2x_1^2 - x_{i-1})^2$
16	[-100, 100]	30	UN	Single-group Shifted and m-rotated Elliptic [69]	$f(x) = F_{\text{rot\_elliptic}}[z(P_1 : P_m)] * 10^6 + F_{\text{elliptic}}[z(P_{m+1} : P_{N_d})]$
17	[-100, 100]	30	UN	Single-group Shifted m-dimensional Schwefel's Problem 1.2 [69]	$f(x) = F_{\text{schwefel}}[z(P_1 : P_m)] * 10^6 + F_{\text{sphere}}[z(P_{m+1} : P_{N_d})]$

No	Rango	D	C	Función	Fórmula
18	[-100, 100]	30	UN	D/2m-group Shifted and m-rotated Elliptic [69]	$f(x) = \sum_{k=1}^{\frac{N_d}{2m}} F_{rot_{elliptic}}\left[z\left(P_{(k-1)*m+1}:P_{k*m}\right)\right] + F_{elliptic}\left[z\left(P_{\frac{N_d}{2}+1}:P_{N_d}\right)\right]$
19	[-100, 100]	30	UN	D/2m-group Shifted m-dimensional Schwefel's Problem 1.2 [69]	$f(x) = \sum_{k=1}^{\frac{N_d}{2m}} F_{schwefel}\left[z\left(P_{(k-1)*m+1}:P_{k*m}\right)\right] + F_{sphere}\left[z\left(P_{\frac{N_d}{2}+1}:P_{N_d}\right)\right]$
20	[-100, 100]	30	UN	D/m-group Shifted and m-rotated Elliptic [69]	$f(x) = \sum_{k=1}^{\frac{N_d}{m}} F_{rot\_elliptic}\left[z\left(P_{(k-1)*m+1}:P_{k*m}\right)\right]$
21	[-100, 100]	30	UN	D/m-group Shifted m-dimensional Schwefel's Problem 1.2 [69]	$f(x) = \sum_{k=1}^{\frac{N_d}{m}} F_{schwefel}\left[z\left(P_{(k-1)*m+1}:P_{k*m}\right)\right]$
22	[-100, 100]	30	UN	Shifted Schwefel's Problem 1.2 [69]	$f(x) = F_{schwefel}(z) = \sum_{i=1}^{N_d}\left(\sum_{j=1}^{i} x_i\right)^2$
23	[-100, 100]	30	UN	Shifted Elliptic [69]	$f(x) = F_{elliptic}(z) = \sum_{i=1}^{N_d}(10^6)^{\frac{i-1}{N_d-1}} z_i^2$
24	[-100, 100]	30	UN	Shifted Schwefel's Problem 1.2 with Noise in Fitness [67]	$f(z) = \left(\sum_{i=1}^{N_d}\left(\sum_{j=1}^{i} z_j\right)^2\right) * (1 + 0.4\lvert N(0,1)\rvert)$
25	[-100, 100]	30	UN	Shifted Rotated High Conditioned Elliptic [67]	$f(z) = \sum_{i=1}^{N_d}(10^6)^{\frac{i-1}{Nd-1}} z_i^2$
26	[-100, 100]	30	UN	Elliptic [69]	$f(x) = \sum_{i=1}^{N_d}(10^6)^{\frac{i-1}{Nd-1}} x_i^2$
27	$[-N_d, N_d]$	10	UN	Trid 10 [18]	$f(x) = \sum_{i=1}^{N_d}(x_i - 1)^2 - \sum_{i=2}^{n} x_i x_{i-1}$
28	[-4, 5]	24	UN	Powell [18]	$f(x) = \sum_{i=1}^{N_d/k}(x_{4i-3} - 10x_{4i-2})^2 + 5(x_{4i-1} - x_{4i})^2 + (x_{4i-2} - 10x_{4i-1})^4 + 10(x_{4i-3} - 10x_{4i})^4$
29	[-100, 100]	2	MS	Bohachevsky 1 [18]	$f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 10$
30	[-10, 10]	2	MS	Booth [18]	$f(x) = (x_1 + 2x_2 - 7)^2 + (2x_2 + x_2 - 5)^2$
31	[-100, 100]	30	MS	Rastrigin[69]	$f(x) = 10N_d + \sum_{i=1}^{N_d}(x_i{}^2 - 10\cos(2\pi x_i))$
31	[-100, 100]	30	MS	Rastrigin[69]	$f(x) = 10N_d + \sum_{i=1}^{N_d}(x_i{}^2 - 10\cos(2\pi x_i))$
32	[-100, 100]	30	MS	Generalized Schwefel [69]	$f(x) = \sum_{i=1}^{N_d}(-x_i \sin(\sqrt{x_i}))$

No	Rango	D	C	Función	Fórmula
33	[-0, π]	30	MS	Michalewicz 10 [69]	$f(x) = -\sum_{i=1}^{N_d} \sin(x_i)\left[\sin\frac{ix_i^2}{\pi}\right]^{2m}$
34	[-5, 5]	30	MS	Shifted Rastrigin [69]	$f(x) = F_{rastrigin}(z) = \sum_{i=1}^{N_d}[z_i^2 - 10\cos(2\pi z_i) + 10]$
35	[-32, 32]	30	MS	Shifted Ackley [69]	$f(x) = F_{ackley}(z) = -20e^{\left(-0.2\sqrt{\frac{1}{Nd}\sum_{i=1}^{N_d} z_i^2}\right)} - e^{\left(\frac{1}{Nd}\sum_{i=1}^{N_d}\cos 2\pi z_i\right)} + 20 + e$
36	[-100, 100]	2	MN	Schaffer [18]	$f(x) = 0.5 + \frac{\sin^2\left(\sqrt{x_1^2 + x_2^2}\right) - 0.5}{\left(1 + 0.001(x_1^2 + x_2^2)\right)^2}$
37	[-5, 5]	2	MN	Six-Hump Camel-Back [18]	$f(x) = 4x_1^2 + 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$
38	[-100, 100]	2	MN	Bohachevsky 2 [18]	$f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1)(4\pi x_2) + 0.3$
39	[-100, 100]	2	MN	Bohachevsky 3 [18]	$f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1 + 4\pi x_2) + 0.3$
40	[-10, 10]	5	MN	Shubert [69]	$f(x) = \left(\sum_{i=1}^{N_d} i\cos((i+1)x_1 + i)\right)\left(\sum_{i=1}^{N_d} i\cos((i+1)x_2 + i)\right)$
41	[-2, 2]	2	MN	Goldstein & Price [70]	$f(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)][30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$
42	[-600, 600]	30	MN	Griewank [69]	$f(x) = \frac{1}{4000}\sum_{i=1}^{N_d} x_i^2 - \prod_{i=1}^{N_d}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$
43	[-32, 32]	30	MN	Ackley [69]	$f(x) = -20e^{\left(-0.2\sqrt{\frac{1}{N_d}\sum_{i=1}^{N_d} x_i^2}\right)} - e^{\left(\frac{1}{Nd}\sum_{i=1}^{N_d}\cos 2\pi x_i\right)} + 20 + e$
44	[-5, 5]	30	MN	Single-group Shifted and m-rotated Rastrigin [69]	$f(x) = F_{rot\_rastrigin}[z(P_1 : P_m)] * 10^6 + F_{rastrigin}[z(P_{m+1} : P_{N_d})]$
45	[-100, 100]		MN	Single-group Shifted and m-rotated Ackley [69]	$f(x) = F_{rot\_ackley}[z(P_1 : P_m)] * 10^6 + F_{ackley}[z(P_{m+1} : P_{N_d})]$
46	[-100, 100]	30	MN	Single-group Shifted m-dimensional Rosenbrock [69]	$f(x) = F_{rosenbrock}[z(P_1 : P_m)] * 10^6 + F_{sphere}[z(P_{m+1} : P_{N_d})]$
47	[-5, 5]	30	MN	D/2m-group Shifted and m-rotated Rastrigin [69]	$f(x) = \sum_{k=1}^{\frac{N_d}{2m}} F_{rot\_rastrigin}[z(P_{(k-1)*m+1} : P_{k*m})] + F_{rastrigin}\left[z\left(P_{\frac{N_d}{2}+1} : P_{N_d}\right)\right]$
48	[-32, 32]	30	MN	D/2m-group Shifted and m-rotated Ackley [69]	$f(x) = \sum_{k=1}^{\frac{N_d}{2m}} F_{rot_{ackley}}[z(P_{(k-1)*m+1} : P_{k*m})] + F_{ackley}\left[z\left(P_{\frac{N_d}{2}+1} : P_{N_d}\right)\right]$
49	[-100, 100]	30	MN	D/2m-group Shifted m-dimensional Rosenbrock [69]	$f(x) = \sum_{k=1}^{\frac{N_d}{2m}} F_{rosenbrock}[z(P_{(k-1)*m+1} : P_{k*m})] + F_{sphere}\left[z\left(P_{\frac{N_d}{2}+1} : P_{N_d}\right)\right]$
50	[-5, 5]	30	MN	D/m-group Shifted and m-rotated Rastrigin [69]	$f(x) = \sum_{k=1}^{\frac{N_d}{m}} F_{rot\_rastrigin}[z(P_{(k-1)*m+1} : P_{k*m})]$
51	[-32, 32]	30	MN	D/m-group Shifted and m-rotated Ackley [69]	$f(x) = \sum_{k=1}^{\frac{N_d}{m}} F_{rot\_ackley}[z(P_{(k-1)*m+1} : P_{k*m})]$

No	Rango	D	C	Función	Fórmula
52	[-100, 100]	30	MN	D/m-group Shifted m-dimensional Rosenbrock [69]	$f(x) = \sum_{k=1}^{\frac{N_d}{m}} F_{rosenbrock}[z(P_{(k-1)*m+1}:P_{k*m})]$
53	[-100, 100]	30	MN	Shifted Rosenbrock [69]	$f(x) = F_{ackley}(z) = \sum_{i=1}^{N_d-1} [100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2]$
54	[-100, 100]	30	MN	Shifted Rotated Expanded Scaffer's F6 [67]	$F(x,y) = 0.5 + \frac{\left(\sin\sqrt{x^2+y^2}\right)^2 - 0.5}{\left(1 + 0.001 * (x^2+y^2)\right)^2},$ $f(x) = F(x_1, x_2) + F(x_2, x_3) + .. + F(x_{N_d-1}, x_{N_d}) + F(x_{N_d}, x_1)$
55	[-5, 5]	30	MN	Shifted Rotated Weierstrass [67]	$f(x) = \sum_{i=1}^{N_d} \left(\sum_{k=0}^{kmax} \left[a^k * \cos\left(2\pi b^k * (x_i + 0.5)\right) \right] \right) - D \sum_{k=0}^{kmax} [a^k * \cos(2\pi b^k * 0.5)]$
56	[-65.536, 65.536]	24	MN	Shekel's Foxholes [70]	$f(x) = \frac{1}{500} + \sum_{j=0}^{N_d} \frac{1}{c_j + \sum_{i=0}^{1} (x_i - A_{i,j})^6}$
57	[-5, 5]	11	MN	Kowalik [18]	$f(x) = \sum_{i=1}^{N_d} \left(a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 x_4} \right)^2$
58	$[-N_d, N_d]$	4	MN	Perm [18]	$f(x) = \sum_{k=1}^{N_d} \left[\sum_{i=1}^{N_d} (i^k + \beta) \left(\left(\frac{x_i}{i}\right)^k - 1 \right) \right]^2$
59	$[0, N_d]$	4	MN	Power Sum [18]	$f(x) = \sum_{i=1}^{N_d} \left[\left(\sum_{j=1}^{Nd} x_j^i \right) - b_i \right]^2$
60	[0, 1]	6	MN	Hartman 6 [18]	$f(x) = -\sum_{i=1}^{4} c_i e^{\left[-\sum_{j=1}^{N_d} a_{ij}(x_j - p_{ij})^2\right]}$
61	[0, 10]	5	MN	Langerman 5 [70]	$f(x) = -\sum_{i=1}^{N_d} c_i \left(e^{\left(-\frac{1}{\pi}\sum_{j=1}^{N_d}(x_j - a_{ij})^2\right)} \cos\left(\pi \sum_{j=1}^{N_d}(x_j - a_{ij})^2\right) \right)$

3.3.7 Métodos estadísticos no paramétricos

Los métodos no paramétricas incluyen las técnicas de selección a utilizarse en condiciones en las que no existen supuestos sobre la distribución de los parámetros de la población. Se aplican con mayor frecuencia a los datos nominales y ordinales [71]. En este trabajo se usaron dos modelos comunes no-paramétricos o de tipo de distribución libre: la pruebas de Wilcoxon y prueba de Friedman. A continuación se especifican las caracterizan principales de éstas pruebas:

- **Prueba de Wilcoxon:** Se caracteriza por que permite contrastar la hipótesis de igualdad entre dos medianas poblacionales y es paralela a la prueba paramétrica de contraste t para muestras relacionadas.

- **Prueba de Friedman:** Se caracteriza por que es una extensión de la prueba de Wilcoxon para incluir datos registrados en más de dos periodos de tiempo o grupos de tres o más sujetos pareados, con un sujeto de cada grupo que ha sido asignado aleatoriamente a una de las tres o más condiciones; y la prueba examina los rangos de los datos generados en cada periodo de tiempo para determinar si las variables comparten la misma distribución continua de su origen.

Las pruebas estadísticas no paramétricas se escogieron debido a las siguientes características:

1. Son más fáciles de aplicar.
2. Son aplicables a los datos jerarquizados.
3. Se pueden usar cuando dos series de observaciones provienen de distintas poblaciones.
4. Son la única alternativa cuando el tamaño de muestra es pequeño.
5. Son útiles a un nivel de significancia previamente especificado.

Las evaluaciones aplicadas a las diferentes versiones del algoritmo CS y sus respectivos resultados aplicando las pruebas estadísticas no paramétricas se presentan en la siguiente sección.

3.3.8 Evaluación 1: Mejor valor óptimo alcanzado en diferentes períodos de tiempo

Esta evaluación tiene como objetivo identificar que versión del algoritmo CS proporciona los mejores resultados (los resultados más cercanos al mínimo global de la función objetivo) para los siguientes períodos de tiempo: 5, 10, 20, 40 y 80

segundos. Las siguientes secciones muestran los resultados en general (sobre todas las funciones de prueba), sobre las funciones unimodales separables, las funciones unimodales no separables, las funciones multimodales separables y las funciones multimodales no separables.

3.3.8.1 Evaluación 1: Resultados generales

En la **Tabla 4** se presentan los resultados generales obtenidos en la Prueba de Friedman por las versiones del algoritmo CS en 5 periodos de tiempo, a saber: 5, 10, 20, 40 y 80 segundos.

Tabla 4. Clasificación de las pruebas de Friedman por regla de oro para la evaluación relacionada con tiempo.

Tiempo	5 Segundos		10 Segundos		20 Segundos		40 Segundos		80 Segundos	
Algoritmo	Rango Promedio	Posición	Rango Promedio	Posición	Rango Promedio	Posición	Rango Promedio	Posición	Rango Promedio	Posición
CS	3.4508	4	3.2951	3	3.3689	3	3.4508	4	3.4508	3
COA	3.6230	6	3.7623	5	3.6803	5	3.4098	3	3.4918	5
ICS	3.4508	3	3.4918	4	3.4590	4	3.4016	2	3.4180	4
MCS	3.5738	5	4.1557	6	4.1721	6	4.1967	6	4.2049	6
¦MCSA	3.4508	2	3.2459	2	3.2869	2	3.4590	5	3.3852	2
CS+LEM	**3.4508**	**1**	**3.0492**	**1**	**3.0328**	**1**	**3.0820**	**1**	**3.0492**	**1**

- En 5 segundos de ejecución, CS+LEM reporta el mejor valor óptimo con una estadística de Friedman (distribuida de acuerdo al chi-cuadrado con 5 grados de libertad) igual a 0.526932 y el valor P calculado por la prueba de Friedman de Friedman igual a 0.9911039254339233 (ver **Tabla 4**). Además, los resultados de CS+LEM superan a COA, MCS y MCSA; e ICS y MCSA superan a COA y MCS con un nivel de confianza igual a 95% en los resultados de la prueba de Wilcoxon.

- En 10 segundos de ejecución, CS+LEM reporta el mejor valor óptimo con una estadística de Friedman (distribuida de acuerdo al chi-cuadrado con 5 grados de libertad) igual a 14.093677 y el valor P calculado por la prueba de Friedman de Friedman igual a 0.015025191367560087 (ver **Tabla 4**). Además, los resultados de CS+LEM superan los resultados de los otros algoritmos; y MCSA supera a CS, COA, ICS y MCS con un nivel de confianza igual a 95% en los resultados de la prueba de Wilcoxon.

- En 20 segundos de ejecución, CS+LEM reporta el mejor valor óptimo con una estadística de Friedman (distribuida de acuerdo al chi-cuadrado con 5 grados de libertad) igual a 13.36534 y el valor P calculado por la prueba de Friedman de Friedman igual a 0.020185395790274008 (ver **Tabla 4**). Además, los resultados de CS+LEM superan los resultados de los otros algoritmos; y MCSA

supera a CS, COA, ICS y MCS con un nivel de confianza igual a 95% en los resultados de la prueba de Wilcoxon.

- En 40 segundos de ejecución, CS+LEM reporta el mejor valor óptimo con una estadística de Friedman (distribuida de acuerdo al chi-cuadrado con 5 grados de libertad) igual a 11.887588 y el valor P calculado por la prueba de Friedman de Friedman igual a 0.03636097851214215 (ver **Tabla 4**). Además, los resultados de CS+LEM superan los resultados de los otros algoritmos, y COA mejora a CS, ICS, MCS y MCSA con un nivel de confianza igual a 95% en los resultados de la prueba de Wilcoxon.

- En 80 segundos de ejecución, CS+LEM reporta el mejor valor óptimo con una estadística de Friedman (distribuida de acuerdo al chi-cuadrado con 5 grados de libertad) igual a 12.592506 y el valor P calculado por la prueba de Friedman de Friedman igual a 0.027511868934805395 (ver **Tabla 4**). Además, los resultados de CS+LEM superan los resultados de los otro algoritmos; y COA supera a CS, ICS, MCS y MCSA con un nivel de confianza igual a 95% en los resultados de la prueba de Wilcoxon.

En general, CS+LEM obtiene los mejores resultados en la mayoría de las funciones de prueba. La prueba estadística de Friedman muestra que CS+LEM es en general la mejor opción para resolver problemas si el diseñador no sabe nada acerca del panorama (landscape) de la función de aptitud, el problema tiene alta dimensionalidad y el tiempo de ejecución es corto (menor de 80 segundos).

En el Anexo B – Experimentación Estudio Comparativo, Sección B.1, se muestra los resultados específicos obtenidos por cada versión del algoritmo CS en diferentes periodos de tiempo de ejecución.

3.3.8.2 Evaluación 1: Resultados funciones unimodales separables

La **Tabla 5** muestra los resultados obtenidos por cada versión del algoritmo CS sobre funciones unimodales separables en cinco periodos de tiempo de ejecución, 5, 10, 20, 40 y 80 segundos. En general, MCSA obtiene los mejores resultados en la mayoría de las funciones unimodales separables.

Tabla 5. Clasificación de las pruebas de Friedman por regla de oro para la evaluación relacionada con tiempo en funciones unimodales separables.

Tiempo	5 Segundos		10 Segundos		20 Segundos		40 Segundos		80 Segundos	
Algoritmo	Rango Promedio	Posición	Rango Promedio	Posición	Rango Promedio	Posición	Rango Promedio	Posición	Rango Promedio	Posición
CS	3.3333	4	3.1667	4	3.1667	4	3.1667	3	3.1667	4
COA	3.9167	6	4.1667	6	4.1667	6	4.1667	6	4.1667	6
ICS	**3.3333**	**1**	3.1667	3	3.1667	3	3.1667	4	3.1667	3
MCS	3.7500	5	4.1667	5	4.1667	5	4.1667	5	4.1667	5

Tiempo	5 Segundos		10 Segundos		20 Segundos		40 Segundos		80 Segundos	
Algoritmo	Rango Promedio	Posición	Rango Promedio	Posición	Rango Promedio	Posición	Rango Promedio	Posición	Rango Promedio	Posición
MCSA	3.3333	2	**3.1667**	**1**	**3.1667**	**1**	**3.1667**	**1**	**3.1667**	**1**
CS+LEM	3.3333	3	3.1667	2	3.1667	2	3.1667	2	3.1667	2

- Para funciones unimodales separables, en 5 segundos de ejecución, ICS reporta el mejor valor óptimo con una estadística de Friedman (distribuida de acuerdo al chi-cuadrado con 5 grados de libertad) igual a 0.595238 y el valor P calculado por la prueba de Friedman igual a 0.988220230321001 (ver **Tabla 5**). Además, ICS supera a COA y MCS con un nivel de confianza igual a 95% en los resultados de la prueba de Wilcoxon.

- Para funciones unimodales separables, en 10 segundos de ejecución, MCSA reporta el mejor valor óptimo con una estadística de Friedman (distribuida de acuerdo al chi-cuadrado con 5 grados de libertad) igual a 2.285714 y el valor P calculado por la prueba de Friedman igual a 0.8083627560841977 (ver **Tabla 5**). Además, CS+LEM y MCSA superan a CS, COA, ICS y MCS con un nivel de confianza igual a 95% en los resultados de la prueba de Wilcoxon y el algoritmo CS supera a COA y MCS con un nivel de confianza igual a 90%.

- Para funciones unimodales separables, en 20 segundos de ejecución, MCSA reporta el mejor valor óptimo con una estadística de Friedman (distribuida de acuerdo al chi-cuadrado con 5 grados de libertad) igual a 2.285714 y el valor P calculado por la prueba de Friedman igual a 0.8083627560841977 (ver **Tabla 5**). Además, MCSA supera a COA, ICS y MCS; y el algoritmo CS+LEM supera a COA y MCS con un nivel de confianza igual a 95% en los resultados de la prueba de Wilcoxon.

- Para funciones unimodales separables, en 40 segundos de ejecución, MCSA reporta el mejor valor óptimo con una estadística de Friedman (distribuida de acuerdo al chi-cuadrado con 5 grados de libertad) igual a 2.285714 y el valor P calculado por la prueba de Friedman igual a 0.8083627560841977 (ver **Tabla 5**). Además, MCSA supera a CS, ICS y MCS con un nivel de confianza igual a 95% en los resultados de la prueba de Wilcoxon y el algoritmo CS supera a COA y a ICS con un nivel de confianza igual a 90%.

- Para funciones unimodales separables, en 80 segundos de ejecución, MCSA reporta el mejor valor óptimo con una estadística de Friedman (distribuida de acuerdo al chi-cuadrado con 5 grados de libertad) igual a 2.285714 y el valor P calculado por la prueba de Friedman igual a 0.8083627560841977 (ver **Tabla**

5). Finalmente, MCSA supera a ICS y MCS con un nivel de confianza igual a 95% en los resultados de la prueba de Wilcoxon.

3.3.8.3 Evaluación 1: Resultados funciones unimodales no separables

La **Tabla 6** muestra los resultados obtenidos por cada versión del algoritmo CS en cinco periodos de tiempo de ejecución, 5, 10, 20, 40 y 80 segundos sobre funciones unimodales no separables. CS+LEM obtiene los mejores resultados en todas las funciones unimodales no separables.

Tabla 6. Clasificación de las pruebas de Friedman por regla de oro para la evaluación relacionada con el tiempo en funciones unimodales no separables.

Tiempo	5 Segundos		10 Segundos		20 Segundos		40 Segundos		80 Segundos	
Algoritmo	Rango Promedio	Posición	Rango Promedio	Posición	Rango Promedio	Posición	Rango Promedio	Posición	Rango Promedio	Posición
CS	3.0227	3	3.3409	3	3.2273	2	3.7045	4	3.7273	4
COA	5.8864	6	3.9318	4	4.0455	5	2.0000	2	2.5000	2
ICS	3.2955	4	4.2273	5	3.6364	4	4.0455	5	3.9091	5
MCS	3.3182	5	5.5227	6	5.1818	6	5.8636	6	5.8409	6
MCSA	2.7273	2	2.5227	2	3.3636	3	3.6818	3	3.5455	3
CS+LEM	**2.7500**	**1**	**1.4545**	**1**	**1.5455**	**1**	**1.7045**	**1**	**1.4773**	**1**

- Para funciones unimodales no separables, en 5 segundos de ejecución, CS+LEM reporta el mejor valor óptimo con una estadística de Friedman (distribuida de acuerdo al chi-cuadrado con 5 grados de libertad) igual a 44.987013 y el valor P calculado por la prueba de Friedman de Friedman igual a 1.4623199540153564E-8 (ver **Tabla 6**). Además, CS+LEM supera a COA y MCS con un nivel de confianza igual a 95% en los resultados de la prueba de Wilcoxon.

- Para funciones unimodales no separables, en 10 segundos de ejecución, CS+LEM reporta el mejor valor óptimo con una estadística de Friedman (distribuida de acuerdo al chi-cuadrado con 5 grados de libertad) igual a 62.675325 y el valor P calculado por la prueba de Friedman de Friedman igual a 4.518241336626261E-11 (ver **Tabla 6**). Además, CS+LEM supera todos los algoritmos; y los algoritmos CS y MCSA superan a COA, ICS y MCS con un nivel de confianza igual a 95% en los resultados de la prueba de Wilcoxon.

- Para funciones unimodales no separables, en 20 segundos de ejecución, CS+LEM reporta el mejor valor óptimo con una estadística de Friedman (distribuida de acuerdo al chi-cuadrado con 5 grados de libertad) igual a 44.363636 y el valor P calculado por la prueba de Friedman de Friedman igual a 1.958243422972572E-8 (ver **Tabla 6**). Además, CS+LEM supera todos los algoritmos con un nivel de confianza igual a 95% en los resultados de la

prueba de Wilcoxon; y el algoritmo CS supera a COA, ICS y MCS con un nivel de confianza igual a 90%.

- Para funciones unimodales no separables, en 40 segundos de ejecución, CS+LEM reporta el mejor valor óptimo con una estadística de Friedman (distribuida de acuerdo al chi-cuadrado con 5 grados de libertad) igual a 71.863636 y el valor P calculado por la prueba de Friedman de Friedman igual a 4.93050045236032E-11 (ver **Tabla 6**). Además, CS+LEM y COA superan a CS, ICS, MCS y MCSA con un nivel de confianza igual a 95% en los resultados de la prueba de Wilcoxon.

- Para funciones unimodales no separables, en 80 segundos de ejecución, CS+LEM reporta el mejor valor óptimo con una estadística de Friedman (distribuida de acuerdo al chi-cuadrado con 5 grados de libertad) igual a 67.837662 y el valor P calculado por la prueba de Friedman de Friedman igual a 4.449141055573591E-11 (ver **Tabla 6**). Finalmente, CS+LEM supera todos los algoritmos; y el algoritmo COA supera a CS, ICS y MCS con un nivel de confianza igual a 95% en los resultados de la prueba de Wilcoxon.

3.3.8.4 Evaluación 1: Resultados funciones multimodales separables

La **Tabla 7** muestra los resultados obtenidos por cada versión del algoritmo CS en cinco periodos de tiempo de ejecución, 5, 10, 20, 40 y 80 segundos sobre funciones multimodales separables. En general, CS+LEM obtiene los mejores resultados en la mayoría de las funciones multimodales separables.

Tabla 7. Clasificación de las pruebas de Friedman por regla de oro para la evaluación relacionada con el tiempo en funciones multimodales separables.

Tiempo	5 Segundos		10 Segundos		20 Segundos		40 Segundos		80 Segundos	
Algoritmo	Rango Promedio	Posición	Rango Promedio	Posición	Rango Promedio	Posición	Rango Promedio	Posición	Rango Promedio	Posición
CS	2.8571	2	3.1429	3	3.5714	3	3.5714	3	3.5714	3
COA	6.0000	6	4.2857	5	4.0000	5	2.4286	2	2.1429	2
ICS	3.0000	5	4.1429	4	3.7143	4	3.8571	5	3.7143	5
MCS	**2.2857**	**1**	5.1429	6	5.8571	6	6.0000	6	6.0000	6
MCSA	3.1429	3	2.8571	2	2.4286	2	3.5714	2	3.7143	4
CS+LEM	3.7143	4	**1.4286**	**1**	**1.4286**	**1**	**1.5714**	**1**	**1.8571**	**1**

- Para funciones multimodales separables, en 5 segundos de ejecución, MCS reporta el mejor valor óptimo con una estadística de Friedman (distribuida de acuerdo al chi-cuadrado con 5 grados de libertad) igual a 17.122449 y el valor P calculado por la prueba de Friedman igual a 0.0042733293740794265 (ver **Tabla 7**). Además, MCS supera a COA con un nivel de confianza igual a 95% en los resultados de la prueba de Wilcoxon.

- Para funciones multimodales separables, en 10 segundos de ejecución, CS+LEM reporta el mejor valor óptimo con una estadística de Friedman (distribuida de acuerdo al chi-cuadrado con 5 grados de libertad) igual a 17.122449 y el valor P calculado por la prueba de Friedman igual a 0.0042733293740796485 (ver **Tabla 7**). Además, CS+LEM supera a COA y MCS; y el algoritmo MCSA supera a ICS y MCS con un nivel de confianza igual a 95% en los resultados de la prueba de Wilcoxon.

- Para funciones multimodales separables, en 20 segundos de ejecución, CS+LEM reporta el mejor valor óptimo con una estadística de Friedman (distribuida de acuerdo al chi-cuadrado con 5 grados de libertad) igual a 22.591837 y el valor P calculado por la prueba de Friedman igual a 4.0392465107896847E-4 (ver **Tabla 7**). Además, MCSA supera a CS, ICS y MCS; y el algoritmo CS +LEM supera a COA y MCS con un nivel de confianza igual a 95% en los resultados de la prueba de Wilcoxon.

- Para funciones multimodales separables, en 40 segundos de ejecución, CS+LEM reporta el mejor valor óptimo con una estadística de Friedman (distribuida de acuerdo al chi-cuadrado con 5 grados de libertad) igual a 22.510204 y el valor P calculado por la prueba de Friedman igual a 4.1866583679028846E-4 (ver **Tabla 7**). Además, CS+LEM supera a MCS y MCSA con un nivel de confianza igual a 95% en los resultados de la prueba de Wilcoxon.

- Para funciones multimodales separables, en 80 segundos de ejecución, CS+LEM reporta el mejor valor óptimo con una estadística de Friedman (distribuida de acuerdo al chi-cuadrado con 5 grados de libertad) igual a 21.77551 y el valor P calculado por la prueba de Friedman igual a 5.776225557250214E-4 (ver **Tabla 7**). Finalmente, CS+LEM supera a MCS con un nivel de confianza igual a 95% en los resultados de la prueba de Wilcoxon.

3.3.8.5 Evaluación 1: Resultados funciones multimodales no separables

La **Tabla 8** muestra los resultados obtenidos por cada versión del algoritmo CS en cinco periodos de tiempo de ejecución, 5, 10, 20, 40 y 80 segundos sobre funciones multimodales no separables. CS+LEM obtiene los mejores resultados en todas las unciones multimodales no separables.

Tabla 8. Clasificación de las pruebas de Friedman por regla de oro para la evaluación relacionada con el tiempo en funciones multimodales no separables.

Tiempo	5 Segundos		10 Segundos		20 Segundos		40 Segundos		80 Segundos	
Algoritmo	Rango Promedio	Posición	Rango Promedio	Posición	Rango Promedio	Posición	Rango Promedio	Posición	Rango Promedio	Posición
CS	3.2115	3	3.1346	3	3.4038	3	3.7115	5	3.6154	4
COA	5.3269	6	4.0577	5	3.8846	5	2.6538	2	2.7885	2
ICS	2.9808	2	3.5577	4	3.6731	4	3.6923	4	3.6346	5
MCS	3.5192	5	5.0577	6	5.0962	6	5.3462	6	5.4231	6
MCSA	3.3269	4	3.0769	2	2.8269	2	3.2115	3	3.3077	3
CS+LEM	**2.6346**	**1**	**2.1154**	**1**	**2.1154**	**1**	**2.3846**	**1**	**2.2308**	**1**

- Para funciones multimodales no separables, en 5 segundos de ejecución, MCS reporta el mejor valor óptimo con una estadística de Friedman (distribuida de acuerdo al chi-cuadrado con 5 grados de libertad) igual a 33.203297 y el valor P calculado por la prueba de Friedman igual a 3.429319770598127E-6 (ver **Tabla 8**). Además, CS+LEM supera a COA, MCS y MCSA con un nivel de confianza igual a 95% en los resultados de la prueba de Wilcoxon.

- Para funciones multimodales no separables, en 10 segundos de ejecución, CS+LEM reporta el mejor valor óptimo con una estadística de Friedman (distribuida de acuerdo al chi-cuadrado con 5 grados de libertad) igual a 36.923077 y el valor P calculado por la prueba de Friedman igual a 6.206206417669335E-7 (ver **Tabla 8**). Además, CS+LEM supera todos los algoritmos; y los algoritmos MCSA y CS superan a COA y MCS con un nivel de confianza igual a 95% en los resultados de la prueba de Wilcoxon.

- Para funciones multimodales no separables, en 20 segundos de ejecución, CS+LEM reporta el mejor valor óptimo con una estadística de Friedman (distribuida de acuerdo al chi-cuadrado con 5 grados de libertad) igual a 37.923077 y el valor P calculado por la prueba de Friedman igual a 3.91012594636031E-7 (ver **Tabla 8**). Además, CS+LEM superan todos los algoritmos; y MCSA supera a CS, COA, ICS y MCS con un nivel de confianza igual a 95% en los resultados de la prueba de Wilcoxon.

- Para funciones multimodales no separables, en 40 segundos de ejecución, CS+LEM reporta el mejor valor óptimo con una estadística de Friedman (distribuida de acuerdo al chi-cuadrado con 5 grados de libertad) igual a 41.104396 y el valor P calculado por la prueba de Friedman igual a 8.943153018137195E-8 (ver **Tabla 8**). Además, CS+LEM supera a CS, ICS, MCS y MCSA; y el algoritmo COA supera a ICS y MCS con un nivel de confianza igual a 95% en los resultados de la prueba de Wilcoxon.

- Para funciones multimodales no separables, en 80 segundos de ejecución, CS+LEM reporta el mejor valor óptimo con una estadística de Friedman (distribuida de acuerdo al chi-cuadrado con 5 grados de libertad) igual a 43.708791 y el valor P calculado por la prueba de Friedman igual a 2.6562509414240765E-8 (ver **Tabla 8**). Además, CS+LEM supera a CS, ICS, MCS y MCSA; y el algoritmo MCSA supera a CS y MCS con un nivel de confianza igual a 95% en los resultados de la prueba de Wilcoxon.

3.3.9 Evaluación 2: Número de evaluaciones de la función objetivo requeridas para alcanzar el óptimo global.

Esta evaluación tiene como objetivo identificar qué versión del algoritmo CS proporciona los mejores resultados (resultados más cercano al mínimo global de la función objetivo con el menor número de evaluaciones de la función objetivo). Todos los algoritmos se ejecutan hasta encontrar el mínimo global o exceder un máximo de 50.000 evaluaciones de la función objetivo. Para más detalles refiérase al Anexo B – Experimentación Estudio Comparativo, Sección B.2, donde se muestra los resultados específicos obtenidos por cada versión del algoritmo CS.

Tabla 9. Clasificación de las pruebas de Friedman por regla de oro para la evaluación relacionada con el número de evaluaciones de la función objetivo.

Algoritmo	Rango Promedio	Posición
CS	3.7951	3
COA	**1.3279**	**1**
ICS	4.0492	5
MCS	4.2787	6
MCSA	3.8197	4
CS+LEM	3.7295	2

En promedio, COA reporta mejores resultados en el número de evaluaciones de la función objetivo requeridas para alcanzar al óptimo global que los otros cinco algoritmos. La clasificación promedio de las evaluaciones muestra que COA reporta el menor número de evaluaciones con una estadística de Friedman (distribuida de acuerdo al chi-cuadrado con 5 grados de libertad) igual a 102.271663 y el valor P calculado por la prueba de Friedman igual a 8.503309167906536E-11 (ver **Tabla 9**). Además, COA supera todos los algoritmos; el algoritmo CS supera a ICS y MCS con un nivel de confianza igual a 95%; y CS+LEM supera a MCS con un nivel de confianza igual a 95% en los resultados de la prueba de Wilcoxon.

Finalmente, es importante resaltar que COA es la mejor versión del algoritmo CS en esta prueba, porque alcanza el valor óptimo de 55 funciones de prueba sobre un total de 61 funciones en un número máximo especificado de evaluaciones de la

función objetivo. Este es un alto número de funciones de prueba resueltas en comparación con las otras versiones del algoritmo CS que solo son capaces de resolver de 7 a 14 funciones de prueba usando 50.000 evaluaciones de la función objetivo. Además, cabe resaltar que el algoritmo propuesto CS+LEM ocupa el segundo puesto con un total de 14 funciones de prueba resueltas.

El problema principal con COA es el tiempo adicional de ejecución, puesto que es 639.91% veces más lento que MCS el cual reporta el menor tiempo promedio por generación. La **Tabla 10** muestra en la primer columna los resultados de tiempo promedio adicional de ejecución obtenido por cada algoritmos, ordenados del más rápido al más lento; y en las columnas siguientes el comportamiento de los algoritmos ICS, CS, MCSA, CS+LEM y COA en términos del tiempo adicional que utilizan en cada generación en relación con MCS. Por lo anterior, COA es la mejor alternativa para entornos donde se puede esperar mucho mayor tiempo para encontrar la solución óptima del problema.

Tabla 10. Promedio de tiempo adicional en cada generación usado por los algoritmos CS en relación con MCS.

Algoritmo	ICS	CS	MCSA	CS+LEM	COA
MCS	39.67%	40.53%	45.89%	76.50%	643.91%
ICS		0.62%	4.46%	26.37%	432.63%
CS			3.82%	25.60%	429.37%
MCSA				20.98%	409.90%
CS+LEM					321.47%

3.3.10 Evaluación 3: Mejor valor óptimo alcanzado en diferente número de evaluaciones de la función objetivo

Esta evaluación tiene como objetivo identificar qué versión del algoritmo CS proporciona mejores resultados (resultados más cercano al mínimo global de la función objetivo) para el siguiente número de evaluaciones de la función objetivo: 5.000, 10.000, 20.000 y 50.000. Las siguientes secciones muestran los resultados en general (sobre todas las funciones), sobre funciones unimodales separables, funciones unimodales no separables, Funciones multimodales separables y funciones multimodales no separables.

3.3.10.1.1 Evaluación 3: Resultados generales

En la **Tabla 11** se presentan los resultados generales obtenidos en la Prueba de Friedman por las versiones del algoritmo CS en diferentes números de evaluaciones de la función objetivo, a saber: 5.000 10.000, 20.000 y 50.000. Para más detalles refiérase al Anexo B – Experimentación Estudio Comparativo, Sección B.3, donde se muestra los resultados específicos obtenidos por cada versión del algoritmo CS.

Tabla 11. Clasificación de las pruebas de Friedman por regla de oro para la evaluación relacionada con el mejor valor óptimo.

Tiempo	5.000 Evaluaciones		10.000 Evaluaciones		20.000 Evaluaciones		50.000 Evaluaciones	
Algoritmo	Rango Promedio	Posición	Rango Promedio	Posición	Rango Promedio	Posición	Rango Promedio	Posición
CS	3.2623	3	3.3443	3	3.4016	3	3.4918	3
COA	4.4016	6	3.6639	5	**3.0574**	**1**	**2.9262**	**1**
ICS	3.0410	2	3.1475	2	3.2131	2	2.9426	2
MCS	**2.9180**	**1**	**3.1311**	**1**	3.4098	4	3.5820	5
MCSA	3.3361	4	3.4016	4	3.4754	5	3.5656	4
CS+LEM	4.0410	5	4.3115	6	4.4426	6	4.4918	6

- En 5.000 evaluaciones de la función objetivo, MCS reporta el mejor valor óptimo con una estadística de Friedman (distribuida de acuerdo al chi-cuadrado con 5 grados de libertad) igual a 30.297424 y el valor P calculado por la prueba de Friedman igual a 1.2887890412760505E-5 (ver **Tabla 11**). Además, los resultados de MCS superan a CS, COA, ICS y CS+LEM; y el algoritmo ICS supera a COA y CS+LEM con un nivel de confianza igual a 95% en los resultados de la prueba de Wilcoxon.

- En 10.000 evaluaciones de la función objetivo, MCS reporta el mejor valor óptimo con una estadística de Friedman (distribuida de acuerdo al chi-cuadrado con 5 grados de libertad) igual a 17.0726 y el valor P calculado por la prueba de Friedman igual a 0.004364148264689782 (ver **Tabla 11**). Además, los resultados de MCS superan a CS, COA, ICS y CS+LEM; y el algoritmo CS supera a COA y CS+LEM con un nivel de confianza igual a 95% en los resultados de la prueba de Wilcoxon.

- En 20.000 evaluaciones de la función objetivo, COA reporta el mejor valor óptimo con una estadística de Friedman (distribuida de acuerdo al chi-cuadrado con 5 grados de libertad) igual a 20.655738 y el valor P calculado por la prueba de Friedman igual a 9.407719759269018E-4 (ver **Tabla 11**). Además, los resultados de COA y ICS superan a CS y CS+LEM con un nivel de confianza igual a 95% en los resultados de la prueba de Wilcoxon; y COA supera a ICS, MCSA y CS+LEM con un nivel de confianza igual a 90%.

- En 50.000 evaluaciones de la función objetivo, COA reporta el mejor valor óptimo con una estadística de Friedman (distribuida de acuerdo al chi-cuadrado con 5 grados de libertad) igual a 28.489461 y el valor P calculado por la prueba de Friedman igual a 2.9195936867121297E-5 (ver **Tabla 11**). Además, los resultados de COA superan a CS, MCSA y CS+LEM; y el

algoritmo ICS supera a CS y CS+LEM con un nivel de confianza igual a 95% en los resultados de la prueba de Wilcoxon.

En general, para un alto número de evaluaciones de la función objetivo COA reporta los mejores resultados, mientras que el algoritmo MCS reporta los mejores resultados para bajo número de evaluaciones.

3.3.10.2 Evaluación 3: Resultados funciones unimodales separables

La **Tabla 12** muestra los resultados obtenidos por cada versión del algoritmo CS en cuatro diferentes números de evaluaciones de la función objetivo, 5.000, 10.000, 20.000 y 50.000 sobre funciones unimodales separables. En general, MCSA obtiene los mejores resultados en todas las evaluaciones para las funciones unimodales separables.

Tabla 12. Clasificación de las pruebas de Friedman por regla de oro para la evaluación relacionada con el mejor valor óptimo en funciones unimodales separables.

Tiempo	5.000 Evaluaciones		10.000 Evaluaciones		20.000 Evaluaciones		50.000 Evaluaciones	
Algoritmo	Rango Promedio	Posición	Rango Promedio	Posición	Rango Promedio	Posición	Rango Promedio	Posición
CS	2.7500	2	2.8333	2	2.9167	2	3.0000	2
COA	4.5833	5	4.7500	6	4.0000	5	3.6667	4
ICS	3.0833	3	3.1667	3	3.2500	3	3.4167	5
MCS	4.5833	6	4.3333	5	4.7500	6	4.7500	6
MCSA	**2.4167**	**1**	**2.5000**	**1**	**2.5833**	**1**	**2.8333**	**1**
CS+LEM	3.5833	4	3.4167	4	3.5000	4	3.3333	3

- Para funciones unimodales separables, en 5.000 evaluaciones de la función objetivo, MCSA reporta el mejor valor óptimo con una estadística de Friedman (distribuida de acuerdo al chi-cuadrado con 5 grados de libertad) igual a 7.309524 y el valor P calculado por la prueba de Friedman igual a 0.19861945441041795 (ver **Tabla 12**).

- Para funciones unimodales separables, en 10.000 evaluaciones de la función objetivo, MCSA reporta el mejor valor óptimo con una estadística de Friedman (distribuida de acuerdo al chi-cuadrado con 5 grados de libertad) igual a 6.547619 y el valor P calculado por la prueba de Friedman igual a 0.25651553046441067 (ver **Tabla 12**). Además, los resultados de CS+LEM superan a COA y MCS con un nivel de confianza igual a 95% en los resultados de la prueba de Wilcoxon.

- Para funciones unimodales separables, en 20.000 evaluaciones de la función objetivo, COA reporta el mejor valor óptimo con una estadística de Friedman

(distribuida de acuerdo al chi-cuadrado con 5 grados de libertad) igual a 5.238095 y el valor P calculado por la prueba de Friedman igual a 0.38751919558455017 (ver **Tabla 12**). Además, los resultados de CS+LEM superan a MCS con un nivel de confianza igual a 95% en los resultados de la prueba de Wilcoxon.

- Para funciones unimodales separables, en 50.000 evaluaciones de la función objetivo, MCSA reporta el mejor valor óptimo con una estadística de Friedman (distribuida de acuerdo al chi-cuadrado con 5 grados de libertad) igual a 3.97619 y el valor P calculado por la prueba de Friedman igual a 0.552849024957503 (ver **Tabla 12**). Además, los resultados de CS+LEM superan a MCS con un nivel de confianza igual a 95% en los resultados de la prueba de Wilcoxon.

3.3.10.3 Evaluación 3: Resultados funciones unimodales no separables

La **Tabla 13** muestra los resultados obtenidos por cada versión del algoritmo CS en cuatro diferentes números de evaluaciones de la función objetivo, 5.000, 10.000, 20.000 y 50.000 sobre funciones unimodales no separables. En general, COA obtiene los mejores resultados para las funciones unimodales no separables.

Tabla 13. Clasificación de las pruebas de Friedman por regla de oro para la evaluación relacionada con el mejor valor óptimo en funciones unimodales no separables.

Tiempo	5.000 Evaluaciones		10.000 Evaluaciones		20.000 Evaluaciones		50.000 Evaluaciones	
Algoritmo	Rango Promedio	Posición	Rango Promedio	Posición	Rango Promedio	Posición	Rango Promedio	Posición
CS	3.2273	2	**3.2045**	**1**	3.2045	2	3.4773	3
COA	3.8864	5	3.5000	5	**2.7273**	**1**	**2.4318**	**1**
ICS	3.2273	3	3.2273	3	3.3409	3	3.0227	2
MCS	**3.0909**	**1**	3.2727	4	3.7045	5	3.7727	5
MCSA	3.2727	4	3.2045	2	3.3409	4	3.5227	4
CS+LEM	4.2955	6	4.5909	6	4.6818	6	4.7727	6

- Para funciones unimodales no separables, en 5.000 evaluaciones de la función objetivo, MCS reporta el mejor valor óptimo con una estadística de Friedman (distribuida de acuerdo al chi-cuadrado con 5 grados de libertad) igual a 7.227273 y el valor P calculado por la prueba de Friedman igual a 0.20427901814386218 (ver **Tabla 13**). Además, los resultados de MCS superan a COA; y los algoritmos CS, ICS y MCSA superan a CS+LEM con un nivel de confianza igual a 95% en los resultados de la prueba de Wilcoxon.

- Para funciones unimodales no separables, en 10.000 evaluaciones de la función objetivo, MCS reporta el mejor valor óptimo con una estadística de Friedman (distribuida de acuerdo al chi-cuadrado con 5 grados de libertad)

igual a 9.37013 y el valor P calculado por la prueba de Friedman igual a 0.09518089187661205 (ver **Tabla 13**). Además, los resultados de CS superan a CS+LEM con un nivel de confianza igual a 95% en los resultados de la prueba de Wilcoxon.

- Para funciones unimodales no separables, en 20.000 evaluaciones de la función objetivo, COA reporta el mejor valor óptimo con una estadística de Friedman (distribuida de acuerdo al chi-cuadrado con 5 grados de libertad) igual a 13.662338 y el valor P calculado por la prueba de Friedman igual a 0.017902709821705054 (ver **Tabla 13**). Además, los resultados de COA superan a CS+LEM; y el algoritmo CS supera a MCSA y CS+LEM con un nivel de confianza igual a 95% en los resultados de la prueba de Wilcoxon.

- Para funciones unimodales no separables, en 50.000 evaluaciones de la función objetivo, COA reporta el mejor valor óptimo con una estadística de Friedman (distribuida de acuerdo al chi-cuadrado con 5 grados de libertad) igual a 19.25974 y el valor P calculado por la prueba de Friedman igual a 0.0017193174249131582 (ver **Tabla 13**). Además, los resultados de COA superan a CS, ICS, MCSA y CS+LEM con un nivel de confianza igual a 95% en los resultados de la prueba de Wilcoxon.

3.3.10.4 Evaluación 3: Resultados funciones multimodales separables

La **Tabla 14** muestra los resultados obtenidos por cada versión del algoritmo CS en cuatro diferentes números de valuaciones de la función objetivo, 5.000, 10.000, 20.000 y 50.000 sobre funciones multimodales separables. En general, MCS obtiene los mejores resultados para funciones multimodales separables.

Tabla 14. Clasificación de las pruebas de Friedman por regla de oro para la evaluación relacionada con el mejor valor óptimo en funciones multimodales separables.

Tiempo	5.000 Evaluaciones		10.000 Evaluaciones		20.000 Evaluaciones		50.000 Evaluaciones	
Algoritmo	Rango Promedio	Posición	Rango Promedio	Posición	Rango Promedio	Posición	Rango Promedio	Posición
CS	3.3571	3	3.9286	4	3.7857	4	3.8571	5
COA	5.1429	6	4.1429	5	4.0000	5	3.5714	3
ICS	3.0714	2	3.0714	2	2.9286	2	**2.3571**	**1**
MCS	**2.2857**	**1**	**2.1429**	**1**	**2.2857**	**1**	3.0714	2
MCSA	3.3571	2	3.5000	3	3.6429	3	3.7143	4
CS+LEM	3.7857	5	4.2143	6	4.3571	6	4.4286	6

- Para funciones multimodales separables, en 5.000 evaluaciones de la función objetivo, MCS reporta el mejor valor óptimo con una estadística de Friedman (distribuida de acuerdo al chi-cuadrado con 5 grados de libertad) igual a 8.959184 y el valor P calculado por la prueba de Friedman igual a

0.11070330083232738 (ver **Tabla 14**). Además, MCS supera a COA con un nivel de confianza igual a 95% en los resultados de la prueba de Wilcoxon.

- Para funciones multimodales separables, en 10.000 evaluaciones de la función objetivo, MCS reporta el mejor valor óptimo con una estadística de Friedman (distribuida de acuerdo al chi-cuadrado con 5 grados de libertad) igual a 6.265306 y el valor P calculado por la prueba de Friedman igual a 0.2812527300539138 (ver **Tabla 14**). Además, MCS supera a COA con un nivel de confianza igual a 95% en los resultados de la prueba de Wilcoxon.

- Para funciones multimodales separables, En 20.000 evaluaciones de la función objetivo, COA reporta el mejor valor óptimo con una estadística de Friedman (distribuida de acuerdo al chi-cuadrado con 5 grados de libertad) igual a 5.77551 y el valor P calculado por la prueba de Friedman igual a 0.3286792215708251 (ver **Tabla 14**). Además, MCS supera a COA con un nivel de confianza igual a 95% en los resultados de la prueba de Wilcoxon.

- Para funciones multimodales separables, in 50.000 evaluaciones de la función objetivo, COA reporta el mejor valor óptimo con una estadística de Friedman (distribuida de acuerdo al chi-cuadrado con 5 grados de libertad) igual a 5.061224 y el valor P calculado por la prueba de Friedman igual a 0.40845409330381544 (ver **Tabla 14**). Además, ICS supera a COA con un nivel de confianza igual a 95% en los resultados de la prueba de Wilcoxon y MCS supera a CS+LEM con un nivel de confianza igual a 90%.

3.3.10.5 Evaluación 3: Resultados funciones multimodales no separables

La **Tabla 15** muestra los resultados obtenidos por cada versión del algoritmo CS en cuatro diferentes números de evaluaciones de la función objetivo, 5.000, 10.000, 20.000 y 50.000 sobre funciones multimodales no separables. En general, ICS reporta mejores resultados en valor mínimo global que los otros cinco algoritmos tomando un alto número de evaluaciones de la función objetivo y el algoritmo MCS para un bajo número de evaluaciones.

Tabla 15. Clasificación de las pruebas de Friedman por regla de oro para la evaluación relacionada con el mejor valor óptimo en funciones multimodales no separables.

Tiempo	5.000 Evaluaciones		10.000 Evaluaciones		20.000 Evaluaciones		50.000 Evaluaciones	
Algoritmo	Rango Promedio	Posición	Rango Promedio	Posición	Rango Promedio	Posición	Rango Promedio	Posición
CS	3.2885	3	3.4808	3	3.5192	4	3.5192	4
COA	4.5577	6	3.5000	4	3.0962	3	2.9423	2
ICS	2.8077	2	2.9615	2	**3.0000**	**1**	**2.7692**	**1**

Tiempo	5.000 Evaluaciones		10.000 Evaluaciones		20.000 Evaluaciones		50.000 Evaluaciones	
Algoritmo	Rango Promedio	Posición	Rango Promedio	Posición	Rango Promedio	Posición	Rango Promedio	Posición
MCS	**2.6538**	**1**	**2.9231**	**1**	3.0577	2	3.3077	3
MCSA	3.5385	4	3.7308	5	3.7692	5	3.7692	5
CS+LEM	4.1538	5	4.4038	6	4.5577	6	4.6923	6

- Para funciones multimodales no separables, en 5.000 evaluaciones de la función objetivo, MCS reporta el mejor valor óptimo con una estadística de Friedman (distribuida de acuerdo al chi-cuadrado con 5 grados de libertad) igual a 20.708791 y el valor P calculado por la prueba de Friedman igual a 9.193523217060351E-4 (ver **Tabla 15**). Además, los resultados de MCS superan todos los algoritmos; y ICS supera a CS y CS+LEM con un nivel de confianza igual a 95% en los resultados de la prueba de Wilcoxon.

- Para funciones multimodales no separables, en 10.000 evaluaciones de la función objetivo, MCS reporta el mejor valor óptimo con una estadística de Friedman (distribuida de acuerdo al chi-cuadrado con 5 grados de libertad) igual a 11.093407 y el valor P calculado por la prueba de Friedman igual a 0.049559137396659625 (ver **Tabla 15**). Además, MCS supera a CS, ICS y CS+LEM; y el algoritmo ICS supera a CS, MCSA y CS+LEM con un nivel de confianza igual a 95% en los resultados de la prueba de Wilcoxon.

- Para funciones multimodales no separables, en 20.000 evaluaciones de la función objetivo, MCS reporta el mejor valor óptimo con una estadística de Friedman (distribuida de acuerdo al chi-cuadrado con 5 grados de libertad) igual a 13.373626 y el valor P calculado por la prueba de Friedman igual a 0.020118066944719848 (ver **Tabla 15**). Además, los resultados de ICS, COA y MCS superan a CS y CS+LEM con un nivel de confianza igual a 95% en los resultados de la prueba de Wilcoxon.

- Para funciones multimodales no separables, en 50.000 evaluaciones de la función objetivo, COA reporta el mejor valor óptimo con una estadística de Friedman (distribuida de acuerdo al chi-cuadrado (chi-square) con 5 grados de libertad) igual a 17.653846.y el valor P calculado por la prueba de Friedman igual a 0.00341303181219077 (ver **Tabla 15**). Además, COA supera a CS y CS+LEM; y el algoritmo ICS supera a CS y CS+LEM con un nivel de confianza igual a 95% en los resultados de la prueba de Wilcoxon.

3.3.11 Análisis y discusión del estudio comparativo

Como resultado de las tres evaluaciones comparativas entre las versiones del algoritmo CS, se concluye lo siguiente: Todas las versiones pueden ser apropiadas para el problema de Agrupamiento de Documentos Web, debido a su sobresaliente desempeño para resolver los problemas de optimización continua, validados por medio de las funciones de pruebas específicas. Sin embargo no todas las versiones se pueden representar en dicho problema, ya que existen características como la definición de grupos de soluciones (hábitats) y la definición dinámica de los parámetros (porcentaje de abandonos y tamaño de los pasos de Lévy) que no son posibles de representar en el dominio del problema. Esto se debe a la complejidad de la programación que esto acarrearía y el coste computacional de dichos procesos, donde se cuenta con un tiempo limitado (menos de 2 segundos) para presentar resultados. Es por esto que la versión del algoritmo de Búsqueda Cucú seleccionada para ser representada en el problema de Agrupamiento de Documentos Web es la versión original de CS, considerando su simplicidad, su facilidad de implementación y su comportamiento general en las evaluaciones durante el proceso de optimización de las funciones de prueba.

Capítulo 4

4 ALGORITMO WDC-CSK Y EXPERIMENTACIÓN

Para dar cumplimiento al segundo y tercer objetivo específico propuesto en este proyecto, este capítulo presenta inicialmente la descripción general del algoritmo propuesto llamado Agrupamiento de Documentos Web basado en el Algoritmo de la Búsqueda Cucú (WDC-CSK), basado en el algoritmo meta-heurístico de la Búsqueda Cucú, el algoritmo K-means, el Criterio de Información Bayesiano Balanceado, los métodos Split y Merge. Finalmente, muestra el proceso de evaluación del algoritmo propuesto WDC-CSK, el cual fue probado con cuatro diferentes conjuntos de datos, a saber: DMOZ-50, AMBIENT, MORESQUE y ODP-239 sobre un total de 447 consultas, y comparado con otros algoritmos de agrupamiento de documentos Web, entre ellos: Suffix Tree Clustering (STC), Lingo y Bisecting K-means.

4.1 ALGORITMO K-MEANS

Este algoritmo pertenece a los métodos de agrupamiento particional y es el algoritmo más popular y más comúnmente utilizado para agrupamiento, debido a su fácil implementación, sencillez conceptual, encuentra el mínimo (o máximo) local en un espacio de búsqueda y la complejidad en el tiempo es de orden O(nkt), donde n es el número de patrones, k es el número de grupos y t el número de ciclos requeridos para converger al óptimo local. K-means en cada una de sus iteraciones, busca minimizar la Suma del Error Cuadrado (Sum of Squared Error, SSE) de los puntos asignados a cada grupo en relación con el centro de cada agrupación. Desafortunadamente, la calidad del resultado depende de los puntos iniciales y puede converger a un mínimo local si la partición inicial no está apropiadamente seleccionada [38, 72-74]. La función de la SSE manejada por este algoritmo se define con la fórmula (9).

$$SSE = \sum_{j=1}^{k} \sum_{i=1}^{n} P_{i,j} \left\| x_i - c_j \right\|^2 \qquad (9)$$

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

A continuación se describe el algoritmo K-means:

Las entradas de K-means son: El número de grupos (valor de K) y un conjunto (tabla, matriz o colección) con n objetos (o registros) en un espacio de

características D dimensional, formalmente definido por $X = \{x_1, x_2, .., x_n\}$(En nuestro caso, x_i es un vector fila, por razones de implementación). Las salidas de K-means son un conjunto que contiene K centros (denotado por c_j). Los pasos en el procedimiento K-means se puede resumir como se muestra en la **Figura 7**.

01	Seleccione una Partición Inicial (K centros)
	Repetir
02	Re calcular las membrecías
03	Actualizar Centros
04	**Hasta** (Criterio de Parada)
05	Retornar Solución

Figura 7. Algoritmo K-means

En el paso 01, hay varios enfoques para seleccionar K centros iniciales [75], por ejemplo Forgey (estrategia seleccionada para el algoritmo propuesto) [76] sugiere seleccionar K instancias aleatorias desde el conjunto de datos y McQueen sugiere seleccionar los primeros K puntos en el conjunto de datos como las semillas preliminares y luego, utilizando una estrategia incremental para actualizar y seleccionar los K centros reales de la solución inicial [75].

En el paso 02, es necesario recalcular membresías de acuerdo a la solución actual. Es decir, asignar cada objeto del conjunto de datos a su centro del grupo más cercano formando un nuevo grupo de datos.

En agrupamiento existe un sinnúmero de medidas de similitud o distancia, pero en el área de recuperación de información, búsqueda Web y agrupamiento de documentos Web, pero la similitud basada en cosenos se destaca mostrando los mejores resultados [30, 77]. En esta investigación se usa la similitud basada en coseno [78, 79] para medir el grado de similitud entre dos documentos o entre un documento y el centroide del grupo, definido con la formula (10).

$$Sim(d,q) = \frac{\sum_{i=1}^{D} W_{i,d} \times W_{i,q}}{\sqrt{\sum_{i=1}^{D} W^2{}_{i,d}} \sqrt{\sum_{i=1}^{D} W^2{}_{i,q}}} \qquad (10)$$

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.

Cuando dos documentos son iguales el coseno tiene un valor de 1 y si no hay ninguna similitud entre documentos el valor es 0, donde el máximo valor del coseno indica una mayor similitud entre los documentos. Esta medida ha sido la más utilizada por la comunidad académica y científica de recuperación de

información y búsqueda Web, tanto para medir la similitud de los documentos entre sí, como para medir la similitud de los documentos con la consulta del usuario. Además, las investigaciones en Agrupamiento de documentos Web la han usado para calcular la distancia de los documentos al documento centro del grupo al que pertenece cada documento [30, 77].

En la literatura de agrupamiento particional, distintos criterios se han usado para comparar dos o más soluciones y decidir cuál es la mejor [35, 80]. Los criterios más populares están basados en las matrices de dispersión de los datos internos a cada grupo y entre los diferentes grupos. Dentro de los criterios para comparar soluciones de agrupamiento, fueron seleccionados para éste proyecto de investigación dos criterios, los cuales permiten calificar cada solución encontrada y definir automáticamente el número de agrupaciones. Estos son el Criterio de Información Bayesiano (Bayesian Information Criterion, BIC) [21] expresado por (11) y el Criterio de Información Bayesiano Balanceado (Balanced Bayesian Information Criterion, BBIC) [81] expresado por (13).

$$\mathrm{BIC} = \mathrm{n} * \mathrm{Ln}\left(\frac{\mathrm{SSE}}{\mathrm{n}}\right) + \mathrm{k} * \mathrm{Ln}(\mathrm{n}) \tag{11}$$

Donde n es el número total de documentos, SSE es la Suma del Error Cuadrado con respecto al centroide de cada agrupación expresado por la formula (9) y k es el número de grupos.

$$\mathrm{ADBC} = \frac{2}{\mathrm{n} \times (\mathrm{n}-1)} \sum_{\mathrm{i}=1}^{\mathrm{k}-1} \sum_{\mathrm{j}=\mathrm{i}+1}^{\mathrm{K}} \left(1 - \mathrm{SimCos}(\mathrm{C_i}, \mathrm{C_j})\right) \tag{12}$$

Donde SimCos es la similitud de cosenos definida por la formula (10).

$$\mathrm{BBIC} = \mathrm{n} * \mathrm{Ln}\left(\frac{\mathrm{SSE}}{\mathrm{n} \times \mathrm{ADBC}}\right) + \mathrm{k} * \mathrm{Ln}(\mathrm{n}) \tag{13}$$

Donde ADBC es el promedio de la distancia entre los centroides expresado por la formula (12)

El objetivo en el algoritmo es minimizar los criterios BIC o BBIC, lo cual implica minimizar SSE, buscando agrupaciones más compactas (distancia muy pequeña entre el centroide y los elementos que perteneces a la agrupación). El factor $k \times Ln(n)$ permite aplicar un costo a las soluciones que tengan muchas agrupaciones (y que en general tienen un valor menor de SSE, que en promedio decrece cuando k aumenta) y permite de esta forma comparar soluciones que tienen distinto número de agrupaciones. En el caso de BBIC además se busca aumentar la distancia (separación) entre los diferentes grupos, lo que además ayuda a definir etiquetas más diferentes para las diferentes agrupaciones.

4.2 ALGORITMO PROPUESTO: WDC-CSK

El nuevo algoritmo propuesto, llamado Agrupamiento de Documentos Web basado en el Algoritmo de la Búsqueda Cucú (Web Document Clustering based on the Cuckoo Search Algorithm, WDC-CSK) es un algoritmo para agrupamiento de documentos Web centrado en la descripción [23] inspirado por el algoritmo meta-heurístico de la Búsqueda Cucú (CS) [82]. El algoritmo K-means se utiliza como una estrategia local en WDC-CSK para mejorar las soluciones globales de CS. Los vuelos de Lévy son reemplazados por dos operaciones o métodos, Split y Merge, los cuales son usados para promover la diversidad en la población y prevenir la convergencia rápida o prematura de la población a la solución óptima local. Finalmente, los criterios BIC o BBIC se emplean como función de aptitud (fitness) o función objetivo para encontrar automáticamente el número de agrupaciones y comparar cual de dos o más soluciones es mejor. La **Figura 8** muestra los principales pasos ejecutados por el algoritmo WDC-CSK.

01	**Inicializar los parámetros del algoritmo**
02	**Pre-procesamiento de documentos**
03	**Ejecutar en paralelo un número (MNI) específico de islas**
04	**Inicializar la población de nidos**; crea aleatoriamente un conjunto de nidos (población de nidos) de la isla actual
05	**Ejecutar K-means** (optimizador local) para cada nido en la población de la isla actual
06	**Calcula el valor de aptitud (fitness)** (BIC o BBIC) de acuerdo con (11) o (13) para todos los nidos en la población de la isla actual
07	**Hacer**
08	**Crear un nuevo nido** usando un abandono o las operaciones (métodos) Split o Merge basado en un nido seleccionado al azar (nido actual) de la isla actual
09	**Ejecutar K-Means** (optimizador local) para el nuevo nido generado
10	**Calcula el valor de aptitud** (BIC o BBIC) de acuerdo con (11) o (13) para el nuevo nido generado
11	**Almacenar la mejor solución**, si el nuevo nido generado es mejor que el nido seleccionado aleatoriamente, este último nido se sustituye en la población para el nuevo nido generado
12	**Hasta** que se cumplan las condiciones de parada (Se alcanza el parámetro MNN o el parámetro MET)
13	**Seleccionar el mejor nido** en la población de nidos de la isla actual
14	**Fin de la ejecución en paralelo**
15	**Seleccionar el mejor nido** de todas las islas
16	**Asignar etiquetas a los grupos** en el mejor nido basado en frases frecuentes en cada grupo.

Figura 8. Resumen del algoritmo WCD-CSK.

4.2.1 Pasos del algoritmo WDC-CSK

A continuación, se presenta una explicación detallada de los pasos más importantes del algoritmo WDC-CSK:

01: Inicializar parámetros del algoritmo: En esta investigación, el problema de optimización radica en minimizar los criterios BBIC o BIC, llamados función de

aptitud o fitness. WDC-CSK necesita los siguientes parámetros: Número Máximo de Islas (Maximum Number of Islands, MNI): número entero entre 1 y 5, Tamaño de la Población (Population Size, PS): número entero entre 5 y 10, Función Objetivo (Objective Function, OF): valor de enumeración entre BBIC y BIC, Probabilidad de Abandonos (Probability of Abandoned, PA): valor real entre 0.1 y 0.25, Umbral de Frecuencia de Términos (Term Frequency Threshold, TFT): valor entero mayor que 2 para el proceso de etiquetado, y finalmente, Máximo Número de Nidos (Maximum Number of Nests, MNN) o Tiempo de Ejecución Máximo (Maximum Execution Time, MET) en milisegundos para detener la ejecución del algoritmo.

02: Pre-procesamiento de documentos. Inicialmente se utiliza Lucene (http://lucene.apache.org) para la etapa de pre-procesamiento de documentos. Esta etapa incluye: tokenización, filtrado de minúsculas, remover palabras vacías (stop words), aplicar el algoritmo de stemming de Porter [23] (disponible sólo en el idioma inglés) y la construcción de la Matriz de Términos por Documento (TDM).

La matriz TDM es la estructura más usada para la representación de documentos en RI, y se basa en el Modelo de Espacio Vectorial (VSM) [29, 30]. En este modelo se conciben los documentos como bolsas de palabras y la colección de documentos se representa por una matriz de D-términos por N-documentos. Cada documento se representa por un vector de frecuencia de términos (tf_i) normalizado por la inversa de la frecuencia del documento para ese término, que se conoce como el valor TF-IDF (expresado por la ecuación (14)), y la similitud de coseno (ver ecuación (10)) se utiliza para medir el grado de similitud entre dos documentos o entre un documento y el centroide del grupo.

$$w_{i,j} = \frac{freq_{i,j}}{\max\left(freq_i\right)} \times \log\left(\frac{N}{n_j}\right) \tag{14}$$

Donde $freq_{i,j}$ es la frecuencia observada del término j en el documento I, $\max(freq_i)$ es la máxima frecuencia observada en el documento I, N es el número total de documentos en la colección, y n_j es el número de documentos donde se presenta j términos.

03: Ejecutar en paralelo un número (MNI) especifico de islas. En el algoritmo propuesto se extiende el concepto de islas, para definir la ejecución de múltiples poblaciones o conjunto de soluciones que pueden evolucionar de forma separada e independiente, con el fin de permitir obtener diversidad de soluciones y lograr seleccionar la mejor solución.

04: Inicializar la población de nidos. El algoritmo WDC-CSK trabaja con nidos, como enfoque de representación de las soluciones. Cada nido tiene un número diferente de grupos, una lista de centroides y el valor de la función objetivo, basada en BBIC o BIC que depende de la ubicación de cada nido y el número de centroides. Los centros de agrupamiento en el nido consiste de D x K números reales, donde K es el número de grupos y D es el total de número de términos (palabras). Por ejemplo, en un dato tridimensional el nido < [0.5|0.1|0.8], [0.2|0.5|0.3], [0.4|0.2|0.8], [0.1|0.7|0.7], 0.819> codificado cuatro (valor de K) grupos con valor de aptitud 0.819. Inicialmente, cada centroide corresponde a un documento diferente seleccionado aleatoriamente en la matriz TDM (estrategia Forgey en el algoritmo K-Means).

El número inicial de grupos, es decir, el valor de K, es aleatoriamente calculado de 2 a Kmax (incluyéndolo), donde K es un número natural y Kmax es el límite superior del número de grupos y se calcula como $\sqrt{N} + 1$, (donde N es el número total de documentos en la matriz TDM, pero este valor no puede ser menor a ocho ni superior al número de documentos), esta es una regla empírica utilizada por muchos investigadores en la literatura de agrupación [62].

05: Ejecutar K-means. Los pasos 02 a 05 de la **Figura 7** se ejecutan basados en los centroides registrados en cada nido. El paso 01 no es necesario porque el nido selecciona previamente los centroides. Este paso es similar al paso 09.

08: Crear un nuevo nido. Para crear un nuevo nido (solución) el algoritmo realiza una operación de abandono, Merge o Split. Con una probabilidad específica (definida por el parámetro PA) el algoritmo crea un nuevo nido con centroides seleccionados aleatoriamente a partir de la matriz TDM. Esta operación corresponde a un abandono y se inspira en la situación cuando un huevo de cucú es descubierto por el ave huésped. En este caso se crea un nido totalmente nuevo para completar la población de nidos de cucú en la isla actual. Esta operación proporciona diversidad y previene que la población de nidos converja demasiado rápido.

Con una probabilidad especifica ((1-PA)/2) se ejecutan las operaciones Split o Merge. Estas operaciones reemplazan los vuelos de Lévy en el algoritmo de búsqueda cucú original, debido a que este método de representación no es posible desplazar a la representación de centroides manejado en el proceso de Agrupamiento Web. Estas operaciones son usadas para promover la diversidad en la población y prevenir la convergencia rápida o prematura de la población a la solución óptima local. Para ambas operaciones (Split y Merge), se selecciona

inicialmente un nido aleatoriamente de la población actual. Este nido se copia en un nuevo nido, llamado **nido base**. En la operación Merge, se seleccionan los dos centroides más similares (medido por la similitud de cosenos) del nido base y se unen. En la operación Split, se selecciona el grupo más disperso y se divide en dos grupos. El grupo más disperso se selecciona basado en el valor SSE (Suma del Error Cuadrado) reportado por cada grupo asociado al centroide en el nido base. Para dividir el grupo, se selecciona el documento más diferente del grupo actual y se crea un nuevo grupo con este documento como centroide.

13: Seleccionar el mejor nido. En este paso el algoritmo encuentra y selecciona la mejor solución de la población de nidos de la isla actual. El mejor nido es el nido con el menor valor de aptitud (minimizar BBIC o BIC). Luego se retorna esta solución como la mejor solución de agrupación (centroides y fitness) de la isla actual.

16: Asignación de etiquetas a los grupos. El algoritmo utiliza un enfoque de frases frecuentes (Frequent Phrases, FPH) para el etiquetado de cada grupo. Este paso corresponde al paso 2 en Lingo [52] (con algunas modificaciones) llamado "Extracción de Frases Frecuentes". En WDC-CSK este método se utiliza para cada grupo generado en los pasos anteriores. El etiquetado de cada grupo funciona de la siguiente manera:

1. **Conversión de la representación:** Cada documento en el grupo actual es convertido de una representación basada en caracteres a representación basada en palabras.
2. **Concatenación de documentos:** Se concatenan los documentos del grupo actual y se crea un nuevo documento con la versión invertida de la concatenación.
3. **Descubrimiento de frases completas:** Se descubren en el grupo actual las frases completas a la derecha y las frases completas a la izquierda, se ordenan alfabéticamente y luego se combinan en un grupo de frases completas.
4. **Selección final:** Se seleccionan los términos y frases ubicadas en el grupo actual que excedan el Umbral de Frecuencia de los Términos (**Term Frequency Threshold, TFT)**. Los términos de la consulta del usuario se eliminan de los términos o frases seleccionadas con el fin de mejorar la calidad del proceso de etiquetado.
5. **Construyendo las etiquetas del grupo "Otros":** Si algunos documentos no alcanzan el TFT, entonces ellos son enviados a otros grupos.

6. **Inducción de la etiqueta al grupo:** En el grupo actual, se construye una matriz TDM. Utilizan la similitud de coseno, se seleccionan los mejores términos o frases candidatas para el grupo (optimizado SSE).

Teniendo en cuenta que WDC-CSK es un algoritmo compacto (evoluciona generando una nueva solución en cada iteración) y que el ingreso de la solución generada se basa en la estrategia de remplazo de la peor solución en la población, logrando con esto que la población mejore progresivamente. Se puede afirmar basado en la demostración realizada por Mahdavi [39] que se basada en cadenas finitas de Markov, que el algoritmo converge al óptimo global.

4.3 IMPLEMENTACIÓN DEL SISTEMA DE LABORATORIO (WDC)

Para el desarrollo de las pruebas del proyecto se realizó un prototipo software que permitió evaluar la propuesta de algoritmo híbrido para agrupamiento de documentos Web, éste prototipo recibe el nombre de Project Web Document Clustering (WDC). Para más detalles sobre el proceso de desarrollo refiérase al Anexo C – Implementación del Sistema de Laboratorio (WDC).

4.4 EXPERIMENTACIÓN

En esta sección se muestra los resultados obtenidos por el algoritmo WDC-CSK en dos diferentes evaluaciones y luego su comparación contra otros algoritmos del estado del arte, entre ellos: Bisecting K-means, STC y Lingo. Los componentes de evaluación y los resultados obtenidos se presentan a continuación.

4.4.1 Pregunta de investigación

A partir de la pregunta de investigación y el tercer objetivo específico del proyecto se planteó la necesidad de definir la calidad del proceso de agrupamiento de documentos Web con el algoritmo propuesto, basado en precisión y recuerdo ponderado sobre una colección de datos de prueba y su comparación con los resultados de algoritmos del estado del arte. La pregunta para éste estudio experimental es:

¿Es posible proporcionar mejores resultados en cuanto a la calidad de los resultados en el proceso de agrupamiento de documentos Web y el proceso de etiquetado a través del algoritmo propuesto WDC-CSK superando los algoritmos del estado del arte?

4.4.2 Objetivo del estudio experimental

El objetivo de este estudio experimental es definir la calidad del proceso de agrupamiento de documentos Web con el algoritmo propuesto en cuanto a la contrastación contra un ideal y la evaluación del comportamiento del usuario.

4.4.3 Contexto del estudio experimental

Para la realización del estudio experimental se empleó el prototipo software WDC que permitió efectuar las evaluaciones del algoritmo propuesto WDC-CSK. Cada evaluación se ejecuta para 30 ciclos y durante un segundo (un escenario real de agrupamiento de documentos Web), todo esto garantizar la confiabilidad de los resultados obtenidos.

4.4.4 Diseño del estudio experimental

De acuerdo al objetivo del estudio experimental, se diseñaron los indicadores, métricas e instrumentos a emplear, la **Tabla 16** relaciona estos elementos para el estudio experimental.

Tabla 16. Parámetros estudio experimental.

Objetivo	Indicadores	Métricas	Instrumentos
El objetivo de este estudio experimental fue definir la calidad del proceso de agrupamiento de documentos Web con el algoritmo propuesto en cuanto a la contrastación contra un ideal y la evaluación del comportamiento del usuario.	Calidad de los resultados del proceso de Agrupamiento	Medida -F	DMOZ-50
		Precisión	
		Recuerdo	AMBIENT
		Fall-Out	
		Exactitud	MORESQUE
	Calidad del proceso de etiquetado	SSLk	ODP-239

4.4.5 Medidas de evaluación

El proceso de evaluación del algoritmo presentado en este proyecto, WDC-CSK, se llevó a cabo incluyendo dos puntos de vista: 1) La contrastación contra un ideal (Validación por Regla de Oro) y 2) La evaluación del comportamiento del usuario (usando la métrica SSL_k). Con estos experimentos se buscaba evaluar el algoritmo propuesto, en cuanto a la calidad de los resultados y la facilidad con que los usuarios pueden utilizarlos y así tener una perspectiva completa de los resultados de agrupamiento obtenidos, A continuación se describen estas dos perspectivas.

4.4.5.1 Contrastación contra un ideal

En esta evaluación se usa la Validación por Regla de Oro, la cual tiene como objetivo evaluar la calidad de los resultados de agrupamiento, es decir, qué tan buenos son los resultados del método de agrupación con respecto a la recuperación de grupos conocidos (denominadas clases) de una partición estándar de oro (ideal). Varias medidas de evaluación están disponibles para esta

tarea, incluyendo: precisión, recuerdo, medida-F, fall-out y exactitud (índice de Rand)[39]. 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 (15) y (16).

(b) Calcular la precisión ponderada (P), recuerdo ponderado (R) y medida-F ponderada (F) basada en (17).

$$P(C, C^i) = \frac{|C \cap C^i|}{|C|} \; y \; R(C, C^i) = \frac{|C \cap C^i|}{|C^i|} \tag{15}$$

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

$$F(C, C^i) = \frac{2 * P(C, C^i) * R(C, C^i)}{P(C, C^i) + R(C, C^i)} \tag{16}$$

$$P = \frac{1}{T}\sum_{j=1}^{h} |C_j^i| * P(C_m, C_j^i), \; R = \frac{1}{T}\sum_{j=1}^{h} |C_j^i| * R(C_m, C_j^i), \text{ y } \; F = \frac{2*P*R}{P+R} \tag{17}$$

donde $T = \sum_{j=1}^{h} |C_j^i|$

4.4.5.2 Evaluación del comportamiento del usuario

En esta evaluación se usa la métrica SSL_k (Subtopic Search Length under k document sufficiency). Esta métrica permite evaluar la facilidad con la que los usuarios pueden utilizar los resultados del proceso de agrupamiento, en decir, evalúa el comportamiento del usuario respecto a los resultados [59, 61, 64]. Esta medida se define como el número promedio de elementos (etiquetas de grupo o resultados de búsqueda) que deben ser examinados antes de encontrar un número suficiente (k) de documentos relevantes para cualquiera de los subtemas de consulta, suponiendo que tanto las etiquetas de los grupos como los resultados de búsqueda se lean secuencialmente de arriba hacia abajo, y se abra ese solo grupo con etiquetas relevantes para los subtemas en cuestión. SSL_k permite una evaluación de recuperación completa de subtemas (la recuperación de varios documentos relacionados para cualquier subtema) en lugar de centrarse en la cobertura de un subtema (es decir, la recuperación de al menos un documento por

algún subtema). SSL_k también permite una modelización realista del comportamiento de búsqueda de los usuarios, porque se tiene en cuenta el papel desempeñado por las etiquetas del grupo.

4.4.6 Parámetros WDC-CSK

En el algoritmo WDC-CSK se destacan dos parámetros, PS y PA, parámetros que debieron ser optimizados, con el fin de tener los mejores resultados. Para ello se realizó un afinamiento de parámetros a través de una combinatoria todos contra todos, con los siguientes valores: PS con 5, 6, 8 y 10; y PA con 10%, 15%, 20% y 25% (valores recopilados de las versiones del algoritmo CS) y evaluados para cada conjunto de datos. Finalmente, los valores de los parámetros se definieron así, MNI se establece en 2, PS en 5, PA en 10%, TFT en 5 y MET en 1000 milisegundos. Los valores de los parámetros son iguales para todos los conjuntos de datos.

En la **Tabla 17** se presentan los resultados obtenidos en el proceso de afinamiento de parámetros sobre el conjunto de datos DMOZ-50, siendo la pareja, PA establecida en 5 y PS en 10%, donde el algoritmo WDC-CSK obtiene resultados superiores para la mayoría de las medidas de evaluación. El comportamiento es similar para los otros conjuntos de datos.

Tabla 17. Afinamiento de parámetros sobre el conjunto de datos DMOZ-50.

PS	PA	K Estimado	Precisión	Recuerdo	Medida-F	Exactitud	Fall-Out
5	**0.10**	**9.08**	90.02	**70.97**	**77.13**	**92.31**	1.49
	0.15	9.19	90.22	70.67	77.10	92.29	1.45
	0.20	9.22	90.15	70.37	76.74	92.16	1.46
	0.25	9.30	90.28	69.94	76.53	92.11	1.45
6	0.10	9.20	90.10	70.31	76.73	92.17	1.46
	0.15	9.29	90.23	69.83	76.46	92.08	1.44
	0.20	9.29	90.24	69.71	76.40	92.06	1.44
	0.25	9.29	90.36	69.93	76.61	92.10	1.44
8	0.10	9.49	90.68	69.04	76.16	91.91	1.35
	0.15	9.57	90.57	68.33	75.58	91.76	1.36
	0.20	9.50	90.40	68.37	75.52	91.73	1.39
	0.25	9.52	**90.85**	68.76	75.97	91.85	1.34
10	0.10	9.69	90.74	67.57	75.12	91.54	1.33
	0.15	9.65	90.77	67.57	75.10	91.56	1.33
	0.20	9.69	90.82	67.69	75.24	91.60	**1.32**
	0.25	9.66	90.82	67.72	75.27	91.60	1.33

4.4.7 Conjunto de datos para validación

En la ejecución de los experimentos el algoritmo propuesto fue utilizado para el agrupamiento de resultados Web sobre cuatro conjuntos de datos de evaluación tradicionales reconocidos dentro de la comunidad académica e investigativa de recuperación de información, estos son: DMOZ-50, AMBIENT, MORESQUE y

ODP-239. En estos datos se encuentran un total de 447 consultas con sus respuestas ideales. A continuación se presenta una descripción general de los conjuntos de datos utilizados.

El conjunto de datos **DMOZ-50** consta de 50 consultas derivadas del Open Directory Project (acrónimo para el directorio de Mozilla). Cada consulta tiene en promedio 129.14 documentos, 6.02 subtemas (significados de temas muy diferentes) y 22.62 resultados relevantes por cada subtema recuperado. Cada consulta es una gran colección de documentos, cada una con una cantidad de clases relativamente pequeña, y un gran número de documentos por cada clase. En este conjunto de datos, no están disponibles las palabras claves de la consulta. La colección se encuentra disponible para descargar en http://artemisa.unicauca.edu.co/~ccobos/wdc/wdc.htm.

El conjunto de datos **AMBIENT** (AMBIguous ENTries) consta de 44 consultas extraídas de entradas *ambiguas* de Wikipedia. Cada consulta tiene en promedio 50.55 resultados de búsqueda clasificados, recolectados de Yahoo! (anotados manualmente con juicios de confianza por subtema a nivel de documento), 7.91 subtemas y 7.72 resultados relevantes recuperados por cada subtema. La mayoría de las consultas en AMBIENT son de una sola palabra (1 palabra clave) y están todas disponibles. El conjunto de datos AMBIENT mide la habilidad de recuperar los subtemas contenidos en los resultados de búsqueda (los documentos obtenidos por Yahoo!), no todos los posibles subtemas de una consulta. Este conjunto de datos se encuentra disponible para su descarga en http://credo.fub.it/ambient.

El conjunto de datos **MORESQUE** (MORE Sense-tagged QUEry results) consta de 114 consultas ambiguas que se desarrollaron como complemento al conjunto de datos AMBIENT. Este conjunto de datos pone a prueba el comportamiento de los algoritmos de búsqueda Web en consultas de diversa longitud, variando entre 1 y 4 palabras. Este conjunto de datos contiene 114 consultas de longitud 2, 3 y 4 palabras (todas disponibles), junto a un promedio de 53,54 resultados principales (documentos) de Yahoo!, 3.82 subtemas y 19.43 resultados relevantes por subtema obtenido. Este conjunto de datos se puede descargar en http://lcl.uniroma1.it/moresque.

El conjunto de datos **ODP-239** consta de 239 consultas derivadas de Open Directory Project. Cada consulta tiene en promedio 106.95 documentos (cada uno compuesto de URL, título y una corta descripción), 9.56 subtemas y 11.38 resultados relevantes por cada subtema encontrado. ODP-239 está compuesto por muchas colecciones pequeñas, cada una con una cantidad de clases

relativamente grande, en lugar de tener una gran colección de documentos con un número pequeño de clases. Los temas, subtemas y documentos asociados fueron seleccionados de tal forma que la distribución de los documentos en los subtemas refleja la importancia relativa de cada uno. El conjunto de datos está disponible para su descarga en http://credo.fub.it/odp239.

4.4.8 Sistemas comparados

En la literatura revisada se destacan algunos algoritmos de agrupación de documentos Web, para el proceso de evaluación en esta investigación se tomaron los resultados de los algoritmos: STC, Lingo y Bisecting K-means; resultados que se compararon con los resultados obtenidos por el algoritmo propuesto WDC-CSK. A continuación se presenta un resumen de dichos métodos de agrupación:

Suffix Tree Clustering (STC) [27] es el enfoque original de agrupamiento de búsquedas Web basado en árboles de sufijos y frases frecuentes.

Lingo [31], que es un sucesor muy conocido de STC. En este algoritmo de agrupamiento Web (implementado en el framework de código abierto Carrot^2) primero se extraen frases frecuentes de los documentos utilizando matrices de sufijos, luego, se seleccionan las mejores frases frecuentes usando Descomposición del Valor Singular (Singular Value Decomposition, SVD), y finalmente, los documentos se asignan a dichas frases frecuentes.

Bisecting K-means [29, 33, 36], algoritmo que 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

Adicionalmente, el algoritmo WDC-CSK se comparó con los resultados de Lingo3G, KeySRC, OPTIMSRC y Yahoo! utilizando los resultados previamente reportados. **Lingo3G** es un algoritmo de agrupamiento Web comercial también disponible en Carrot^2. Este algoritmo es muy diferente a Lingo, utiliza una meta-heurística hecha a la medida para seleccionar las etiquetas del grupo bien definidas y diversas. **KeySRC** [59] es un motor de agrupamiento Web construido sobre STC con una poda basada en técnicas de detección de las partes de un discurso (part-of-speech) y la selección dinámica del nivel de corte en el dendograma de agrupamiento. **OPTIMSRC** [61] es un algoritmo de agrupamiento de documentos Web basado en la generación de grupos usando un algoritmo estocástico de ascenso de colinas seguido de un proceso de etiquetado basado en Lingo, STC y KeySRC. Y los resultados de Yahoo! que son los resultados de

búsqueda originales devueltos por el motor de búsqueda Yahoo!. En la referencia [61] se presentan los resultados de SSL para Yahoo! en el conjunto de datos AMBIENT.

4.4.9 Resultados y discusión

Para el proceso de evaluación el algoritmo propuesto WDC-CSK fue ejecutado durante un tiempo máximo de un segundo (un escenario real de agrupamiento de documentos Web) en cada conjunto de datos. Este proceso se repitió 30 veces sobre cada conjunto de datos individual. Adicional a los parámetros especificados el algoritmo se ejecutó empleando como función de aptitud tanto BBIC como BIC.

Las especificaciones del equipo de cómputo utilizado son las siguientes: Procesador Intel Celeron de 2.20 GHz, 4 GB de Memoria RAM y Sistema operativo Windows 7 de 64 Bits.

A continuación se presentan los resultados obtenidos por el algoritmo WDC-CSK para cada una de las evaluaciones definidas anteriormente.

4.4.9.1 Resultados de la validación por regla de oro

Luego de efectuar esta prueba, con el objetivo de evaluar la calidad de los resultados de agrupamiento, en la **Tabla 18** se muestra los resultados para cada conjunto de datos y algoritmo. Esta tabla presenta los resultados promedio para todos los algoritmos, debido a que cada algoritmo fue ejecutado 30 veces sobre cada conjunto de datos individual; además, WDC-CSK fue ejecutado durante aproximadamente un segundo (un escenario real de agrupamiento de documentos Web) en cada conjunto de datos.

Tabla 18. Resultados de la validación por regla de oro.

Conjunto de Datos	Algoritmo	K Estimado	Diferencia con el k ideal	Precisión	Recuerdo	Medida F	Exactitud	Fall-Out
DMOZ-50	WDC-CSK BBIC	**9.08**	**3.06**	**90.02**	**70.97**	**77.13**	**92.31**	1.49
	WDC-CSK BIC	9.59	3.57	**90.08**	67.19	74.59	91.36	**1.44**
	Lingo	34.29	28.27	83.85	37.88	48.23	83.41	4.76
	STC	16.00	9.98	84.82	57.85	65.12	88.81	3.08
	Bisecting K-means	10.98	4.96	70.94	43.37	50.32	84.42	4.58
AMBIENT	WDC-CSK BBIC	**7.32**	**0.59**	77.46	**58.94**	**61.79**	**82.80**	4.04
	WDC-CSK BIC	7.40	**0.59**	77.58	57.33	60.51	82.22	4.02
	Lingo	20.86	12.95	**86.75**	50.21	58.68	80.43	**3.06**
	STC	11.00	3.09	72.40	53.14	55.38	81.89	5.64
	Bisecting K-means	11.39	3.48	76.46	40.65	45.97	77.12	3.54
MORESQUE	WDC-CSK BBIC	7.78	1.78	88.16	40.10	49.77	58.75	5.78
	WDC-CSK BIC	**7.61**	1.95	87.86	39.17	48.70	58.12	5.68
	Lingo	20.16	10.60	**90.50**	39.35	50.55	59.18	6.13
	STC	11.17	1.61	82.83	**49.96**	**57.18**	**65.45**	12.76

Conjunto de Datos	Algoritmo	K Estimado	Diferencia con el k ideal	Precisión	Recuerdo	Medida F	Exactitud	Fall-Out
	Bisecting K-means	10.36	**0.80**	87.36	30.05	38.69	53.47	**4.36**
ODP-239	WDC-CSK BBIC	9.67	2.13	60.46	**44.15**	**46.23**	**81.74**	5.59
	WDC-CSK BIC	**9.22**	**1.69**	59.59	43.03	44.98	81.32	**5.39**
	Lingo	31.39	23.85	**71.56**	32.93	41.01	79.15	6.60
	STC	15.98	8.44	57.33	39.74	41.80	80.65	9.90
	Bisecting K-means	11.75	4.21	55.60	32.12	34.92	78.16	6.21
Promedio	WDC-CSK BBIC	8.89	1.99	72.51	**47.58**	**52.12**	77.16	5.03
	WDC-CSK BIC	**8.67**	**1.86**	71.98	46.16	50.77	76.62	4.89
	Lingo	27.81	19.89	**79.26**	36.82	45.99	74.66	5.93
	STC	14.27	6.34	68.39	45.69	49.67	**77.81**	9.45
	Bisecting K-means	9.63	2.09	76.23	41.19	47.17	71.64	**4.59**

En el conjunto de datos DMOZ-50, el algoritmo propuesto WDC-CSK (BBIC y BIC) supera los algoritmos Lingo y STC en las medidas de evaluación: precisión, recuerdo, medida-F, exactitud y fall-out. En el conjunto de datos AMBIENT, WDC-CSK (BBIC y BIC) supera en recuerdo, medida-F y precisión a Lingo y STC. Además el valor de fall-out es competitivo (la diferencia con el mejor valor en esta medida de evaluación no es estadísticamente significativa) en este conjunto de datos. Los resultados en el conjunto de datos MORESQUE son favorables para STC. WDC-CSK (BBIC y BIC) y Lingo tienen reportes similares en todas las medidas. En el conjunto de datos ODP-239, WDC-CSK (BBIC y BIC) supera en recuerdo, medida-F, fall-out y exactitud a los otros algoritmos.

En general, WDC-CSK (BBIC y BIC) obtiene un mejor número de grupos en todos los conjuntos de datos y la diferencia es estadísticamente significativa, por lo tanto, se contribuye a que el usuario encuentre más rápidamente el tópico de información que está buscando. En promedio WDC-CSK (BBIC y BIC) es diferente al número ideal de grupos por cerca de 1.86-1.99 grupos, mientras Lingo es diferente por 19.89, STC por 6.34 y Bisecting K-means por 2.09. Los resultados promedio reportan que WDC-CSK supera en cuanto a recuerdo, medida-F, fall-out, y exactitud a todos los algoritmos. En la siguiente tabla se muestran los resultados obtenidos en la prueba Friedman por cada medida de evaluación.

Tabla 19. Clasificación de la prueba de Friedman por regla de oro.

Medida	Precisión		Recuerdo		Medida-F		Exactitud		Fall-Out	
Algoritmo	Rango Promedio	Posición	Rango Promedio	Posición	Rango Promedio	Posición	Rango Promedio	Posición	Rango Promedio	Posición
WDC-CSK BBIC	2.7483	2	**1.8736**	**1**	**1.9843**	**1**	**1.9978**	**1**	2.7036	2
WDC-CSK BIC	2.9821	3	2.3087	2	2.5168	2	2.4586	2	**2.5246**	**1**
Lingo	**1.962**	**1**	3.8121	4	3.104	4	3.3378	4	2.8389	3
STC	3.3859	4	2.6353	3	2.8289	3	2.8859	3	3.6477	5

Medida	Precisión		Recuerdo		Medida-F		Exactitud		Fall-Out	
Algoritmo	Rango Promedio	Posición	Rango Promedio	Posición	Rango Promedio	Posición	Rango Promedio	Posición	Rango Promedio	Posición
Bisecting K-means	3.9217	5	4.3702	5	4.566	5	4.3199	5	3.2852	4

Los resultado de la medida-F usando la prueba de Friedman muestran que WDC-CSK BBIC es el mejor algoritmo con una estadística de Friedman (distribuido de acuerdo a chi-cuadrado con 4 grados de libertad) igual a 671.844743 y el valor de P calculado por la prueba de Friedman igual a 2.1438339992130295E-10 (ver **Tabla 19**). Además, los resultados de la prueba de Wilcoxon muestran que WDC-CSK BBIC supera a todos los algoritmos, WDC-CSK BIC, supera los otros tres algoritmos y STC supera a Lingo y Bisecting K-means con un nivel de confianza igual a 90%.

El valor promedio de recuerdo muestra que WDC-CSK BBIC es el mejor algoritmo con una estadística de igual a 789.697987 y el valor de P igual a 2.532015708212043E-10 (ver **Tabla 19**). Además, aplicando la prueba de Wilcoxon, WDC-CSK BBIC supera a todos los algoritmos con un nivel de confianza de 95%, WDC-CSK BIC supera a Lingo, STC y Bisecting K-means con un nivel de confianza igual a 90% en la misma prueba. Finalmente, Lingo supera a STC y Bisecting K-means con un nivel de confianza igual a 90%.

El valor promedio de exactitud muestra que WDC-CSK BBIC es el mejor algoritmo con una estadística de Friedman (igual a 566.237136 y el valor de P igual a 2.0056112326471975E-10 (ver **Tabla 19**). Además, WDC-CSK BBIC supera todos los algoritmos con un nivel de confianza de 90% en la prueba de Wilcoxon, y supera a WDC-CSK BIC, Lingo y Bisecting K-means con un nivel de confianza igual a 95% en la misma prueba. Finalmente, WDC-CSK BIC supera a Lingo y Bisecting K-means con un nivel de confianza igual a 95%.

El valor promedio de fall-out muestra que WDC-CSK BIC es el mejor algoritmo con una estadística de Friedman igual a 150.302461 y el valor de P igual a 1.0063594402254239E-10 (ver **Tabla 19**). Además, WDC-CSK BIC supera a todos los algoritmos, WDC-CSK BIC supera a Lingo, STC y Bisecting K-means con un nivel de confianza igual a 95% en los resultados de la prueba de Wilcoxon. Finalmente, Bisecting K-means supera a STC y Lingo con un nivel de confianza igual a 90% en los resultados de la prueba de Wilcoxon.

El valor promedio de precisión muestra que Lingo es el mejor algoritmo con una estadística de Friedman igual a 382.565101 y el valor de P igual a 1.56726742694957E-10 (ver **Tabla 19**). Además, Lingo supera a todos los algoritmos, WDC-CSK BBIC supera los otros tres algoritmos, y WDC-CSK BIC

supera a STC y Bisecting K-means con un nivel de confianza igual a 95% en los resultados de la prueba de Wilcoxon.

Es importante destacar que aunque los resultados obtenidos por Lingo en precisión superan a los demás algoritmos, este resultado es sesgado, debido a que el número de grupos obtenidos es demasiado alto (excediendo en 19.89 al número de grupos ideal de referencia) y el valor de recuerdo fue inferior, es así como el valor de precisión sólo muestra que Lingo es capaz de asignar un número pequeño de documentos relacionados con el mismo tema en un pequeño número de agrupaciones generadas. Lo que implica que el usuario emplee mucho más tiempo en encontrar los resultados deseados.

En la **Figura 9**, se presentan las curvas de precisión, recuerdo y medida-F a través de diferente número de iteraciones.

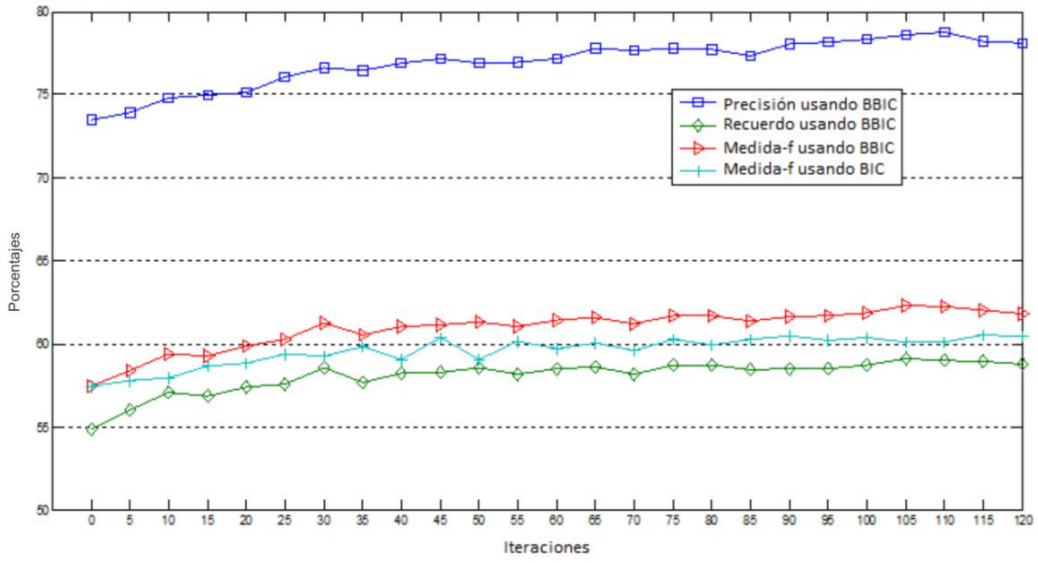

Figura 9. Precisión, recuerdo y medida-F para WDC-CSK BBIC a través de diferentes iteraciones en el conjunto de datos AMBIENT.

Los resultados muestran que todos los valores de las medidas de evaluación incrementaron con el número de iteraciones. Por lo tanto, cuando los usuarios pueden esperar más tiempo por los resultados, WDC-CSK organiza de una mejor manera los grupos de documentos y prueba ser la mejor opción. La utilización de BBIC con similitud de cosenos es una buena opción para el agrupamiento de resultados Web porque la precisión y el recuerdo aumentan cuando WDC-CSK optimiza BBIC, pero en algunas iteraciones (por ejemplo, iteración 40 a 50) esta relación positiva fracasa. En consecuencia, el grupo de investigadores se propone definir una mejor función de aptitud para los algoritmos evolutivos en el agrupamiento de resultados Web basados en programación genética. Además

análisis posteriores mostraron que, en general, WDC-CSK BBIC incrementa la calidad de los grupos (basada en precisión, recuerdo, y medida-F) cuando se utiliza más iteraciones sin importar el número de documentos, número de temas, o el número de atributos en el conjunto de datos. Sólo el conjunto de datos de MORESQUE no cumple con esta regla, debido a la ambigüedad de las consultas, donde la alta similitud entre los subtemas no permite una precisa definición de las nombres de los grupos en el proceso de etiquetado.

Al generar nuevos nidos usando un abandono, los métodos de Split y Merge del algoritmo propuesto WDC-CSK, la función de aptitud BBIC incrementan su efectividad a través de las iteraciones. La **Figura 10** muestra un 32% de efectiva de la nueva solución en las primeras cinco iteraciones, es decir, la nueva solución es mejor que otras soluciones en la población de nidos. Después, la efectividad aumenta en un 54% en cinco iteraciones más. Luego aumenta a 67% en la iteración 15, y finalmente esta alrededor del 90% en la décima octava iteración. El comportamiento en la **Figura 10** es para el conjunto de datos AMBIENT, pero este es similar para los otros conjuntos de datos. El comportamiento de WDC-CSK utilizando BIC como función de aptitud es similar, pero en promedio es un 2.7% menos efectiva en cada iteración.

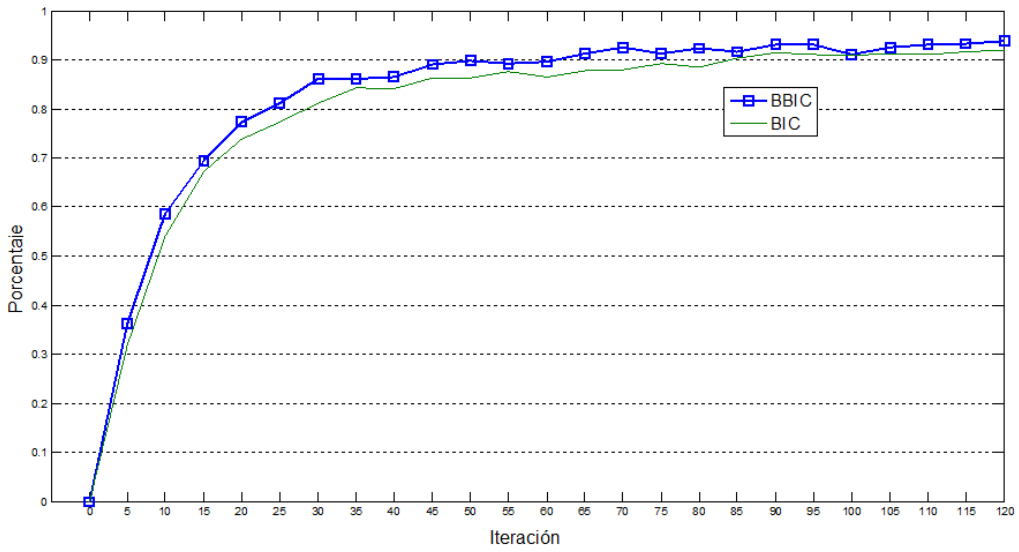

Figura 10. Efectividad de nuevos nidos generados en diferente número de iteraciones sobre el conjunto de datos AMBIENT.

El número estimado de agrupaciones en WDC-CSK (BBIC y BIC) está mejor definido que el de los algoritmos Lingo, STC y Bisecting K-means. DMOZ-50 tiene un promedio de 6.02 subtemas, WDC-CSK encuentra en promedio, 9.08 a 9.59 grupos, mientras que Lingo encuentra 34.29, STC encuentra 16 y Bisecting K-means encuentra 10.98 grupos. En AMBIENT se encuentra un comportamiento

similar. AMBIENT tiene en promedio 7.91 subtemas, WDC-CSK encuentra en promedio 7.32 a 7.40 mientras Lingo encuentra 20.86, STC 11 y Bisecting K-means 11.39, y finalmente ODP-239 tiene en promedio 9.56 subtemas, WDC-CSK encuentra en promedio 9.22 a 9.67 mientras Lingo encuentra 31.39, STC encuentra 15.98 y Bisecting K-means 11.75. Con un alto número de agrupaciones el algoritmo incrementa la precisión, pero es más difícil para los usuarios encontrar la información requerida.

4.4.9.2 Resultados de la evaluación del comportamiento del usuario

La medida SSL_k se utilizó para evaluar la facilidad con la que los usuarios pueden utilizar los resultados de las búsquedas. Como se explicó anteriormente, esta medida evalúa la cantidad promedio de elementos (documentos) que un usuario debe examinar antes de encontrar una cantidad adecuada (k) de documentos relevantes al tema seleccionado.

En esta evaluación no se tiene en cuenta los resultados de SSL_K del algoritmo Bisecting K-means debido a que usa términos estadísticamente representativos y no frases frecuentes, lo que conlleva a que la asignación de etiquetas a los grupos no sea la adecuada. En el estado del arte ya está definido que la presentación con términos no es la adecuada para sistemas que agrupan documentos Web.

Los algoritmos se probaron sobre todas las consultas de los diferentes conjuntos de datos y el desempeño de la salida correspondiente fueron evaluados utilizando SSL_k, con *k = 1, 2, 3, 4*. Los resultados, promediados del total de las pruebas, se reportan en la **Tabla 20**.

Tabla 20. Evaluación comportamiento del usuario.

Conjunto de Datos	Algoritmo	SSL_1	SSL_2	SLL_3	SSL_4	Suma de SSL_k
DMOZ-50	WDC-CSK BBIC	17,23	20,97	23,66	25,93	87,78
	WDC-CSK BIC	16,87	20,82	23,45	25,79	86,93
	Lingo	14,2	16,6	18,5	21,9	71,2
	STC	**12,1**	**16,4**	**18,6**	**21,3**	**68,4**
AMBIENT	WDC-CSK BBIC	**15,33**	**26,58**	**33,18**	**37,80**	**112,89**
	WDC-CSK BIC	15,60	26,84	33,48	38,02	113,93
	Lingo	22,4	36,5	47,2	54,3	160,5
	STC	27,2	44,9	54,8	60,4	187,4
	Best combination*	21,7	29,3	33,2	**37,3**	121,4
	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	138,9
	Yahoo!*	21,6	35,5	42,0	47,6	146,6
MORESQUE	WDC-CSK BBIC	**11,65**	**19,13**	**24,63**	**28,37**	**83,78**
	WDC-CSK BIC	12,00	19,60	25,01	28,67	85,28
	Lingo	16,5	26,4	33,9	39,2	116,0

Conjunto de Datos	Algoritmo	SSL_1	SSL_2	SLL_3	SSL_4	Suma de SSL_k
	STC	19,6	32,3	40,2	45,2	137,2
ODP-239	WDC-CSK BBIC	20,18	**30,09**	**39,71**	**51,64**	**141,63**
	WDC-CSK BIC	**20,16**	30,23	40,00	51,92	142,30
	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	75,0	195,2

* Tomado de [61]

** Tomado de [26]

El promedio de los valores individuales de SSL_k y la suma de SSL_k muestra que WDC-CSK BBIC es el mejor algoritmo con una estadística de Friedman (distribuida de acuerdo al chi-cuadrado con 3 grados de libertad) igual a 339.541611 y el valor de P calculado por la prueba de Friedman igual a 1.5249290719054898E-10. En promedio los rangos de la medida fueron: 1.9172 para WDC-CSK BBIC, 1.9642 para WDC-CSK BIC, 2.9765 para Lingo y 3.1421 para STC. Además, WDC-CSK (BBIC y BIC) supera todos los algoritmos, y Lingo supera a STC con un nivel de confianza igual a 95% en los resultados de la prueba de Wilcoxon.

El conjunto de datos DMOZ-50 presenta pobres resultados en el algoritmo WDC-CSK porque este conjunto de datos no contiene palabras claves, mientras que el conjunto de datos MORESQUE tiene 2, 3 o 4 palabras claves para describir la consulta y por lo tanto en este conjunto de datos el algoritmo propuesto supera los resultados de SSL_k en un mayor grado. Las palabras claves en las consultas son de gran importancia para el proceso de etiquetado en el algoritmo WDC-CSK.

Capítulo 5

5 CONCLUSIONES, RECOMENDACIONES Y TRABAJO FUTURO

5.1 CONCLUSIONES

Se presenta una nueva versión del algoritmo CS llamado Búsqueda Cucú usando Modelos Evolutivos que Aprenden, CS+LEM. El algoritmo propuesto utiliza técnicas de LEM para crear un conjunto de reglas que permita inferir nuevos candidatos en la población que no surjan solamente a partir de la exploración aleatoria. Se sometió el algoritmo a 61 funciones de optimización clásicas y en la mayoría supera los resultados obtenidos. En general, CS+LEM es la mejor opción para resolver los problemas si el diseñador no sabe nada sobre el panorama de la función de aptitud, el problema tiene una alta dimensionalidad y el tiempo de ejecución es corto (Prueba 1). El diseño y evaluación de ésta propuesta de algoritmo fue adicional al proyecto, permitiendo tener una visión más amplia del algoritmo general en el proceso de optimización continua y la viabilidad en el problema de agrupamiento de documentos Web.

La hibridación de CS con el algoritmo K-means (COA) reporta los mejores resultados en la prueba 2 (número de evaluaciones de la función objetivo requeridas para alcanzar el óptimo global) que los otro algoritmos evaluados. COA alcanza el valor óptimo global de 55 funciones de prueba sobre un total 61 funciones, éste es un número alto de funciones resueltas en comparación con las otras versiones del algoritmo CS que solo son capaces de resolver de 7 a 14 funciones de prueba usando 50.000 evaluaciones de la función objetivo. Desafortunadamente, COA requiere mucho más tiempo de ejecución que los otros algoritmos CS, siendo hasta 6 veces más lento que MCS el cual reporta el menor tiempo promedio por generación. Por lo tanto, COA es la mejor opción para ser utilizado en escenarios fuera de línea donde los usuarios pueden esperar mucho más tiempo para la solución óptima global del problema. Además, es importante resaltar que el algoritmo propuesto CS+LEM ocupa el segundo lugar en esta prueba alcanzando el óptimo global de 14 funciones de prueba.

COA y MCS reportan en la prueba 3 (Mejor valor óptimo alcanzado en diferente número de evaluaciones de la función) mejores resultados que los otros algoritmos. Para un alto número de evaluaciones de la función objetivo COA reporta los mejores resultados, mientras tanto el algoritmo MCS reporta los mejores resultados para un bajo número de evaluaciones.

El algoritmo WDC-CSK muestra mejores resultados cuando se utiliza el Criterio de Información Bayesiano Balanceado (BBIC) para decidir qué solución es mejor que otra, por que incrementan su efectividad a través de las iteraciones.

El algoritmo WDC-CSK trabaja con nidos como esquema de representación de las soluciones. Cada nido tiene un número diferente de grupos, una lista de centroides, y el valor de la función objetivo (basada en BBIC o BIC) que depende de la ubicación de los centroides y del número de centroides en cada nido. Este esquema de representación, aunque usa atributos individuales de valor real – que en principio genera un espacio de búsqueda infinito - funciona apropiadamente con el uso del K-means como optimizador local y la estrategia de Forgey como inicializador de la población, debido a que se generan nuevas soluciones que cada vez son mejores y entran a reemplazar las peores de la población, teniendo una mejora incremental.

Se diseñó, implementó y evaluó el algoritmo propuesto WDC-CSK, algoritmo centrado en la descripción para el agrupamiento de documentos Web basado en el algoritmo meta-heurístico de la Búsqueda Cucú (estrategia de búsqueda global) y el algoritmo K-means (estrategia local para mejorar la solución) con la capacidad de definir de forma automática el número de grupos. El algoritmo WDC-CSK muestra excelentes resultados experimentales en conjuntos de datos de prueba tradicionales del área de investigación y su comparación con varios algoritmos del estado del arte muestra que se obtuvo las siguientes mejoras en las principales medidas de evaluación, en Medida-F se obtuvo una mejora de entre el 4.5% y el 13%, entre 3.5% y el 28% en recuerdo, entre 1% y 3% en exactitud y entre el 18% y el 49% en fall-out. Los experimentos también muestran mejoras sobre SSL_k acumulado valores de entre el 21% y el 31%.

5.2 RECOMENDACIONES Y TRABAJO FUTURO

Como recomendación, se propone la realización de convenios con universidades y centros de investigación del país o del exterior con el fin de realizar evaluaciones más exhaustivas usando súper-computadoras, por ejemplo con el CINVESTAV de México, aprovechando de esta forma los recientes acercamientos con el profesor José Torres Jiménez del CINVESTAV-Tamaulipas.

Como trabajo futuro, se propone comparar las versiones del algoritmo CS con otras heurísticas Optimización del Enjambre de Partículas (Particle Swarm Optimization, PSO), Evolución Diferencial (Differential Evolution, DE), Búsqueda Armónica (Harmony Search, HS) y Colonia Artificial de Abejas (Artificial Bee Colony, ABC) de manera detallada como en el presente trabajo. Además de hacer

una propuesta del algoritmo Búsqueda Cucú que combine las fortalezas de cada uno de los algoritmos presentados en uno sólo capaz de usar inteligentemente la estrategia más adecuada en cada iteración.

Hay varias tareas para trabajos futuros en agrupamiento de documentos Web, entre ellas: 1) Diseñar otros algoritmos bio-inspirados para el agrupamiento de resultados Web y comparar los resultados con WDC-CSK, 2) El uso de técnicas de desambiguación con la finalidad de mejorar la calidad de los resultados del grupo y comparar los resultados con otros algoritmos, y 3) El uso de WordNet u otra herramienta semántica para trabajar con conceptos en vez de términos y compararlos con otros algoritmos en el estado del arte,

GLOSARIO

Optimización: Es un proceso conocido también como programación matemática cuyo objetivo es intentar dar respuesta a un tipo general de problemas donde se desea elegir la mejor solución dentro de un conjunto de soluciones [83].

Sistemas IR: Proceso que accede a información previamente almacenada, mediante herramientas informáticas que permiten establecer ecuaciones de búsqueda específicas.

Snippet: En el contexto del tema de investigación de éste proyecto, hace referencia al fragmento de texto que describe cada documento resultado de una búsqueda.

Grupo (Clúster): Este término se aplica a una colección de objetos que son "similares" entre si y son "distintos" a los objetos pertenecientes a otros grupos.

Etiquetas: En el proceso de agrupamiento de documentos Web, este término hace referencia a los elementos encargados de identificar o describir un clúster.

Centroide: En el tema de esta investigación, este término hacer referencia al punto que define el centro del clúster.

SVD: Abreviatura que referencia “Singular Value Descomposition”, la cual es una técnica que sirve para realizar factorización de matrices.

Lucene: Es un API de código abierto para recuperación de información, originalmente implementada en Java por Doug Cutting. Está apoyado por el Apache Software Foundation y se distribuye bajo la Apache Software License. Esta API tiene versiones para otros lenguajes incluyendo C#, es útil para cualquier aplicación que requiera indexado y búsqueda de texto completo. Lucene ha sido ampliamente usado por su utilidad en la implementación de motores de búsqueda.

Función Unimodal: Es una función matemática para la cual existe una única solución óptima (expresada como un mínimo o un máximo).

Funciones Multimodal: Es una función matemática para la cual existe más de una solución óptima.

BIBLIOGRAFÍA

[1] C. M. a. A. Amouroux, "Governing the Internet, Freedom and Regulation in the OSCE Region," *The Representative on Freedom of the Media,* p. 231, 2007.

[2] J. A. Anita Ferreira, "Disminución de la sobrecarga de información en la World Wide Web a partir de interacciones dialógicas hombre-computador," p. 19, 2009.

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

[4] J. Moore, E.-H. S. Han, D. Boley, M. Gini, R. Gross, K. Hastings, G. Karypis, V. Kumar, and B. Mobasher, "Web Page Categorization and Feature Selection Using Association Rule and Principal Component Clustering," in *Workshop on Information Technologies and Systems*, 1997.

[5] J. Madrid and S. Gauch, "Incorporating Conceptual Matching in Search," in *Conference on Information and Knowledge Management* McLean, VA 2002.

[6] D. H. Wolpert and W. G. Macready, "No Free Lunch Theorems for Optimization," *IEEE Transactions on Evolutionary Computation 1,* vol. 1, p. 16, 1997.

[7] X.-S. Yang and S. Deb, "Engineering Optimisation by Cuckoo Search," *International Journal Mathematical Modelling and Numerical Optimisation,* vol. 1, p. 17, 2010.

[8] Y. Xin-She and S. Deb, "Cuckoo Search via Lévy flights," in *World Congress on Nature & Biologically Inspired Computing*, 2009, pp. 210-214.

[9] R. Rajabioun, "Cuckoo Optimization Algorithm," *Applied Soft Computing,* vol. 11, pp. 5508-5518, 2011.

[10] E. Valian, S. Mohanna, and S. Tavakoli, "Improved Cuckoo Search Algorithm for Feedforward Neural Network Training," *International Journal of Artificial Intelligence & Applications (IJAIA),* vol. 2, p. 8, 2011.

[11] S. Walton, O. Hassan, K. Morgan, and M. R. Brown, "Modified cuckoo search: A new gradient free optimisation algorithm," *Chaos, Solitons & Fractals,* vol. 44, pp. 710-718, 2011.

[12] M. Tuba, M. Subotic, and N. Stanarevic, "Modified cuckoo search algorithm for unconstrained optimization problems," presented at the Proceedings of the 5th European conference on European computing conference, Paris, France, 2011.

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

[14] T. Bäck, *Evolutionary algorithms in theory and practice : evolution strategies, evolutionary programming, genetic algorithms*. New York: Oxford University Press, 1996.

[15] F. Glover and R. Marti, "Tabu Search," in *Metaheuristic Procedures for Training Neutral Networks*. vol. 36, E. Alba and R. Martí, Eds., ed: Springer US, 2006, pp. 53-69.

[16] N. Bacanin, "An object-oriented software implementation of a novel cuckoo search algorithm," presented at the Proceedings of the 5th European conference on European computing conference, Paris, France, 2011.

[17] X.-S. Yang, *Nature-Inspired Metaheuristic Algorithms*, Second ed.: Luniver Press, 2010.

[18] D. Karaboga and B. Akay, "A comparative study of Artificial Bee Colony algorithm," *Applied Mathematics and Computation*, vol. 214, pp. 108-132, 2009.

[19] I. Landa-Torres, S. Gil-Lopez, J. Del Ser, S. Salcedo-Sanz, D. Manjarres, and J. A. Portilla-Figueras, "Efficient citywide planning of open WiFi access networks using novel grouping harmony search heuristics," *Engineering Applications of Artificial Intelligence*, vol. 26, pp. 1124-1130, 2013.

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

[21] Q. H. Nguyen, Y. S. Ong, and N. Krasnogor, "A study on the design issues of Memetic Algorithm," in *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, 2007, pp. 2390-2397.

[22] X.-S. Yang and S. Deb, "Cuckoo Search via Lévy Flights," p. 7, 2009.

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

[24] C. J. V. Rijsbergen, *Information Retrieval*: Butterworth-Heinemann, 1979.

[25] G. Mecca, S. Raunich, and A. Pappalardo, "A new algorithm for clustering search results," *Data & Knowledge Engineering*, vol. 62, pp. 504-522, 2007.

[26] 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.

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

[28] L. Jing, "Survey of Text Clustering," ed. The University of Hong Kong, 2008.

[29] K. Hammouda, "Web Mining: Clustering Web Documents A Preliminary Review," ed. University of Waterloo, 2001.

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

[31] S. Osiński and D. Weiss, "A concept-driven algorithm for clustering search results," *Intelligent Systems, IEEE*, vol. 20, pp. 48-54, 2005.

[32] W. Song, C. H. Li, and S. C. Park, "Genetic algorithm for text clustering using ontology and evaluating the validity of various semantic similarity measures," *Expert Systems with Applications*, vol. 36, pp. 9095-9104, 2009.

[33] Y. Li, S. M. Chung, and J. D. Holt, "Text document clustering based on frequent word meaning sequences," *Data & Knowledge Engineering*, vol. 64, pp. 381-404, 2008.

[34] L. Xiang-Wei, H. Pi-Lian, and W. Hui-Ying, "The research of text clustering algorithms based on frequent term sets," in *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on*, 2005, pp. 2352-2356 Vol. 4.

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

[36] M. Steinbach, G. Karypis, and V. Kumar, "A comparison of document clustering techniques," University of Minnesota Technical Report #00-034, 2000.

[37] P. Berkhin, J. Kogan, C. Nicholas, and M. Teboulle, "A Survey of Clustering Data Mining Techniques," in *Grouping Multidimensional Data*, ed: Springer-Verlag, 2006, pp. 25-71.

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

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

[40] 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.

[41] M. H. Chehreghani, H. Abolhassani, and M. H. Chehreghani, "Density link-based methods for clustering web pages," *Decision Support Systems*, vol. 47, pp. 374-382, 2009.

[42] 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.

[43] 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.

[44] T. Matsumoto and E. Hung, "Fuzzy clustering and relevance ranking of web search results with differentiating cluster label generation," in *Fuzzy Systems (FUZZ), 2010 IEEE International Conference on*, 2010, pp. 1-8.

[45] E. Fersini, E. Messina, and F. Archetti, "A probabilistic relational approach for web document clustering," *Information Processing & Management*, vol. 46, pp. 117-130, 2010.

[46] I. Lee and B.-W. On, "An effective web document clustering algorithm based on bisection and merge," *Artif. Intell. Rev.*, vol. 36, pp. 69-85, 2011.

[47] X. He, J.-B. Wang, Z.-X. Zhang, and Y.-R. Cai, "Clustering web documents based on Multiclass spectral clustering," in *Machine Learning and Cybernetics (ICMLC), 2011 International Conference on*, 2011, pp. 1466-1471.

[48] D. Zhang and Y. Dong, "Semantic, Hierarchical, Online Clustering of Web Search Results," in *Advanced Web Technologies and Applications*, ed, 2004, pp. 69-78.

[49] B. Fung, K. Wang, and M. Ester, "Hierarchical document clustering using frequent itemsets," in *Proceedings of the SIAM International Conference on Data Mining*, 2003.

[50] F. Beil, M. Ester, and X. Xu, "Frequent term-based text clustering," in *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, Edmonton, Alberta, Canada, 2002, pp. 436-442.

[51] S. Osiński, "An Algorithm for clustering of web search results," Master, Poznań University of Technology, Poland, 2003.

[52] S. Osiński, J. Stefanowski, and D. Weiss, "Lingo: Search results clustering algorithm based on Singular Value Decomposition," in *Proceedings of the*

International Conference on Intelligent Information Systems (IIPWM), 2004, pp. 359-368.

[53] S. Osiński and D. Weiss, "Conceptual clustering using Lingo algorithm: Evaluation on Open Directory Project data," in *Proceedings of the International Conference on Intelligent Information Systems (IIPWM)*, 2004, pp. 369-377.

[54] S. Osiński and D. Weiss, "Carrot 2: Design of a Flexible and Efficient Web Information Retrieval Framework," in *Advances in Web Intelligence*, ed, 2005, pp. 439-444.

[55] S. Osiński, "Improving quality of search results clustering with approximate matrix factorizations," in *28th European Conference on IR Research (ECIR 2006)*, London, UK, 2006, pp. 167-178.

[56] X. Wei, L. Xin, and G. Yihong, "Document clustering based on non-negative matrix factorization," presented at the Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval, Toronto, Canada, 2003.

[57] Z. Zhong-Yuan and J. Zhang, "Survey on the Variations and Applications of Nonnegative Matrix Factorization," in *The Ninth International Symposium on Operations Research and Its Applications (ISORA'10)*, Chengdu-Jiuzhaigou, China, 2010, pp. 317–323.

[58] 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.

[59] 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.

[60] R. Navigli and G. Crisafulli, "Inducing word senses to improve web search result clustering," presented at the Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, Cambridge, Massachusetts, 2010.

[61] C. Carpineto and G. Romano, "Optimal meta search results clustering," presented at the Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval, Geneva, Switzerland, 2010.

[62] C. Cobos, J. Andrade, W. Constain, M. Mendoza, and E. León, "Web document clustering based on Global-Best Harmony Search, K-means, Frequent Term Sets and Bayesian Information Criterion," in *IEEE Congress on Evolutionary Computation (IEEE CEC)*, Barcelona, Spain, 2010, pp. 4637-4644.

[63] C. Cobos, M. Mendoza, and E. Leon, "A hyper-heuristic approach to design and tuning heuristic methods for web document clustering," in *2011 IEEE Congress on Evolutionary Computation (CEC)*, New Orleans, USA., 2011, pp. 1350-1358.

[64] U. Scaiella, P. Ferragina, A. Marino, and M. Ciaramita, "Topical clustering of search results," presented at the Proceedings of the fifth ACM international conference on Web search and data mining, Seattle, Washington, USA, 2012.

[65] R. S. Michalski, "LEARNABLE EVOLUTION MODEL: Evolutionary Processes Guided by Machine Learning," *Mach. Learn.*, vol. 38, pp. 9-40, 2000.

[66] C. Cobos, D. Estupiñán, and J. Pérez, "GHS + LEM: Global-best Harmony Search using learnable evolution models," *Applied Mathematics and Computation*, vol. 218, pp. 2558-2578, 2011.

[67] P. N. Suganthan, N. Hansen, J. J. Liang1, K. Deb, Y.-P. Chen, A. Auger, and S. Tiwari, "Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization," 2005.

[68] K. Tang, X. Yao, P. N. Suganthan, C. MacNish, Y. P. Chen, C. M. Chen, and Z. Yang, "Benchmark Functions for the CEC' 2008 Special Session and Competition on Large Scale Global Optimization," p. 18, 2007.

[69] K. Tang, X. Li, P. N. Suganthan, Z. Yang, and T. Weise, "Benchmark Functions for the CEC' 2010 Special Session and Competition on Large-Scale Global Optimization," 2009.

[70] M. Molga and C. Smutnicki, "Test functions for optimization needs," 2005.

[71] M. H. Badii, A. Guillen, L.A., E. Araiza, and J. Cerna, "Métodos No-Paramétricos de Uso Común " *Daena: International Journal of Good Conscience,* April 2012 2012.

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

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

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

[75] 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.

[76] E. W. Forgey, "Cluster analysis of multivariate data: efficiency versus interpretability of classifications," *Biometrics*, vol. 21, pp. 768-769, 1965.

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

[78] R. M. Aliguliyev, "Clustering of document collection - A weighting approach," *Expert Systems with Applications,* vol. In Press, Corrected Proof.

[79] M. Li and M. Qi, "MAPBOT: a Web based map information retrieval system," *Information and Software Technology*, vol. 45, pp. 691-698, 2003.

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

[81] 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, ed: Springer Berlin Heidelberg, 2012, pp. 179-188.

[82] Y. Xin-She and S. Deb, "Cuckoo Search via Lévy flights," in *Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*, 2009, pp. 210-214.

[83] J. Nocedal and S. Wright, "Numerical Optimization," *Springer Series in Operations Research,* p. 656, 28, April 2000.