Algoritmos para la Generación Automática de Resúmenes Extractivos de un Documento Basados en los Procedimientos de Saltos de Ranas Revueltas y Competencia de Liga de Fútbol

Julián Andrés Noguera Agredo
Juan David Yip Herrera

Trabajo de Grado para optar al título de Ingeniero de Sistemas

Director: Dr. Martha Eliana Mendoza Becerra
Co-Director: Dr. Carlos Alberto Cobos

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Sistemas
Grupo de I+D en Tecnologías de la Información (GTI) Línea Investigación: Gestión de la Información, Recuperación de la Información
Popayán
2018

Algoritmos para la Generación Automática de Resúmenes Extractivos de un Documento Basados en los Procedimientos de Saltos de Ranas Revueltas y Competencia de Liga de Fútbol

Julián Andrés Noguera Agredo
Juan David Yip Herrera

Director: Dr. Martha Eliana Mendoza Becerra
Co-Director: Dr. Carlos Alberto Cobos

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Sistemas Grupo de I+D en Tecnologías de la Información (GTI) Línea Investigación: Gestión de la Información, Recuperación de la Información

Popayán
2018

Agradecimientos

Agradecemos a nuestros padres por brindarnos la oportunidad de tener una excelente educación, por su guía y su apoyo incondicional a lo largo de este camino.

Martha Mendoza y Carlos Cobos, por compartir de su conocimiento, tiempo, apoyo y mucha de su paciencia con nosotros para realizar este proceso que permitió llevarnos a finalizar esta etapa, también gracias por enriquecer nuestra experiencia como futuros ingenieros. Así mismo, a todos los profesores que nos orientaron a lo largo de esta carrera, mil gracias.

Gracias a todos nuestros compañeros que estuvieron con nosotros en toda esta etapa tan maravillosa, por el apoyo que nos ofrecieron y por los grandes momentos que pasamos.

CONTENIDO

Lista de tablas

Lista de figuras

Presentación

Actualmente la información textual que se encuentra en la Web está aumentando de forma exponencial, por lo cual, cuando un usuario requiere información de un tema específico debe invertir mucho tiempo para encontrar los documentos que sean de utilidad. Bajo este contexto el área de investigación de generación automática de resúmenes extractivos aborda este problema buscando algoritmos que permitan obtener un resumen con la información más relevante en un documento.

En este documento, se plantean dos algoritmos que permiten la generación automática de resúmenes de un documento basados en las metaheurísticas, uno de ellos en Saltos de Ranas Revueltas adaptado con una memoria tabú explicita (SFLA-MT-SingleDocSum) y el otro en Competencia de Liga de Futbol adaptado con ascenso de la colina (SLC-ASC-SingleDocSum). A lo largo de este documento se explica el proceso de adaptación de estos algoritmos y las bases teóricas necesarias para ésta.

El capítulo 1 presenta el problema que se aborda en este proyecto, la importancia del desarrollo del mismo, los objetivos planteados para desarrollar una solución alternativa al problema planteado y los resultados obtenidos al concluir el desarrollo de esta investigación.

En el capítulo 2 se abordan los conceptos más relevantes en el área de investigación de generación de resúmenes, los métodos del estado del arte en esta área, las medidas de calidad de resúmenes automáticos aceptadas por la comunidad científica, la representación de documentos en el modelo espacio vectorial con las medidas asociadas para ponderación de términos y la medida de similitud de cosenos. También la descripción general de los algoritmos de Saltos de Ranas Revueltas (SFLA), Competencia de Liga de Futbol (SLC), búsqueda local de Ascenso de Colina y Memoria Tabú.

En el capítulo 3 se presenta el proceso que se llevó a cabo para la adaptación de los algoritmos SFLA-MT-SingleDocSum y SLC-ASC-SingleDocSum y la descripción de los tres ciclos planteados. En primer ciclo incluye la definición de la función objetivo y las adaptaciones de los algoritmos originales al problema de generación automática de resúmenes, junto a las configuraciones y afinamientos de parámetros. El segundo y tercer ciclo están enfocados en la inclusión de ascenso de la colina y una memoria tabú respectivamente, en los algoritmos SFLA y SLC junto con la configuración y afinamiento de parámetros de cada algoritmo.

En el capítulo 4 se presentan los algoritmos propuestos SFLA-MT-SingleDocSum y SLC-ASC-SingleDocSum, describiendo la representación de las soluciones y la función objetivo con los parámetros definidos para cada algoritmo. Además, los cambios realizados a los algoritmos originales para el problema de generación automática de resúmenes de un documento.

En el capítulo 5 se presentan las tareas de pre-procesamiento realizadas a los documentos originales, los conjuntos de datos (DUC2001, DUC2002) y métricas (ROUGE-1, ROUGE-2, ROUGE-SU) usados para la evaluación, y el afinamiento de los parámetros de cada algoritmo. Luego se muestran los resultados de evaluación de la calidad de los resúmenes

generados por los dos algoritmos propuestos con respecto a otros métodos del estado del arte, y por último se muestra una comparación de los algoritmos descritos en los tres ciclos del proceso de adaptación de los algoritmos.

En el capítulo 6 se presentan las conclusiones del proceso de adaptación de estos algoritmos al problema de generación automática de resúmenes de un documento y del resultado de las evaluaciones de estos algoritmos propuestos. Además, se exponen recomendaciones para trabajos de investigación similares y probables líneas de trabajo futuro que nacen de la presente investigación.

Al final se presentan las referencias bibliográficas usadas durante todo el proceso del proyecto.

Capítulo 1

1 INTRODUCCIÓN

1.1 PLANTEAMIENTO DEL PROBLEMA

Actualmente la información textual que se encuentra en la Web está aumentando de forma exponencial, por lo cual, cuando un usuario requiere información de un tema específico debe invertir mucho tiempo para encontrar los documentos que sean de su utilidad. En este contexto el área de investigación de generación automática de resúmenes extractivos aborda este problema desarrollando diferentes métodos que buscan seleccionar la información más relevante contenida en un documento. Se ha aplicado en algunas áreas como: motores de búsqueda [1], que obtienen un pequeño resumen del documento o página web; E-mail [2], para resumir las discusiones en un conjunto de correos electrónicos; E-learning [3], seleccionando la información más relevante de un texto.

En la generación automática de resúmenes existen dos técnicas teniendo en cuenta la forma en que se obtiene el resumen: extractiva o abstractiva [4]. Los sistemas extractivos forman el resumen extrayendo y seleccionando las oraciones más relevantes de un documento original, por lo general estas oraciones se incluyen en el resumen en el mismo orden como aparecen en el texto original. Por otro lado, los sistemas abstractivos pueden transformar las oraciones (fusionar oraciones o incluir nuevas) para mejorar la coherencia del resumen, utilizando herramientas de análisis lingüísticos que requieren mayor capacidad de procesamiento y memoria. Actualmente las técnicas extractivas son más usadas por su simplicidad y reducción en el tiempo de computación.

Estos sistemas extractivos han sido investigados ampliamente y en la literatura se encuentran diferentes métodos para la generación automática de este tipo de resúmenes para un documento, como: estadísticos [4]–[7]; aprendizaje de máquina [4], [8]–[19]; grafos [20]–[26]; conectividad de texto [4], [27]–[32]. Otros métodos son los algoritmos metaheurísticos como: programación genética [4], [33], [34]; algoritmos genéticos [4], [35]–[39]; optimización de enjambres de partículas [4], [40], [41]; algoritmos de búsqueda armónica [42]; algoritmos meméticos [43], [44] y evolución diferencial [45], [46]. También existen técnicas híbridas que combinan dos o más métodos para la generación automática de resúmenes [47]–[54].

Los métodos basados en metaheurísticas abordan la generación automática de resúmenes como un problema de optimización global, buscando seleccionar las mejores oraciones para formar el resumen. Utilizar este tipo de enfoque ha conseguido muy buenos resultados siendo un área prometedora de investigación, como es el caso del algoritmo metaheurístico Procedimiento de Búsqueda del Pescador (FSP, Fisherman Search Procedure) que fue adaptado al problema de generación automática de resúmenes extractivos de un solo documento [55], ocupando el primer puesto en el estado del arte. En este algoritmo se definió

una función objetivo normalizada con cinco características: posición, longitud, relación con el título, cobertura y cohesión.

Un aspecto importante en los algoritmos metaheurísticos es la definición de la función objetivo, la cual para el problema de generación automática de resúmenes no ha sido establecida debido a su complejidad (por lo que sólo se cuenta con aproximaciones), por lo tanto, para este proyecto se tomó como base la función definida por el algoritmo FSP, debido a los buenos resultados obtenidos.

De otro lado, el problema de la mochila viajera también ha sido abordado con algoritmos basados en metaheurísticas, como Saltos de Ranas Revueltas (SFLA, Shuffled Frog Leaping Algorithm) [56] y Competencia de Liga de Fútbol (SLC, Soccer League Competition) [57] obteniendo muy buenos resultados. El problema de la mochila tiene características similares al problema de generación automática de resúmenes, ya que en ambos casos se deben seleccionar los mejores elementos (oraciones con mayor relevancia en el caso de la generación de resúmenes, objetos con mayor valor y menor peso para la mochila) y tienen una restricción de capacidad (cantidad de oraciones y peso de la mochila).

Teniendo en cuenta lo anterior, en este proyecto se proponen dos algoritmos metaheurísticos para abordar el problema de generación automática de resúmenes de un documento, uno basado en SFLA con memoria tabú y otro en SLC con ascenso de la colina, obteniendo muy buenos resultados en la calidad de los resúmenes generados sobre los conjuntos de datos de DUC2001 y DUC2002 (Obtenidos de la Conferencia de Entendimiento del Documento), comparado con métodos del estado del arte. Las métricas usadas fueron las de ROUGE (aceptadas por la comunidad científica para medir calidad de los resúmenes), que miden la coincidencia de palabras entre resúmenes elaborados por humanos y el resumen generado automáticamente.

Por lo tanto, la pregunta de investigación que se planteó en este proyecto fue: ¿qué tipo de adaptaciones con respecto a los esquemas de inicialización, operadores evolutivos, generación del vecindario y características de la función objetivo, se deben hacer a los algoritmos SFLA y SLC para abordar el problema de generación automática de resúmenes extractivos de un solo documento?

1.2 APORTES

Con el presente proyecto se proporciona nuevo conocimiento en el área de generación automática de resúmenes de un solo documento basada en metaheurísticas, con la adaptación a este problema de los algoritmos SFLA con memoria tabú y SLC con ascenso de la colina, con relación a los esquemas de inicialización, los operadores evolutivos y la generación del vecindario. Además, conocimiento con respecto a las características usadas en la función objetivo que se tomó como base. Este conocimiento es de gran importancia para la comunidad de Recuperación de la Información y Sistemas Inteligentes. Este aporte se hace teniendo en cuenta que no se encontró ningún reporte de que los algoritmos SFLA y SLC hayan abordado el problema de generación automática de resúmenes de un solo documento a la fecha de finalización de este proyecto.

Además, las líneas de investigación de Gestión de la Información y Sistemas Inteligentes del Grupo de I+D en Tecnologías de la Información cuentan con una nueva solución informática basada en metaheurísticas que permite generar resúmenes automáticos de un solo documento con mayor calidad que otros métodos del estado del arte, permitiendo que se pueda seguir realizando investigación en esta área al interior del grupo.

1.3 OBJETIVOS

1.3.1 Objetivo general

Proponer los algoritmos saltos de ranas revueltas y competencia de liga de fútbol, adaptados al problema de generación automática de resúmenes extractivos de un solo documento, incluyendo búsqueda local con Ascenso de la Colina y/o una Memoria Tabú.

1.3.2 Objetivos específicos

- Definir las características de una función objetivo para la generación automática de resúmenes extractivos de un solo documento basados en la función objetivo planteada en [55].

- Adaptar los algoritmos de saltos de rana revueltas y competencia de liga de fútbol al problema de generación automática de resúmenes extractivos de un solo documento, incluyendo búsqueda local con Ascenso de la Colina y/o una Memoria Tabú.

- Evaluar la calidad de los resúmenes generados con los algoritmos propuestos utilizando documentos de noticias de la Conferencia de Entendimiento del Documento (DUC2001 y DUC2002) y medidas ROUGE[1], comparando los resultados obtenidos con los reportados por otros algoritmos del estado del arte.

1.4 RESULTADOS OBTENIDOS

- **Monografía del trabajo de grado**, en la que se presentan los conceptos teóricos necesarios para el desarrollo del proyecto, el proceso de afinamiento de la función objetivo y la adaptación de los algoritmos SFLA y SLC al problema de generación automática de resúmenes de un documento. También, la descripción de los algoritmos propuestos SFLA-MT-SingleDocSum y SLC-ASC-SingleDocSum y la evaluación de la calidad de los resúmenes obtenidos por estos algoritmos y su comparación con otros métodos del estado del arte. Finalmente, las conclusiones, recomendaciones y el trabajo futuro que el GTI espera desarrollar en base a esta investigación.

- **Código de los algoritmos propuestos**, junto con las especificaciones de su lógica y componentes.

- **Artículo**, que presenta la descripción de los algoritmos propuestos SFLA-MT-SingleDocSum y SLC-ASC-SingleDocSum, la representación de las soluciones, la configuración de sus parámetros, la función objetivo, y las modificaciones realizadas a los algoritmos SFLA y SLC originales junto a la memoria tabú explícita y ascenso de la colina, respectivamente, para adaptarlos al problema de generación automática de resúmenes de un solo documento. Además, se presentan los resultados obtenidos al

evaluar los resúmenes generados y compararlos con los resultados obtenidos por otros algoritmos del estado del arte. El artículo se enviará a una revista nacional/internacional para su evaluación.

Capítulo 2

2 CONTEXTO TEÓRICO Y ESTADO DEL ARTE

En este capítulo se presentan los conceptos más relevantes en el área de investigación de generación automática de resúmenes, los trabajos más destacados en esta área, las medidas de calidad de generación de resúmenes reconocidas por la comunidad científica, la representación de documentos en el modelo espacio vectorial, medidas asociadas para ponderación de términos y la medida de similitud de cosenos. También la descripción general de los algoritmos de Saltos de Ranas Revueltas, Competencia de Liga de Futbol, búsqueda local de Ascenso de la Colina y Memoria Tabú.

2.1 GENERACIÓN AUTOMÁTICA DE RESÚMENES DE TEXTO

La generación automática de resúmenes de textos es una tarea del procesamiento de lenguaje natural, que tiene por objetivo crear una versión breve y corta de uno o más documentos de texto, que represente la información más esencial de los documentos [4].

Existen diferentes formas de clasificar los resúmenes, teniendo en cuenta [4]:

- Número de documentos: un documento, múltiples documentos.
- Lenguaje del documento: mono lenguaje, multilenguaje.
- Tipo de documento: científico, noticia, blog, entre otros.
- Propósito: (i) Indicativo, brinda una corta descripción de la idea principal del documento. (ii) Informativo, busca reemplazar el documento original generando una versión más corta con el contenido más relevante. (iii) Crítico, reúne la perspectiva del autor del resumen.
- Audiencia a la que va dirigido: (i) Genérico, da igual valor a los temas principales del documento y está dirigido a público en general. (ii) Basados en consultas, el resumen contiene la información más relevante con respecto a una pregunta específica. (iii) Enfocados en el usuario, elaborados teniendo en cuenta las características del usuario.
- Profundidad de procesamiento: (i) Estrategias poco profundas o superficiales, emplean características superficiales como: frecuencia de términos, palabras clave, entre otras. (ii) Estrategias profundas, usan métodos avanzados de procesamiento de lenguaje natural para identificar las entidades presentes en el texto y sus relaciones.
- Forma como el resumen es extraído: (i) Técnicas abstractivas, genera un resumen a través del análisis de la semántica y la gramática, usando conocimientos lingüísticos. (ii) Técnicas extractivas, selecciona palabras, oraciones o párrafos más importantes del documento original para ser parte del resumen.

2.1.1 Métodos de generación automática de resúmenes de un solo documento

En la literatura se encuentran diferentes métodos de generación automática de resúmenes de un solo documento, a continuación, se presentan algunos de los más representativos:

2.1.1.1 Estadísticos

Las primeras investigaciones en el área de la generación automática de resúmenes extractivos utilizaron características estadísticas para calificar una oración e identificar las palabras clave u oraciones más relevantes, algunas características son: posición en el documento, número de palabras clave, frecuencia de palabras u oraciones, relación con el título, entre otras [5]. En general estos métodos efectúan un pre-procesamiento de oraciones con el fin de segmentar y filtrar las oraciones del documento, luego calcula el puntaje de cada oración de acuerdo a las características estadísticas y al final las oraciones con los puntajes más altos se incluirán en el resumen. La efectividad de algunas características estadísticas dependen del formato y estilo de escritura del documento, sin embargo estos métodos siguen siendo usados debido a su facilidad de implementación y menor complejidad computacional.

En [6] aplican un sistema que en tres niveles selecciona las mejores oraciones que formaran el resumen, en cada nivel evalúa y filtra las oraciones de acuerdo al puntaje obtenido según la característica respectiva al nivel en este orden: frecuencia de término y frecuencia inversa de la oración, presencia de nombres de entidades y nombres propios.

2.1.1.2 Aprendizaje de máquina

Es un enfoque supervisado que necesita realizar un entrenamiento previo con un conjunto de datos para calcular las probabilidades o pesos optimizados de las características apoyándose en diversos clasificadores: bayesiano [8], [12], [17], redes neuronales artificiales [14], [17], modelos ocultos de Markov [13], [58], campos aleatorios condicionales [15], árboles de decisión o lógica difusa [9], [16]. Cada una de las palabras clave u oraciones seleccionadas del documento original se ordenan de acuerdo a la probabilidad dada por la evaluación de las características, de mayor a menor valor, para ser incluidas en el resumen [4], [10], [11], [18]. Para este tipo de métodos es indispensable contar con datos de entrenamiento y además estos métodos dependen del lenguaje en que están escritos los documentos de entrenamiento.

2.1.1.3 Grafos

El documento es representado mediante un grafo no dirigido conformado por nodos que a su vez representan las oraciones. Si dos oraciones son semejantes, estas se conectan con un arco que tiene un peso relacionado a la similitud entre ellas. De esta forma las conexiones entre todas las oraciones del documento son modeladas en un grafo que permite ser iterado hasta un criterio específico, para luego ordenar y elegir las oraciones con mayor puntaje en el grafo [20]. Hay investigaciones donde las diferentes palabras que componen una oración aportan importancia al momento de evaluarla, algo similar a las palabras claves que se destacan en el artículo [26]. En otras, además del peso en el arco por similitud se adicionan otros tres arcos de pesos (similitud semántica, resolución de la correferencia y relaciones del Discurso) [23]. A diferencia de los estudios nombrados anteriormente en [24], las secuencias

frecuentes máximas (MFS, Maximal Frequent Secuences) son representadas como nodos. Estas MFS son los n-gramas frecuentes que no pertenecen a ningún otro n-grama frecuente, que ofrecen la información más relevante de un documento. En [21] se aborda de manera unificada la generación automática de resúmenes de un documento y múltiples documentos con un algoritmo basado en grafos, en el resumen se incluyen los nodos u oraciones que tengan los mejores puntajes. Estos métodos no supervisados son independientes del lenguaje y mejoran la cohesión en los resúmenes generados, pero por otro lado la complejidad computacional aumenta a medida que el número de nodos y arcos del grafo se incrementa.

2.1.1.4 Conectividad de texto

Buscan identificar las relaciones entre conceptos del documento, utilizando estrategias como las cadenas léxicas y las estructuras retóricas. La extracción de palabras candidatas del documento en los métodos basados en cadenas léxicas se define utilizando un tesauro como WordNet, posteriormente se inspecciona palabra por palabra analizando si se pueden agregar en una cadena léxica existente o por el contrario generar una nueva, finalmente se incluyen en el resumen las oraciones más relevantes extraídas de las cadenas léxicas más fuertes encontradas. Por otra parte, los métodos basados en estructuras retóricas realizan la extracción de segmentos retóricos del documento, permitiendo generar una estructura de árbol donde las unidades de texto conforman los nodos, que pueden ser catalogados como satélite o núcleo, basándose en el nivel de relevancia para el discurso, posteriormente, de acuerdo a las métricas definidas por el algoritmo se asigna el puntaje de cada estructura retórica y se eligen las estructuras con mayor puntaje para ser incluidas en el resumen [4]. Si bien el contexto y la cohesión se mantienen en este enfoque, a medida que aumenta la cantidad de texto su precisión se reduce, para superar esta limitación en [31] se presenta un modelo híbrido que combina estructuras retóricas y el modelo de espacio vectorial. Estos métodos presentan desventajas como el uso de técnicas complejas y la dependencia del lenguaje para el procesamiento del texto [4].

2.1.1.5 Metaheurísticas

Los métodos basados en metaheurísticas se han utilizado de dos formas: (i) calcular los pesos de las características incluidas en una ecuación matemática que evalúa la importancia de cada oración respecto al documento original [33], [34], [36], [40], en este caso las soluciones candidatas representan la combinación de pesos de estas características, al final de la evolución el algoritmo retorna la mejor combinación encontrada, luego el resumen se conforma por aquellas oraciones con mayor puntuación utilizando esta combinación. (ii) generar automáticamente el resumen [35], [37], [38], [41], [42], [44], [45], [59], abordando el problema de generación de resúmenes como un problema de optimización, en este caso, por lo general las soluciones candidatas representan las oraciones que formarán parte del resumen, cuando la ejecución del algoritmo termina, el resumen se conforma con las oraciones presentes en la mejor solución encontrada. A continuación, se describen varias investigaciones que utilizan esta segunda forma.

En [55] se propone un algoritmo memético basado en el procedimiento de búsqueda del pescador y una memoria tabú llamado FSP-MT (Fisherman Search Procedure – Memoria Tabú), cuyas soluciones candidatas (puntos de captura) representan de forma binaria la presencia o ausencia de las oraciones en el resumen y utiliza una memoria tabú para cada

punto de captura, la cual permite almacenar las soluciones visitadas recientemente. Para cada punto de captura se generan vecinos (puntos de red) mediante un procedimiento que deshabilita la peor oración utilizando el criterio de posición-cobertura y habilita otra oración de forma aleatoria, luego se selecciona la solución con mayor valor de aptitud (óptimo local) y si esta nueva solución es mejor que el punto de captura éste es reemplazado por el óptimo local. Este procedimiento se repite un cierto número de iteraciones establecida. Al final se selecciona el punto de captura con mayor valor de aptitud (óptimo global) y las oraciones presentes en esta solución forman parte del resumen. La función objetivo que utiliza está compuesta por las características de: posición de la oración en el documento, longitud de la oración, relación con el título, cohesión y cobertura.

En [42] emplean un algoritmo de búsqueda armónica y utilizan una función objetivo ponderada que evalúa las oraciones basado en las características: similitud entre las oraciones del resumen, similitud de las oraciones con el título del documento y grado de relación entre las oraciones consecutivas. Las oraciones de la solución con mayor aptitud se incluirán en el resumen.

En [34] se plantea un modelo de programación genética que permite crear una función de clasificación de oraciones que a través de una etapa de entrenamiento, optimiza y define el tipo de relación entre sus características (ubicación de la oración en el párrafo y en el documento, longitud de la oración y frecuencia de palabras) usando operadores básicos como suma, resta, multiplicación y división (+, -, *, /). Luego durante la etapa de prueba, se aplica la función ya optimizada al conjunto de oraciones o al documento a evaluar.

En [51] se propone un modelo que combina un procedimiento para filtrar las oraciones semejantes y escoger las que tengan mayor diversidad, luego a través de un conjunto de datos de entrenamiento optimiza los pesos de las características que determinan el modo de evaluar cada oración del documento mediante PSO y finalmente aplica lógica difusa para mejorar la precisión. El método se presenta de dos maneras: en la primera la diversidad no se impone en el modelo y en la segunda la diversidad predomina en el método.

2.1.1.6 Híbridos

En [47] se plantea un modelo que realiza análisis semántico latente (LSA, Latent Semantic Analysis) representando el documento como una matriz de palabras por oración. Luego construyen un mapa de relaciones de texto que se basa en la representación de la oración semántica derivada de la matriz semántica. Por último, la selección de oraciones establece un camino óptimo de acuerdo con el mapa y selecciona las mejores oraciones para generar el resumen.

Babar y Patil [53] proponen utilizar la lógica difusa donde se asigna una puntuación a las oraciones para luego seleccionar las mejores y generar un primer resumen. Luego utilizan LSA para comparar la similitud semántica entre oraciones del documento original y se genera un segundo resumen. De los dos resúmenes generados, se extraen las oraciones comunes y las que son muy diferentes para ser parte del resumen final.

En [54] proponen usar: Autómatas Celulares de Aprendizaje (CLA, Cellular Learning Automata) para calcular la similitud de las oraciones, Optimización por Enjambre de Partículas y Algoritmo Genético (PSO-GA, Particle Swarm Optimization – Genetic Algorithm)

para establecer los pesos adecuados a las características según su importancia y lógica difusa para puntuar las oraciones.

2.1.2 Métodos de evaluación de la calidad de los resúmenes

En el desarrollo de cualquier sistema o método es muy importante la evaluación, en el caso de los sistemas de generación automática de resúmenes es una tarea compleja ya que la definición de lo que es un buen resumen varía entre las personas y las irregularidades de los lenguajes (humanos). Los métodos para la evaluación de sistemas de generación automática de resúmenes se dividen en dos grandes categorías que son intrínsecas y extrínsecas [60].

2.1.2.1 Evaluación intrínseca

La mayoría de los esquemas de evaluación de resúmenes son intrínsecos, este evalúa el sistema sin tener presente a su audiencia objetivo, y usualmente utilizan un conjunto de resúmenes ideales[1] con los cuales se compara, que pueden ser generados por humanos expertos o sistemas de referencia. Sin embargo, debido a la subjetividad que acompaña la creación de un resumen, se utiliza un puntaje promedio de varios resúmenes ideales generados por humanos por cada documento original [60].

La evaluación intrínseca se enfoca principalmente en la coherencia y la capacidad informativa de los resúmenes, midiendo de ese modo únicamente la calidad de salida [60]. De esta forma, surgen algunas medidas utilizadas comúnmente en este tipo de evaluación, como la *precisión* y el *recuerdo* [61]. La precisión se establece como el número de oraciones en el resumen generado que están contenidas en el resumen de referencia. Análogamente, el recuerdo es la proporción del número de oraciones comunes entre el resumen generado y el de referencia, sobre el número total de oraciones del resumen generado. Precisión y recuerdo son medidas estándar para la recuperación de información y, algunas veces se unen para formar la denominada medida F.

Actualmente el paquete de medidas de ROUGE [62] se utiliza como una forma automatizada de evaluación de resúmenes, que se basa en la cantidad de unidades comunes entre un resumen generado y un resumen ideal.

2.1.2.2 Evaluación extrínseca

Este tipo de evaluación está pensada en el usuario final. Por lo tanto, evalúa la calidad del resumen con base en la eficacia y la aceptabilidad en alguna tarea como: comprensión de lectura, seguimiento de instrucciones, recopilación de información, entre otras. Además, si el resumen contiene algún tipo de instrucciones, es posible medir la calidad del mismo a través del seguimiento de las instrucciones y el resultado del mismo. Otras tareas medibles son la recopilación de información de un conjunto de documentos, el tiempo y esfuerzo necesario para enviar a editar el resumen generado por una máquina con un propósito específico, o el impacto del sistema de generación de resúmenes en un sistema del que

[1] Este tipo de conjuntos suele conocerse como gold-standard corpus, y usualmente son vistos como modelos de excelencia que representan el límite más alto al que razonablemente se puede llegar por medios automáticos. Dentro de éste trabajo, este tipo de resúmenes serán mencionados como resúmenes ideales, resúmenes modelo o resúmenes de referencia

forma parte, por ejemplo relevancia retroalimentación (ampliación de consultas) en un motor de búsqueda o un sistema de pregunta-respuesta [60].

2.1.2.3 Evaluación ROUGE

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) es un paquete de software que determina la calidad de un resumen generado comparado con resúmenes ideales creados por humanos a través de un conjunto de medidas. Las medidas cuentan el número de unidades superpuestas, tales como pares de palabras y n-gramas de palabras, entre el resumen generado y los resúmenes ideales [62], llevando a cabo la evaluación de coincidencias entre los resúmenes.

El paquete de evaluación de ROUGE incluye varias medidas como ROUGE-N, ROUGE-L, ROUGE-W, ROUGE-S y ROUGE-SU. Las más usadas por la comunidad científica son ROUGE-1, ROUGE-2 y ROUGE-SU4.

- **ROUGE-N**

ROUGE-N es una medida basada en el recuerdo de n-gramas entre un resumen generado y un conjunto de resúmenes de referencia. En la Ecuación (2.1) se presenta el cálculo de esta medida.

$$\mathrm{ROUGE-N} = \frac{\sum_{S\in\{ResúmenesDeReferencia\}} \sum_{grama_n \in S} Conteo_{Coincidencias}(grama_n)}{\sum_{S\in\{ResúmenesDeReferencia\}} \sum_{grama_n \in S} Conteo(grama_n)} \qquad (2.1)$$

Donde n representa la longitud del n-grama ($grama_n$) y $Conteo_{Coincidencias}(grama_n)$ es el máximo número de n-gramas coincidentes entre un resumen candidato y un conjunto de resúmenes de referencia. El denominador de esta fórmula corresponde a la suma de la cantidad de n-gramas en el resumen de referencia, de ahí que su valor aumente a medida que el número de resúmenes ideales se incrementa. De esta forma, obtendrá un mejor valor el resumen generado que comparta palabras con más de un resumen de referencia para la medida ROUGE-N.

- **ROUGE-S**

ROUGE-S calcula la superposición de saltos de bigramas (bigramas-skip) entre un resumen candidato y un conjunto de resúmenes de referencia. Un bigrama-skip se refiere a un par de palabras, en el orden en que están en la oración, permitiendo saltos arbitrarios. Dadas una oración de referencia X, longitud m, y una oración candidata Y, longitud n, el cálculo de la medida-F basada en bigramas-skip corresponde al cálculo de ROUGES como se indica en las Ecuaciones (2.2), (2.3), (2.4).

$$R_{skip1} = \frac{SKIP2(X,Y)}{C(m,2)} \qquad (2.2)$$

$$P_{skip2} = \frac{SKIP2(X,Y)}{C(n,2)} \qquad (2.3)$$

$$F_{skip2} = \frac{(1+\beta^2)R_{skip2}\,P_{skip2}}{R_{skip2}+\beta^2 P_{skip2}} \qquad (2.4)$$

Donde $SKIP2(X,Y)$ es la cantidad de bigramas-skip que coinciden entre X e Y, β controla la importancia relativa de P_{skip2} y R_{skip2}, y C es la función de combinación que calcula la cantidad de bigramas-skip presentes en una oración[2]. Considerando el siguiente ejemplo:

S_1: police killed the gunman
S_2: the gunman police killed

Se deduce que hay 6 bigramas-skip[3] en cada oración. Para S_1 los bigramas-skip son {"police killed", "police the", "police gunman", "killed the", "killed gunman", "the gunman"} y para S_2 los bigramas-skip son {"the gunman", "the police", "the killed", "gunman police", "gunman killed", "police killed"}. Entonces existen dos bigramas-skip de S_2 que coinciden con S_1 y son {"police killed", "the gunman"}. De esta forma, P_{skip2} y R_{skip2} entre S_1 y S_2 son igual a 0.3333, así ROUGE-S se calcula como 0.3333.

- **ROUGE-SU**

Un problema que tiene ROUGE-S es que si una oración candidata no tiene ningún par de palabras coincidentes con otro par en las oraciones de referencia esta no tendrá ningún valor. ROUGE-SU evita este problema tomando como punto de partida ROUGE-S, pero incluyendo el manejo de unigramas como conteo de unidades. De esta forma, ROUGE-SU adiciona un indicador al inicio de las oraciones candidatas y de referencia que no tengan ningún par de palabras coincidentes.

2.1.2.4 Colección de documentos de evaluación

La Conferencia de Entendimiento del Documento (DUC, Document Understanding Conference), establece resúmenes "ideales" creados por humanos expertos sobre un conjunto de documentos originales, con el fin de ofrecer a la comunidad académica y científica conjuntos de datos que permitan evaluar y comparar los resultados obtenidos por sistemas de generación automática de resúmenes. La comunidad académico-científica tiene una gran aceptación de DUC, ya que su conjunto de documentos es constantemente utilizado como conjunto de referencia por diversos estudios [4], [44], [46].

2.2 REPRESENTACIÓN DE LOS DOCUMENTOS

A continuación, se presenta el modelo de representación de oraciones del documento en el espacio vectorial, las técnicas de ponderación de términos en una oración y la medida de similitud de coseno.

2.2.1 Modelo de espacio vectorial

El modelo de espacio vectorial ha sido usado desde inicios de los 70 [63] en los sistemas de recuperación de información para representar oraciones como vectores, en tareas como: agrupamiento de documentos u oraciones, clasificación de documentos u oraciones, etc. Este modelo también ha sido utilizado ampliamente en la generación automática de

[2] La fórmula general para calcular las combinaciones que se pueden obtener con n elementos, tomados de r en r, es C(n, r) = (n!/r!*(n-r)!)
[3] C (4,2) = (4!/2!*2!) = 6

resúmenes, en este caso, representando las oraciones de un documento por medio de un vector de pesos de los términos que la componen (Ver Figura 2-1). De esta manera, si un término pertenece a una oración, obtiene un valor de acuerdo a su relevancia dentro de la oración dependiendo de la técnica de ponderación de términos utilizada [64].

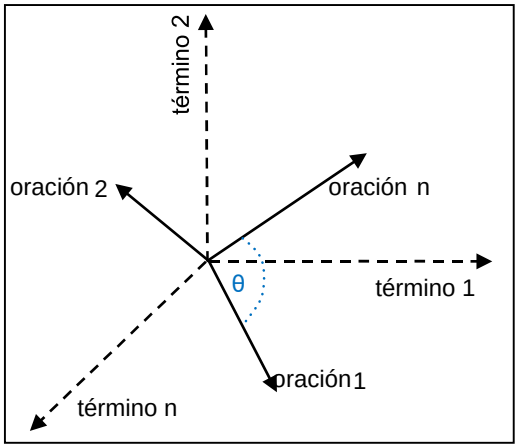

Figura 2-1 Representación Modelo Espacio Vectorial

Las coordenadas determinadas por los términos de la oración lo sitúan dentro del espacio vectorial. Basándose en este modelo, dada una oración S_i de n términos, su representación vectorial corresponde a $S_i = (w_{i1}, w_{i2}, w_{i3}, \dots, w_{in})$, donde w_{ij} es el peso, relevancia o importancia del término j en la oración S_i.

Con esta representación se obtiene la matriz de términos por oración (ver Figura 2-2), donde las columnas representan cada uno de los términos existentes en cualquiera de las oraciones (un peso puede ser cero) y las filas representan las oraciones.

$$Matriz_{oración,término} = \begin{Bmatrix} w_{1,1} & \dots, & w_{1,n} \\ \dots, & \dots, & \dots, \\ w_{m,1} & \dots, & w_{m,n} \end{Bmatrix}$$

Figura 2-2 Matriz de Términos Por Oración

En la matriz, se observa en un espacio multidimensional los vectores de pesos como proyecciones, donde la dimensión está dada por el número de oraciones m y el número de términos n. De esta forma se puede medir la similitud léxica entre las oraciones, a través del cálculo de alguna medida de similitud. Por ejemplo, si se quiere medir la similitud entre dos oraciones, se pueden representar con los siguientes vectores:

$$\overrightarrow{S_i} = (w_{i,1}, \quad w_{i,2}, \dots, w_{i,n})$$

$$\overrightarrow{S_j} = (w_{j,1}, \quad w_{j,2}, \dots, w_{j,n})$$

Figura 2-3 Representación de oraciones como vectores

En esta notación $\overrightarrow{S_i}$ y $\overrightarrow{S_j}$ son los vectores que representan las dos oraciones, respectivamente. De esta manera se puede calcular la similitud por medio de la medida Similitud Coseno (ver sección 2.2.3). Esta técnica es usada en la mayoría de estudios del estado del arte [21], [42], [51], [65].

2.2.2 Técnicas de ponderación de términos en una oración

En esta sección, se presentan algunas de las técnicas más utilizadas en la generación automática de resúmenes para la ponderación de términos en una oración.

2.2.2.1 Booleana

Este esquema binario da la misma importancia a cada término que está en la oración. Puede ser utilizado cuando el número de apariciones del término no es considerado importante. El peso $w_{ij} \in \{0,1\}$ indica la ausencia o presencia del término j dentro de la oración i. Se define como muestra la Ecuación (2.5), [66].

$$w_{ij} = \begin{cases} 1, & si\ el\ término\ j\ está\ en\ la\ oración\ i \\ 0, & en\ caso\ contrario \end{cases} \quad (2.5)$$

2.2.2.2 Frecuencia del término

Este esquema cuenta cuántas veces aparece el término en una oración. Entre mayor sea la frecuencia del término (TF, Term Frequent) en la oración, mayor probabilidad de que el término sea importante. Como se observa en la Ecuación (2.6), en este esquema la importancia de un término se basa en el número de veces que aparezca en la oración.

$$w_{ij} = f_{ij} \quad (2.6)$$

Donde f_{ij} es la frecuencia del término j en la oración i.

Existen términos muy comunes que pueden aparecer en cualquier parte de la oración lo que limita utilizar este método, sin que por ello, contenga información importante para caracterizarlo o diferenciarlo. Este tipo de términos tendrían una alta importancia incluso cuando son menos representativos que otros. Así mismo, los términos pertenecientes a oraciones con mayor longitud tendrían mayor frecuencia que aquellos presentes en oraciones más cortas [67]. De esta manera, el cálculo de frecuencia más frecuentemente utilizado, es el que se observa en la Ecuación (2.7).

$$w_{ij} = \frac{f_{ij}}{MáxFreq_i} \quad (2.7)$$

Donde $MáxFreq_i$ indica la cantidad de ocurrencias del término más frecuente en la oración i.

2.2.2.3 Frecuencia inversa de un término

Es un mecanismo que se utiliza para disminuir el efecto de los términos demasiado frecuentes en una oración, la idea es reducir el peso TF de un término con un factor que aumenta con su frecuencia de aparición. La frecuencia inversa de documento (IDF, Inverse

document frequency), hace referencia a la frecuencia de un término dentro de un conjunto de oraciones [68]. La Ecuación (2.8) expresa esta definición.

$$w_{ij} = \log \frac{N}{n_j} \tag{2.8}$$

Donde N es la cantidad de oraciones del documento y n_j es la cantidad de oraciones donde aparece el término j.

2.2.2.4 Frecuencia relativa de un término

Este esquema combina la definición de los métodos descritos en las secciones 2.2.2.2 y 2.2.2.3, para generar un peso compuesto [68]. El ponderado TF-IDF asigna al término i un peso en la oración j, como se muestra en la Ecuación (2.9).

$$w_{ij} = TF_{ij} \times IDF_j \tag{2.9}$$

Donde TF_{ij} es la frecuencia del término j en la oración i, IDF_j es la frecuencia inversa del término j, como se presentó en las Ecuaciones (2.7) y (2.8), respectivamente [68]. Reemplazando se tiene la Ecuación (2.10).

$$w_{ij} = \frac{f_{ij}}{MáxFreq_i} \times \log \frac{N}{n_j} \tag{2.10}$$

De esta manera, TF-IDF asigna al término j un peso en la oración i que es:

- Más grande cuando j está más presente dentro de un pequeño número de oraciones.
- Pequeño cuando el término aparece pocas veces en una oración, o aparece en muchas oraciones (se toma como una señal de baja relevancia).
- Más pequeño cuando el término aparece en casi todas las oraciones del documento.

2.2.3 Medida de similitud: Similitud de Coseno

En la generación automática de resúmenes, la medida de similitud de coseno se utiliza de forma estándar para calcular la similitud entre las representaciones vectoriales de oraciones. Esta similitud evalúa el valor del coseno del ángulo comprendido entre dos vectores a comparar, teniendo en cuenta la propiedad del coseno que retorna un valor de uno cuando los vectores son iguales y cero cuando son diferentes [64]. De esta forma, entre más pequeño sea el ángulo, mayor será la similitud y su valor tiende a uno. De esta manera, dadas las dos oraciones representadas por los vectores $\overrightarrow{S_i}$ y $\overrightarrow{S_j}$ (ver Figura 2-3), la medida de similitud de cosenos será calculada como se observa en la Ecuación (2.11).

$$sim_{cos}(\overrightarrow{S_i}, \overrightarrow{S_j}) = \frac{\sum_{k=1}^{m} w_{ik} w_{jk}}{\sqrt{\sum_{k=1}^{m} w_{ik}^2 * \sum_{k=1}^{m} w_{jk}^2}} \tag{2.11}$$

Donde m es el número total de términos en la oración, w_{ik} se refiere al peso del término k en la oración $\overrightarrow{S_i}$ y w_{jk} es el peso del término k en la oración $\overrightarrow{S_j}$.

2.3 ALGORITMOS BASE

En esta sección se presentan los algoritmos metaheurísticos base que se adaptaron para el problema de generación automática de resúmenes de texto de un solo documento.

2.3.1 Saltos de Ranas Revueltas (SFLA)

SFLA es un algoritmo metaheurístico de optimización basado en la imitación y el modelamiento del comportamiento de un grupo de ranas en búsqueda de refugio, alimento o pareja dentro de un estanque [56]. Inicialmente SFLA inicializa un conjunto de soluciones candidatas de manera aleatoria (población de ranas inicial) para todo el espacio de búsqueda y luego son ordenadas descendientemente según su valor de aptitud. Cada rana es representada por un vector de posición x_i dentro del espacio de búsqueda, además el algoritmo almacena la mejor solución encontrada (G_{best}). En la Ecuación (2.12) se muestra el conjunto de vectores de posición de las ranas.

$$X = [x_1, x_2, \dots, x_i, \dots, x_P], i = 1,2, \dots, P \qquad (2.12)$$

Donde x_i es la *i*-ésima rana del conjunto X y P es el número total de ranas en el estanque.

Después de tener las ranas ya ordenadas, la población es barajada en los m memeplex. El procedimiento de baraja consiste en que la primera rana va al primer memeplex, la segunda rana al segundo memeplex, la rana m va al m-ésimo memeplex, la rana $m+1$ va al primer memeplex y así sucesivamente. La Figura 2-4 muestra el procedimiento de baraja en los memeplex.

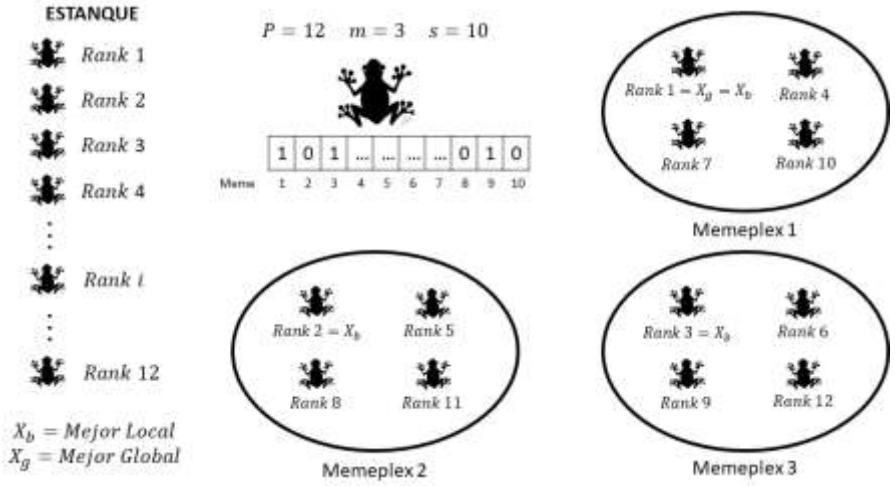

Figura 2-4 Procedimiento de baraja en los memeplex

Dentro de cada memeplex se realiza un procedimiento denominado ***Búsqueda Local*** durante una cantidad de iteraciones determinadas, donde inicialmente se identifica la mejor y la peor rana del memeplex identificadas como x_b y x_w respectivamente, además de la rana con mejor valor de aptitud del estanque que se identifica como x_g. Luego, en cada

iteración la peor rana del memeplex (x_w) realiza un salto guiado, inicialmente hacia la rana con mejor valor de aptitud (x_b) con la finalidad de modificarse y mejorar su valor de aptitud. La posición de x_w se ajusta como expresa la Ecuación (2.13).

$$S_j = rand() * (X_{bj} - X_{wj}) \qquad (2.13)$$

Donde S_i es el cambio en la *j*-ésima posición de la rana candidata al restar el valor de cada meme entre la mejor y la peor rana multiplicado por un valor aleatorio entre [0,1]. La nueva posición de la rana es dada por la Ecuación (2.14).

$$x_{wj}(nuevo) = x_{wj} + S_j \qquad (2.14)$$

Si este procedimiento de salto produce una mejor solución, esta reemplaza la peor rana. De lo contrario, la peor rana (x_w) realiza nuevamente un salto guiado pero ahora hacia la mejor rana del estanque (x_g) usando nuevamente las Ecuaciones (2.15)(2.13) y (2.14) pero reemplazando x_b por x_g. Si después del cambio tampoco hay mejora, entonces x_w hará un salto aleatorio que consiste en generar una rana aleatoriamente para reemplazarla.

El resultado ideal de los saltos en la búsqueda local es que la mayor cantidad de ranas converjan en la mejor solución. La Figura 2-5 muestra diferentes instantes del algoritmo haciendo visible el cambio de posición de las ranas después de la búsqueda local, de reagruparse y ser barajadas en los memeplex. En el *cuadro a* se observa el estanque con la población inicial generada aleatoriamente, el *cuadro b* muestra las ranas del cuadro anterior ya barajadas en los diferentes memeplex según su valor de aptitud diferenciadas con figuras geométricas (Cuadrado un memeplex, Triángulos otro memeplex, etc.). En el *cuadro c* se reflejan los cambios por los saltos en cada memeplex de la peor rana en cada iteración, y finalmente en el *cuadro d* se muestra la convergencia de varias ranas en un punto de espacio. Este procedimiento se repite hasta completar el número de iteraciones definido o hasta cumplir con el criterio de parada.

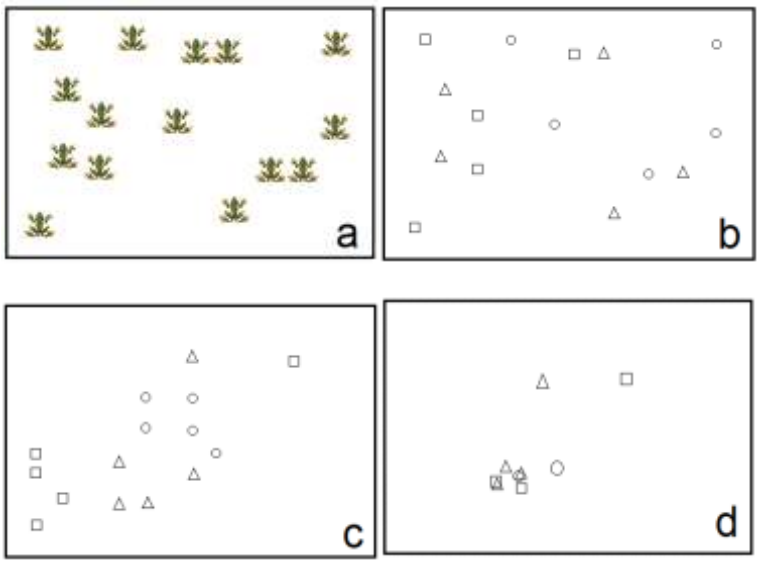

Figura 2-5 Grafico de SFLA

El esquema del algoritmo SFLA se presenta en la Figura 2-6, donde inicialmente se crea cada rana ($Inicializar\ x_i$), se evalúa ($Evaluar\ x_i$) y se verifica si es mejor que la mejor solución (G_{best}) para actualizarla si es el caso. Este procedimiento se repite hasta generar las P ranas de la población.

Dentro de cada iteración del algoritmo ($itMax$), se realiza un rankeo de las ranas que consiste en ordenarlas descendientemente según su valor de aptitud ($RankearRanas$), luego son barajadas en los memeplex ($BarajarRanas$) donde se realizará la ***búsqueda local*** en cada uno de ellos.

Esta búsqueda local se hace durante una cantidad de iteraciones (it) en cada memeplex, haciendo lo siguiente: (i) identificar la mejor local como x_b y la peor local como x_w, también la mejor solución global como x_g; (ii) hacer que x_w se modifique con los saltos y si al calcular su valor de aptitud mejora, entonces reemplazarla. Esta modificación se hace primero saltando hacia x_b ($Saltar\ x_b$), si no mejora entonces saltará hacia x_g ($Saltar\ x_g$). Finalmente, si con los saltos guiados no se reemplazó, entonces x_w hace un salto aleatorio ($Inicializar$) que consiste en inicializar la rana de manera aleatoria nuevamente.

Al terminar la búsqueda local en los memeplex, las ranas en los memeplex, se reagrupan ($ReagruparRanas$) y se ordenan nuevamente ($RankearRanas$) para finalmente aplicarles un método de mutación ($MutarRanas$) que consiste en alterar 2 genes.

```
INICIO SFLA
    HACER
        Inicializar x_i                 /*Crear rana i */
        Evaluar x_i                     /*Evaluar la calidad de la rana i */
    HASTA Crear P Ranas
    PARA k = 0 HASTA itMax              /*Número de iteraciones*/
        RankearRanas                    /*Ordenar las ranas descendientemente*/
        BarajarRanas                    /*Repartir las ranas en los memeplex*/
        BusquedaLocal                   /*Generar nuevas soluciones con los saltos*/
        ReagruparRanas                  /*Reunir ranas desde los memeplex*/
        MutarRanas                      /*Aplicar mutación a la población*/
    FIN PARA
    Retornar G_best
FIN SFLA
```

Figura 2-6 Esquema general SFLA

2.3.2 Competencia de Liga de Futbol (SLC)

SLC es una metaheurística inspirada en el comportamiento que se lleva a cabo en una liga profesional de futbol [57]. Inicialmente el algoritmo SLC genera los jugadores (soluciones candidatas iniciales), que pueden ser titulares o suplentes, que jugarán en la liga (espacio de búsqueda). Cada jugador es representado por un vector de posición x_i (localizado en el espacio de búsqueda). En la Ecuación (2.15) se muestra el conjunto de vectores de posición de los jugadores.

$$X = [x_1, x_2, \dots, x_i, \dots, x_N], i = 1,2, \dots, P \qquad (2.15)$$

Donde x_i es el *i*-ésimo jugador del conjunto X y P es el número de jugadores. Luego, los jugadores se asignan a los equipos basándose en su desempeño y son organizados de

forma descendente con el fin de agrupar a los mejores jugadores en un equipo. La asignación de jugadores se describe mediante la Ecuación (2.16).

$$EQUIPO = \begin{bmatrix} FP_1 \\ FP_2 \\ . \\ FP_{nF} \\ S_1 \\ S_2 \\ . \\ S_{nS} \end{bmatrix} = \begin{bmatrix} xF_1^1 & xF_2^1 & . & . & . & . & . & xF_K^1 \\ xF_1^2 & xF_2^2 & . & . & . & . & . & xF_K^2 \\ . & . & . & . & . & . & . & . \\ xF_1^{nF} & . & . & . & . & . & . & xF_K^{nF} \\ xR_1^1 & xR_2^1 & . & . & . & . & . & xR_K^1 \\ xR_1^2 & xR_2^2 & . & . & . & . & . & xR_K^2 \\ . & . & . & . & . & . & . & . \\ xR_1^{nS} & . & . & . & . & . & . & xR_K^{nS} \end{bmatrix} \quad (2.16)$$

Donde FP_i es el *i*-ésimo jugador titular del equipo, S_i es el *i*-ésimo jugador suplente del equipo, nF es el número de jugadores titulares del equipo, nS es el número de jugadores suplentes del equipo, xF_K^i representa la *k*-ésima posición del vector solución del *i*-ésimo jugador titular y xR_K^i representa la *k*-ésima posición del vector solución del *i*-ésimo jugador suplente.

Una vez asignados todos los jugadores a sus equipos se realizan los enfrentamientos entre equipos. Para definir el equipo ganador de cada enfrentamiento se calcula el poder de los dos equipos, que se calcula como el poder promedio de sus jugadores titulares calculado con la función objetivo, y se describe en la Ecuación (2.17)

$$TP(i) = \left(\frac{1}{nF}\right)\sum_{j=1}^{nF} PP(i,j) \quad (2.17)$$

Donde nF es el número total de jugadores titulares del *i*-ésimo equipo y PP es el poder del *j*-ésimo jugador del *i*-ésimo equipo.

En cada encuentro, el equipo con más poder tendrá más oportunidad de ganar. La probabilidad de ganar de cada equipo es dada por las Ecuaciones (2.18) y (2.19).

$$Pv(k) = \frac{TP(k)}{(TP(i) + TP(k))} \quad (2.18)$$

$$Pv(i) = \frac{TP(i)}{(TP(i) + TP(k))} \quad (2.19)$$

Donde Pv representa la probabilidad de victoria de los equipos k e i que se van a enfrentar, la suma de estas dos probabilidades es igual a 1.

Después de cada enfrentamiento el equipo ganador realiza unos cambios en algunos de sus jugadores (soluciones candidatas), tanto titulares como suplentes. Estos cambios, tienen el objetivo de mejorar el desempeño de los jugadores y equipos, y son simulados con los procedimientos de imitación en los jugadores titulares y provocación en los jugadores suplentes.

Cada vez que un jugador aplica uno de los procedimientos se verifica que la nueva solución encontrada sea mejor que su predecesora, esto con el fin de reemplazarla en caso de que sea mejor. Este procedimiento se lleva a cabo hasta terminar todos los enfrentamientos de la liga.

En la Figura 2-7 se presenta el procedimiento de asignación de los jugadores a los equipos. Primero se crean los jugadores de forma aleatoria, luego se organizan de forma descendente por su desempeño (aptitud) y finalmente de acuerdo al desempeño se asigna el mismo número de jugadores a cada equipo donde se organizan de acuerdo al orden en el que hayan quedado después de ordenarlos descendentemente con el fin de agrupar a los mejores jugadores en un mismo equipo.

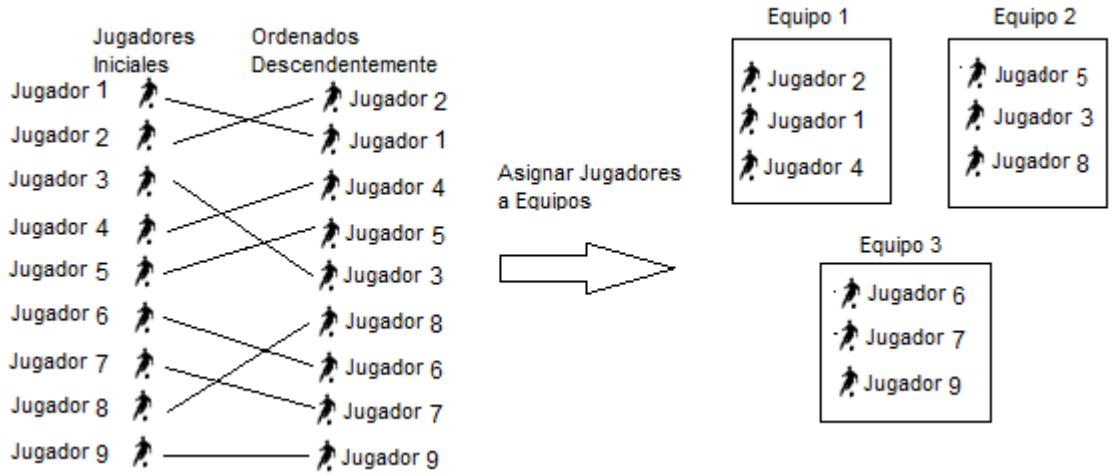

Figura 2-7 Procedimiento de asignación de equipos

La Figura 2-8 muestra la transición de los jugadores después de cada enfrentamiento entre equipos, en la que la mayoría de ellos converge hacia el mejor jugador de la liga (SSP). En el *primer cuadro*, los jugadores se encuentran en el espacio de búsqueda repartidos aleatoriamente, en el *segundo cuadro* se observan los jugadores asignados a un equipo diferenciándolos por color, después en el *tercer cuadro* se tienen los cambios (imitación y provocación) que aplican a mejorar el rendimiento de los jugadores, y finalmente en el *cuarto cuadro* se observa el proceso de convergencia de los jugadores hacia el mejor jugador de la liga o muy próxima a ella, este comportamiento se efectúa después de varias ligas. Este procedimiento se realiza un número de iteraciones definido o hasta cumplir el criterio de parada.

El esquema del algoritmo SLC se presenta en la Figura 2-9, inicialmente se genera aleatoriamente un jugador ($Inicializar\ x_i$), luego es evaluado ($Evaluar\ x_i$) y se verifica si el jugador es mejor que la mejor solución global ($Actualizar\ G_{best}$) para reemplazarla, este procedimiento se repite hasta generar los n jugadores. Luego, se organizan los jugadores de forma descendente ($Ordenar$) por su valor de aptitud para ser asignados a sus equipos ($asignar$) y establecer el mejor jugador del equipo o mejor solución local ($Instarciar\ p_i$). Posteriormente el algoritmo itera k enfrentamientos, donde en cada enfrentamiento se verifica el equipo ganador, se realiza una explotación por medio los procedimientos de imitación y provocación a los jugadores del equipo ganador. Dentro de cada procedimiento,

se evalúa y se actualiza las soluciones creadas para verificar si es mejor que su predecesor, luego se verifica si es mejor que la mejor solución local, y finalmente se verifica si es mejor que la mejor global.

Figura 2-8 Grafico del SLC

```
INICIO SLC
        HACER
            Inicializar x_i                    /*Crear jugador i */
            Evaluar x_i                        /*Evaluar la calidad del jugador i*/
            Actualizar G_best                  /*Actualizar la mejor solución global*/
        HASTA Crear n Jugadores
        Ordenar                                /*Ordenar jugadores descendentemente*/
        Asignar                                /*Asignar jugadores a los equipos*/
        Instanciar p_i                         /*Instanciar mejor solución local*/
        PARA k = 0 HASTA Nenfre                /*Número de enfrentamientos*/
            Selección Equipo 1 y 2             /*Selección de equipos a enfrentar*/
            SI Ganador Equipo1                 /*El equipo 1 resulta ganador*/
                Imitación Equipo1              /*Imitación a jugadores titulares*/
                Provocación Equipo1            /*Provocación a jugadores suplentes*/
            SINO                               /*El equipo 2 resulta ganador*/
                Imitación Equipo2              /*Imitación a jugadores titulares*/
                Provocación Equipo2            /*Provocación a jugadores suplentes*/
        FIN PARA
        Retornar mejor solución G_best
FIN SLC
```

Figura 2-9 Esquema general SLC

2.4 BÚSQUEDA LOCAL

La búsqueda local es una heurística donde se modifica levemente una solución inicial utilizando operadores de ajuste que permiten explorar mejores soluciones en su vecindad. En general se puede aplicar en diferentes problemas específicos de optimización combinatoria [69].

2.4.1 Ascenso de Colina

Es un algoritmo de optimización muy simple, en el cual se genera una solución inicial y luego se realiza un bucle que produce una nueva solución candidata, si esta nueva solución es mejor reemplaza a su predecesora convirtiéndose en la solución actual [70]. Un alto número de iteraciones logra aumentar las probabilidades de conseguir mejores soluciones, explotando una heurística de proximidad donde realizar cambios pequeños a una solución candidata frecuentemente, dan como resultado soluciones similares que tienen un valor de aptitud igual o mejor a la actual, permitiendo ascender la colina paso a paso hasta llegar a un óptimo local [69]. Al ser fácil de implementar y el poco uso de memoria que requiere (almacena solo el estado actual) son algunas de las ventajas de este algoritmo.

En la Figura 2-10, se presenta un esquema del algoritmo Ascenso de la Colina en [69], donde una solución inicial (S) es copiada $(Copiar(S))$ y transformada $(Ajustar\ (Copiar(S))$ para generar una nueva solución candidata (R), que es evaluada y comparada con S $(Calidad(R) > Calidad(S))$, de modo que si R es mejor que S, S se cambia con la información de R, de lo contrario S se mantiene. El algoritmo se repite hasta efectuar una condición de salida que puede ser encontrar una solución S ideal, llevar a cabo un número fijo de iteraciones o realizar un número máximo de evaluaciones. Por último el algoritmo retorna la mejor solución encontrada $(Retornar\ S)$.

$S = Solucion\ candidata\ inicial$
Repetir
    $R = Ajustar\ (Copiar(S))$
    Si $Calidad(R) > Calidad(S)$ **entonces**
        $S = R$
Hasta que S sea la solución ideal o se cumpla la condición de salida.
Retornar S.

Figura 2-10 Esquema general del algoritmo Ascenso de la Colina

2.4.2 Memoria Tabú

La memoria tabú es uno de los componentes principales del algoritmo de búsqueda Tabú, la cual puede ser: (i) Explícita, almacena soluciones completas visitadas durante la búsqueda; (ii) Atributiva, guardando información de ciertos atributos o características de las soluciones visitadas que cambian al moverse de una solución a otra; (iii) Memoria a corto plazo, su objetivo es evitar el ciclado, para ello almacena los últimos movimientos realizados y pueden ser utilizados para “recordar” aquellos que hacen caer de nuevo en soluciones ya exploradas; (iv) Memoria a largo plazo, registra la frecuencia de ocurrencias de los movimientos, las soluciones o sus atributos, evitando ciclos en óptimos locales a largo plazo. El objetivo de utilizar la memoria tabú es identificar soluciones candidatas ya visitadas para marcarlas como tabú de modo que el algoritmo no vuelva a visitar esas posibles soluciones durante una cantidad determinada de tiempo o de movimientos.

Capítulo 3

3 PROCESO DE ADAPTACIÓN: ALGORITMOS SFLA y SLC

Para el desarrollo de este proyecto se utilizó el Patrón de Investigación Iterativa (PII) propuesto por Pratt [71] diseñado para proyectos de investigación que involucran una solución computacional. Está compuesto por cuatro etapas principales que son: observación, identificación del problema, desarrollo de la solución y prueba de la solución. Además de la propuesta de Pratt, en este proyecto se contempla una etapa adicional de documentación y divulgación que es transversal a PII. En el primer ciclo se realiza la definición de la función objetivo y la adaptación de los algoritmos SFLA y SLC originales al problema de generación automática de resúmenes, con su respectiva configuración y afinamiento de parámetros. Para la búsqueda local se tiene en cuenta ascenso de la colina y memoria tabú, ya que al tener un límite de evaluaciones de la función objetivo, estos dos procedimientos usan menos evaluaciones permitiendo que el algoritmo evolucione de mejor manera, lo que no ocurre en otros procedimientos como ascenso de colina por la máxima pendiente, temple simulado y búsqueda tabú. En el segundo ciclo se adiciona ascenso de la colina a los algoritmos SFLA y SLC, se configuran y se afinan los parámetros de ascenso de la colina. Y en tercer ciclo se adiciona una memoria tabú a estos mismos algoritmos junto a la configuración y afinamiento de los parámetros asociados a la memoria tabú.

3.1 CICLO I: ADAPTACIÓN DE ALGORITMOS

Para hacer la adaptación de los algoritmos SFLA y SLC al problema de generación automática de resúmenes se toma como base la función objetivo planteado en [55] generando diferentes versiones de ella; la configuración y afinamiento de los parámetros de los algoritmos y los asociados al problema. Además se realiza el afinamiento de los pesos de la configuración de la función objetivo para establecer la función definitiva.

3.1.1 Características de la función objetivo

La función objetivo es un mecanismo importante para el área de generación automática de resúmenes que ayuda a identificar las oraciones más relevantes del documento, estableciendo la calidad de la oración a través de unas características que conforman la función objetivo. Se determina usar como configuración base el conjunto de características utilizadas en [55] teniendo en cuenta los buenos resultados obtenidos en este artículo. Estas características son independientes al lenguaje y son: posición de la oración en el documento, longitud de la oración, relación con el título, cohesión y cobertura.

Además, se realizó una revisión del estado del arte para identificar las características más utilizadas en el problema de generación automática de resúmenes y con estas generar diferentes versiones de la función objetivo donde se reemplaza una característica a la vez.

A continuación se describen las características.

3.1.1.1 Posición de la oración en el documento

De acuerdo a [72] la información más importante suele encontrarse en las primera posiciones del documento. Se han aplicado en [7], [18], [39], [44], [54] diferentes esquemas que evalúan las oraciones respecto a su posición en el documento mostrando buenos resultados para establecer la importancia de una oración.

- Posición 1

En [55], se define un esquema que utiliza la distancia que hay entre la posición de la oración y el inicio del documento, el puntaje asignado disminuye a medida que la oración se encuentre más lejos del inicio del documento (Ver Ecuación (3.1)). Esta característica tiende a $\frac{2}{O}$ cuando las oraciones en el resumen candidato corresponden a las primeras oraciones del documento original y tiende a cero cuando las oraciones están en las últimas posiciones.

$$PF_s = \frac{\sum_{\forall S_i \in S} PositionRanking(S_i)}{O} \qquad (3.1)$$

Donde O es el número de oraciones en el resumen S y $PositionRanking(S_i)$ es la clasificación de la posición de cada oración S_i.

Esta clasificación de la posición se basa en el método de selección lineal de rango usada en algoritmos genéticos (Ver Ecuación (3.2)). El mejor en la clasificación recibe un valor de $\frac{2}{n}$ y el más bajo en la clasificación un valor cercano a cero.

$$PositionRanking(S_i) = \frac{2 - 2 * \left(\frac{pos_i - 1}{n - 1}\right)}{n} \qquad (3.2)$$

Donde pos_i es la posición de la oración i según el orden de aparición en el documento, y n es el número total de oraciones en el documento.

- Posición 2

El artículo [2] plantea la característica posición con una fórmula matemática diferente donde las oraciones que aparezcan en las primeras posiciones en un párrafo tendrán mayor valor acercándose a uno y a medida de que se alejan de las primeras posiciones su valor desciende tendiendo a cero, como se muestra en la Ecuación (3.3).

$$PFs = \frac{(n - i)}{n} \qquad (3.3)$$

Donde PF indica el puntaje de la oración s, n representa el número de oraciones del documento e i indica la posición de la oración en el documento (arrancando la primera posición en cero).

3.1.1.2 Longitud de la oración

Esta característica hace referencia a la importancia de una oración en el documento teniendo en cuenta la longitud de la misma, medida en el número de palabras. Algunos estudios plantean que el resumen de un documento debe tener menos en cuenta las oraciones cortas debido a que estas pueden contener menos información significativa [8].

- Longitud 1

En [55] se aplica la desviación estándar para obtener una evaluación relativa de las oraciones en relación a su longitud, que evalúa favorablemente los resúmenes compuestos de oraciones largas y medianas (Ver Ecuaciones (3.4) y (3.5)), calculando el promedio de las longitudes de las oraciones del resumen (dividendo entre el número de oraciones N), de tal manera que a medida que N crece (oraciones cortas), disminuye el valor de esta característica en la función objetivo, en relación con un resumen cuyo N es más pequeño (oraciones largas).

$$L = \frac{\sum_{\forall S_i \in S} \frac{1 - e^{-\alpha_i}}{1 + e^{-\alpha_i}}}{N} \tag{3.4}$$

$$\alpha_i = \frac{l_i - \mu(l_i)}{std(l_i)} \tag{3.5}$$

Donde l_i es la longitud de la oración S_i , $\mu(l_i)$ es la longitud media de las oraciones y $std(l_i)$ es la desviación estándar de las longitudes de las oraciones.

- Longitud 2

En [18] se describe la característica longitud penalizando las oraciones más cortas según el número de palabras que la componen. La longitud de la oración se compara con la longitud de la oración más larga del documento, como se muestran en la Ecuación (3.6).

$$L = \frac{nSi}{nLongDocumento} \tag{3.6}$$

Donde nSi representa el número de palabras que componen la oración i y $nLongDocumento$ es el número de palabras que componen la oración más larga del documento.

3.1.1.3 Relación con el título

Esta característica evalúa la similitud entre el título del documento y las oraciones del resumen, ya que se asume que el título contiene las palabras más relevantes y por ende las oraciones más relevantes son similares a éste [73].

- Relación con el título 1

En [55] esta característica calcula la similitud de coseno entre los vectores que representan cada oración s_i que forman el resumen y el vector que representa el título t, (Ver Ecuaciones (3.7) y (3.8)).

$$TRF_s = \frac{TR_s}{máximo_{\forall S} TR} \tag{3.7}$$

$$TR_s = \frac{\sum_{S_i \in S} sim(S_i, t)}{L} \tag{3.8}$$

Donde TR_s es el promedio de la similitud de las oraciones con el título en el resumen S, L es el número de oraciones del resumen, TRF_s es el factor de similitud de las oraciones del resumen S con el título y $máximo_{\forall S} TR$ es el promedio de los máximos valores obtenidos de las similitudes de las oraciones con el título. Si TR_s es cercano a cero las oraciones del resumen son distintas al título y recíprocamente cuando el valor es cercano a 1.

- Relación con el título 2

En [20], [26] presentan la característica de relación al título destacando el hecho que el título siempre resume los temas más importantes del documento. Por lo tanto, cuantas más palabras tengan en común una oración con el título, más importante es, obteniendo un valor cercano a uno. Para una oración este puntaje se define en la Ecuación (3.9).

$$TRs = \frac{words\ in\ S\ \cap words\ in\ Title}{words\ in\ S\ \cup words\ in\ Title} \tag{3.9}$$

Donde TR es el puntaje de una oración s en relación al título, se intersectan las palabras que tengan en común la oración s y el título, para luego dividirlo entre el número de términos que hay entre la oración s y el título.

3.1.1.4 Cohesión

Esta característica permite generar un resumen con oraciones relacionadas, midiendo la similitud que existe entre las diferentes oraciones que conforman el resumen para establecer si tratan sobre la misma información.

En [55] la evaluación de cohesión de un resumen se realiza midiendo la similitud de cosenos entre los vectores que representan las oraciones que lo componen, como se muestra en las Ecuaciones (3.10) - (3.11).

$$CF = \frac{\sum_{\forall S_i, S_j \in S} sim(S_i, S_j)}{N_s} \tag{3.10}$$

$$N_s = \frac{(L)\ *\ (L-1)}{2} \tag{3.11}$$

Donde CF es el promedio de las similitudes de todas las oraciones en el resumen S, que corresponde a la cohesión del resumen S, $sim(S_i, S_j)$ es la similitud de cosenos entre las

oraciones S_i y S_j, N_s es el número de comparaciones de similitud entre las oraciones del resumen, L es el número de oraciones del resumen N_s, como se presenta en la Ecuación (3.11). De este modo CF tiende a cero cuando las oraciones del resumen son muy diferentes entre ellas, mientras que CF tiende a uno cuando las oraciones son muy similares entre ellas.

3.1.1.5 Cobertura

La cobertura intenta medir que tanta información del texto original es cubierta por las oraciones que componen el resumen [74].

En [55] se presenta un esquema para la evaluación de esta característica, donde se calcula la similitud de cosenos entre el vector de términos que representa el resumen y el vector de términos del documento, como se muestra en la Ecuación (3.12).

$$Cov = sim_{cos}(R, D) \tag{3.12}$$

Donde R representa el vector del resumen y D el vector de todo el documento. Cuando Cov tiene un valor cercano a uno significa que el resumen cubre una gran proporción del texto original y por lo tanto tiene una alta cobertura, mientras un valor cercano a cero indica que el resumen tiene baja cobertura.

3.1.1.6 Centralidad de la oración

Ésta es una de las características más usadas en la revisión del estado del arte [18], [20], [26] y mide la similitud que hay entre una oración y el resto del documento, como el grado de superposición de vocabulario entre una oración y las demás. Esto quiere decir, que si una oración contiene más términos idénticos a las demás oraciones, es más significativa. En general, la oración con la centralidad más alta denota el centroide del documento. Para una oración s, su centralidad se calcula de acuerdo a la Ecuación (3.13).

$$SentCent(s) = \frac{words\ in\ S\ \cap\ words\ in\ Other\ Sentences}{words\ in\ S\ \cup\ words\ in\ Other\ Sentences} \tag{3.13}$$

Donde $SentCent(s)$ es la centralidad de la oración s, $words\ in\ s$ son las palabras de la oración s y $words\ in\ Others\ Sentences$ las palabras de las demás oraciones del documento.

3.1.2 Configuraciones de la Función Objetivo

Se toma como referencia la función objetivo definitiva planteada en [55] para el desarrollo de este proyecto adaptándola a los algoritmos SFLA y SLC. Además, se generan diferentes versiones de ésta haciendo algunos cambios en sus características (ver Tabla 1).

Versión	Adaptación
1	Función objetivo original sin cambios
2	Cambio de la fórmula matemática de la característica posición
3	Cambio de la fórmula matemática de la característica longitud
4	Cambio de la fórmula matemática de la característica relación con el título
5	Cambio de la característica cohesión por centralidad de la oración

Tabla 1 Cambios en la función objetivo

Primera versión: Versión Original

Esta configuración es la versión original que utiliza las cinco características usadas en [55]; y es calculada como se presenta en la Ecuación (3.14), con su correspondiente fórmula matemática en (3.15).

$$f(x) = Posición1 + Longitud1 + Relación\ con\ el\ título1 + Cohesió + Cobertura \tag{3.14}$$

$$f(x) = \sum_{\forall S_i \in Summary} \left[\frac{PositionsRanking(s_i)}{O} + \frac{\frac{1 - e^{-\frac{l(s_i)-\mu(l)}{std(l)}}}{1 + e^{-\frac{l(s_i)-\mu(l)}{std(l)}}}}{N} + \frac{sim(s_i, t)}{máximo_{\forall summary} TR} + \sum_{\forall S_j \in Summary} \frac{sim_{cos}(s_i, s_j)}{N_s} \right] + sim_{cos}(R, D) \tag{3.15}$$

Segunda versión: Variación posición

En esta versión se toma la primera versión y se modifica la fórmula para el cálculo de la característica de la posición (Posición2).

$$f(x) = \boldsymbol{Posición2} + Longitud1 + Relación\ con\ el\ título1 + Cohesión + Cobertura \tag{3.16}$$

$$f(x) = \sum_{\forall S_i \in Summary} \left[\frac{(\boldsymbol{n} - \boldsymbol{i})}{\boldsymbol{n}} + \frac{\frac{1 - e^{-\frac{l(s_i) - \mu(l)}{std(l)}}}{1 + e^{-\frac{l(s_i) - \mu(l)}{std(l)}}}}{N} + \frac{sim(s_i, t)}{máximo_{\forall summary} TR} + \sum_{\forall S_j \in Summary} \frac{sim_{cos}(s_i, s_j)}{N_s} \right] + sim_{cos}(R, D) \tag{3.17}$$

Tercera versión: Variación longitud

En esta versión se toma la primera versión y se modifica la fórmula para el cálculo de la característica de la longitud (Longitud2).

$$f(x) = Posición1 + \boldsymbol{Longitud2} + Relación\ con\ el\ título1 + Cohesión + Cobertura \tag{3.18}$$

$$f(x) = \sum_{\forall S_i \in Summary} \left[\frac{PositionsRanking(s_i)}{O} + \frac{\boldsymbol{nSi}}{\boldsymbol{nLongDocumento}} + \frac{sim(s_i, t)}{máximo_{\forall summary} TR} + \sum_{\forall S_j \in Summary} \frac{sim_{cos}(s_i, s_j)}{N_s} \right] + sim_{cos}(R, D) \tag{3.19}$$

Cuarta versión: Variación Relación con el Título.

En esta versión se toma la primera versión y se modifica la fórmula para el cálculo de la característica de relación con el título (Relación con el título2).

$$f(x) = Posición1 + Longitud1 + \boldsymbol{Relación\ con\ el\ título2} + Cohesión + Cobertura \tag{3.20}$$

$$f(x) = \sum_{\forall S_i \in Summary} \left[\frac{PositionsRanking(s_i)}{O} + \frac{\frac{1 - e^{-\frac{l(s_i)-\mu(l)}{std(l)}}}{1 + e^{-\frac{l(s_i)-\mu(l)}{std(l)}}}}{N} + \frac{\boldsymbol{words\ in\ S_i\ \cap words\ in\ Title}}{\boldsymbol{words\ in\ S_i\ \cup words\ in\ Title}} + \sum_{\forall S_j \in Summary} \frac{sim_{cos}(s_i, s_j)}{N_s} \right] + sim_{cos}(R, D) \tag{3.21}$$

Quinta versión: Cambio de Cohesión por Centralidad de la oración.

En esta versión se toma la primera versión y se cambia la característica cohesión por la característica centralidad (Centralidad).

$$f(x) = Posición1 + Longitud1 + Relación\ con\ el\ título1 + \boldsymbol{Centralidad} + Cobertura \tag{3.22}$$

$$f(x) = \sum_{\forall S_i \in Summary} \left[\frac{PositionsRanking(s_i)}{O} + \frac{\frac{1 - e^{-\frac{l(s_i)-\mu(l)}{std(l)}}}{1 + e^{-\frac{l(s_i)-\mu(l)}{std(l)}}}}{N} + \frac{sim(s_i, t)}{máximo_{\forall summary} TR} + \frac{\boldsymbol{words\ in\ S_i\ \cap words\ in\ Other\ Sentences}}{\boldsymbol{words\ in\ S_i\ \cup words\ in\ Other\ Sentences}} \right] + sim_{cos}(R, D) \tag{3.23}$$

3.1.3 Configuración de parámetros

Se tomó como referencia los parámetros utilizados para un problema continuo en [75] y [76] para definir los parámetros de los algoritmos SFLA y SLC respectivamente, adaptándolos al problema de generación automática de resúmenes, el cual es discreto.

3.1.3.1 Algoritmo SFLA

El algoritmo utiliza cuatro parámetros independientes: el número de ranas, el número de memeplex, el número de iteraciones dentro del memeplex y el número de iteraciones máximas.

- Número de ranas (*P*): este parámetro es fundamental ya que establece la diversidad dentro del espacio de búsqueda, se recomienda valores cercanos a la cantidad de dimensiones del problema.

- Número de memeplex (*m*): es el número de grupos en los que se divide la población dentro del espacio de búsqueda, este valor es importante ya que define cuántas ranas hay dentro de cada grupo y así mismo indica que tan intensa se realiza la explotación.

- Número de iteraciones (*it*): el número de iteraciones controla la extensión de la explotación, debido a esto se recomienda usar un número pequeño para lograr una búsqueda rápida.

- Número de iteraciones máximas (*itMax*): este valor controla la extensión de la exploración, permitiendo que el algoritmo pueda combinar diferentes tipos de poblaciones y lograr una convergencia hacia la mejor solución.

3.1.3.2 Algoritmo SLC

El algoritmo utiliza cuatro parámetros independientes: el número de equipos, el número de jugadores titulares, el número de jugadores suplentes y el número de iteraciones.

- Número de Equipos (nT): este parámetro indica el número de equipos en los que se divide la población de jugadores, esto con el fin de realizar explotación en el espacio de búsqueda a cada uno de los jugadores que conforman el grupo.

- Número de Jugadores Titulares (nF): es un parámetro importante para lograr variedad en el algoritmo ya que establece los puntos de exploración del espacio de búsqueda. Se recomienda utilizar valores grandes, aunque para realizar la experimentación de este problema se recomienda utilizar valores pequeños debido a la restricción del número de evaluaciones de la función objetivo y que la extensión del espacio de búsqueda de este problema es más pequeña en comparación a la de un problema continuo.

- Número de Jugadores Suplentes (nS): al igual que el número de jugadores titulares es un parámetro que se utiliza para lograr variedad en el algoritmo, aunque se toman en segundo plano. Se recomienda valores pequeños ya que estos se toman como respaldos en caso de necesitar cambios en los jugadores titulares.

- Iteraciones (nI): el número de ligas o iteraciones controla la extensión de búsqueda, debido a esto se recomienda usar un número pequeño de iteraciones que se refleja en una búsqueda rápida y encontrar una respuesta dentro de los límites esperados.

Utilizando la función objetivo original con la combinación final de pesos presentada en [55] se realizó la afinación de parámetros asociados a los algoritmos SFLA y SLC para definir la configuración preliminar de estos, que se muestra en la Ecuación (3.24).

$$f(x) = 0{,}15 * Posición1 + 0{,}05 * Longitud1 + 0{,}10 * Relación\ con\ el\ título1 + 0{,}05 * Cohesión + 0{,}65 * Cobertura \quad (3.24)$$

Por lo tanto la afinación preliminar obtuvo una combinación de valores para estos parámetros que se listan en la Tabla 2, junto a su abreviatura.

SFLA		SLC	
Parámetro	Valor	Parámetro	Valor
P	200	nT	40
m	10	nF	10
It	10	nS	5
itMax	150	nI	1

Tabla 2 Configuración preliminar de los parámetros de SFLA y SLC

3.1.3.3 Asociados al problema

Teniendo en cuenta que se trata el mismo problema que FSP, se toman los mismos parámetros asociados al problema presentados en [55]. Estos parámetros son:

- Máximo Número de Evaluaciones de la función objetivo: permite realizar una comparación en igual condiciones con otros métodos metaheurísticos del estado del arte, estableciendo el valor de este parámetro en 1600 evaluaciones.

- Límite de cantidad de palabras del resumen: establece el número máximo de palabras que puede tener el resumen generado. Para permitir la evaluación de los resúmenes generados mediante las medidas ROUGE sobre los conjuntos de datos DUC2001 y DUC2002, se estableció el valor de este parámetro en 100 palabras.

- Máxima longitud a evaluar de un resumen: con el fin de lograr una mayor exploración del espacio de búsqueda, durante la evolución de una solución candidata se permite que ésta supere el número de palabras hasta el valor definido por este parámetro. Sin embargo, al finalizar la ejecución del algoritmo, si es necesario se eliminan oraciones para cumplir con la restricción del número de palabras que deben conformar el resumen (límite de cantidad de palabras del resumen).

- Criterio para deshabilitar una oración: establece que característica(s) de la función objetivo son utilizadas para calcular el puntaje de cada oración del documento y deshabilitar la oración con el puntaje más bajo.

- Criterio de selección de oraciones del resumen final: define que característica(s) de la función objetivo serán utilizadas al finalizar la ejecución del algoritmo para definir el orden en que las oraciones aparecen en el resumen generado.

La configuración preliminar de estos parámetros se lista en la Tabla 3.

Parámetro	Valor
Máximo Número de Evaluaciones de la función objetivo	1600
Máxima longitud a evaluar de un resumen	130
Criterio para deshabilitar una oración	Posición-Cobertura
Criterio de selección de oraciones del resumen final	Posición

Tabla 3 Configuración preliminar de los parámetros asociados al problema

3.1.4 Afinación de la función objetivo

Para afinar la función objetivo, se deben afinar los pesos asociados a las características que la componen con la finalidad de lograr un equilibrio en los valores obtenidos (ya que estos no se encuentran en el mismo rango) y de esta forma lograr que las características obtengan valores semejantes, de manera que al evaluar la función objetivo esta sea de forma proporcional. Para realizar el afinamiento de los pesos en la configuración de la función objetivo, se tomó la configuración preliminar de los parámetros de los algoritmos SFLA y SLC (véase Tabla 2) y los parámetros asociados al problema (véase Tabla 3).

El afinamiento de estos pesos asociados a las características se realizó utilizando como base los pesos utilizados en [55], luego se modifican los pesos de cada característica en un rango entre 0.01 y 0.05, teniendo en cuenta que las suma de los pesos de todas las características debe ser igual a 1, por lo cual al incrementar el peso de una característica se debe reducir el mismo valor en una o varias características de la función objetivo.

La configuración de la función objetivo está definida en la Ecuación (3.25), con los pesos asociados a cada característica.

$$f(x) = \alpha * Posición1 + \beta * Longitud1 + \gamma * Relación\ con\ el\ título1 + \mu * Cohesión + \rho * Cobertura \tag{3.25}$$

Después de realizar el proceso de afinación, la combinación de pesos que obtuvo mejores resultados se muestra en la
Tabla 4.

Algoritmo	Posición	Relación con el título	Longitud	Cohesión	Cobertura
	α	β	γ	μ	ρ
SFLA	0,15	0,04	0,09	0,07	0,65
SLC	0,19	0,05	0,06	0,05	0,65

Tabla 4 Pesos de la Tercera Función Objetivo

3.1.5 Configuración Definitiva de la Función Objetivo

Los resultados de la evaluación de las diferentes versiones de la función objetivo se presentan en la Tabla 5, donde la versión original obtiene los mejores resultados en todas las medidas ROUGE sobre DUC2001 (solo con la excepción de ROUGE-2) y DUC2002 tanto en SFLA como en SLC. Por esto, se establece la versión original (ver Ecuación (3.19)) como la función objetivo definitiva, con sus correspondientes pesos.

Algoritmo	Versión	DUC2001			DUC2002		
		Rouge-1 Recall	Rouge-2 Recall	Rouge-SU4 Recall	Rouge-1 Recall	Rouge-2 Recall	Rouge-SU4 Recall
SFLA	**Original**	**0,45953**	0,20653	**0,22883**	**0,48738**	**0,22934**	**0,24587**
	Segunda	0,43543	0,18050	0.20463	0,46598	0,20556	0,22458
	Tercera	0,45800	**0,20746**	0,22792	0,48429	0,22820	0,24433
	Cuarta	0,44673	0,18913	0,21431	0,47370	0,21440	0,23339
	Quinta	0,45809	0,20279	0,22643	0,48634	0,22800	0,24463
SLC	**Original**	**0,46045**	**0,20757**	**0,23003**	**0,48741**	**0,22997**	**0,24632**
	Segunda	0,44606	0,19636	0,21862	0,47541	0,21778	0,23588
	Tercera	0,45964	0,20724	0,22946	0,48530	0,22886	0,24505
	Cuarta	0,44936	0,19461	0,21870	0,47459	0,21704	0,23572
	Quinta	0,45886	0,20495	0,22769	0,48635	0,22904	0,24547

Tabla 5 Resultados de afinación de las funciones objetivo

3.1.6 Afinación de parámetros

Utilizando la configuración definitiva (versión original) de la función objetivo, el proceso de afinación de parámetros de los algoritmos y los asociados al problema, se muestran a continuación.

3.1.6.1 Algoritmos SFLA y SLC

El proceso de afinamiento de los parámetros de los algoritmos SFLA y SLC fue el siguiente:

- Primer paso. Conformación de un conjunto de combinaciones de valores para los parámetros (P, m, It y itMax) de SFLA y (nT, nF, nS y nI) de SLC sin ningún tipo de restricción. Una vez generado este conjunto, se descartan aquellas combinaciones que no presentan un buen balance entre exploración y explotación de acuerdo a la sección 3.1.3 (Configuración de parámetros algoritmos SFLA y SLC).

- Segundo paso. Evaluación del conjunto definido en el primer paso, para luego, analizar los resultados obtenidos y determinar que combinación de parámetros obtiene el mejor resultado. En la Tabla 6 se presenta el valor de cada uno de los parámetros SFLA y SLC con los que se obtuvieron los mejores resultados.

SFLA		SLC	
Parámetro	Valor	Parámetro	Valor
P	20	nT	40
m	5	nF	10
It	10	nS	5
itMax	150	nI	1

Tabla 6 Mejor Combinación de los parámetros de SFLA y SLC

3.1.6.2 Asociados al Problema

Para el proceso de afinamiento de los parámetros relacionados con el problema de generación de resúmenes de un documento se utilizaron los parámetros asociados a los algoritmos SFLA y SLC presentados en la Tabla 6.

- Máxima longitud a evaluar de un resumen: se tomaron valores cercanos al parámetro de *Límite de cantidad de palabras del resumen* definido en 100. Por lo tanto, los valores fueron 100, 110, 120 y 130. Para SLFA y SLC, los mejores resultados se obtuvieron con el valor de 120 para el DUC2002 y de 130 para DUC2001. Para SLFA, se toma el valor de 120 teniendo en cuenta que DUC2001 es más propenso a mejorar con este algoritmo que DUC2002. Para SLC, se toma el valor de 130 porque con este algoritmo DUC2002 es más propenso a mejorar que DUC2001 (ver Tabla 7).

Algoritmo	Experimento	DUC2001			DUC2002		
		R1R	R2R	RSU4R	R1R	R2R	RSU4R
SFLA	100	0,44842	0,20303	0,22405	0,47223	0,22015	0,23670
	110	0,45690	0,20447	0,22725	0,48411	0,22590	0,24272
	120	0,45654	0,20493	0,22682	**0,48824**	**0,23051**	**0,24648**
	130	**0,45978**	**0,20702**	**0,22932**	0,48687	0,22934	0,24589
SLC	100	0,44714	0,20214	0,22271	0,47217	0,22075	0,23693
	110	0,45757	0,20478	0,22722	0,48565	0,22748	0,24420
	120	0,45773	0,20472	0,22720	**0,48897**	**0,23093**	**0,24681**
	130	**0,46045**	**0,20757**	**0,23003**	0,48740	0,22997	0,24632

Tabla 7 Mejores resultados Máxima Longitud a Evaluar

- Criterio para deshabilitar una oración: se seleccionaron individualmente las características de Posición, Relación con el Título, Longitud, Cohesión y Cobertura; y la combinación de las características que obtuvieron mejores resultados. Los resultados obtenidos se presentan en la Tabla 8, con el *Criterio de selección de oraciones del resumen final* fijado en *Posición.* Se observa que para el caso de SFLA la característica de *Posición* obtuvo mejores resultados en 5 de 6 medidas, y en SLC la *Posición-Cobertura* obtuvo los mejores resultados en todas las medidas.

Algoritmo	Experimento	DUC2001			DUC2002		
		R1R	R2R	RSU4R	R1R	R2R	RSU4R
SFLA	Posición Cobertura	0,45654	0,20493	0,22682	0,48824	0,23051	0,24648
	Cobertura	0,45645	0,20421	0,22631	0,48710	0,22932	0,24559
	Posición	0,45643	**0,20592**	**0,22765**	**0,48990**	**0,23215**	**0,24793**
	Cohesión	**0,45729**	0,20496	0,22719	0,48763	0,22895	0,24539
	Relación con el Título	0,45524	0,20232	0,22499	0,48706	0,22879	0,24514
	Longitud	0,45471	0,20299	0,22514	0,48667	0,22937	0,24543
SLC	**Posición Cobertura**	**0,46045**	**0,20757**	**0,23003**	**0,48740**	**0,22997**	**0,24632**
	Cobertura	0,46014	0,20686	0,22945	0,48700	0,22973	0,24604
	Posición	0,45938	0,20691	0,22940	0,48623	0,22883	0,24533
	Cohesión	0,45759	0,20483	0,22765	0,48536	0,22794	0,24463
	Relación con el Título	0,46015	0,20717	0,22968	0,48679	0,22920	0,24560
	Longitud	0,45972	0,20678	0,22934	0,48681	0,22940	0,24585

Tabla 8 Mejor resultado Criterio para deshabilitar una oración

- Criterio de selección de oraciones del resumen final: se experimentó con las características individuales y la combinación que obtuvo mejor resultado de acuerdo a la Tabla 8, donde el *Criterio para deshabilitar una oración* se estableció en *Posición* para SFLA y *Posición-Cobertura* para SLC. Los resultados obtenidos al finalizar el proceso de afinación del parámetro *Criterio de selección de oraciones del resumen final*, se presentan en la Tabla 9, en la cual se observa que *Posición* fue la característica con mejor desempeño en SFLA obteniendo mejores resultados en 5 de 6 medidas y en SLC obteniendo los mejores resultados en todas las medidas.

Algoritmo	Experimento	DUC2001			DUC2002		
		R1R	R2R	RSU4R	R1R	R2R	RSU4R
SFLA	Posición Cobertura	0,45481	0,20436	0,22654	0,48785	0,22908	0,24498
	Cobertura	0,45402	0,20284	0,22481	0,48540	0,22627	0,24237
	Posición	0,45643	**0,20592**	**0,22765**	**0,48990**	**0,23215**	**0,24793**
	Cohesión	0,45523	0,20255	0,22471	0,48612	0,22611	0,24242
	Relación con el Título	**0,45671**	0,20409	0,22566	0,48418	0,22409	0,24064
	Longitud	0,45318	0,20291	0,22505	0,48433	0,22561	0,24196
SLC	Posición Cobertura	0,45703	0,20417	0,22678	0,48477	0,22639	0,24245
	Cobertura	0,45603	0,20205	0,22502	0,48230	0,22357	0,23990
	Posición	**0,46045**	**0,20757**	**0,23003**	**0,48740**	**0,22997**	**0,24632**
	Cohesión	0,45827	0,20317	0,22678	0,48393	0,22406	0,24051
	Relación con el Título	0,45920	0,20444	0,22660	0,48268	0,22285	0,23944
	Longitud	0,45309	0,20013	0,22280	0,48023	0,22271	0,23945

Tabla 9 Mejor resultado Criterio de selección de oraciones del resumen final

Se debe resaltar que en el proceso de afinación de los parámetros *Criterio para deshabilitar una oración* y *Criterio de selección de oraciones del resumen final*, no fue necesario una nueva afinación de los pesos de la función objetivo y se utilizaron los mismos parámetros asociados a los algoritmos presentados en la Tabla 6. Los valores de los parámetros que corresponden al experimento que obtuvo el mejor resultado se presentan en la Tabla 10.

Parámetro	Valor	
Máximo Número de Evaluaciones de la función objetivo		1600
Máxima longitud a evaluar de un resumen	SFLA	120
	SLC	130
Criterio para deshabilitar una oración	SFLA	Posición
	SLC	Posición Cobertura
Criterio de selección de oraciones del resumen final		Posición

Tabla 10 Configuración final de los parámetros asociados al problema

3.1.7 Esquema Algoritmos Adaptados

3.1.7.1 SFLA-SingleDocSum

El algoritmo SFLA original (ver Figura 2-6) fue adaptado al problema de generación automática de resúmenes de textos (SLC-SingleDocSum, ver Figura 3-1) realizando los siguientes cambios: (i) se establece la representación de la soluciones de forma binaria para facilitar las operaciones de los algoritmos (ver sección 4.1) y además todas las soluciones generadas de forma aleatoria deben ser soluciones viables para que el algoritmo inicie con mejores soluciones; (ii) en la búsqueda local los diferentes tipos de saltos tienen una modificación en el método de reparación para mejorar la fase de explotación al incluir conocimiento del problema; (iii) el método de mutación no incluye a la mejor rana con el fin de conservar la mejor solución encontrada para la siguiente iteración y se aplica reparación a las ranas mutadas para mejorar la exploración; y (iv) el algoritmo retorna la mejor solución ordenada según el criterio de selección de oraciones del resumen final.

```
INICIO SFLA
        HACER
            Inicializar solución viable x_i        /*Crear rana i*/
            Evaluar x_i                            /*Evaluar la calidad de la rana i*/
        HASTA Crear P Ranas
        PARA k = 0 HASTA itMax                     /*Número de iteraciones*/
            RankearRanas                           /*Ordenar las ranas descendientemente*/
            BarajarRanas                           /*Repartir las ranas en los memeplex*/
            BusquedaLocal                          /*Generar nuevas soluciones con los saltos*/
            ReagruparRanas                         /*Reunir ranas desde los memeplex*/
            PARA j = 1 HASTA P                     /*Mutación desde la segunda rana */
                MutarRana x_j                      /*Aplicar mutación a la rana j*/
            FIN PARA
        FIN PARA
        Retornar G_best
FIN SFLA
```

Figura 3-1 Esquema Algoritmo SFLA-SingleDocSum

La representación de las soluciones de forma binaria se hace a través de la normalización de los valores dentro de los saltos como se observa en las Ecuaciones (3.26) y (3.27). Donde t_i es el resultado de utilizar una función sigmoide que normaliza los valores de S y luego se utiliza la función x_{w_i} donde según el intervalo en que t_i corresponda, tomará como valor cero, uno o se mantendrá el valor.

$$t_i = \frac{1}{1 + e^{-S_i}} \tag{3.26}$$

$$y_i = \begin{cases} 0 \quad , \quad t_i \leq \alpha \\ y_i \, , \quad \alpha < t_i \leq \frac{1}{2}(1+\alpha) \\ 1 \quad , \quad \frac{1}{2}(1+\alpha) < t_i \end{cases} \tag{3.27}$$

Dentro de la Búsqueda Local descrita en la Figura 3-2, en cada salto se genera una solución copia (y_i) de la rana a ajustar, luego se aplica uno de los 2 tipos de reparación que se incluyeron para modificar el vector solución que incluyen conocimiento del problema. Inicialmente se evalúan las oraciones no activas en el conjunto de soluciones de la copia de acuerdo a las características establecidas en el *Criterio para deshabilitar una oración*, luego se tiene en cuenta el tipo de salto: los saltos guiados hacia la mejor rana local (x_b) o la mejor rana global (x_g) hacen la $Reparacion2$ que consiste en seleccionar una oración de forma aleatoria para ser habilitada; y en el caso del salto aleatorio, se hace la $Reparacion1$ que selecciona la oración con la mayor evaluación para ser habilitada. Una vez la solución copia ya ha creado el nuevo punto y se ha hecho la reparación, se evalúa su calidad mediante la función objetivo definida en la sección 3.1.4 y a continuación se compara con la mejor solución global (x_g), de este modo, si la nueva rana es mejor, x_g se actualiza con su información.

En el procedimiento $MutarRanas$ ya no se incluye la rana con mayor aptitud (G_{best} o x_g), esto quiere decir que el procedimiento de mutación se hace de igual forma, aplicando con cierta probabilidad la mutación a la población desde la segunda rana hasta la última, con el fin de no alterar la mejor solución encontrada en el ciclo.

Cabe destacar que al retornar la *mejor solución* global (G_{best}), el algoritmo SFLA-SingleDocSum ordena las oraciones según el *Criterio de selección de oraciones del resumen final* para que aparezcan en ese orden en el resumen generado.

```
INICIO Búsqueda Local
    PARA im = 0 HASTA m                    /*Número de Memeplex*/
        PARA itm = 0 HASTA it              /*Número de iteraciones en cada memeplex*/
            Determinar x_w, x_b y x_g      /*Determinar la peor y mejor rana*/
            y_i = Saltar x_b               /*Saltar usando la mejor local*/
            Reparacion2                    /*Habilitar oración aleatoriamente*/
            Evaluar y_i
            SI Aptitud y_i > Aptitud x_w
                x_w = y_i                  /*Actualizar la peor rana*/
            SINO
                y_i = Saltar x_g           /*Saltar usando la mejor global*/
                Reparacion2                /*Habilitar oración aleatoriamente*/
                Evaluar y_i
                SI Aptitud y_i > Aptitud x_w
                    x_w = y_i              /*Actualizar la peor rana*/
                SINO
                    x_w = Inicializar y_i  /*Inicializar x_w*/
                    Reparacion1            /*Habilitar mejor oración*/
                    Evaluar y_i
                    SI Aptitud y_i > Aptitud x_g
                        x_g = y_i          /*Actualizar la mejor rana global*/
        FIN PARA
    FIN PARA
FIN Búsqueda Local
```

Figura 3-2 Esquema Procedimiento Búsqueda Local

3.1.7.2 SLC-SingleDocSum

El algoritmo SLC original (ver Figura 2-9) fue adaptado al problema de generación automática de resúmenes de textos (SLC-SingleDocSum, ver Figura 3-3), cuyos cambios son: (i) se establece la representación de la soluciones de forma binaria para facilitar las operaciones de los algoritmos (ver sección 4.1) y además todas las soluciones generadas de forma aleatoria deben ser soluciones viables para que el algoritmo inicie con mejores soluciones; (ii) se modifican los procedimientos de mejora (imitación y provocación) que se aplican al equipo ganador para mejorar la fase de explotación; (iii) se incluyen dos nuevos procedimientos (mutación y sustitución) que se aplican en los jugadores del equipo perdedor incluyendo conocimiento del problema mejorando la fase de explotación en mutación y la fase de exploración en sustitución; y (iv) el algoritmo retorna la mejor solución ordenada según el criterio de selección de oraciones del resumen final.

```
INICIO SLC
    HACER
        Inicializar solución viable xi          /*Crear jugador i*/
        Evaluar x_i                             /*Evaluar la calidad del jugador i*/
        Actualizar G_best                       /*Actualizar la mejor solución global*/
    HASTA Crear n Jugadores
    Ordenar                                     /*Ordenar jugadores descendentemente*/
    Asignar                                     /*Asignar jugadores a los equipos*/
    Instanciar p_i                              /*Instanciar mejor solución local*/
    PARA k = 0 HASTA Nenfre                     /*Número de enfrentamientos*/
        Selección equipo 1y2                    /*Selección de equipos a enfrentar*/
        SI Ganador Equipo1
            Imitación Equipo1                   /*Imitación a jugadores titulares*/
            Provocación Equipo1                 /*Provocación a jugadores suplentes*/
            Mutación Equipo2                    /*Mutación a jugadores titulares*/
            Sustitución Equipo2                 /*Sustitución a jugadores suplentes*/
        SINO
            Imitación Equipo2                   /*Imitación a jugadores titulares*/
            Provocación Equipo2                 /*Provocación a jugadores suplentes*/
            Mutación Equipo1                    /*Mutación a jugadores titulares*/
            Sustitución Equipo1                 /*Sustitución a jugadores suplentes*/
    FIN PARA
    Retornar mejor Solución G_best
FIN SLC
```

Figura 3-3 Esquema Algoritmo SLC-SingleDocSum

En el procedimiento original de *imitación* (ver Figura 3-4) los jugadores titulares (x_i) del equipo ganador tratan de imitar al mejor jugador de la liga (G_{best}) a través de una probabilidad y luego de acuerdo a otra probabilidad se le realiza un ajuste. Esto cambia para que solo se realice uno de los dos pasos anteriores, tratar de imitar el mejor jugador de la liga o realizar el ajuste.

```
INICIO Imitación                                    /*Imitación del jugador x_i */
    PARA i = 0 HASTA n                              /*Número de jugadores titulares por equipo*/
        PARA j = 0 HASTA N                          /*Número de oraciones*/
            SI aleatorio < probabilidad1 (HCMR)     /*Probabilidad de aplicar imitación hacia G_best,j*/
                y_i,j = G_best,j                    /*Imitación de la posición j de x_i hacia G_best,j*/
            SINO
                SI aleatorio < probabilidad2 (PAR)          /*Probabilidad de aplicar ajuste*/
                    y_i,j = Ajuste x_i,j            /*Ajuste de la posición j de x_i*/
        FIN PARA
        Evaluar y_i                                 /*Evaluar la calidad del nuevo jugador i*/
        SI Aptitud y_i > Aptitud x_i                /*Si y_i es mejor que su predecesora x_i*/
            x_i = y_i                               /*Actualizar su predecesora*/
        SINO
            PARA j = 0 HASTA N                      /*Número de oraciones*/
                SI aleatorio < probabilidad1 (HCMR)         /*Probabilidad de aplicar imitación hacia p_i*/
                    y_i,j = p_i,j                   /*Imitación de la posición j de x_i hacia p_i*/
                SINO
                    SI aleatorio < probabilidad2 (PAR)      /*Probabilidad de aplicar ajuste*/
                        y_i,j = Ajuste x_i,j        /*Ajuste de la posición j de x_i*/
            FIN PARA
            Evaluar y_i                             /*Evaluar la calidad de nuevo jugador i*/
            SI Aptitud y_i > Aptitud x_i            /*Si y_i es mejor que su predecesora x_i */
                x_i = y_i                           /*Actualizar su predecesora*/
        Actualizar p_i                              /*Actualizar la mejor solución local*/
        Actualizar G_best                           /*Actualizar la mejor solución global*/
    FIN PARA
FIN Imitación
```

Figura 3-4 Esquema Procedimiento de Imitación

Para el caso de $Provocación$ (ver Figura 3-5) sigue los mismos pasos de imitación donde se realiza un cálculo de provocación hacia el mejor jugador de la liga (G_{best}), en caso de no ser mejor se realiza la provocación hacia el mejor jugador del equipo (p_i) y si el anterior paso tampoco arroja mejores resultados se toma como última opción la provocación hacia un jugador aleatorio del equipo. El cambio que se realizó a este procedimiento es la adición de una probabilidad para ser utilizado solo en algunos jugadores suplentes.

```
INICIO Provocación
        PARA i = 0 HASTA n                          /*Número de jugadores suplentes por equipo*/
            SI aleatorio < probabilidad             /*Probabilidad para aplicar provocación*/
                PARA j = 0 HASTA N                  /*Número de oraciones*/
                    y_{i,j} = Provocación G_{best,j}  /*Provocación de la posición j de x_i hacia G_{best,j}*/
                FIN PARA
                Evaluar y_i                         /*Evaluar la calidad del nuevo jugador i*/
                SI Aptitud y_i > Aptitud x_i        /*Si y_i es mejor que su predecesora x_i */
                    x_i = y_i                       /*Actualizar su predecesora*/
                SINO
                    PARA j = 0 HASTA N              /*Número de oraciones*/
                        y_{i,j} = Provocación p_{i,j}  /*Provocación de la posición j de x_i hacia p_{i,j}*/
                    FIN PARA
                    Evaluar y_i                     /*Evaluar la calidad del nuevo jugador i*/
                    SI Aptitud y_i > Aptitud x_i    /*Si y_i es mejor que su predecesora x_i */
                        x_i = y_i                   /*Actualizar su predecesora*/
                    SINO
                        PARA j = 0 HASTA N              /*Número de oraciones*/
                            y_{i,j} = Provocación alea_{i,j}  /*Provocación de la posición j de x_i hacia alea_{i,j}*/
                        FIN PARA
                        Evaluar y_i                 /*Evaluar la calidad del nuevo jugador i*/
                        SI Aptitud y_i > Aptitud x_i  /*Si y_i es mejor que su predecesora x_i */
                            x_i = y_i               /*Actualizar su predecesora*/
                Actualizar p_i                      /*Actualizar la mejor solución local*/
                Actualizar G_best                   /*Actualizar la mejor solución global*/
        FIN PARA
 FIN Provocación
```

Figura 3-5 Esquema Procedimiento de Provocación

Este nuevo procedimiento de *mutación* (ver Figura 3-6) utiliza la mutación de intercambio para incluir conocimiento del problema, la cual consiste en seleccionar del vector solución la peor oración que se encuentre activa (valor en uno) en el resumen basándose en el *Criterio para deshabilitar una oración* y apagarla (valor en cero), luego selecciona de forma aleatoria una posible oración del documento que no se encuentre dentro de las oraciones del resumen y activarla (valor en uno).

```
INICIO Mutación
        PARA i = 0 HASTA n                          /*Número de jugadores titulares por equipo*/
            y_i = mutación x_i                      /*Mutación del jugador titular*/
            Evaluar y_i                             /*Evaluar la calidad del jugador i*/
            SI Aptitud y_i > Aptitud x_i            /*Si y_i es mejor que su predecesora x_i */
                x_i = y_i                           /*Actualizar su predecesora*/
            Actualizar p_i                          /*Actualizar la mejor solución local*/
            Actualizar G_best                       /*Actualizar la mejor solución global*/
        FIN PARA
 FIN Mutación
```

Figura 3-6 Esquema Procedimiento de Mutación

Por último el nuevo procedimiento de *sustitución* (ver Figura 3-7) realiza un cruce entre dos jugadores (vector solución), estos se seleccionan de forma aleatoria entre los jugadores suplentes del equipo perdedor, de los cuales se generan dos nuevas soluciones, en caso de que estas dos nuevas soluciones sean mejores que los dos peores titulares se reemplazan por estos.

```
INICIO Sustitución
        Seleccionar x_a y x_b                    /*Seleccionar dos jugadores aleatoriamente*/
        y_a = cruce (x_a, x_b)                   /*Cruce entre los jugadores */
        y_b = cruce (x_b, x_a)                   /*Cruce entre los jugadores */
        Evaluar y_a                              /*Evaluar la calidad del jugador a*/
        Evaluar y_b                              /*Evaluar la calidad del jugador b*/
        SI Aptitud y_a > (Aptitud x_a || Aptitud x_b)  || Aptitud y_b > (Aptitud x_a || Aptitud x_b)
            Actualizar x_a,x_b                   /*Actualizar predecesores*/
        Actualizar p_i                           /*Actualizar la mejor solución local*/
        Actualizar G_best                        /*Actualizar la mejor solución global*/
FIN Sustitución
```

Figura 3-7 Esquema Procedimiento de Sustitución

Una vez que los procedimientos hayan creado el nuevo jugador (solución candidata), se evalúa la calidad mediante la función objetivo definida en la sección 3.1.4 y se compara con su predecesora verificando que su aptitud sea mejor para reemplazarla, de lo contrario se mantiene.

Hay que resaltar que al retornar la *mejor Solución* (G_{best}), el algoritmo SLC-SingleDocSum ordena las oraciones según el *Criterio de selección de oraciones del resumen final* para que aparezcan en el orden que se establezca en el resumen generado.

3.2 CICLO II: ADICIÓN ASCENSO DE LA COLINA

En este ciclo se establece la configuración de los parámetros del algoritmo Ascenso de la Colina y tres esquemas de optimización para adaptarlo a los algoritmos SFLA y SLC. También se afinan los parámetros del Ascenso de la Colina por cada esquema. Por último se adaptan los algoritmos SFLA y SLC al esquema que obtuvo mejor resultado el cual determina la configuración definitiva del algoritmo para este ciclo.

3.2.1 Configuración parámetros de Ascenso de la Colina

Para incluir el optimizador de ascenso de la colina en los algoritmos SFLA y SLC, es necesario agregar los parámetros definidos en la sección 3.1.3, y otros específicos correspondientes a la optimización, que son:

- Número de Optimizaciones (Opt). Determina el número de veces que el método de optimización itera ajustando una solución.

- Puntos a Optimizar (PuntosOpt). Indica el número de soluciones candidatas que serán optimizados.

3.2.2 Afinación de parámetros con Ascenso de la Colina

El proceso de afinación utiliza la configuración de la función objetivo (sección 3.1.4), la configuración de los parámetros asociados a los algoritmos que se muestra en la Tabla 6 y la configuración de los parámetros asociados al problema presentada en la Tabla 10 en los siguientes esquemas de optimización:

- Optimización del peor al inicializar. La optimización se aplica sobre la peor solución de la población inicial.

- Optimización aleatoria en cada iteración. La optimización se aplica sobre un número de soluciones de manera aleatoria al finalizar cada iteración.
- Optimización del mejor en cada iteración. La optimización se aplica sobre la mejor solución global al finalizar cada iteración.

De acuerdo a lo observado en la Tabla 11 se establece que el esquema *OptMejorAlIterar* en SFLA y el esquema *OptPeorAlInicializar* en SLC, obtienen los mejores resultados en cinco de las seis medidas, con los valores de los parámetros de SFLA y SLC presentados en la Tabla 12.

Algoritmos	Experimento	DUC2001			DUC2002		
		R1R	R2R	RSU4R	R1R	R2R	RSU4R
SFLA	PeorAlInicializar	0,45728	0,20628	0,22808	0,49006	0,23228	0,24804
	AleatorioAlIterar	**0,45734**	0,20625	0,22827	0,49025	0,23209	0,24799
	MejorAlIterar	0,45722	**0,20663**	**0,22839**	**0,49034**	**0,23258**	**0,24830**
SLC	**PeorAlInicializar**	**0,46060**	0,20771	**0,23015**	**0,48738**	**0,22994**	**0,24630**
	AleatorioAlIterar	0,46048	0,20753	0,23000	0,48718	0,22956	0,24601
	MejorAlIterar	0,46051	**0,20776**	0,23012	0,48712	0,22970	0,24611

Tabla 11 Mejor Resultado esquemas de optimización

SFLA		SLC	
Parámetro	**Valor**	**Parámetro**	**Valor**
P	20	nT	40
M	5	nF	10
It	10	nS	5
itMax	150	nI	1
Número de Optimizaciones	5	Número de Optimizaciones	6
Puntos a Optimizar	1	Puntos a Optimizar	1
Optimización	Mejor Al Iterar	Optimización	El Peor Luego de Inicializar

Tabla 12 Mejor configuración de parámetros SFLA y SLC con Ascenso Colina

3.2.3 Esquema Algoritmos

3.2.3.1 SLFA-ASC-SingleDocSum

En la Figura 3-8 **Esquema Algoritmo SFLA-ASC-SingleDocSum** se presenta el algoritmo SFLA-SingleDocSum con la adaptación de Ascenso de la Colina, ahora denominado SFLA-ASC-SingleDocSum. Esta adaptación incluye el método de $Optimizar\ mejor\ rana\ al\ iterar\ con\ Ascenso\ de\ la$ colina.

```
INICIO SFLA
        HACER
            Inicializar solución viable x_i          /*Crear rana i*/
            Evaluar x_i                              /*Evaluar la calidad de la rana i*/
        HASTA Crear P Ranas
        PARA k = 0 HASTA itMax                       /*Número de iteraciones*/
            RankearRanas                             /*Ordenar las ranas descendientemente*/
            BarajarRanas                             /*Repartir las ranas en los memeplex*/
            BusquedaLocal                            /*Generar nuevas soluciones con los saltos*/
            ReagruparRanas                           /*Reunir ranas desde los memeplex*/
            PARA j = 1 HASTA P                       /*Mutación desde la segunda rana */
                MutarRana x_j                        /*Aplicar mutación a la rana j*/
            FIN PARA
            Optimizar mejor rana con Ascenso de la Colina
        FIN PARA
        Retornar G_best
FIN SFLA
```

Figura 3-8 Esquema Algoritmo SFLA-ASC-SingleDocSum

Este método adapta el algoritmo Ascenso de la Colina optimizando la mejor rana del estanque al final de cada iteración, en este caso particular solamente se tuvo en cuenta el parámetro Opt ya que solo se optimizó una rana. El esquema de *Optimizar mejor rana con Ascenso de la Colina* se presenta en la Figura 3-9 **Procedimiento de Optimización con Ascenso de Colina**.

```
INICIO Optimizar mejor rana con Ascenso de la colina
        REPETIR
            S = copia G_best                         /*Solución candidata */
            PARA k = 0 HASTA Opt HACER               /*Número de Optimizaciones*/
                Puntuar Oraciones S
                R = DeshabilitarOracion S
                R = HabilitarOracion S
                Evaluar R
                SI Aptitud R > Aptitud S
                    S =  R
            FIN PARA
            G_best = S
        HASTA Optimizar (N*PuntosOpt) ranas
FIN Optimizar mejor rana con Ascenso de la colina
```

Figura 3-9 Procedimiento de Optimización con Ascenso de Colina

Para iniciar la optimización, se hace una copia de la mejor rana del estanque (G_{best}) y se establece como solución candidata (S). Después, se evalúa la solución candidata S ($PuntuarOracion\ S$) y a continuación se genera una nueva solución candidata R deshabilitando una oración de forma aleatoria de entre las peores de la solución candidata y habilitando la mejor oración de la oraciones posibles (ver sección 4.5.1) que tenga la misma o menor cantidad de palabras que la oración que se acaba de deshabilitar ($HabilitarOracion\ S$), las oraciones no activas que pueden ser habilitadas están ordenadas de acuerdo a la característica establecida en el *Criterio de Deshabilitar una oración*. Luego, se evalúa la calidad de la nueva solución R ($Evaluar\ R$) con la función objetivo definida en la sección 3.1.4 y se comparan los valores de aptitud de R y S, de modo que si la aptitud de R es mayor que la de S, R reemplaza a S. Este procedimiento se realiza Opt veces y finaliza reemplazando el valor de G_{best} con S.

3.2.3.2 SLC-ASC-SingleDocSum

El esquema del algoritmo SLC-ASC-SingleDocSum presentado en la Figura 3-10 es la adaptación del algoritmo Ascenso de la Colina al algoritmo SLC-SingleDocSum. El algoritmo propuesto en este ciclo difiere respecto al SLC-SingleDocSum (ver Figura 3-3) en que agrega el procedimiento $Optimizar\ el\ peor\ jugador\ al\ inicializar\ con\ Ascenso\ de\ la$ colina.

```
INICIO SLC
        HACER
            Inicializar solución viable x_i          /*Crear jugador i*/
            Evaluar x_i                              /*Evaluar la calidad del jugador i*/
            Actualizar G_best                        /*Actualizar la mejor solución global*/
        HASTA Crear n Jugadores
        Ordenar                                      /*Ordenar jugadores descendentemente*/
        Asignar                                      /*Asignar jugadores a los equipos*/
        Instanciar p_i                               /*Instanciar mejor solución local*/
        Optimizar peor jugador con Ascenso de la Colina
        PARA k = 0 HASTA Nenfre                      /*Número de enfrentamientos*/
            Selección equipo 1y2                     /*Selección de equipos a enfrentar*/
            SI Ganador Equipo1
                Imitación Equipo1                    /*Imitación a jugadores titulares*/
                Provocación Equipo1                  /*Provocación a jugadores suplentes*/
                Mutación Equipo2                     /*Mutación a jugadores titulares*/
                Sustitución Equipo2                  /*Sustitución a jugadores suplentes*/

            SINO
                Imitación Equipo2                    /*Imitación a jugadores titulares*/
                Provocación Equipo2                  /*Provocación a jugadores suplentes*/
                Mutación Equipo1                     /*Mutación a jugadores titulares*/
                Sustitución Equipo1                  /*Sustitución a jugadores suplentes*/
        FIN PARA
        Retornar mejor Solución G_best
  FIN SLC
```

Figura 3-10 Esquema Algoritmo SLC-ASC-SingleDocSum

El esquema de *Optimizar peor jugador con Ascenso de la Colina* se presenta en la Figura 3-11. Este procedimiento adapta el algoritmo Ascenso de la Colina para optimizar el peor jugador luego de ser generado. A continuación selecciona el peor jugador de la liga y lo establece como solución candidata S.

Para iniciar la optimización, se evalúan las oraciones activas en la solución candidata S y las oraciones posibles que no están activas, de acuerdo a la característica establecida en el *Criterio de Deshabilitar una oración*. A continuación se genera una nueva solución candidata R deshabilitando la peor oración de la solución candidata y habilitando la mejor oración de la frases posibles que tenga menor o igual número de palabras que la oración que se acaba de deshabilitar, por último se evalúa la calidad de la nueva solución R con la función objetivo definida en la sección 3.1.4 y se compara con la evaluación de S, de este modo, si la evaluación de R es mayor que la evaluación de S, S se actualiza con R, de lo contrario S se mantiene. Así, S se optimiza Opt veces para al final actualizar el peor jugador x_i.

```
INICIO Optimizar peor jugador con Ascenso de la colina
        REPETIR
            xi = peor jugador de la liga
            S = copia xi                          /*Solución candidata*/
            PARA k = 0 HASTA Opt HACER            /*Número de Optimizaciones*/
                Puntuar Oraciones S
                R = Deshabilitar peor oración S
                R = Habilitar la mejor oración S
                Evaluar R                         /*Evaluar la calidad del jugador R*/
                SI Aptitud R > Aptitud S
                    S = R
            FIN PARA
            xi = S
        HASTA Optimizar (N*PuntosOpt) jugadores
 FIN Optimizar peor jugador con Ascenso de la colina
```

Figura 3-11 Procedimiento de Optimización con Ascenso de Colina

3.3 CICLO III: ADICIÓN DE UNA MEMORIA TABÚ

En este ciclo se adiciona una memoria tabú en los algoritmos SFLA y SLC, definiendo la configuración de los parámetros asociados a la memoria tabú. Además se realiza el afinamiento de los parámetros de la memoria tabú para dos tipos de memoria. Por último se adapta la memoria tabú que obtuvo mejores resultados en los algoritmos SFLA y SLC para definir la configuración definitiva del algoritmo para este ciclo.

3.3.1 Configuración de parámetros de la memoria tabú

Los parámetros definidos para esta configuración, son los descritos en la sección 3.1.3 y los específicos de la memoria tabú son:

- Tenure. Es el tiempo o número de iteraciones que una solución candidata permanece en la lista tabú.

- Explícita global. Comportándose como una lista FIFO (First In, First Out), esta guarda las soluciones visitadas.

- Explícita por conjunto. Almacena por cada conjunto de soluciones (rana en SFLA, equipo en SLC) las soluciones visitadas durante su explotación. También funciona como una lista FIFO.

3.3.2 Afinación de parámetros de la memoria tabú

Este proceso utiliza la configuración final de la función objetivo (3.1.4) y la configuración de los parámetros asociados al problema (Tabla 10), donde se evalúan los algoritmos SFLA y SLC adaptado a cada tipo de memoria tabú, sin optimizar y optimizando.

Sin Optimización. Se realiza de igual forma que la afinación de parámetros de los algoritmos (3.1.6), diferenciándose al introducir el tipo de memoria tabú. A continuación se presentan los resultados obtenidos por cada tipo de memoria tabú:

- Explícita global. El valor del parámetro $tenure$ para este tipo de memoria varía en incrementos de uno, en el rango entre uno y veinte. El valor de $tenure$ se establece en

ocho para SFLA y en quince para SLC para el tipo Explícita Global ya que obtuvieron los mejores resultados como se observa en la Tabla 13.

Algoritmo	Experimento	DUC2001			DUC2002		
		R1R	R2R	RSU4R	R1R	R2R	RSU4R
SFLA	**8**	**0,45761**	**0,20665**	**0,22836**	**0,49037**	**0,23217**	**0,24810**
	9	0,45717	0,20654	0,22806	0,49005	0,23186	0,24780
	10	0,45728	0,20639	0,22810	0,49029	0,23214	0,24799
SLC	10	0,45984	0,20706	0,22956	0,48698	0,22939	0,24591
	15	**0,45997**	**0,20707**	**0,22961**	0,48679	0,22918	0,24571
	20	0,45990	0,20700	0,22951	**0,48708**	**0,22959**	**0,24603**

Tabla 13 Mejores Resultados Explícita Global

- Explícita por conjunto. El valor del parámetro $tenure$ está en el rango entre uno y diez con incrementos de uno para SFLA, y entre cincuenta y ochenta con incrementos de cinco para SLC. Los mejores resultados obtenidos son presentados en la Tabla 14, mostrando que el valor de $tenure$ para SFLA es de cuatro y para SLC es cincuenta para el tipo Explícita por conjunto.

Algoritmo	Experimento	DUC2001			DUC2002		
		R1R	R2R	RSU4R	R1R	R2R	RSU4R
SFLA	**4**	**0,45748**	**0,20668**	**0,22834**	0,49014	0,23189	0,24791
	5	0,45719	0,20644	0,22798	**0,49026**	**0,23201**	**0,24795**
	6	0,45709	0,20619	0,22787	0,49004	0,23184	0,24782
SLC	**50**	0,46030	**0,20710**	0,22964	0,48737	**0,22962**	**0,24609**
	75	**0,46053**	0,20705	**0,22971**	**0,48744**	0,22954	0,24603
	80	0,46031	0,20683	0,22946	0,48710	0,22924	0,24756

Tabla 14 Mejores Resultados Explícita por conjunto

Una vez finalizado este proceso, se presentan los resultados obtenidos para cada tipo de memoria tabú en la Tabla 15, donde se observa que en SFLA la memoria *Explícita Global* y en SLC la memoria *Explícita por Conjunto (equipo),* obtienen los mejores resultados en cinco de las seis medidas.

Algoritmo	Experimento	DUC2001			DUC2002		
		R1R	R2R	RSU4R	R1R	R2R	RSU4R
SFLA	**Explícita Global**	**0,45761**	0,20665	**0,22836**	**0,49037**	**0,23217**	**0,24810**
	Explícita por conjunto (rana)	0,45748	**0,20668**	0,22834	0,49014	0,23189	0,24791
SLC	Explícita Global	0,45997	**0,20707**	0,22961	0,48679	0,22918	0,24571
	Explícita por conjunto (equipo)	**0,46053**	0,20705	**0,22971**	**0,48744**	**0,22954**	**0,24603**

Tabla 15 Mejor Resultado de cada esquema de la memoria tabú

Con Optimización: se utiliza Ascenso de la Colina como método de optimización de forma similar a la afinación de parámetros de SFLA y SLC (secciones 3.2.2), pero adicionando los parámetros del tipo de memoria. Al terminar el proceso para cada uno de los algoritmos, se selecciona el mejor resultado que obtuvo cada uno de los esquemas de la memoria tabú después de su evaluación, tal y como se presentan en la Tabla 16.

Algoritmo	Experimento	DUC2001			DUC2002		
		R1R	R2R	RSU4R	R1R	R2R	RSU4R
SFLA	**Explícita Global - MejorAlIterar**	**0,45746**	**0,20678**	**0,22836**	**0,49040**	**0,23214**	**0,24804**
	Explícita por rana - MejorAlIterar	0,45698	0,20611	0,22788	0,49030	0,23192	0,24792
SLC	**Explícita Global - PeorAlInicializar**	**0,46031**	**0,20712**	**0,22959**	0,48695	**0,22954**	**0,24592**
	Explícita por equipo - PeorAlInicializar	0,46017	0,20692	0,22953	**0,48704**	0,22936	0,24587

Tabla 16 Mejores Resultados SFLA y SLC con Memoria tabú y Ascenso de la Colina

En la Tabla 16 se observa que para SFLA el algoritmo Ascenso de la Colina con el esquema de optimización *MejorAlIterar* y una memoria *Explicita Global* obtiene los mejores resultados en todas las medidas y para SLC el algoritmo Ascenso de la Colina con el esquema de optimización *PeorAlInicializar* y una memoria *Explicita Global* obtiene los mejores resultados en cinco de las seis medidas.

Al finalizar se presenta en la Tabla 17 el mejor resultado de cada una de las Tabla 15 y Tabla 16. Se observa que en SFLA la memoria *Explicita Global,* sin aplicar optimización, obtiene mejores resultados en cuatro de las seis medidas ROUGE y para SLC la memoria *Explicita por Conjunto (equipo),* sin aplicar optimización, obtiene los mejores resultados en cinco de las seis medidas ROUGE.

Algoritmo	Experimento	DUC2001			DUC2002		
		R1R	R2R	RSU4R	R1R	R2R	RSU4R
SFLA	**Explícita Global**	**0,45761**	0,20665	**0,22836**	0,49037	**0,23217**	**0,24810**
	Explícita Global - MejorAlIterar	0,45746	**0,20678**	0,22836	**0,49040**	0,23214	0,24804
SLC	**Explícita por conjunto (equipo)**	**0,46053**	0,20705	**0,22971**	**0,48744**	**0,22954**	**0,24603**
	Explícita Global - PeorAlInicializar	0,46031	**0,20712**	0,22959	0,48695	0,22954	0,24592

Tabla 17 Mejores resultados SFLA y SLC con memoria tabú

Los valores de los parámetros de los algoritmos SFLA y SLC que corresponden a la mejor configuración encontrada en este ciclo, son los presentados en la Tabla 18.

SFLA		SLC	
Parámetro	**Valor**	**Parámetro**	**Valor**
P	20	nT	40
M	5	nF	10
It	10	nS	5
itMax	150	nI	1
Número de Optimizaciones	0	Número de Optimizaciones	0
Puntos a Optimizar	0	Puntos a Optimizar	0
Algoritmo de Optimización Local	Ninguno	Algoritmo de Optimización Local	Ninguno
Optimización	Ninguna	Optimización	Ninguna
Memoria Tabú	Explícita Global	Memoria Tabú	Explícita por Conjunto

Tabla 18 Mejor Combinación de parámetros de los algoritmos con memoria tabú

3.3.3 Esquema Algoritmos

3.3.3.1 SFLA-MT-SingleDocSum

El esquema del algoritmo SFLA-MT-SingleDocSum presentado en la Figura 3-12 es la adaptación de una memoria tabú tipo explícita Global que permite soluciones Repetidas en el algoritmo SFLA-SingleDocSum. El algoritmo propuesto en este ciclo difiere respecto al SFLA-SingleDocSum (ver Figura 3-1) implementando una lista tabú $(Inicializar\ LT)$ que permite almacenar las soluciones que se crean a partir de su ajuste. Luego de crear y evaluar cada rana (x_i) se agrega a la lista tabú (LT). Además en cada procedimiento de salto dentro de la búsqueda local (ver Figura 3-13), se agrega un ciclo condicional que verifica $(VerificarMemoriTabu\ aux_i)$ si la solución creada (y_i) se encuentra en la lista tabú (LT), de manera que si la solución (y_i) está en la lista $(Retorna\ Verdadero)$ se deben repetir las operaciones hasta obtener una nueva solución que sea diferente.

```
INICIO SFLA
    Tenure = 8                                  /*Tamaño máximo de la tabú*/
    Inicializar LT                              /*Crear Lista Tabú */
    HACER
        Inicializar solución viable x_i         /*Crear rana i*/
        Evaluar x_i                             /*Evaluar la calidad de la rana i*/
        Agregar x_i dentro de LT                /*Agrega la solución i a la lista tabú*/
    HASTA Crear P Ranas
    PARA k = 0 HASTA itMax                      /*Número de iteraciones*/
        RankearRanas                            /*Ordenar las ranas descendientemente*/
        BarajarRanas                            /*Repartir las ranas en los memeplex*/
        BusquedaLocal                           /*Generar nuevas soluciones con los saltos*/
        ReagruparRanas                          /*Reunir ranas desde los memeplex*/
        PARA j = 1 HASTA P                      /*Mutación desde la segunda rana */
            MutarRana x_j                       /*Aplicar mutación a la rana j*/
        FIN PARA
    FIN PARA
    Retornar G_best
FIN SFLA
```

Figura 3-12 Esquema Algoritmo SFLA-MT-SingleDocSum

```
INICIO Búsqueda Local
    PARA im = 0 HASTA m                         /*Número de Memeplex*/
        PARA itm = 0 HASTA it                   /*Número de iteraciones en cada memeplex*/
            Determinar x_w, x_b y x_g           /*Determinar la peor y mejor rana*/
            REPETIR
                y_i = Saltar x_b                /*Saltar usando la mejor local*/
                Agregar y_i dentro de LT        /*Agrega y_i a la lista tabú*/
            HASTA VerificarMemoriaTabu y_i = Falso /*Verifica si la solución es Tabú*/
            Reparacion2                         /*Habilitar oración aleatoriamente*/
            Evaluar y_i
            SI Aptitud y_i > Aptitud x_w
                x_w = y_i                       /*Actualizar la peor rana*/
            SINO
                REPETIR
                    y_i = Saltar x_g            /*Saltar usando la mejor global*/
                    Agregar y_i dentro de LT    /*Agrega y_i a la lista tabú*/
                HASTA VerificarMemoriaTabu y_i = Falso /*Verifica si la solución es Tabú*
                Reparacion2                     /*Habilitar oración aleatoriamente*/
                Evaluar y_i

                SI Aptitud y_i > Aptitud x_w
                    x_w = y_i                   /*Actualizar la peor rana*/
                SINO
                    x_w = Inicializar y_i       /*Inicializar x_w*/
                    Reparacion1                 /*Habilitar mejor oración*/
                    Evaluar y_i
                    SI Aptitud y_i > Aptitud x_g
                        x_g = y_i               /*Actualizar la mejor rana global*/
        FIN PARA
    FIN PARA
FIN Búsqueda Local
```

Figura 3-13 Esquema Procedimiento Búsqueda Local con memoria tabú

3.3.3.2 SLC-MT-SingleDocSum

El esquema del algoritmo SLC-MT-SingleDocSum presentado en la Figura 3-14 es la adaptación de una memoria tabú tipo explícita por conjunto (equipo) en el algoritmo SLC-SingleDocSum. El algoritmo propuesto en este ciclo difiere respecto al SLC-SingleDocSum (ver Figura 3-3) en que implementa listas tabú $(Inicializar\ LT_j)$ que permiten almacenar por cada equipo (z_i) las soluciones que se crean a partir de su ajuste. Luego de crear y evaluar el jugador (x_i) se agrega a la lista tabú (LT_j). Además en los procedimientos de imitación, provocación, mutación y sustitución presentado en los esquemas de la Figura 3-15, Figura 3-16, Figura 3-17 y Figura 3-18 respectivamente, se agrega un ciclo condicional que verifica $(VericarMemoriaTabu(y_i))$ si la solución creada (y_i) no se encuentra en la lista tabú (LT_j) del equipo (z_i), de manera que, si la solución (y_i) está en la lista tabú $(Retorna\ Verdadero)$ no se tiene en cuenta como nueva solución por lo que se repiten las operaciones para obtener uno diferente, si por el contrario, la solución no está en la lista tabú es tomada como nueva solución y se agrega a la lista tabú (LT_j).

```
INICIO SLC
    Tenure = 60                          /*Tamaño máximo de la tabú*/
    Inicializar LT_j                     /*Crear lista tabú del equipo j*/
    HACER
        Inicializar solución viable xi   /*Crear jugador i*/
        Evaluar x_i                      /*Evaluar la calidad del jugador i*/
        Agregar x_i dentro de LT_j       /*Agrega la solución i a la lista tabú j*/
        Actualizar G_best                /*Actualizar la mejor solución global*/
    HASTA Crear n Jugadores
    Ordenar                              /*Ordenar jugadores descendentemente*/
    Asignar                              /*Asignar jugadores a los equipos*/
    Instanciar p_i                       /*Instanciar mejor solución local*/
    PARA k = 0 HASTA Nenfre              /*Número de enfrentamientos*/
        Selección equipo 1y2             /*Selección de equipos a enfrentar*/
        SI Ganador Equipo1
            Imitación Equipo1            /*Imitación a jugadores titulares*/
            Provocación Equipo1          /*Provocación a jugadores suplentes*/
            Mutación Equipo2             /*Mutación a jugadores titulares*/
            Sustitución Equipo2          /*Sustitución a jugadores suplentes*/
        SINO
            Imitación Equipo2            /*Imitación a jugadores titulares*/
            Provocación Equipo2          /*Provocación a jugadores suplentes*/
            Mutación Equipo1             /*Mutación a jugadores titulares*/
            Sustitución Equipo1          /*Sustitución a jugadores suplentes*/
    FIN PARA
    Retornar mejor Solución G_best
    FIN SLC
```

Figura 3-14 Esquema Algoritmo SLC-MT-SingleDocSum

```
INICIO Imitación                                    /*Imitación del jugador x_i */
    PARA i = 0 HASTA n                              /*Número de jugadores titulares por equipo*/
        REPETIR
            PARA j = 0 HASTA N                      /*Número de oraciones*/
                SI aleatorio < probabilidad1 (HCMR)     /*Probabilidad de aplicar imitación hacia G_best,j*/
                    y_i,j = G_best,j                /*Imitación de la posición j de x_i hacia G_best,j*/
                SINO
                    SI aleatorio < probabilidad2 (PAR)  /*Probabilidad de aplicar ajuste*/
                        y_i,j = Ajuste x_i,j        /*Ajuste de la posición j de x_i*/
            FIN PARA
        HASTA VerificarMemoriaTabu y_i = Falso      /*Verifica si la solución es Tabú*/
        Evaluar y_i                                 /*Evaluar la calidad del nuevo jugador i*/
        SI Aptitud y_i > Aptitud x_i                /*Si y_i es mejor que su predecesora x_i*/
            x_i = y_i                               /*Actualizar su predecesora*/
        SINO
            REPETIR
                PARA j = 0 HASTA N                  /*Número de oraciones*/
                    SI aleatorio < probabilidad1 (HCMR)  /*Probabilidad de aplicar imitación hacia p_i*/
                        y_i,j = p_i,j               /*Imitación de la posición j de x_i hacia p_i*/
                    SINO
                        SI aleatorio < probabilidad2 (PAR)   /*Probabilidad de aplicar ajuste*/
                            y_i,j = Ajuste x_i,j    /*Ajuste de la posición j de x_i*/
                FIN PARA
            HASTA VerificarMemoriaTabu y_i = Falso  /*Verifica si la solución es Tabú*/
            Evaluar y_i                             /*Evaluar la calidad de nuevo jugador i*/
            SI Aptitud y_i > Aptitud x_i            /*Si y_i es mejor que su predecesora x_i */
                x_i = y_i                           /*Actualizar su predecesora*/
        Actualizar p_i                              /*Actualizar la mejor solución local*/
        Actualizar G_best                           /*Actualizar la mejor solución global*/
    FIN PARA
FIN Imitación
```

Figura 3-15 Esquema Procedimiento de Imitación con memoria tabú

```
INICIO Provocación
        PARA i = 0 HASTA n                          /*Número de jugadores suplentes por equipo*/
            SI aleatorio < probabilidad             /*Probabilidad para aplicar provocación*/
                REPETIR
                    PARA j = 0 HASTA N              /*Número de oraciones*/
                        y_{i,j} = Provocación G_{best,j}   /*Provocación de la posición j de x_i hacia G_{best,j}*/
                    FIN PARA
                HASTA VerificarMemoriaTabu y_i = Falso     /*Verifica si la solución es Tabú*/
                Evaluar y_i                         /*Evaluar la calidad del nuevo jugador i*/
                SI Aptitud y_i > Aptitud x_i        /*Si y_i es mejor que su predecesora x_i */
                    x_i = y_i                       /*Actualizar su predecesora*/
                SINO
                    REPETIR
                        PARA j = 0 HASTA N          /*Número de oraciones*/
                            y_{i,j} = Provocación p_{i,j}   /*Provocación de la posición j de x_i hacia p_{i,j}*/
                        FIN PARA
                    HASTA VerificarMemoriaTabu y_i = Falso     /*Verifica si la solución es Tabú*/
                    Evaluar y_i                     /*Evaluar la calidad del nuevo jugador i*/
                    SI Aptitud y_i > Aptitud x_i    /*Si y_i es mejor que su predecesora x_i */
                        x_i = y_i                   /*Actualizar su predecesora*/
                    SINO
                        REPETIR
                            PARA j = 0 HASTA N      /*Número de oraciones*/
                                y_{i,j} = Provocación alea_{i,j}/*Provocación de la posición j de x_i hacia alea_{i,j}*/
                            FIN PARA
                        HASTA VerificarMemoriaTabu y_i = Falso     /*Verifica si la solución es Tabú*/
                        Evaluar y_i                 /*Evaluar la calidad del nuevo jugador i*/
                        SI Aptitud y_i > Aptitud x_i /*Si y_i es mejor que su predecesora x_i */
                            x_i = y_i               /*Actualizar su predecesora*/
                Actualizar p_i                      /*Actualizar la mejor solución local*/
                Actualizar G_best                   /*Actualizar la mejor solución global*/
        FIN PARA
 FIN Provocación
```

Figura 3-16 Esquema Procedimiento de Provocación con memoria tabú

```
INICIO Mutación
        PARA i = 0 HASTA n                          /*Número de jugadores titulares por equipo*/
            REPETIR
                y_i = mutación x_i                  /*Mutación del jugador titular*/
            HASTA VerificarMemoriaTabu y_i = Falso     /*Verifica si la solución es Tabú*/
            Evaluar y_i                             /*Evaluar la calidad del jugador i*/
            SI Aptitud y_i > Aptitud x_i            /*Si y_i es mejor que su predecesora x_i */
                x_i = y_i                           /*Actualizar su predecesora*/
            Actualizar p_i                          /*Actualizar la mejor solución local*/
            Actualizar G_best                       /*Actualizar la mejor solución global*/
        FIN PARA
 FIN Mutación
```

Figura 3-17 Esquema Procedimiento de Mutación con memoria tabú

INICIO Sustitución
Seleccionar x_a y x_b /*Seleccionar dos jugadores aleatoriamente*/
REPETIR
y_a = cruce (x_a, x_b) /*Cruce entre los jugadores */
y_b = cruce (x_b, x_a) /*Cruce entre los jugadores */
HASTA *VerificarMemoriaTabu* y_a, y_b = *Falso* /*Verifica si y_a y y_b son Tabú*/
Evaluar y_a /*Evaluar la calidad del jugador a*/
Evaluar y_b /*Evaluar la calidad del jugador b*/
SI Aptitud y_a > (Aptitud x_a || Aptitud x_b) || Aptitud y_b > (Aptitud x_a || Aptitud x_b)
Actualizar x_a,x_b /*Actualizar predecesores*/
Actualizar p_i /*Actualizar la mejor solución local*/
Actualizar G_{best} /*Actualizar la mejor solución global*/
FIN Sustitución

Figura 3-18 Esquema Procedimiento de Sustitución con memoria tabú

Finalizados los tres ciclos de diseño se presenta en la Tabla 19 el mejor resultado obtenido por los algoritmos propuestos en cada ciclo, donde se observa que en SFLA el algoritmo con mejor desempeño fue SFLA-MT-SingleDocSum debido a que mostró los mejores resultados en tres de seis medidas (ROUGE-1 Y ROUGE-2 sobre DUC2001, y ROUGE-1 sobre DUC 2002), las cuales son tres de las cuatro medidas que se tienen en cuenta al momento de compararse con el estado del arte y en SLC el algoritmo con mejor desempeño fue SLC-ASC-SingleDocSum ya que obtuvo los mejores resultados en tres de las seis medidas, por lo tanto se seleccionan como propuestas definitivas.

Algoritmo	Experimento	DUC2001			DUC2002		
		R1R	R2R	RSU4R	R1R	R2R	RSU4R
SFLA	SFLA-SingleDocSum	0,45643	0,20592	0,22765	0,48990	0,23215	0,24793
	SFLA-ASC-SingleDocSum	0,45722	0,20663	**0,22839**	0,49034	**0,23258**	**0,24830**
	SFLA-MT-SingleDocSum	**0,45761**	**0,20665**	0,22836	**0,49037**	0,23217	0,24810
SLC	SLC-SingleDocSum	0,46045	0,20757	0,23003	0,48740	**0,22997**	**0,24632**
	SLC-ASC-SingleDocSum	**0,46060**	**0,20771**	**0,23015**	0,48738	0,22994	0,24630
	SLC-MT-SingleDocSum	0,46053	0,20705	0,22971	**0,48744**	0,22954	0,24603

Tabla 19 Mejor Resultado de cada Ciclo

Capítulo 4

4 ALGORITMOS PROPUESTOS: SFLA-MT-SINGLEDOCSUM y SLC-ASC-SINGLEDOCSUM

A continuación, se describen los algoritmos propuestos para el problema de generación automática de resúmenes extractivos de un documento llamados SFLA-MT-SingleDocSum y SLC-ASC-SingleDocSum, los cuales están basados en los algoritmos SFLA de Muzaffar Eusuff [75] adaptando una memoria tabú explicita y SLC de Naser Moosavian [77] adaptando la búsqueda local Ascenso de la Colina. En las siguientes secciones se presenta: la representación de las soluciones candidatas, la función objetivo utilizada para evaluar la calidad de los resúmenes generados, la descripción de los componentes de los algoritmos SFLA-MT-SingleDocSum y SLC-ASC-SingleDocSum, y el esquema general del proceso para generar un resumen.

4.1 REPRESENTACIÓN DE LAS SOLUCIONES

Al trabajar con algoritmos metaheurísticos es necesario representar los objetos solución dentro del espacio de búsqueda, esta representación se describe como el mecanismo que establece la estructura que permite mapear cada punto del espacio de búsqueda en una solución candidata (codificación) y obtener desde la solución candidata el punto del espacio que está representando (descodificación). De esta manera se usa la codificación en los procedimientos de construcción y ajuste de las soluciones, y la decodificación en la evaluación de las soluciones candidatas y en la transformación de la solución final [70].

Teniendo en cuenta lo anterior, SFLA-MT-SingleDocSum y SLC-ASC-SingleDocSum realizan la representación de las soluciones a través de la codificación binaria. La codificación binaria usa un vector binario de tamaño (n), donde (n) es el número de oraciones que conforman un documento representado como $\{s_1, s_2, \dots, s_n\}$ donde cada elemento del vector toma un valor de uno o cero que indica la presencia o ausencia de una oración del documento en el resumen. Por ejemplo, si un documento tiene diez oraciones $(n = 10)$, la representación de un vector solución puede ser la siguiente $[1,0,1,0,1,0,1,0,1,0]$, en este caso, indicando que las oraciones primera, tercera, quinta, séptima y novena están habilitadas para ser parte de la solución candidata (resumen candidato) [44], [78].

La decodificación consiste en buscar las oraciones ordenadamente dentro del vector solución donde tengan el valor uno, que indican la presencia de la oración en el resumen generado. De esta forma, la evaluación de una solución candidata y la conformación del resumen final se realizan con las oraciones obtenidas después de la decodificación [44].

4.2 FUNCIÓN OBJETIVO

La función objetivo que se utiliza en los algoritmos SFLA-MT-SingleDocSum y SLC-ASC-SingleDocSum está conformada por características que dependen de la oración: *posición en el documento*, *longitud* (medida en palabras) y *relación con el título*; y otras que dependen

de la similitud con las oraciones del resumen candidato: *cohesión*, que es la similitud entre las oraciones del resumen y *cobertura*, que es la similitud de las oraciones del resumen con respecto al documento. Estas características fueron calculadas con las ecuaciones: *Posición1* (3.1), *Longitud1* (3.4), *Relación con el Título1* (3.7), *Cohesión* (3.10) y *Cobertura* (3.12), tal y como se presenta en la Sección 3.1.1. De este modo la función objetivo es calculada de acuerdo a la Ecuación (4.1), con su correspondiente fórmula matemática en la Ecuación (4.2) y los pesos de cada característica por algoritmo en la Tabla 20.

$$f(x) = \alpha * Posición1 + \beta * Longitud1 + \gamma * Relación\ con\ el\ título1 + \mu * Cohesión + \rho * Cobertura \tag{4.1}$$

$$f(x) = \sum_{\forall S_i \in Summary} \left[\alpha * \frac{PositionsRanking(s_i)}{O} + \beta * \frac{\frac{1-e^{-\frac{l(s_i)-\mu(l)}{std(l)}}}{1+e^{-\frac{l(s_i)-\mu(l)}{std(l)}}}}{N} + \gamma * \frac{sim(s_i,t)}{máximo_{\forall summary} TR} + \mu * \sum_{\forall S_j \in Summary} \frac{sim_{cos}(s_i,s_j)}{N_s} \right] + \rho * sim_{cos}(R,D) \tag{4.2}$$

Algoritmo	Posición	Relación con el título	Longitud	Cohesión	Cobertura
	α	β	γ	μ	ρ
SFLA	0,15	0,04	0,09	0,07	0,65
SLC	0,19	0,05	0,06	0,05	0,65

Tabla 20 Pesos de la Función ObjetivoTabla 20 la suma de cada uno de los factores presentes en esta función objetivo debe ser igual a uno en cualquiera de las combinaciones establecidas.

4.3 ADAPTACIÓN DE SFLA CON MEMORIA TABÚ

Las modificaciones que se realizaron al algoritmo SFLA original para adaptarlo al problema de generación automática de resumes de texto fueron las siguientes:

- Se estableció la representación de las soluciones candidatas de forma binaria y viables.
- En los saltos de la búsqueda local, al momento de activar una frase en las reparaciones se hace: aleatoriamente para los saltos guiados y la mejor para el salto aleatorio.
- El procedimiento de mutación no se aplica a la mejor rana del estanque y aplica reparación a las ranas mutadas.
- La adaptación de una *memorita Tabú Explicita Global con soluciones Repetidas*.

- Al retornar la mejor solución global G_{best} el algoritmo SFLA-MT-SingleDocSum ordena las oraciones según el *Criterio de Selección de Oraciones al generar el Resumen* para que aparezcan en ese orden en el resumen generado.

En la Tabla 21 se presenta como se realizaba en la versión original y el cambio realizado.

SFLA		
Referente a	**Original**	**Cambio**
Esquemas de inicialización	Soluciones candidatas no viables continuas.	Soluciones candidatas viables y binarias.
Generación del vecindario	Reparación agrega las dos mejores frases.	Reparación se hace de dos formas: la primera forma agrega la mejor frase, mientras la segunda forma agrega una frase aleatoria.
Operadores evolutivos	Mutación se aplica a toda la población.	Mutación excluye a la mejor global y se aplica reparación.
Generación del vecindario	Sin memoria tabú	Se adiciona memoria tabú explicita global
Operadores evolutivos	Retorna soluciones sin organizar	Retorna soluciones organizadas bajo el criterio de selección de oraciones del resumen final

Tabla 21 Cambios del algoritmo base SFLA

El esquema del algoritmo SFLA-MT-SingleDocSum es presentado en la Figura 4-1.

```
INICIO SFLA
        Tenure = 8                                  /*Tamaño máximo de la tabú*/
        Inicializar LT                              /*Crear Lista Tabú */
        HACER
            Inicializar soluciones viables x_i      /*Crear rana i*/
            Evaluar x_i                             /*Evaluar la calidad de la rana i*/
            Agregar x_i dentro de LT                /*Agrega la solución i a la lista tabú*/
        HASTA Crear P Ranas
        PARA i = 0 HASTA itMax                      /*Número de iteraciones*/
            RankearRanas                            /*Ordenar las ranas descendientemente*/
            BarajarRanas                            /*Repartir las ranas en los memeplex*/
            BusquedaLocal_Modificada                /*Generar nuevas soluciones con los saltos*/
            ReagruparRanas                          /*Reunir ranas desde los memeplex*/
            PARA j = 1 HASTA P                      /*Mutación desde la segunda rana */
                MutarRana x_j                       /*Aplicar mutación a la rana j*/
            FIN PARA
        FIN PARA
        Retornar G_best
FIN SFLA
```

Figura 4-1 Esquema Algoritmo SFLA-MT-SingleDocSum

El procedimiento incluido de *Memoria Tabú explicita global* en la búsqueda local es presentado en la Figura 4-2 y efectúa algunas tareas adicionales al momento de crear una nueva rana candidata:

- Se hace una copia de la peor rana del memeplex (x_w) en y_i, siendo esta quien hace los saltos sin alterar la solución original.

- Se evalúa la nueva solución ($evaluar(y_i)$) y se compara con su predecesora, en caso de ser mejor se reemplaza.
- Al final se actualiza x_w con la solución copia y_i.

```
INICIO Búsqueda Local
    PARA im = 0 HASTA m                          /*Número de Memeplex*/
        PARA itm = 0 HASTA it                    /*Número de iteraciones en cada memeplex*/
            Determinar x_w, x_b y x_g            /*Determinar la peor y mejor rana*/
            REPETIR
                y_i = Saltar x_b                 /*Saltar usando la mejor local*/
                Agregar y_i dentro de LT         /*Agrega y_i a la lista tabú*/
            HASTA VerificarMemoriaTabu y_i = Falso /*Verifica si la solución es Tabú*/
            Reparacion2                          /*Habilitar oración aleatoriamente*/
            Evaluar y_i
            SI Aptitud y_i > Aptitud x_w
                x_w = y_i                        /*Actualizar la peor rana*/
            SINO
                REPETIR
                    y_i = Saltar x_g             /*Saltar usando la mejor global*/
                    Agregar y_i dentro de LT     /*Agrega y_i a la lista tabú*/
                HASTA VerificarMemoriaTabu y_i = Falso /*Verifica si la solución es Tabú*
                Reparacion2                      /*Habilitar oración aleatoriamente*/
                Evaluar y_i

                SI Aptitud y_i > Aptitud x_w
                    x_w = y_i                    /*Actualizar la peor rana*/
                SINO
                    x_w = Inicializar y_i        /*Inicializar x_w*/
                    Reparacion1                  /*Habilitar mejor oración*/
                    Evaluar y_i
                    SI Aptitud y_i > Aptitud x_g
                        x_g = y_i                /*Actualizar la mejor rana global*/
        FIN PARA
    FIN PARA
FIN Búsqueda Local
```

Figura 4-2 Esquema Procedimiento Búsqueda Local con memoria tabú

4.3.1 Búsqueda Local (SFLA)

Este procedimiento se hace en cada memeplex m y se realiza un número de iteraciones (it) que indica la cantidad de veces que se va a modificar a la peor rana del memeplex hasta lograr mejorarla o generar una nueva solución.

En cada iteración para realizar el ajuste, lo primero es hacer una copia de x_w en una variable temporal (y_i), esta variable es la que realiza los siguientes métodos: Saltar y Reparar.

4.3.1.1 Saltar

Este método consiste en hacer que y_i salte en una primera instancia hacia una solución considerada como la mejor local o global (x_b y x_g, respectivamente) utilizando la Ecuación (4.3):

$$S_i = rand() * (x_{bi} - y_i) \tag{4.3}$$

Donde S_i es el resultado de la diferencia entre la mejor solución y la peor multiplicada por un número aleatorio [0,1]. Se debe tener en cuenta que las posiciones cuyos valores sean iguales, no se verán afectadas por esta ecuación.

Después, este valor que se encuentra entre [-1,+1] se debe normalizar para luego poder representarlo de forma binaria. Las Ecuaciones (4.4) y (4.5) muestran cómo hacerlo:

$$t_i = \frac{1}{1 + e^{-S_i}} \tag{4.4}$$

$$y_i = \begin{cases} 0 \quad , \quad t_i \leq \alpha \\ y_i \, , \quad \alpha < t_i \leq \frac{1}{2}(1+\alpha) \\ 1 \quad , \quad \frac{1}{2}(1+\alpha) < t_i \end{cases} \tag{4.5}$$

Donde t_i es la normalización de S_i usando una función sigmoide y y_i es el valor definitivo en la posición i de la rana y_i teniendo en cuenta los rangos donde se evalúa t_i delimitados por el valor constante α.

La Ecuación (4.3) muestra el salto hacia la mejor local y reemplazaría a la peor si después del salto logra mejorar su valor de aptitud, que en caso de no hacerlo se repetirán las Ecuaciones (4.3), (4.4) y (4.5) reemplazando x_b por x_g. Finalmente, si no hay mejora con los saltos guiados entonces x_w saltará de manera aleatoria que consiste en inicializar nuevamente la rana en cualquier posición del espacio de búsqueda.

4.3.1.2 Reparación

Existen dos tipos de reparación incluidas en SFLA-MT-SingleDocSum y consisten en modificar el vector solución de y_i después de hacer un salto incluyendo conocimiento del problema, esto se logra habilitando una de las oraciones que no están activas en cada rana modificada de la siguiente forma:

- *Reparación 1:* Esta reparación se realiza después del salto aleatorio, donde se puntúan las oraciones de la solución candidata (ver 4.5.1) con el fin de habilitar (ver 4.5.2) la mejor oración de las no activas.

- *Reparación 2:* Consiste en habilitar (ver 4.5.2) una oración de la solución candidata de manera aleatoria de entre las no activas con la finalidad de incrementar la exploración.

4.3.1.3 Verificar Memoria Tabú

El algoritmo SFLA-MT-SingleDocSum adapta una memoria tabú de tipo explícita global a través de la implementación de una lista (LT) que almacena los vecinos generados a partir de su ajuste, con el objetivo de aumentar la diversidad de las soluciones candidatas al generar nuevas soluciones. Esta lista almacena 8 soluciones candidatas $(tenure = 8)$,

actuando como una lista de tipo FIFO cada vez que se supera el $tenure$, como se presenta en la Ecuación (4.6).

$$LT = [y_1, y_2, \dots, y_{tabú}, \dots, y_{tenure}], tabu = 1,2, \dots, tenure \quad (4.6)$$

Donde LT representa la lista tabú explícita global, $y_{tabú}$ representa una rana en la lista tabú y $tenure$ indica el número máximo de soluciones que puede almacenar la lista tabú.

De acuerdo a la Ecuación (4.7) verifica si la rana (y_k) es o no una solución tabú. Si el resultado es verdadero, la solución es tabú y por lo tanto se debe generar una nueva solución, ya que esta ha sido visitada con anterioridad. De modo contrario, la rana no es solución tabú y es válida como nueva solución candidata. En cualquiera de los casos anteriores, la solución es agregada a la lista tabú (LT).

$$VerificarMemoriaTabú(Y_k) = \begin{Bmatrix} Verdadero, & Sí\ Y_k \in LT \\ Falso, & De\ otro\ modo \end{Bmatrix} \quad (4.7)$$

4.3.1.4 Mutación

La mutación consiste en deshabilitar una frase (ver 4.5.3) y habilitar otra (ver 4.5.2) de forma aleatoria. Para cada rana x_i se calcula una probabilidad para aplicar o no la mutación, este método excluye a la primera rana que es la mejor solución global (G_{best}).

4.4 ADAPTACIÓN DE SLC CON ASCENSO DE LA COLINA

Las modificaciones que se realizaron al algoritmo SLC original para adaptarlo al problema de generación automática de resumes de texto fueron las siguientes:

- Se estableció la representación de las soluciones candidatas de forma binaria y viables.
- Se cambiaron las probabilidades dentro del procedimiento de imitación con el fin de realizar más imitación hacia los mejores jugadores y reducir la probabilidad de ajuste.
- Se introduce una probabilidad dentro del procedimiento de provocación, haciendo que el procedimiento solo aplique en algunos jugadores.
- Se agrega el procedimiento de mutación de intercambio en donde deshabilita la peor oración y habilita otra oración de forma aleatoria.
- Se adiciona el procedimiento de sustitución donde se realiza el cruce entre dos jugadores suplentes del equipo perdedor para generar dos nuevas soluciones candidadtas.
- La adaptación de la búsqueda local de *ascenso de la colina* aplicada en el peor jugador de la liga una vez se hayan inicializado las soluciones.
- Al retornar la mejor solución global G_{best}el algoritmo SLC-ASC-SingleDocSum ordena las oraciones según el *Criterio de Selección de Oraciones al generar el Resumen* para que aparezcan en ese orden en el resumen generado.

En la **Tabla 22** se presentan como se realizaba en la versión original y el cambio realizado.

SLC		
Referente a	**Original**	**Cambio**
Esquemas de inicialización	Soluciones candidatas continuas y no viables	Soluciones candidatas binarias y viables
Operadores evolutivos	Imitación y ajuste al jugador titular	Imitación o ajuste al jugador titular
Operadores evolutivos	Provocación en todos los jugadores suplentes del equipo	Provocación en algunos jugadores suplentes del equipo
Operadores evolutivos	No aplica mutación	Aplica mutación de intercambio en los jugadores titulares del equipo perdedor
Operadores evolutivos	No aplica sustitución	Aplica sustitución en los jugadores suplentes del equipo perdedor.
Generación del vecindario	Sin ascenso de la colina	Se adiciona ascenso de la colina a la peor solución al inicializar
Operadores evolutivos	Retorna soluciones sin organizar	Retorna soluciones organizadas bajo el criterio de selección de oraciones del resumen final

Tabla 22 Cambios del algoritmo base SLC

El esquema del algoritmo SLC-ASC-SingleDocSum es presentado en la Figura 4-3.

```
INICIO SLC
        HACER
            Inicializar soluciones viables x_i          /*Crear jugador i*/
            Evaluar x_i                                 /*Evaluar la calidad del jugador i*/
            Actualizar G_best                           /*Actualizar la mejor solución global*/
        HASTA Crear n Jugadores
        Ordenar                                         /*Ordenar jugadores descendentemente*/
        Asignar                                         /*Asignar jugadores a los equipos*/
        Instanciar p_i                                  /*Instanciar mejor solución local*/
        Optimizar peor jugador con Ascenso de la Colina
        PARA i = 0 HASTA Nenfre                         /*Número de enfrentamientos*/
            Selección equipo 1y2                        /*Selección de equipos a enfrentar*/
            SI Ganador Equipo1
                Imitación Equipo1                       /*Imitación a jugadores titulares*/
                Provocación Equipo1                     /*Provocación a jugadores suplentes*/
                Mutación Equipo2                        /*Mutación a jugadores titulares*/
                Sustitución Equipo2                     /*Sustitución a jugadores suplentes*/

            SINO
                Imitación Equipo2                       /*Imitación a jugadores titulares*/
                Provocación Equipo2                     /*Provocación a jugadores suplentes*/
                Mutación Equipo1                        /*Mutación a jugadores titulares*/
                Sustitución Equipo1                     /*Sustitución a jugadores suplentes*/
        FIN PARA
        Retornar mejor Solución G_best
  FIN SLC
```

Figura 4-3 Esquema Algoritmo SLC-ASC-SingleDocSum

El procedimiento *Optimizar peor jugador con Ascenso de la Colina* presentado en la Figura 4-4 efectúa las siguientes tareas adicionales para crear un nuevo jugador:

- Inicialmente se selecciona el peor jugador de la liga x_i para realizar el ajuste.

- Se hace una copia del peor jugador ($Copia\ x_i$) en S para no perder los valores iniciales de la solución.
- Se evalúan las oraciones en la solución del peor jugador (S) de acuerdo a las características establecidas en el *Criterio para deshabilitar una oración* ($Puntuar\ oraciones\ S$).
- Se selecciona la oración con la menor evaluación para ser deshabilitada ($Deshabilitar\ peor\ oración\ S$).
- A continuación se habilita la mejor oración que tenga menor o igual número de palabras que la oración que se acaba de deshabilitar ($Habilitar\ mejor\ oración\ S$).
- Se evalúa la nueva solución ($Evaluar\ R$) y se compara con su predecesora, en caso de ser mejor se reemplaza.
- Al final actualiza al peor jugador de la liga (x_i) con la solución copia (S), en otras palabras el nuevo jugador.

```
INICIO Optimizar peor jugador con Ascenso de la colina
        REPETIR
            xi = Peor jugador de la liga
            S = Copia xi                          /*Solución candidata*/
            PARA i=0 HASTA Opt HACER              /*Número de Optimizaciones*/
                Puntuar Oraciones S
                R = Deshabilitar peor oración S
                R = Habilitar la mejor oración S
                Evaluar R                         /*Evaluar la calidad del jugador R*/
                SI Aptitud R > Aptitud S
                    S = R
            FIN PARA
            xi = S
        HASTA Optimizar (N*PuntosOpt) jugadores
 FIN Optimizar peor jugador con Ascenso de la colina
```

Figura 4-4 Procedimiento de Optimización con Ascenso de Colina SLC

4.4.1 Optimizar peor jugador con Ascenso de la Colina (SLC)

Una vez se haya generado la población inicial N, esta es organizada descendentemente por su aptitud para ubicar el peor jugador de esta población y aplicarle la optimización un número de iteraciones (optimizaciones Opt) que indica las veces que se va a modificar el peor jugador hasta lograr una mejora o en su defecto hasta terminar el número de iteraciones.

Las tareas llevadas a cabo en este procedimiento, se describen en las siguientes secciones.

4.4.1.1 Puntuar oraciones

El puntaje $PuntuarOracion(s)$ de la s-ésima oración se obtiene sumando el ranking de posición de la oración con el valor obtenido al evaluar la similitud de cosenos entre la oración y el documento (cobertura) (4.8).

$$PuntuarOracion(s) = \frac{2 - 2 * \left(\frac{pos_s - 1}{n - 1}\right)}{n} + sim_{cos}(V_s, D) \qquad (4.8)$$

Donde $s = 1,2, \dots, N$, (N es el número de oraciones del documento), s representa el índice de la s-ésima oración del documento, pos_s representa la posición de la oración s en el

documento, V_s y D son los vectores de términos que representan la oración s y al documento, respectivamente.

4.4.1.2 Puntuar oraciones solución

Este método se basa en evaluar las oraciones habilitadas en la solución candidata x_i y obtener el índice de la oración con el puntaje más bajo (b), de acuerdo a la Ecuación (4.9).

$$b = \begin{cases} s, Sí\ x_{i,s} = 1 \wedge x_{i,b} = 1 \wedge PuntuarOracion(s) < PuntuarOracion(b) \\ b, \qquad De\ otro\ modo \end{cases} \qquad (4.9)$$

Donde $b \in [1, N]$ (N es el número de oraciones del documento) representa el índice de la oración con menos puntaje de x_i, $x_{i,s}$ representa la s-ésima oración en x_i y $s = 1,2, \dots, N$.

4.4.1.3 Deshabilitar peor oración

Este método consiste en deshabilitar (ver 4.5.3) en la solución candidata la oración habilitada con el puntaje más bajo, el cual aporta conocimiento del problema de generación automática de resúmenes al identificar la peor oración del documento para no ser incluida en el resumen.

4.4.1.4 Habilitar la mejor oración

Este método consiste en seleccionar en la solución candidata la mejor oración deshabilitada del documento y habilitarla (ver 4.5.2), verificando que la oración a habilitar tenga menor o igual número de palabras que la oración que se deshabilita. Al igual que el método anterior (4.4.1.3) aporta conocimiento del problema de generación automática de resúmenes al identificar la mejor oración del documento para ser incluida en el resumen.

4.4.2 Procedimientos de mejora (SLC)

Cada vez que se realiza un enfrentamiento entre equipos a los jugadores se les aplica los procedimientos de mejora (imitación, provocación, mutación y sustitución) con el objetivo de mejorar el rendimiento de los jugadores.

4.4.2.1 Imitación

Este procedimiento de mejora (ver Figura 3-4) es aplicado a cada uno de los jugadores titulares (x_i) del equipo ganador, el cual consiste en copiar el vector solución que representa al mejor jugador de la liga (G_{best}) o del equipo (p_i), a través de probabilidades. Se recorre el vector solución del jugador titular (x_i), y si cumple la probabilidad $(HCMR)$ establecida, la posición en la que se encuentra toma el valor exacto (0,1) del vector solución del mejor jugador de la liga o del equipo, en caso de no cumplir esta probabilidad se pasa a evaluar la segunda probabilidad (PAR) donde se realiza un ajuste del valor que se encuentra en la posición del vector solución (x_i), esto se representa en la Ecuación (4.10).

$$\mathrm{x_{i,j}} = \begin{cases} \mathrm{G_{best,j}} & \text{Sí Probabilidad1} > \mathrm{HCMR} \\ \max(\min(\mathrm{j} + \mathrm{round}\,(2 * \mathrm{rand} - 1), 1), 0) & \text{Sí Probabilidad 2} > \mathrm{PAR} \end{cases} \qquad (4.10)$$

Donde $\boldsymbol{x_{i,j}}$ representa la posición $\boldsymbol{j}$ dentro del vector solución del jugador $\boldsymbol{x_i}$, $\boldsymbol{G_{best,j}}$ es el valor que contiene el mejor jugador de la liga en la posición $\boldsymbol{j}$ del vector solución, $\boldsymbol{HCMR}$ indica la probabilidad que se debe cumplir para poder realizar la copia exacta hacia el mejor jugador de la liga y en caso de no cumplirse se pasa a evaluar la segunda probabilidad $\boldsymbol{PAR}$ para poder realizar el ajuste. Cabe recordar que este procedimiento es idéntico con el mejor jugador del equipo en caso de que la solución que se obtenga imitando al mejor jugador de la liga no supere a su predecesor.

4.4.2.2 Provocación

El procedimiento de provocación (ver Figura 3-5) es aplicado solo a algunos jugadores suplentes (x_i) del equipo ganador de acuerdo a una probabilidad, por medio de un ajuste a cada valor dentro del vector solución tomando al mejor jugador de la liga (G_{best}), al mejor jugador del equipo (p_i) o a un jugador aleatorio dentro del equipo ($RP2$), estos se combinan junto con otro jugador del equipo ($RP1$) como se ven en las Ecuaciones (4.11),(4.12) y (4.13) respectivamente para ajustar los valores dentro del vector solución del jugador suplente.

$$x_{i,j} = \max(\min(x_{i,j} + \tau \left(G_{best_j} - RP1_{i,j}\right), 1),0) \qquad (4.11)$$

$$x_{i,j} = \max(\min(x_{i,j} + \tau \left(P_{i,j} - RP1_{i,j}\right), 1),0) \qquad (4.12)$$

$$x_{i,j} = \max(\min(x_{i,j} + \tau \left(RP2_{i,j} - RP1_{i,j}\right), 1),0) \qquad (4.13)$$

Donde $x_{i,j}$ representa la posición j dentro del vector solución del jugador x_i, G_{best_j} es el valor que contiene el mejor jugador de la liga en la posición j, $P_{i,j}$ es el valor que contiene el mejor jugador del equipo i en la posición j, $RP1_{i,j}$ es el valor que contiene un jugador aleatorio 1 del equipo i en la posición j, $RP2_{i,j}$ es el valor que contiene un jugador aleatorio 2 del equipo i en la posición j y τ es un valor aleatorio que puede tomar el valor de cero o uno.

4.4.2.3 Mutación

El procedimiento de mejora de mutación (ver Figura 3-6) se aplica a los jugadores titulares del equipo perdedor (x_i), dando la posibilidad de generar soluciones vecinas para tratar de mejorarlas. Para este procedimiento se realiza una mutación de intercambio que deshabilita la peor oración (d) de (x_i) y se habilita de forma aleatoria una frase no seleccionada (q) como se muestran en la Ecuación (4.14).

$$\mathrm{x_{i,j}} = \left\{ \begin{array}{lll} 0 & \text{Sí} & \mathrm{s = d} \\ 1 & \text{Sí} & \mathrm{s = q} \end{array} \right\} \qquad (4.14)$$

Donde $x_{i,j}$ representa la *j*-ésima oración del *i*-ésimo jugador,$i = 1, 2, \ldots, n$ (n es el número de jugadores), $s = 1, 2, \ldots, N$, (N es el número de oraciones del documento), $d \in [1, N]$, representa el índice de la oración a deshabilitar y $q \in [1, N]$, representa el índice de la oración a habilitar en $x_{i,j}$.

4.4.2.4 Sustitución

El procedimiento (ver Figura 3-7) se basa en el cruce de dos jugadores suplentes (x_i, x_g) seleccionados aleatoriamente del equipo perdedor, utilizando los vectores soluciones de cada jugador se emplea un cruce para generar dos nuevas soluciones (y_1, y_2) con las Ecuaciones (4.15) y (4.16).

$$y_{1,j} = (\alpha * x_{i,j}) + \left((1 - \alpha) * x_{g,j}\right) \quad (4.15)$$

$$y_{2,j} = (\alpha * x_{g,j}) + \left((1 - \alpha) * x_{i,j}\right) \quad (4.16)$$

Donde $y_{1,j}$ y $y_{2,j}$ representan la *j*-ésima posición dentro del vector solución del nuevo jugador 1 y 2, $x_{i,j}$ representa la *j*-ésima oración del *i*-ésimo jugador aleatorio, $i = 1,2, \dots, n$, (n es el número de jugadores), $x_{g,j}$ representa la *j*-ésima oración del *g*-ésimo jugador aleatorio, $g = 1,2, \dots, n$, (n es el número de jugadores) y α es un número aleatorio que puede tomar el valor de cero o uno.

4.5 ADAPTACIONES COMPARTIDAS

4.5.1 Puntuar oraciones documento

Este método se basa en evaluar las oraciones deshabilitadas del documentos y obtener el índice de la oración con el puntaje más alto (a), de acuerdo a la Ecuación (4.17).

$$a = \begin{cases} s, Sí\ d_s = 0 \wedge d_a = 0 \wedge PuntuarOracion(s) > PuntuarOracion(a) \\ a, \quad De\ otro\ modo \end{cases} \quad (4.17)$$

Donde $a \in [1, N]$ (N es el número de oraciones del documento) representa el índice de la oración con menos puntaje del documento d, d_s representa la *s*-ésima oración en d y $s = 1,2, \dots, N$.

4.5.2 Habilitar oración

Este método consiste en habilitar una oración (a) deshabilitada del documento y habilitarla en x_i, de acuerdo a la Ecuación (4.18).

$$\mathrm{x_{i,s}} = \begin{cases} 1, \quad Sí\ s = a \\ \mathrm{x_{i,s}}, \mathrm{De\ otro\ modo} \end{cases} \quad (4.18)$$

Donde $x_{i,s}$ representa la *s*-ésima oración de la *i*-ésima solución, $i = 1, 2, \dots, n$ (n es el número de soluciones), $s = 1,2, \dots, N$ (N es el número de oraciones del documento), $a \in [1, N]$, representa el índice de la oración a habilitar en x_i.

4.5.3 Deshabilitar oración

Este método consiste en deshabilitar en x_i una oración (b) habilitada de acuerdo a la Ecuación (4.19).

$$x_{i,s} = \begin{cases} 0, \quad Sí\ s = b \\ x_{i,s}, De\ otro\ modo \end{cases} \quad (4.19)$$

Donde $x_{i,s}$ representa la *s*-ésima oración de la *i*-ésima solución, $i = 1, 2, \dots, n$, (n es el número de soluciones), $s = 1,2, \dots, N$, (N es el número de oraciones del documento), $b \in [1, N]$, representa el índice de la oración a deshabilitar en x_i.

4.5.4 Condición de parada

La ejecución de los algoritmos termina cuando se realizan las *itMax* en SFLA *y nI* iteraciones en SLC, o cuando se alcanza el número máximo de evaluaciones de la función objetivo.

4.5.5 Generación de las Soluciones Iniciales

Al inicializar (tiempo $t = 0$) se generan n soluciones candidatas (x_i) que se representan de la siguiente forma, de acuerdo a la Ecuación (4.20).

$$x_i = \left[x_{i,1}, x_{i,2}, \dots, x_{i,s}, \dots, x_{i,N}\right] \tag{4.20}$$

Donde N es el número de oraciones en el documento, tal que $x_{i,s} \epsilon \{0,1\}$ es un binario.

Para crear cada x_i, se selecciona aleatoriamente una oración del documento, luego se verifica que la longitud de la oración más la suma de la longitud de las oraciones que se encuentran habilitadas en el vector solución no sobrepase el máximo número de palabras del resumen, si se cumple esta restricción, se habilita esta oración en x_i. Este procedimiento se repite hasta que se cumpla el máximo número de palabras del resumen de acuerdo a la Ecuación (4.21).

$$x_{i,a} = \begin{Bmatrix} 1, si \left(\sum_{s_j \epsilon x_i} l_j\right) + l_a \leq L_{max} \\ 0, \quad De\ otro\ modo \end{Bmatrix} \tag{4.21}$$

Donde α es un número entero aleatorio entre $[1, N]$, l_j es la longitud (medida en palabras) de la *j*-ésima oración del resumen, l_a es la longitud (medida en palabras) de la oración escogida según α y L_{max} es el máximo de número de palabras del resumen.

Al generar el primer x_i con $i = 0$, se establece como la mejor solución global, después al generar cada uno de los otros x_i, se verifica si tiene mejor calidad que la mejor solución global y la reemplaza (ver sección 4.5.8).

4.5.6 Actualización de Predecesor

Para cada solución (rana en SFLA, jugador en SLC), se generan nuevas soluciones candidatas cierto número de iteraciones t. Al final de cada iteración t, se verifica si la evaluación de la nueva $solución\ candidata\ (y_i)$ es mayor a la evaluación del predecesor (x_i), si se cumple entonces x_i se reemplaza con la nueva solución candidata, de acuerdo a la Ecuación (4.22).

$$x_i = \begin{Bmatrix} y_i, & Sí\ f(y_i) > f(x_i) \\ x_i\ , & De\ otro\ modo \end{Bmatrix} \tag{4.22}$$

Donde x_i representa el *i*-ésimo solución,$i = 1, 2, \dots, n$ donde n es la cantidad de población, y_i es la nueva solución encontrada en el vecindario de x_i, $f(y_i)$ y $f(x_i)$ representan la evaluación obtenida por y_i y x_i, en la Ecuación (4.22) respectivamente.

4.5.7 Actualización de la Mejor Solución Local

Una vez generado una nueva solución candidata (y_i), se verifica si la evaluación de y_i es mayor que la evaluación de la $mejor\ solución\ local$, si es así, la $mejor\ solución\ local$ se reemplaza con y_i, de lo contrario se mantiene. Esta tarea se muestra en la Ecuación (4.23).

$$mejor\ solución\ local = \begin{Bmatrix} y_i, & Sí\ f(y_i) > f(mejor\ solución\ local) \\ mejor\ solución\ local\ , & De\ otro\ modo \end{Bmatrix} \quad (4.23)$$

Donde y_i representa la *i*-ésima solución candidata generada a través del ajuste de x_i, $i = 1,2, \dots, n$, n es la cantidad de población, $f(y_i)$ y $f(mejor\ solución\ local)$ representa la evaluación de y_i y $mejor\ solución\ local$ con la Ecuación (4.23), respectivamente.

4.5.8 Actualización de la Mejor Solución Global

Cuando se genera un x_i que tenga mejor evaluación que G_{best} se reemplaza G_{best} por x_i, de otro modo G_{best} se mantiene, como se muestra en la Ecuación (4.24).

$$G_{best} = \begin{Bmatrix} x_i, & Sí\ f(x_i) > f(G_{best}) \\ G_{best}, & De\ otro\ modo \end{Bmatrix} \quad (4.24)$$

Donde x_i representa la *i*-ésimo solución, $i = 1,2, \dots, n$, n es la cantidad de población, G_{best} es la mejor solución global, $f(G_{best})$ y $f(x_i)$ representan el puntaje obtenido al evaluar G_{best} y x_i en la Ecuación (4.24) respectivamente.

4.6 ESQUEMA DE GENERACIÓN DE RESÚMENES

El proceso de generación de un resumen (ver Figura 4-5) inicia con el pre-procesamiento del documento original, primero segmentando el texto en las oraciones que lo conforman, luego normalizando las oraciones convirtiendo mayúsculas a minúsculas y eliminando palabras vacías, después se aplica el proceso de lematización, y finalmente las oraciones son indexadas en una estructura de datos (ver sección 5.1).

El siguiente paso del proceso es llevar cada uno de los términos de las oraciones que se obtienen del pre-procesamiento a la representación del modelo espacio vectorial descrito en la Sección 2.2.1. Por cada término de la oración se almacena su frecuencia relativa en una matriz de pesos, de igual forma se realiza este proceso con los términos del título del documento.

A continuación, con base en la matriz de pesos, se determina la similitud de cosenos entre oraciones, entre cada oración y el documento, y entre cada oración con el título del documento para ser almacenados en una matriz de similitudes. Estos valores de similitud sirven para calcular algunos factores de la función objetivo como la cobertura, cohesión y relación con el título.

Por último se ejecutan los algoritmos SLFA-MT-SingleDocSum y SLC-ASC-SingleDocSum, obteniendo al final una solución candidata (vector solución), donde las posiciones con valor uno son organizadas descendentemente basándose en su posición original en el documento. Posteriormente el vector solución es decodificado para obtener las oraciones originales del documento, generando el resumen final, el cual es truncado a cien palabras para realizar la evaluación de calidad y compararlos con otros métodos del estado del arte.

Figura 4-5 Esquema de Generación de Resúmenes

A continuación se presenta un ejemplo con la noticia original y el resumen generado por el algoritmo SLFA-MT-SingleDocSum.

Documento de la noticia:

"US INSURERS expect to pay out an estimated Dollars 7.3bn (Pounds 3.7bn) in Florida as a result of Hurricane Andrew - by far the costliest disaster the industry has ever faced. The figure is the first official tally of the damage resulting from the hurricane, which ripped through southern Florida last week. In the battered region it is estimated that 275,000 people still have no electricity and at least 150,000 are either homeless or are living amid ruins. President George Bush yesterday made his second visit to the region since the hurricane hit. He pledged the government would see through the clean-up 'until the job is done'. Although there had already been some preliminary guesses at the level of insurance claims, yesterday's figure comes from the Property Claims Services division of the American Insurance Services Group, the property-casualty insurers' trade association. It follows an extensive survey of the area by the big insurance companies. Mr Gary Kerney, director of catastrophe services at the PCS, said the industry was expecting about 685,000 claims in Florida alone. It is reckoned the bulk of the damage - over Dollars 6bn in insured claims - is in Dade County, a rural region to the south of Miami. However, the final cost of Hurricane Andrew will be higher still. Yesterday's estimate does not include any projection for claims in Louisiana, which was also affected by the storm, although less severely than Florida. An estimate of the insured losses in this second state will be released later this week. But on the Florida losses alone, Hurricane Andrew becomes the most costly insured catastrophe in the US. Hurricane Hugo, which hit the east coast in September 1989, cost the insurance industry about Dollars 4.2bn. The Oakland fire disaster, in California last year, cost Dollars 1.2bn. By contrast, insurance claims resulting from the Los Angeles riots earlier this year - the most expensive civil disturbance in the US - totalled just Dollars 775m. Hurricane Andrew leaves the US property-casualty insurers facing their worst-ever year for catastrophe losses. The LA riots and a series of tornadoes, wind and hailstorms in states such as Kansas, Oklahoma and Iowa had already produced insured losses of Dollars 3.9bn. With Florida's Hurricane Andrew losses added in, the total rises to Dollars 11.2bn. This easily exceeds the record Dollars 7.6bn of

catastrophe losses seen in 1989, when the industry paid out on both Hurricane Hugo and the Loma Prieta earthquake in California. Wall Street, however, has reacted calmly to the record losses expected, and insurers' shares - although lower initially - have been firming recently. The property-casualty industry is thought to have adequate reserves to cover the disaster."

Resumen generado:

"US INSURERS expect to pay out an estimated Dollars 7.3bn (Pounds 3.7bn) in Florida as a result of Hurricane Andrew - by far the costliest disaster the industry has ever faced. Although there had already been some preliminary guesses at the level of insurance claims, yesterday's figure comes from the Property Claims Services division of the American Insurance Services Group, the property-casualty insurers' trade association. Mr Gary Kerney, director of catastrophe services at the PCS, said the industry was expecting about 685,000 claims in Florida alone. Hurricane Hugo, which hit the east coast in September 1989, cost the insurance industry about Dollars 4.2bn. Hurricane Andrew leaves the US property-casualty insurers facing their worst-ever year for catastrophe losses."

Capítulo 5

5 EVALUACIÓN

En este capítulo se describen todas las tareas involucradas con la experimentación, incluyendo el pre-procesamiento de los documentos, la descripción de los conjuntos de datos y métricas usadas, los valores de los parámetros de los algoritmos propuestos SFLA y SLC, los resultados obtenidos en la calidad de los resúmenes generados por los algoritmo en los tres ciclos de adaptación de los algoritmos, y la comparación de estos resultados con los métodos del estado del arte.

5.1 NORMALIZACIÓN E INDEXACIÓN DE DOCUMENTOS

La normalización consiste en eliminar espacios adicionales y signos de puntuación extras, segmentar el texto en párrafos, identificar los límites de la oración, eliminar los saltos de línea adicionales, entre otras [68], todo esto con el fin no perder información importante del texto original.

La normalización reduce los términos del documento a una forma canónica[4] permitiendo agrupar los términos conceptualmente relacionados y la indexación que logra identificar los términos clave que son usados por el sistema, facilitando el ordenamiento y la búsqueda de estos términos [64]. A continuación, se describen los procesos de normalización e indexación utilizados en este proyecto.

5.1.1 Segmentación

Es un proceso que ayuda a identificar la relevancia y el valor de los diferentes elementos de un texto dividiéndolo en palabras u oraciones, haciendo fácil la recuperación y la evaluación de los mismos. En este proyecto se usó una herramienta de segmentación de fuente abierta denominada "*splitta*"[5], la cual trata de controlar el problema de ambigüedad en la detección de los límites de las oraciones de un texto, y cuyo desempeño ha sido reportado en un estándar de *Wall Street Journal*, con buenos resultados [79].

5.1.2 Eliminación de mayúsculas y signos ortográficos

Para facilitar el emparejamiento entre palabras y oraciones, se normaliza el texto a través de la conversión de mayúsculas a minúsculas y la eliminación de signos ortográficos, como las tildes o diéresis [68].

[4] La forma canónica de una palabra hace referencia a su forma estándar que, por convención, representa a todas sus flexiones. Esta forma canónica varía según el idioma, por ejemplo, los verbos en inglés se representan mediante la raíz no flexionada, mientras en francés o español se representan con el infinitivo del verbo.

[5] Esta herramienta se encuentra disponible en *http://code.google.com/p/splitta*.

5.1.3 Eliminación de palabras vacías

Son términos que tienen una gran frecuencia dentro del texto, consideradas como ruido ya que no aportan al momento de identificar las oraciones más relevantes del documento. En este proyecto esta tarea se realizó usando un filtro de términos por medio de una lista de palabras vacías elaborada para el sistema de recuperación de información *SMART*[6] [80].

5.1.4 Lematización

Existen palabras que derivan de una palabra raíz (es la parte de la palabra que no cambia) y su significado está relacionado con ella, se denominan derivadas y son usadas con el fin de añadir valor gramatical, como por ejemplo "flor", "florista", y "floricultura" [64]. La lematización busca disminuir la cantidad de palabras derivadas y las formas de inflexión, para reducir la diferencia entre conceptos semejantes y disminuir los recursos de almacenamiento, tiempo de ejecución y el tamaño de la estructura de indexación [81]. En este proyecto, el algoritmo usado para realizar la eliminación de sufijos y recodificar la cadena de texto es Porter[7] [82].

5.1.5 Indexación

Una vez normalizadas las oraciones son almacenadas en una estructura de datos para realizar la búsqueda y emparejamiento de términos u oraciones agilizando el acceso rápido y exacto en la recuperación de información [83]. En este proyecto, la librería de Lucene[8] (de código abierto bajo la licencia Apache Software Licence) facilita la indexación de términos, además permite realizar las tareas de lematización, normalización y eliminación de palabras vacías. Esta herramienta permite abstraer los documentos como un conjunto de campos de texto, útil para la representación de los documentos al ser acoplados con sistemas basados en el Modelo de Espacio Vectorial. Actualmente este se encuentra disponible en diferentes lenguajes de programación como Java, C#, C++, Delphi, PHP, Phyton y Ruby [84], para este proyecto se utilizó Lucene en el lenguaje C# ya que los algoritmos se desarrollaron en este mismo lenguaje.

5.2 COLECCIÓN DE DOCUMENTOS DE EVALUACIÓN

Con el objetivo de ofrecer a la comunidad académica y científica conjuntos de datos que permitan evaluar y comparar los resultados obtenidos por los sistemas de generación automática de resúmenes, la Conferencia de Entendimiento del Documento (DUC, Document Understanding Conference) define unos resúmenes "ideales" creados por humanos expertos sobre un conjunto de documentos originales para este fin.

Para evaluar los algoritmos SFLA-MT-SingleDocSum y SLC-ASC-SingleDocSum, se utilizan los conjuntos de datos DUC2001 y DUC2002, que están compuestos por artículos de noticias en el idioma inglés. DUC2001 es una colección de 309 artículos divididos en 30 tópicos, y DUC2002 es una colección de 567 artículos con 59 tópicos. En los dos conjuntos de datos, los artículos tienen un resumen de referencia con una longitud de 100 palabras (Tabla 23).

[6] Esta lista se encuentra disponible en *ftp://ftp.cs.cornell.edu/pub/smart/english.stop*.
[7] Disponible en http://tartarus.org/martin/PorterStemmer/
[8] Esta librería se encuentra disponible en http://lucenenet.apache.org/

	DUC2002	DUC2001
Número de conjuntos	59	30
Número de documentos	567	309
Fuente de datos	TREC[9]	TREC
Longitud del resumen	100 palabras	100 palabras

Tabla 23 Resumen de los conjuntos de datos utilizados

5.3 MÉTRICAS DE EVALUACIÓN

La evaluación de la calidad de los resúmenes generados por los algoritmos SFLA-MT-SingleDocSum y SLC-ASC-SingleDocSum se realizó utilizando el conjunto de métricas ROUGE (Recall-Oriented Understudy for Gisting Evaluation) [62] en su versión 1.5.5. En este proyecto se usaron las medidas ROUGE-N (con N=1 y N=2) y ROUGE SU4 ya que son las más usadas por los métodos del estado del arte, permitiendo hacer la comparación con los mismos.

5.4 AFINACION DE PARAMETROS

Los valores de los parámetros de SFLA-MT-SingleDocSum y SLC-ASC-SingleDocSum se obtuvieron después de realizar el proceso de afinación, así mismo los asociados al problema, ascenso de la colina y la memoria tabú.

En primer lugar se afinaron los parámetros asociados a los algoritmos; ***P***, ***m***, ***it*** e ***itMax*** en SFLA y ***nF***, ***nS***, ***nT*** y ***nI*** en SLC. Utilizando la configuración preliminar de la Tabla 3, la afinación de parámetros asociados al problema se realizó de la siguiente manera:

- *Máxima longitud a evaluar de un resumen*, tomando los valores de 100, 110, 120 y 130, donde el mejor resultado se encontró en 120 para SFLA y 130 para SLC.
- *Criterio para deshabilitar una oración*, se utilizaron las características de la función objetivo y algunas combinaciones de ellas, dando Posición en SFLA y Posición-Cobertura en SLC los mejores resultados.
- *Criterio de selección de oraciones del resumen final*, donde se tomó cada una de las características de la función objetivo combinadas con las del anterior criterio. El mejor resultado se obtuvo con la característica Posición para SLFA y SLC.
- *Máximo Número de Evaluaciones de la función objetivo,* no requirió afinación ya que su valor tenía la restricción de ser 1600 para hacer la comparación de los algoritmos propuestos con los del estado del arte.

Por último la afinación de los parámetros de ascenso de la colina y la memoria tabú, utilizó las configuraciones de los parámetros de los algoritmos (SFLA, SLC) y asociados al problema presentadas en las Tabla 6 y Tabla 10, respectivamente.

En SFLA se experimentó con los dos tipos de memoria tabú: Explícita Global, Explícita por Rana combinado con la mejor configuración de ascenso de colina que fue *optimizar el mejor*

[9] http://trec.nist.gov/overview.html

al iterar. Cada memoria tomó diferentes valores del parámetro $tenure$. El mejor resultado se obtuvo con la Memoria Tabú *Explícita Global* y el parámetro $tenure$ con valor ocho.

Para SLC se experimentó con la búsqueda local Ascenso de la Colina con las diferentes configuraciones donde se optimiza al mejor, peor, aleatorio o aleatorio por equipo, al inicializar la población o al iterar. La configuración de *optimizar el peor al inicializar* obtuvo los mejores resultados, con el número de optimizaciones establecido en seis y el número de optimización en uno al ser solo un jugador.

Las configuraciones finales de los parámetros de SFLA-MT-SingleDocSum y SLC-ASC-SingleDocSum, se obtuvieron al finalizar los ciclos de experimentación y se presenta en la Tabla 24 y Tabla 25 respectivamente.

	Parámetro	Valor
Parámetros SFLA	Número de Memeplex (m)	5
	Número de Ranas (P)	20
	Número de iteraciones en el memeplex (it)	10
	Número de iteraciones máxima (itMax)	150
Parámetros Asociados al Problema	Máximo Número de Evaluaciones de la función objetivo	1600
	Máxima Longitud a Evaluar de un resumen	120
	Criterio para deshabilitar una oración	Posición
	Criterio de selección de oraciones del resumen final	Posición
Parámetros Memoria Tabú	Tenure	8
	Tipo de Memoria Tabú	Explícita Global

Tabla 24 Parámetros Finales SFLA-MT-SingleDocSum

	Parámetro	Valor
Parámetros SLC	Número de Equipos (nT)	40
	Número de Jugadores Titulares (nF)	10
	Número de Jugadores Suplentes (nS)	5
	Número de Iteraciones (nI)	1
Parámetros Asociados al Problema	Máximo Número de Evaluaciones de la función objetivo	1600
	Máxima Longitud a Evaluar de un resumen	130
	Criterio para deshabilitar una oración	Posición y Cobertura
	Criterio de selección de oraciones del resumen final	Posición
Parámetros Ascenso de la Colina	Número de optimizaciones	6
	Puntos a Optimizar	1
	Optimización	El Peor Luego de Inicializar

Tabla 25 Parámetros Finales SLC-ASC-SingleDocSum

SFLA-MT-SingleDocSum y SLC-ASC-SingleDocSum fueron ejecutados treinta veces por documento, evaluando los resúmenes generados en cada ejecución a través de las métricas

de ROUGE, obteniendo al final un resultado promedio de todo el conjunto de documentos por cada métrica. La implementación de los algoritmos se realizó en el lenguaje de programación C# de la plataforma .NET, las ejecuciones de los algoritmos se realizaron en un computador de escritorio con procesador Intel Core i5 3.2 GHz, RAM de 8 GB, sistema operativo Windows 10.

En la Figura 5-1 se pueden observar los diferentes parámetros para ejecutar los algoritmos como son: *DataSet*, conjunto de documentos a evaluar; *#Experimentos*, cantidad de experimentos a realizar; *Método Optimización*, método a utilizar en la búsqueda local (Ascenso de Colina o Memoria Tabú); *Donde Optimizar?* y *Que Optimizar?*, definen en que parte se va a utilizar la búsqueda local (al inicializar o al iterar) y a que o cuales soluciones se va a aplicar (el mejor, el peor o aleatorio); *#SolucionesOptimizar* y *#Optimizaciones*, definen cuantas soluciones candidatas y el número de vecinos que se le van a generar; *Tipo Memoria Tabú* y *Tenure*, tipo de memoria y tamaño de la misma; *Criterio para deshabilitar,* puntúa y ordena las oraciones para habilitar y deshabilitar en una solución candidata; *Criterio de resumen final*, ordenada las oraciones de la mejor solución encontrada. En la parte derecha, están los porcentajes que pueden definirse para las diferentes características asociadas a la función objetivo como son: *Posición, Relación con el título, Longitud, Cohesión y Cobertura.*

Al dar click en el botón *Evaluar*, el programa toma los parámetros de la interfaz y al finalizar la ejecución retorna en un documento Excel, los resultados de calidad en las medidas Rouge-1 y Rouge-2.

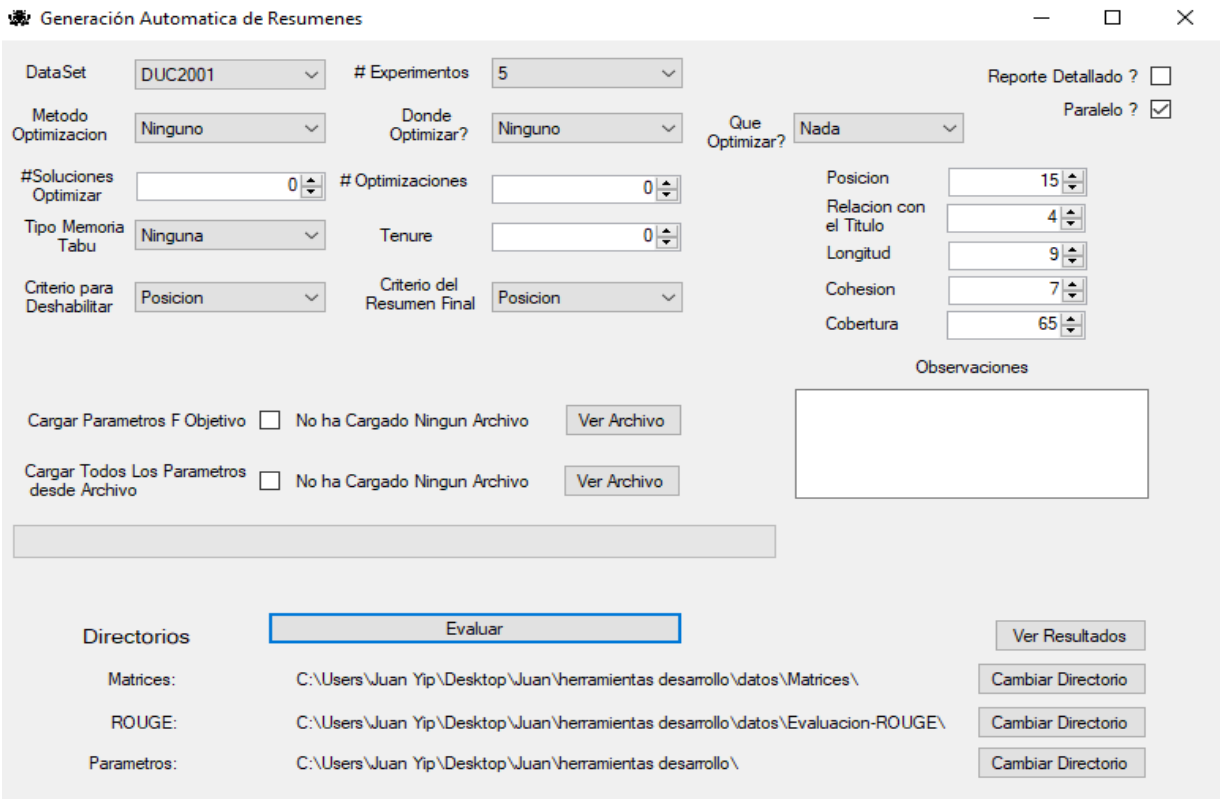

Figura 5-1 Pantalla principal para configurar el algoritmo

5.5 COMPARACIÓN CON OTROS MÉTODOS DEL ESTADO DEL ARTE

Los resultados obtenidos por SFLA y SLC en las medidas ROUGE-1 y ROUGE-2 sobre los conjuntos de datos DUC2001 y DUC2002 son comparados y presentados en la Tabla 26 y Tabla 27 respectivamente, con los siguientes métodos del estado del arte de generación automática de resúmenes de un documento, basados en: *aprendizaje de maquina* NetSum [14], CRF [15] y QCS [58]; *reducción algebraica* SVM [47]; *grafos* UnifiedRank [21] y Manifold Ranking [85]; *metaheurísticas* FEOM [86], MA-SingleDocSum [44], DE [45] y FSP-MT-SingleDocSum [55].

Método	ROUGE-1		ROUGE-2	
SFLA-MT-SingleDocSum	6	0,45746	2	0,20678
SLC-ASC-SingleDocSum	4	0,46060	**1**	**0,20771**
FSP-MT-SingleDocSum	5	0,46004	3	0,20582
MA-SingleDocSum	9	0,44862	4	0,20142
DE	**1**	**0,47856**	6	0,18528
UnifiedRank	8	0,45377	9	0,17646
FEOM	2	0,47728	5	0,18549
NetSum	3	0,46427	8	0,17697
CRF	7	0,45512	10	0,17327
QSC	10	0,44852	7	0,18523
SVM	11	0,44628	11	0,17018
Manifold Ranking	12	0,43359	12	0,16635

Tabla 26 Puntajes ROUGE de los métodos sobre DUC2001

Método	ROUGE-1		ROUGE-2	
SFLA-MT-SingleDocSum	**1**	**0,49040**	**1**	**0,23214**
SLC-ASC-SingleDocSum	3	0,48738	3	0,22994
FSP-MT-SingleDocSum	2	0,48862	2	0,23056
MA-SingleDocSum	5	0,48280	4	0,22840
DE	6	0,46694	8	0,12368
UnifiedRank	4	0,48487	5	0,21462
FEOM	7	0,46575	7	0,12490
NetSum	8	0,44963	9	0,11167
CRF	10	0,44006	10	0,10924
QSC	9	0,44865	6	0,18766
SVM	11	0,43235	11	0,10867
Manifold Ranking	12	0,42325	12	0,10677

Tabla 27 Puntajes ROUGE de los métodos sobre DUC2002

Para calcular el porcentaje de mejora de los resultados obtenidos por SFLA-MT-SingleDocSum y SLC-ASC-SingleDocSum con respecto a los otros métodos, se utiliza la Ecuación (5.1).

$$\frac{MetodoPropuesto - OtroMetodo}{OtroMetodo} \times 100 \qquad (5.1)$$

En la Tabla 28 se presentan los porcentajes de mejora de SFLA-MT-SingleDocSum en las medidas ROUGE-1 y ROUGE-2 para DUC2002.

Método	% Mejora obtenida por SFLA-MT-SingleDocSum (%)	
	ROUGE-1	ROUGE-2
	DUC2002	DUC2002
SLC-ASC-SingleDocSum	0,62	0,96
FSP-MT-SingleDocSum	0,36	0,69
MA-SingleDocSum	1,57	1,64
DE	5,02	87,69
UnifiedRank	1,14	8,16
FEOM	5,29	85,86
NetSum	9,07	107,88
CRF	11,44	112,50
QSC	9,31	23,70
SVM	13,43	113,62
Manifold Ranking	15,87	117,42

Tabla 28 Comparación de SFLA-MT-SingleDocSum con otros métodos

En la Tabla 29 se presentan los porcentajes de mejora de SLC-MT-SingleDocSum en la medida ROUGE-2 para DUC2001.

Método	% Mejora obtenida por SLC-ASC-SingleDocSum (%)
	ROUGE-2
	DUC2001
SFLA-MT-SingleDocSum	0,45
FSP-MT-SingleDocSum	0,92
MA-SingleDocSum	3,12
DE	12,11
UnifiedRank	17,71
FEOM	11,98
NetSum	17,37
CRF	19,88
QSC	12,14
SVM	22,05
Manifold Ranking	24,86

Tabla 29 Comparación de SLC-ASC-SingleDocSum con otros métodos

El porcentaje de mejora obtenido por DE en la medida ROUGE-1 sobre DUC2001 respecto a los otros métodos, se presenta en la Tabla 30.

Método	Mejora obtenida por DE (%)
	ROUGE-1
	DUC2001
SFLA-MT-SingleDocSum	4,61
SLC-ASC-SingleDocSum	3,90
FSP-MT-SingleDocSum	4,03
MA-SingleDocSum	6,67
Unified Rank	5,46
FEOM	0,27
NetSum	3,08
CRF	5,15
QSC	6,70
SVM	7,23
Manifold Ranking	10,37

Tabla 30 Comparación de DE con otros métodos

Debido a que los resultados no identifican cual es el mejor método en general, teniendo en cuenta ambos conjuntos de datos, se utiliza un ranking unificado que tiene en cuenta la posición que ocupa cada método en cada medida, de acuerdo a la fórmula de la Ecuación (5.2) tomada de [87].

$$Ran(Metodo) = \sum_{r=1}^{12} \frac{(12 - r + 1) \times R_r}{12} \qquad (5.2)$$

Donde R_r indica el número de veces que el método aparece clasificado en la posición r. El número doce representa el número total de métodos que están siendo comparados.

Observando los resultados de la Tabla 31, se evidencia que el algoritmo SFLA-MT-SingleDocSum se ubica de primero en este ranking, con dos primeros lugares, un segundo lugar y un sexto lugar superando a los otros métodos del estado del arte, en el caso de SLC-ASC- SingleDocSum consigue ubicarse en segundo lugar del ranking, con un primer lugar, dos terceros lugares y un cuarto lugar. También que los algoritmos basados en metaheurísticas: SFLA-MT-SingleDocSum, SLC-ASC-SingleDocSum, FSP-MT-SingleDocSum, MA-SingleDocSum, DE y FEOM, superan a los otros métodos del estado del arte al tratar la generación automática de resúmenes como un problema de optimización.

Método	R_r												Clasificación Resultante
	1	2	3	4	5	6	7	8	9	10	11	12	
SFLA-MT-SingleDocSum	2	1	0	0	0	1	0	0	0	0	0	0	**3,50**
SLC-ASC-SingleDocSum	1	0	2	1	0	0	0	0	0	0	0	0	**3,42**
FSP-MT-SingleDocSum	0	2	1	0	1	0	0	0	0	0	0	0	3,33
MA-SingleDocSum	0	0	0	2	1	0	0	0	1	0	0	0	2,50
DE	1	0	0	0	0	2	0	1	0	0	0	0	2,58
FEOM	0	1	0	0	1	0	2	0	0	0	0	0	2,58
UnifiedRank	0	0	0	1	1	0	0	1	1	0	0	0	2,17
NetSum	0	0	1	0	0	0	0	2	1	0	0	0	2,00
QSC	0	0	0	0	0	1	1	0	1	1	0	0	1,67
CRF	0	0	0	0	0	0	1	0	0	3	0	0	1,25
SVM	0	0	0	0	0	0	0	0	0	0	4	0	0,67
Manifold Ranking	0	0	0	0	0	0	0	0	0	0	0	4	0,33

Tabla 31 Ranking de Clasificación de los algoritmos

5.6 COMPARACIONES ADICIONALES

Con el propósito de analizar los resultados obtenidos al adaptar los algoritmos SFLA y SLC con los algoritmos Ascenso de Colina y una memoria tabú, se presentan en la Tabla 32 los mejores resultados obtenidos por los siguientes algoritmos:

- SFLA-SingleDocSum y SLC-SingleDocSum: adaptación de SFLA y SLC a la generación automática de resúmenes, obtenido al finalizar el ciclo I (ver sección 3.1).

- SFLA-ASC-SingleDocSum y SLC-ASC-SingleDocSum: adaptación de SFLA y SLC con el optimizador local Ascenso de la Colina a la generación automática de resúmenes, obtenido al finalizar el ciclo II (ver sección 3.2).

- SFLA-MT-SingleDocSum y SLC-MT-SingleDocSum: adaptación de SFLA y SLC con una memoria tabú a la generación automática de resúmenes, obtenido al finalizar el ciclo III (ver sección 3.3).

Algoritmo	**DUC2001**			**DUC2002**		
	R1R	R2R	RSU4R	R1R	R2R	RSU4R
SFLA-SingleDocSum	0,45643	0,20592	0,22765	0,48990	0,23215	0,24793
SFLA-ASC-SingleDocSum	0,45722	0,20663	**0,22839**	0,49034	**0,23258**	**0,24830**
SFLA-MT-SingleDocSum	**0,45761**	**0,20665**	0,22836	**0,49037**	0,23217	0,24810
SLC-SingleDocSum	0,46045	0,20757	0,23003	0,48740	**0,22997**	**0,24632**
SLC-ASC-SingleDocSum	**0,46060**	**0,20771**	**0,23015**	0,48738	0,22994	0,24630
SLC-MT-SingleDocSum	0,46053	0,20705	0,22971	**0,48744**	0,22954	0,24603

Tabla 32 Resultados de la adaptación de los algoritmos

En la Tabla 33 se muestra el mejoramiento relativo, calculado como en la Ecuación (5.1), de los algoritmos SFLA-ASC-SingleDocSum y SFLA-MTSingleDocSum, respecto al algoritmo SFLA-SingleDocSum. También de los algoritmos SLC-ASC-SingleDocSum y SLC-MT-SingleDocSum, respecto al algoritmo SLC-SingleDocSum. Donde se observa el porcentaje de mejora en cada una de las medidas ROUGE.

Algoritmo	DUC2001			DUC2002		
	R1R %	R2R %	RSU4R %	R1R %	R2R %	RSU4R %
SFLA-ASC-SingleDocSum	0,17	0,34	0,33	0,09	0,19	0,15
SFLA-MT-SingleDocSum	0,26	0,35	0,31	0,10	0,01	0,07
SLC-ASC-SingleDocSum	0,03	0,07	0,05	0,00	-0,01	-0,01
SLC-MT-SingleDocSum	0,02	-0,25	-0,14	0,01	-0,19	-0,12

Tabla 33 Mejoramiento relativo al aplicar estrategias de búsqueda local

Conforme a los resultados presentados en la Tabla 33 se observa que:

- El algoritmo SFLA-ASC-SingleDocSum y SFLA-MT- SingleDocSum muestran un mejor desempeño respecto al algoritmo SFLA-SingleDocSum, ya que aumentan el valor en todas las medidas ROUGE.

- El algoritmo SLC-ASC-SingleDocSum muestra un mejor desempeño respecto a SLC-SingleDocSum, debido a que obtuvo mejores resultado en tres de las seis medias (ROUGE-1 Recall, ROUGE-2 Recall y ROUGE-SU4 Recall sobre 2001), la medida ROUGE-1 Recall sobre DUC2002 no tuvo modificación alguna y presentó resultados levemente más bajos en las demás medidas (ROUGE-2 Recall y ROUGE-SU4 Recall sobre 2002).

- El algoritmo SLC-MT-SingleDocSum muestra un peor desempeño en relación a SLC-SingleDocSum, ya que obtuvo peores resultados en cuatro medidas (ROUGE-2 Recall y ROUGE-SU4 Recal sobre DUC2001 y DUC2002) y solo en dos medidas (ROUGE-1 Recall sobre DUC2001 y DUC2002) aumentó el valor levemente.

- El algoritmo SFLA-MT-SingleDocSum muestra un mejor desempeño respecto al algoritmo SFLA-SingleDocSum, debido a que al adaptar una memoria tabú explícita logra mejorar la fase de explotación de cada salto.

- La mejora del desempeño del algoritmo SFLA-ASC-SingleDocSum en comparación con SFLA-SingleDocSum se debe a que la adaptación de la búsqueda local ascenso de la colina intensifica la búsqueda de nuevas soluciones alrededor de la mejor solución encontrada al iterar.

- El algoritmo SLC-ASC-SingleDocSum muestra una leve mejora respecto al algoritmo base SLC-SingleDocSum debido a que la adaptación de ascenso de colina ayuda a la población inicial a tener mejores soluciones con las cuales trabajar, optimizando al peor jugador generado al inicializar la población.

- El algoritmo SLC-MT-SingleDocSum muestra un peor desempeño respecto al algoritmo SLC-SingleDocSum, debido a que la lógica del algoritmo requiere de una población muy grande, lo que conlleva a que se realicen muchas evaluaciones de la función objetivo y esta se encuentra limitada, lo que no permite desarrollar de manera correcta la optimización para encontrar mejores soluciones.

Además se realizó una comparativa de tiempo de ejecución entre los algoritmos SFLA-MT-SingleDocSum, SLC-ASC-SingleDocSum y FSP-MT-SingleDocSum (ver Tabla 34), este tiempo corresponde a un solo experimento, es decir, solo se genera un resumen de cada artículo de noticia.

Algoritmo	DUC2001	DUC2002
	Tiempo	
SFLA-MT-SingleDocSum	25 Segundos	49 Segundos
FSP-MT-SingleDocSum	25 Segundos	50 Segundos
SLC-ASC-SingleDocSum	42 Segundos	89 Segundos

Tabla 34 Resultados de tiempo de ejecución

Capítulo 6

6 CONCLUSIONES Y TRABAJO FUTURO

6.1 CONCLUSIONES

Las adaptaciones necesarias para resolver el problema de la generación automática de resúmenes extractivos de un documento como un problema de optimización binaria, utilizando los algoritmos SFLA y SLC fueron:

6.1.1 SFLA

El cambio de la representación de soluciones continuas en forma binaria que facilitó el trabajo con múltiples variables; la validación de soluciones candidatas viables para evitar la evaluación innecesaria de la función objetivo; la adaptación del método reparación dividiéndolo en dos, donde la reparación 1 habilita la mejor oración utilizando el *Criterio para deshabilitar una Oración* que estableció la característica *Posición* para puntuar las oraciones y seleccionar la de mayor puntaje para ser habilitada, la reparación 2 que habilita una oración de forma aleatoria; la exclusión de la mejor solución global en el procedimiento de mutación; la adaptación de una memoria tabú explícita global que permitió mejorar la explotación en la búsqueda local de cada rana, evitando la evaluación innecesaria en soluciones candidatas ya visitadas; y por último la organización de la mejor solución al terminar el algoritmo bajo el criterio de selección de oraciones del resumen final.

La modificación de los procedimientos de reparación permitió mejorar el proceso de selección de oraciones para el resumen generado añadiendo conocimiento específico del problema dentro de la búsqueda local, ya que en la reparación 1 habilita siempre la mejor frase deshabilitada garantizando que siempre estén incluidas las mejores frases, y la reparación 2 añade un poco de exploración generando soluciones diferentes al habilitar una oración de forma aleatoria.

6.1.2 SLC

Al igual que en SFLA se realizó el cambio de la representación de soluciones continuas a binarias y la validación de soluciones candidatas viables; el cambio del procedimiento de imitación que se limita a realizar solo uno de los procesos, imitar o ajustar; la adaptación del procedimiento de provocación al retornar soluciones binarias y adicionando una probabilidad para ser aplicado solo en algunos jugadores; la inclusión del procedimiento de mutación, donde añade conocimiento específico del problema, generando nuevas soluciones candidatas ordenadas según las características *Posición y Cobertura* para puntuar las oraciones y de esta manera habilitar la oración con mayor puntaje y deshabilitar la de menor puntaje; la inclusión del procedimiento de sustitución que genera exploración al crear nuevas soluciones realizando cruce entre dos soluciones candidatas; la adaptación de ascenso de la colina que mejora la fase de inicialización de la población de jugadores, ya que al mejorar la peor solución generada aleatoriamente permite que el algoritmo cuente con mejores

soluciones desde el inicio antes de comenzar la fase de explotación y exploración; y por último la organización de la mejor solución al terminar el algoritmo bajo el criterio de selección de oraciones del resumen final.

La modificación de los procedimientos de mejora ayudaron a obtener mejores resultados en el problema de generación automática de resúmenes, ya que la imitación trató de copiar la mejor solución global o local mejorando la fase de explotación, los procedimientos de provocación y sustitución realizaron cruce entre los jugadores haciendo un poco más de exploración al generar nuevas y diferentes soluciones, y la mutación aplicó conocimiento específico del problema al deshabilitar la peor oración de la solución candidata mejorando la fase de explotación.

6.1.3 Función Objetivo

Dentro de la configuración de la función objetivo (3.1.2) se presentaron diferentes versiones con cambios en la forma de calcular algunas de las características como posición de la oración, relación al título y longitud, así mismo se reemplazó la característica *cohesión* por *centralidad de la oración* obteniendo en todas ellas peores resultados respecto a la versión original, por lo que se decidió trabajar con la función objetivo original planteada en [55], la cual permite tener una proximidad de la calidad de los resúmenes generados por los algoritmos. El uso de esta función objetivo fue efectivo ya que los algoritmos SFLA-MT-SingleDocSum y SLC-ASC-SingleDocSum obtuvieron mejores resultados que los métodos del estado del arte. Este resultado es importante porque, aunque es una función objetivo aproximada, indica que las características usadas son prometedoras para el problema de generación automática de resúmenes de un documento, ya que han sido usadas en varios algoritmos metaheurísticos.

En cuanto a los pesos de la función objetivo el resultado fue muy similar en ambos algoritmos al igual que en el algoritmo que se tomó como referencia [55]. La característica cobertura no tuvo ninguna modificación y fue la que más aportó, las otras características no se afectaron en gran medida, por lo que se muestra una cierta estabilidad de la función objetivo bajo el mismo rango de pesos dando buenos resultados en algoritmos metaheurísticos para el problema de generación automática de resúmenes de un solo documento.

La calidad de los resúmenes generados automáticamente por los algoritmos SFLA-MT-SingleDocSum y SLC-ASC-SingleDocSum fue evaluada utilizando las medidas ROUGE-1 y ROUGE-2 sobre los conjuntos de datos DUC2001 y DUC2002. Para comparar los resultados obtenidos con los otros métodos del estado del arte se utilizaron únicamente las medidas ROUGE-1 y ROUGE-2 sobre ambos conjuntos de datos, donde se observó que el algoritmo SFLA-MT-SingleDocSum supera en ambas medidas sobre el conjunto de datos DUC2002 a todos los algoritmos del estado del arte con porcentajes que se muestran en la Tabla 28, SLC-ASC-SingleDocSum supera en ROUGE-2 sobre DUC2001 a todos los algoritmos del estado del arte con porcentajes que se muestran en la Tabla 29. También se utilizó un método unificado que clasificó a los algoritmos según las posiciones que ocuparon en las medidas ROUGE-1 y ROUGE-2 sobre DUC2001 y DUC2002, donde SFLA-MT-SingleDocSum obtuvo el primer lugar y SLC-ASC-SingleDocSum el segundo lugar de la clasificación, superando los otros métodos del estado del arte (ver Tabla 31).

La comparación entre las diferentes versiones de los algoritmos de SFLA (SFLA-SingleDocSum, SFLA-ASC-SingleDocSum, SFLA-MT-SingleDocSum) y SLC (SLC-SingleDocSum, SLC-ASC-SingleDocSum, SLC-MT-SingleDocSum), se observó que el algoritmo SFLA-MT-SingleDocSum obtuvo mejores resultados en comparación con SFLA-SingleDocSum y SFLA-ASC-SingleDocSum en las medidas (ROUGE-1, ROUGE-2 sobre DUC2001 y ROUGE-1 sobre DUC2002) que se tienen en cuenta al momento de realizar las comparaciones con el estado del arte y SLC-ASC-SingleDocSum obtuvo los mejores resultados en todas las medidas sobre DUC2001 en relación a SLC-MT-SingleDocSum que obtuvo solo el mejor resultado en una medida (ROUGE-1 sobre DUC 2002) y SLC-SingleDocSum que obtuvo el mejor resultado en dos medidas (ROUGE-2, ROUGE-SU4 sobre DUC 2002).

6.2 RECOMENDACIONES Y TRABAJO FUTURO

Se espera realizar más pruebas de afinación de parámetros asociados a SFLA como el tamaño de la población de ranas, la cantidad de memeplexes y las iteraciones dentro de la búsqueda local. También la adaptación de otros tipos de ajuste para los procedimientos de saltos guiados, reparaciones y de mutación.

Para SLC se recomienda realizar pruebas de afinación asociados al algoritmo como número de jugadores, titulares y sustitutos, número de equipos y número de ligas. Además utilizar diferentes probabilidades en los procedimientos de imitación y provocación, utilizar otros tipos de mutación y otros tipos de cruce en el procedimiento de sustitución.

También se espera realizar más pruebas de afinación de parámetros asociados al problema como aplicar otros tipos de características en el criterio de selección de oraciones y el número máximo de palabras a evaluar, entre otros aspectos.

Llevar a cabo un estudio del comportamiento de la función objetivo con otras características que se encuentran en métodos del estado del arte como: nombres propios, datos numéricos, palabras claves (positivas, negativas), entre otras, así mismo afinar los pesos.

Adaptar los algoritmos SFLA-SingleDocSum y SLC-SingleDocSum en el problema de generación de resúmenes de múltiples documentos y analizar su comportamiento.

Evaluar el comportamiento de SFLA-MT-SingleDocSum y SLC-ASC-SingleDocSum con otros conjuntos de datos como CNN Corpus y evaluar la calidad de los resúmenes generados con otras medidas ROUGE como ROUGE-3, ROUGE-4, ROUGE-W, entre otras.

Se recomienda para este tipo de proyectos llevar un buen orden en el momento de realizar la afinación de parámetros, ya que entre los parámetros asociados a los algoritmos, asociados al problema de generación de resúmenes y asociados a los procedimientos de búsqueda local, se generan muchas pruebas y en caso de no llevar bien este orden, puede llevar a tener perdida de información o repetir pruebas.

Se recomienda llevar un informe de las adaptaciones que se vayan realizando al algoritmo especificando los cambios y el porqué de ellos, con el fin de tener una guía y saber por qué el algoritmo cambia su comportamiento y que cambios adicionales se pueden aplicar.

Por último se recomienda utilizar equipos con alta capacidad de procesamiento y memoria, esto se debe a que la utilización de este tipo de algoritmos requiere mucho de estos recursos para poder ejecutar y realizar las pruebas en poco tiempo. En caso de no contar con los recursos, ejecutar las pruebas en varios equipos paralelamente.

BIBLIOGRAFÍA

[1] A. Porselvi and S. Gunasundari, “Survey on web page visual summarization,” *Int. J. Emerg. Technol. Adv. Eng.*, vol. 3, no. 1, pp. 26–32, 2013.

[2] D. M. Zajic, B. J. Dorr, and J. Lin, “Single-document and multi-document summarization techniques for email threads using sentence compression,” *Inf. Process. Manag.*, vol. 44, no. 4, pp. 1600–1610, 2008.

[3] N. Kumaresh and B. S. Ramakrishnan, “Graph Based Single Document Summarization,” in *Data Engineering and Management: Second International Conference, ICDEM 2010, Tiruchirappalli, India, July 29-31, 2010. Revised Selected Papers*, R. Kannan and F. Andres, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 32–35.

[4] M. Mendoza and L. Elizabeth, “Una Revisión de la Generación Automática de Resúmenes Extractivos,” *Rev. UIS Ing.*, vol. 12, no. 1, pp. 7–27, 2013.

[5] H. P. Edmundson, “New Methods in Automatic Extracting,” *J. ACM*, vol. 16, no. 2, pp. 264–285, 1969.

[6] Y. K. Meena and D. Gopalani, “Feature Priority Based Sentence Filtering Method for Extractive Automatic Text Summarization,” in *Procedia Computer Science*, 2015, vol. 48, pp. 728–734.

[7] H. Oliveira *et al.*, “Assessing shallow sentence scoring techniques and combinations for single and multi-document summarization,” *Expert Syst. Appl.*, vol. 65, pp. 68–86, 2016.

[8] J. Kupiec, J. Pedersen, and F. Chen, “A Trainable Document,” *SIGIR ’95 Proc. 18th Annu. Int. ACM SIGIR Conf. Res. Dev. Inf. Retr.*, vol. 1, pp. 68–73, 1995.

[9] M. E. Hannah, T. V. Geetha, and S. Mukherjee, “Automatic Extractive Text Summarization Based on Fuzzy Logic: A Sentence Oriented Approach,” *Swarm Evol. Memetic Comput. Conf.*, vol. 7076, pp. 530–538, 2011.

[10] F. Kiyoumarsi, “Evaluation of Automatic Text Summarizations based on Human Summaries,” *Procedia - Soc. Behav. Sci.*, vol. 192, pp. 83–91, Jun. 2015.

[11] M. A. Tayal, M. M. Raghuwanshi, and L. G. Malik, “ATSSC: Development of an approach based on soft computing for text summarization,” *Comput. Speech Lang.*, vol. 41, pp. 214–235, 2017.

[12] and B. L. C. Aone, M. E. Okurowski, J. Gorlinsky, “A trainable summarizer with knowledge acquired from robust nlp techniques,” *Adv. Autom. Text Summ. , I. Mani M. T. Maybury*, vol. MIT Press, p. 71–80+, 1999.

[13] J. M. Conroy and D. P. O'leary, "Text summarization via hidden Markov models," *24th Annu. Int. ACM SIGIR Conf. Res. Dev. Inf. Retr.*, pp. 406–407, 2001.

[14] K. M. Svore, M. Way, L. Vanderwende, and C. J. C. Burges, "Enhancing Single-document Summarization by Combining RankNet and Third-party Sources," *Comput. Linguist.*, no. June, pp. 448–457, 2007.

[15] D. Shen, J.-T. Sun, H. Li, Q. Yang, and Z. Chen, "Document summarization using conditional random fields," *20th Int. Jt. Conf. Artifical Intell.*, pp. 2862–2867, 2007.

[16] F. Kyoomarsi, H. Khosravi, E. Eslami, P. K. Dehkordy, and A. Tajoddin, "Optimizing Text Summarization Based on Fuzzy Logic," in *Seventh IEEE/ACIS International Conference on Computer and Information Science (icis 2008)*, 2008, vol. 5, pp. 347–352.

[17] K.-F. Wong, M. Wu, and W. Li, "Extractive Summarization Using Supervised and Semi-supervised Learning," *Proc. 22nd Int. Conf. Comput. Linguist. 1. Assoc. Comput. Linguist. 2008.*, no. August, pp. 985–992, 2008.

[18] M. A. Fattah and F. Ren, "GA, MR, FFNN, PNN and GMM based models for automatic text summarization," *Comput. Speech Lang.*, vol. 23, no. 1, pp. 126–144, 2009.

[19] D. Muratore, M. Hagenbuchner, F. Scarselli, and A. C. Tsoi, "Sentence extraction by graph neural networks," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 6354 LNCS, no. PART 3, pp. 237–246, 2010.

[20] R. Mihalcea and P. Tarau, "TextRank: Bringing order into texts," *Proc. EMNLP*, vol. 85, pp. 404–411, 2004.

[21] M. Summarizations and X. Wan, "Towards a Unified Approach to Simultaneous Single-Document and," *Proc. 23rd Int. Conf. Comput. Linguist.*, no. August, pp. 1137–1145, 2010.

[22] D. R. Amancio, M. G. V. Nunes, O. N. Oliveira, and L. D. F. Costa, "Extractive summarization using complex networks and syntactic dependency," *Phys. A Stat. Mech. its Appl.*, vol. 391, no. 4, pp. 1855–1864, 2012.

[23] R. Ferreira *et al.*, "A Four Dimension Graph Model for Automatic Text Summarization," *2013 IEEE/WIC/ACM Int. Jt. Conf. Web Intell. Intell. Agent Technol.*, no. November, pp. 389–396, 2013.

[24] Y. Ledeneva, R. A. García-Hernández, and A. Gelbukh, "Graph ranking on maximal frequent sequences for single extractive text summarization," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 8404 LNCS, no. PART 2, pp. 466–480, 2014.

[25] N. Chatterjee and P. K. Sahoo, "Random Indexing and Modified Random Indexing based approach for extractive text summarization," *Comput. Speech Lang.*, vol. 29, no. 1, pp. 32–44, 2015.

[26] C. Fang, D. Mu, Z. Deng, and Z. Wu, “Word-sentence co-ranking for automatic extractive text summarization,” *Expert Syst. Appl.*, vol. 72, pp. 189–195, Apr. 2017.

[27] K. Ono, K. Sumita, and S. Miike, “Abstract Generation Based on Rhetorical Extraction,” *15th Conf. Comput. Linguist.*, pp. 344–348, 1994.

[28] R. Barzilay and M. Elhadad, “Using Lexical Chains for Text Summarization,” *ACL/EACL Work. Intell. Scalable Text Summ.*, pp. 10–17, 1997.

[29] D. Marcu, “Improving summarization through rhetorical parsing tuning,” *Proc. 6th Work. Very Large Corpora*, pp. 206–215, 1998.

[30] A. Louis, A. Joshi, and A. Nenkova, “Discourse indicators for content selection in summarization,” *11th Annu. Meet. Spec. Interes. Gr. Discourse Dialogue*, no. September 24-25, pp. 147–156, 2010.

[31] A. Ibrahim and T. Elghazaly, “Improve the automatic summarization of arabic text depending on rhetorical structure theory,” *Proc. - 2013 12th Mex. Int. Conf. Artif. Intell. MICAI 2013*, pp. 223–227, 2013.

[32] A. R. Pal and D. Saha, “An Approach to Automatic Text Summarization using WordNet,” *Adv. Comput. Conf. (IACC), 2014 IEEE Int. IEEE, 2014.*, pp. 1169–1173, 2014.

[33] P.-K. Dehkordi, F. Kumarci, and H. Khosravi, “Text Summarization Based on Genetic Programming,” *Int. J. Comput. ICT Res.*, vol. 3, no. 1, pp. 57–64, 2009.

[34] N. Q. Uy, P. T. Anh, T. C. Doan, and N. X. Hoai, “A Study on the Use of Genetic Programming for Automatic Text Summarization,” in *The Fourth International Conference on Knowledge and Systems Engineering, KSE 2012*, 2012, pp. 93–98.

[35] V. Qazvinian, L. S. Hassanabadi, and R. Halavati, “Summarising text with a genetic algorithm-based sentence extraction,” *Int. J. Knowl. Manag. Stud.*, vol. 2, no. 4, p. 426, 2008.

[36] M. Litvak, M. Last, and M. Friedman, “A new Approach to Improving Multilingual Summarization using a Genetic Algorithm,” *48th Annu. Meet. Assoc. Comput. Linguist.*, no. July, pp. 927–936, 2010.

[37] A. M. and S. G. Niladri Chatterjee, “Single Document Extractive Text Summarization Using Genetic Algorithms,” *Third Int. Conf. Emerg. Appl. Inf. Technol. Single 2012*, pp. 225–254, 2012.

[38] R. A. García-hernández and Y. Ledeneva, “LNCS 7914 - Single Extractive Text Summarization Based on a Genetic Algorithm,” *Pattern Recognit.*, vol. 7914, pp. 374–383, 2013.

[39] Y. Kumar Meena and D. Gopalani, “Evolutionary algorithms for extractive automatic text summarization,” *Procedia Comput. Sci.*, vol. 48, no. C, pp. 244–249, 2015.

[40] M. S. Binwahlan, N. Salim, and L. Suanmali, “Swarm Based Text Summarization,” *2009 Int. Assoc. Comput. Sci. Inf. Technol. - Spring Conf.*, pp. 145–150, 2009.

[41] H. Asgari, B. Masoumi, and O. S. Sheijani, “Automatic text summarization based on multi-agent particle swarm optimization,” *2014 Iran. Conf. Intell. Syst.*, pp. 1–5, 2014.

[42] E. Shareghi and L. S. Hassanabadi, “Text summarization with harmony search algorithm-based sentence extraction,” in *CSTST*, 2008.

[43] X. Ji, “Research on the Automatic Summarization Model based on Genetic Algorithm and Mathematical Regression,” in *Electronic Commerce and Security, 2008 International Symposium on*, 2008, pp. 488–491.

[44] M. Mendoza, S. Bonilla, C. Noguera, C. Cobos, and E. León, “Extractive single-document summarization based on genetic operators and guided local search,” *Expert Syst. Appl.*, vol. 41, no. 9, pp. 4158–4169, 2014.

[45] R. M. Aliguliyev, “A new sentence similarity measure and sentence based extractive technique for automatic text summarization,” *Expert Syst. Appl.*, vol. 36, no. 4, pp. 7764–7772, 2009.

[46] A. Abuobieda, N. Salim, Y. J. Kumar, and A. H. Osman, “An Improved Evolutionary Algorithm for Extractive Text Summarization,” in *Intelligent Information and Database Systems: 5th Asian Conference, ACIIDS 2013, Kuala Lumpur, Malaysia, March 18-20, 2013, Proceedings, Part II*, A. Selamat, N. T. Nguyen, and H. Haron, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 78–89.

[47] J. Y. Yeh, H. R. Ke, W. P. Yang, and I. H. Meng, “Text summarization using a trainable summarizer and latent semantic analysis,” *Inf. Process. Manag.*, vol. 41, no. 1, pp. 75–95, 2005.

[48] E. Gonzàlez and M. Fuentes, “A new lexical chain algorithm used for automatic summarization,” *Front. Artif. Intell. Appl.*, vol. 202, no. 1, pp. 329–338, 2009.

[49] W. Song, L. Cheon Choi, S. Cheol Park, and X. Feng Ding, “Fuzzy evolutionary optimization modeling and its applications to unsupervised categorization and extractive summarization,” *Expert Syst. Appl.*, vol. 38, no. 8, pp. 9112–9121, 2011.

[50] L. Suanmali, N. Salim, and M. S. Binwahlan, “Fuzzy genetic semantic based text summarization,” *Proc. - IEEE 9th Int. Conf. Dependable, Auton. Secur. Comput. DASC 2011*, pp. 1184–1191, 2011.

[51] M. S. Binwahlan, N. Salim, and L. Suanmali, “Fuzzy swarm diversity hybrid model for text summarization,” *Inf. Process. Manag.*, vol. 46, no. 5, pp. 571–588, 2010.

[52] R. A. Ghalehtaki, H. Khotanlou, and M. Esmaeilpour, “A combinational method of fuzzy, particle swarm optimization and cellular learning automata for text summarization,” *2014 Iran. Conf. Intell. Syst. ICIS 2014*, vol. 15, no. 1, 2014.

[53] S. A. Babar and P. D. Patil, “Improving Performance of Text Summarization,” *Procedia Comput. Sci.*, vol. 46, pp. 354–363, 2015.

[54] R. Abbasi-ghalehtaki, H. Khotanlou, and M. Esmaeilpour, “Fuzzy evolutionary cellular learning automata model for text summarization,” *Swarm Evol. Comput.*, vol. 30, pp. 11–26, Oct. 2016.

[55] A. Emiro, M. Piamba, and C. Cobos, “Algoritmo para Generación Automática de Resúmenes Extractivos Genéricos de un Documento basado en el Procedimiento de Búsqueda del Pescador,” 2016.

[56] K. K. Bhattacharjee and S. P. Sarmah, “Shuffled frog leaping algorithm and its application to 0/1 knapsack problem,” *Appl. Soft Comput. J.*, vol. 19, pp. 252–263, 2014.

[57] N. Moosavian, “Soccer league competition algorithm for solving knapsack problems,” *Swarm Evol. Comput.*, vol. 20, pp. 14–22, 2015.

[58] D. M. Dunlavy, D. P. O’Leary, J. M. Conroy, and J. D. Schlesinger, “QCS: A system for querying, clustering and summarizing documents,” *Inf. Process. Manag.*, vol. 43, no. 6, pp. 1588–1605, 2007.

[59] A. Abuobieda, N. Salim, Y. J. Kumar, and A. H. Osman, “Opposition Differential Evolution Based Method for Text Summarization,” *Intell. Inf. Database Syst.*, vol. 7802, pp. 487–496, 2013.

[60] M. Hassel, *Resource Lean and Portable Automatic Text Summarization*. 2007.

[61] C. Rijsbergen, “Information Retrieval,” vol. 573, 1979.

[62] C. Lin, “Rouge: A package for automatic evaluation of summaries,” *Proc. Work. text Summ. branches out (WAS 2004)*, no. 1, pp. 25–26, 2004.

[63] G. Salton, A. Wong, and C. S. Yang, “A vector space model for information retrieval,” *Commun. ACM*, vol. 18, no. 11, pp. 613–620, 1975.

[64] A. Singhal, “Modern Information Retrieval: A Brief Overview,” *Bull. IEEE Comput. Soc. Tech. Comm. Data Eng. Vol. 24, No. 4. (2001), pp. 35-42*, vol. 24, pp. 35–42, 2001.

[65] E. V. Tello, “Generación Automática de Resúmenes de Múltiples Documentos,” *Inst. Nac. Astrofísica, Óptica y Electrónica*, 2007.

[66] R. Montiel Soto, R. A. García-Hernández, Y. Ledeneva, and R. Cruz Reyes, “Comparison of three text models for automatic generation of summaries,” *Proces. del Leng. Nat.*, vol. 43, pp. 303–311, 2009.

[67] E. Greengrass, “Information Retrieval: A Survey,” in *Information Retrieval*, 2000, pp. 141–163.

[68] C. D. Manning, P. Ragahvan, and H. Schutze, *An Introduction to Information Retrieval*, no. Manifold-Ranking Based Topic-Focused Multi-Document Summarization. 2008.

[69] S. Luke, *Essentials of Metaheuristics: A Set of Undergraduate Lecture Notes*. 2013.

[70] T. Weise, *Global Optimization Algorithms - Theory and Application*. 2009.

[71] K. Pratt, “Design Patterns for Research Methods: Iterative Field Research,” *Assoc. Adv. Artif. Intell.*, no. 1994, 2009.

[72] C. Lin and E. Hovy, “Identifying Topics by Position,” *Proc. Fifth Conf. Appl. Nat. Lang. Process.*, pp. 283–290, 1997.

[73] C. N. S. Jr, G. L. Pappa, A. A. Freitas, C. A. A. Kaestner, P. Universidade, and P. Pucpr, “Automatic Text Summarization with Genetic Algorithm-Based Attribute Selection,” *Lect. Notes Artif. Intell.*, vol. 3315, pp. 305–314, 2004.

[74] R. M. Alguliev, R. M. Aliguliyev, M. S. Hajirahimova, and C. A. Mehdiyev, “Expert Systems with Applications MCMR : Maximum coverage and minimum redundant text summarization model,” *Expert Syst. Appl.*, vol. 38, no. 12, pp. 14514–14522, 2011.

[75] M. Eusuff, K. Lansey, and F. Pasha, “Shuffled frog-leaping algorithm: A memetic meta-heuristic for discrete optimization,” *Eng. Optim.*, vol. 38, no. 2, pp. 129–154, 2005.

[76] N. Moosavian and B. Kasaee Roodsari, “Soccer league competition algorithm: A novel meta-heuristic algorithm for optimal design of water distribution networks,” *Swarm Evol. Comput.*, vol. 17, pp. 14–24, 2014.

[77] N. Moosavian and B. K. Roodsari, “Soccer League Competition Algorithm, a New Method for Solving Systems of Nonlinear Equations,” *Int. J. Intell. Sci.*, vol. 04, no. 01, pp. 7–16, 2013.

[78] M. Mendoza, C. Cobos, and E. León, “Extractive Single-Document Summarization Based on Global-Best Harmony Search and a Greedy Local Optimizer,” *Adv. Artif. Intell. Its Appl. 14th Mex. Int. Conf. Artif. Intell. MICAI 2015, Cuernavaca, Morelos, Mex. Oct. 25-31, 2015, Proceedings, Part II*, pp. 52–66, 2015.

[79] D. Gillick, “Sentence Boundary Detection and the Problem with the U.S.,” no. June, pp. 241–244, 2009.

[80] G. Salton, “The Smart environment for retrieval system evaluation- advantages and problem areas,” pp. 316–329, 1971.

[81] B. Lovins, “Development of a Stemming Algorithm,” vol. 11, no. June, pp. 22–31, 1968.

[82] M. F. Porter, “An algorithm for suffix stripping,” vol. 40, pp. 211–218, 2006.

[83] H. Huang and B. Zhang, “Text Indexing and Retrieval,” in *Encyclopedia of Database Systems*, L. Liu and M. T. Özsu, Eds. Springer {US}, 2009, pp. 3055–3058.

[84] A.S Foundation, “Apache Lucene Core,” 2011. [Online]. Available: http://lucene.apache.org/.

[85] X. Wan, J. Yang, and J. Xiao, “Manifold-Ranking Based Topic-Focused Multi-Document Summarization,” *Proc. 20th Int. Jt. Conf. Artifical Intell.*, vol. IJCAI 2007, pp. 2903–2908, 2007.

[86] J. Steinberger and K. Jezek, “Sentence compression for the LSA-based summarizer,” *Proc. 7th Int. Conf. Inf. Syst. Implement. Model.*, pp. 141–148, 2006.

[87] R. M. Aliguliyev, “Performance evaluation of density-based clustering methods,” *Inf. Sci. Comput. Sci. Intell. Syst. Appl. An Int. J.*, vol. 179, no. 20, pp. 3583–3602, 2009.