UNIVERSIDAD JUÁREZ AUTÓNOMA DE TABASCO

DIVISIÓN ACADÉMICA DE INFORMÁTICA Y SISTEMAS

INTERFAZ PARA EL AJUSTE DE PARÁMETROS DEL ALGORITMO BASADO EN EL FORRAJEO DE BACTERIAS MODIFICADO

Trabajo recepcional bajo la modalidad de Tesis

Que para obtener el grado de

Licenciado en Sistemas Computacionales

Presenta:

José Adrian García López

Director:

Dra. Betania Hernández Ocaña

Cuerpo Académico:

Inteligencia Artificial

Líneas de Generación y Aplicación del Conocimiento:

Representación y manejo del conocimiento

Cunduacán, Tabasco Julio 2019

UNIVERSIDAD JUÁREZ AUTÓNOMA DE TABASCO

DIVISIÓN ACADÉMICA DE INFORMÁTICA Y SISTEMAS

INTERFAZ PARA EL AJUSTE DE PARÁMETROS DEL ALGORITMO BASADO EN EL FORRAJEO DE BACTERIAS MODIFICADO

Trabajo recepcional bajo la modalidad de Tesis

Que para obtener el grado de

Licenciado en Sistemas Computacionales

Presenta:

José Adrian García López

Director:

Dra. Betania Hernández Ocaña

Jurado Revisor:

Dra. Juana Canul Reich
M.I.E. Adán García Gómez
Dr. José Hernández Torruco
Lic. Carlos González Zacarías
Dr. Oscar Alberto Chavéz Bosquez

Cuerpo Académico:

Inteligencia Artificial

Oficio Núm.1099/19/DAIS/D
03 de Junio de 2019

Dra. Betania Hernández Ocaña
Profesora Investigadora
Presente

De acuerdo al artículo 72 del Reglamento de Titulación, vigente en nuestra Universidad, me permito designarla Directora del Trabajo Recepcional titulado, **"Interfaz para el ajuste de parámetros del algoritmo basado en el forrajeo de bacterias modificado",** a desarrollarse por él C. José Adrian García López, para obtener el título de Licenciado en Sistemas Computacionales, bajo la modalidad de tesis. No omito manifestarle que con fundamento en el artículo 78 del reglamento antes mencionado, a partir de la fecha expedida tiene como plazo máximo un año para concluir la tesis de lo contrario, los estudiantes deberá optar por otra modalidad de titulación.

Sin otro particular, aprovecho la oportunidad para enviarle un cordial saludo.

Atentamente

UNIVERSIDAD JUAREZ AUTONOMA DE TABASCO

MTE. Oscar Alberto González González

Director

DIVISION ACADEMICA DE INFORMATICA Y SISTEMAS

Recibido
6/Junio/2019

C.c.p. L.C. Carlos González Zacarías.- Coordinador de Estudios Terminales
C.C.P. Tesistas
Archivo
Consecutivo.

MTE.*OAGG/LC*CGZ/VGTV

Carretera Cunduacán-Jalpa Km. 1, Colonia Esmeralda, C.P. 86690. Cunduacán, Tabasco, México.
E-mail: direccion.dais@ujat.mx
Teléfonos: (993) 358 1500 ext. 6727; (914) 336 0616; Fax: (914) 336 0870

F6: Liberación de Director de Tesis 2019

Cunduacán, Tabasco., a 10 de Junio de 2019.

Asunto: Liberación de dirección de tesis.

MTE. Óscar Alberto González González
Director de la DAIS
Presente

Por medio de la presente me permito comunicarle que después de haber realizado las asesorías correspondientes al Trabajo Recepcional: **"Interfaz para el ajuste de parámetros del algoritmo basado en el forrajeo de bacterias modificado",** con folio: 201701-IA06; elaborado por CC. José Adrian García López, de la Licenciatura en Sistemas Computacionales, considero que lo ha concluido satisfactoriamente, por lo que pueden continuar con los trámites de titulación.

Sin otro particular, aprovecho la ocasión para enviarle un cordial saludo.

Atentamente

Dra. Betania Hernández Ocaña

DIVISIÓN ACADÉMICA DE INFORMÁTICA Y SISTEMAS

c.c.p. Lic. Carlos González Zacarías.- Coordinador de Estudios Terminales
Integrantes de la Comisión Revisora
Estudiantes

RECIBIDO 10 JUN 2019 DAIS

F7: Solicitud de Jurado 2019

Cunduacán, Tabasco, a 10 de Junio de 2019.

Asunto: Solicitud de Jurado

MTE. Óscar Alberto González González
Director de la DAIS
Presente

Por este medio nos permitimos informarle que el trabajo recepcional: "Interfaz para el ajuste de parámetros del algoritmo basado en el forrajeo de bacterias modificado", *bajo la modalidad de* ***Tesis***, ha sido liberado por mi directora: Dra. Betania Hernández Ocaña, por lo que en atención a ello me dirijo a usted con la finalidad de solicitarle tenga a bien nombrar al jurado para que evalúe el citado trabajo.

Sin otro particular, aprovecho la ocasión para enviarle un cordial saludo.

Atentamente

José Adrian García López

DIVISIÓN ACADÉMICA
DE INFORMÁTICA Y SISTEMAS
COORDINACIÓN DE ESTUDIOS TERMINALES

Licenciatura:	Sistemas Computacionales
Matrícula:	152h8071
Domicilio:	Carretera el Aguacate
Localidad:	R/a. Culico 2da. Sección, Cunduacán, Tabasco.
Teléfono:	9141200497
E-mail:	joseadrian_g97@hotmail.com

RECIBIDO
10 JUN 2019
DAIS

c.c.p. Lic. Carlos González Zacarias. - Coordinador de Estudios Terminales.
Estudiantes.

Cunduacán, Tabasco, Junio 18 de 2019

En la Universidad Juárez Autónoma de Tabasco, de acuerdo al artículo 85 del Reglamento de Titulación vigente, se revisó el trabajo recepcional titulado, **"Interfaz para el ajuste de parámetros del algoritmo basado en el forrajeo de bacterias modificado"**, a desarrollarse por el C. José Adrian García López, para obtener el título de Licenciado en Sistemas Computacionales, bajo la modalidad de Tesis.

Los integrantes del jurado, después de revisar el trabajo, lo declararon aceptado.

Dr. Óscar Alberto Chávez Bosquez
Profesor Investigador
D.A.I.S. – U.J.A.T
P r e s e n t e.

Dra. Juana Canul Reich
Profesora Investigadora
D.A.I.S. – U.J.A.T
P r e s e n t e.

Dr. José Hernández Torruco
Profesor Investigador
D.A.I.S. – U.J.A.T
P r e s e n t e.

MIE. Adán García Gómez
Profesor Investigador
D.A.I.S. – U.J.A.T
P r e s e n t e.

LC. Carlos González Zacarías
Profesor Investigador
D.A.I.S. – U.J.A.T
P r e s e n t e.

Carretera Cunduacán-Jalpa Km. 1, Colonia Esmeralda, C.P. 86690. Cunduacán, Tabasco, México.
E-mail: direccion.dais@ujat.mx
Teléfonos: (993) 358 1500 ext. 6727; (914) 336 0616; Fax: (914) 336 0870

UNIVERSIDAD JUÁREZ AUTÓNOMA DE TABASCO

"ESTUDIO EN LA DUDA. ACCIÓN EN LA FE"

DAIS
11111000011

INSTITUTO JUÁREZ
1879-2019

Oficio Núm. 1290/2019/DAIS/D
18 de Junio de 2019

C. José Adrian García López

En virtud de que cumple satisfactoriamente los requisitos establecidos en el Reglamento de Titulación vigente en la Universidad, informo a Usted que se autoriza la impresión del trabajo recepcional **"Interfaz para el ajuste de parámetros del algoritmo basado en el forrajeo de bacterias modificado",** para presentar Examen Profesional y obtener el título de Licenciado en Sistemas Computacionales, bajo la modalidad de Tesis.

Sin otro particular aprovecho la oportunidad para saludarle.

Atentamente

UNIVERSIDAD JUAREZ AUTONOMA DE TABASCO

MTE. Oscar Alberto González González

Director

DIVISION ACADEMICA DE INFORMATICA Y SISTEMAS

C.c.p. Director de Tesis 1: Dra. Betania Hernández Ocaña

Archivo
Consecutivo

MTE.*OAGG/LC*CGZ/VGTV

Miembro CUMEX desde 2008
Consorcio de Universidades Mexicanas
UNA ALIANZA DE CALIDAD POR LA EDUCACIÓN SUPERIOR

Carretera Cunduacán-Jalpa Km. 1, Colonia Esmeralda, C.P. 86690. Cunduacán, Tabasco, México.
E-mail: direccion.dais@ujat.mx
Teléfonos: (993) 358 1500 ext. 6727; (914) 336 0616; Fax: (914) 336 0870

Cesión de Derechos

Villahermosa Tabasco, a 08 de julio de 2019.

A quien corresponda:

Los abajo firmantes, en nuestra calidad de alumnos de la DAIS, en esta ciudad de Cunduacán, Tabasco, declaramos que el trabajo recepcional desarrollado en la Universidad Juárez Autónoma de Tabasco titulado, **"Interfaz para el ajuste de parámetros del algoritmo basado en el forrajeo de bacterias modificado"** es de nuestra autoría intelectual y por lo tanto cedemos todos los **derechos** sobre este proyecto a la **"Dra. Betania Hernández Ocaña"**, quien participó como director en el desarrollo del mismo y al cual relevamos de cualquier sanción y asumimos responder a cualquier reclamo de derechos de autor ante las autoridades competentes.

Atentamente

Autores:

José Adrian García López	Ra. Culico 2da. Sección, Cunduacán, Tabasco.	
Nombre	**Domicilio**	**Firma autógrafa**

Carta de Autorización

A quien corresponda:

El que suscribe, autoriza por medio del presente escrito a la Universidad Juárez Autónoma de Tabasco para que utilice tanto física como digitalmente la tesis de grado denominada, **"Interfaz para el ajuste de parámetros del algoritmo basado en el forrajeo de bacterias modificado"**, de la cual soy autor y titular de los Derechos de Autor.

La finalidad del uso por parte de la Universidad Juárez Autónoma de Tabasco de la tesis antes, mencionada, será única y exclusivamente para difusión, educación y sin fines de lucro; autorización que se hace de manera enunciativa más no limitativa para subirla a la Red Abierta de Bibliotecas Digitales (RABID) y a cualquier otra red académica con las que la Universidad tenga relación institucional.

Por lo antes manifestado, libero a la Universidad Juárez Autónoma de Tabasco de cualquier reclamación legal que pudiera ejercer respecto al uso y manipulación de la tesis mencionada y para los fines estipulados en éste documento.

Se firma la presente autorización en la ciudad de Villahermosa, Tabasco a los 08 días del mes de julio del año 2019.

Autorizo

José Adrian García López

Agradecimientos

A Dios porque está conmigo a cada paso que doy, cuidándome y
dándome fortaleza para continuar.

A mis padres quienes a lo largo de la vida están conmigo ofreciéndome bienestar y educación depositando su entera confianza en cada reto que se me presenta.

A toda mi familia, a mis amigos y todas esas personas especiales
que están conmigo en todo momento.

A todos profesores a quienes les debo gran parte de mis conocimientos, gracias a su paciencia y enseñanza.

A la Dra. Betania Hernández Ocaña, Dr. Oscar Alberto Chavéz Bosquez,
Dr. José Hernández Torruco y la Dra. Juana Canúl Reich por su
participación en la elaboración de la tesis y la revisión para la
obtención de un buen trabajo.

Finalmente, un eterno agradecimiento a la prestigiosa Universidad Juárez Autónoma de Tabasco la cual abre sus puertas a jóvenes como nosotros, preparándonos para un futuro competitivo y formándonos como personas de bien.

Dedicatorias

A Dios por haberme dado inteligencia,
paciencia y ser el mejor guía durante mi vida.

A mis padres que me han enseñado que los sueños
son posibles de alcanzar y que gracias a ellos
tengo el esmero de poder luchar por mis
sueños gracias al amor que me brindan.

.

A mis hermanos, Dora María y Santo Azael por creer
en mí y darme motivo de superación.

A la Dra. Betania Hernández Ocaña por su motivación, apoyo
y consejos para la realización de esta tesis.

A mi amigo Juan Carlos Vicente Martínez por haberme ayudado
a lograr nuestro gran objetivo con gran perseverancia.

A todos mis amigos y aquellas personas especiales por darme
animo de seguir adelante y estar conmigo dando
su apoyo incondicionalmente.

Resumen

Existen algoritmos de inteligencia colectiva basado en el forrajeo de bacterias, pero con la dificultad de que no cualquier usuario pueda utilizar debido a la interacción directa con el código que se necesita para poder calibrar los parámetros propios del algoritmo. Tomando en cuenta la sensibilidad de los parámetros, la calibración requiere de mucho tiempo y conocimientos de programación.

En este trabajo de tesis se diseña y se programa una interfaz gráfica en la cual el usuario final pueda calibrar de una manera más rápida y sin necesidad de manipular las líneas de código del algoritmo. Este diseño fue implementado utilizando diagramas UML y como resultado se obtiene una interfaz aplicada a un problema conocido como resorte tensión/compresión, donde el usuario puede configurar la ejecución y calibrar los parámetros propios del algoritmo que son sugeridos para una buena calibración. Además, la interfaz tiene como función mostrar los resultados obtenidos después de la ejecución, exportarlos a una hoja de cálculo de Microsoft Excel, mostrar gráfica de convergencias y estadísticas básicas que son necesarias para que el usuario final tome una decisión de los resultados considerados a utilizar. Se realizaron 30 ejecuciones independientes del algoritmo TS-MBFOA con el problema de prueba y fue comparado con el algoritmo MBFOA (versión anterior), obteniendo que TS-MBFOA tiene mejor resultados con menor número de evaluaciones.

Introducción

Los algoritmos de inteligencia colectiva son algoritmos que simulan el comportamiento de especies simples e inteligentes, como las abejas, hormigas, bacterias, etc. Se trata de un grupo diferente a los algoritmos populares llamados evolutivos. Entre los algoritmos de inteligencia colectiva encontramos el algoritmo de forrajeo de baterías, el cual permite que las bacterias maximicen su concentración de nutrientes en un espacio de búsqueda, todo en cuatro pasos: quimiotáxis (nado-giro), agrupamiento, reproducción y eliminación-dispersión. Este algoritmo ha sido estudiado y analizado para resolver problemas de optimización sin restricciones y con restricciones usando reglas de factibilidad que no requiere el aumento de parámetros a calibrar por el usurario final. Sin embargo, cabe destacar que este algoritmo tiene muchos parámetros a calibrar y uno de los más sensibles es el tamaño de paso.

El algoritmo de forrajeo de bacterias ha sido poco estudiado e implementado para la solución de problemas de optimización, además de que no se cuenta con un código abierto en donde los usuarios puedan consultar, administrarlo o configurarlo para su propia necesidad.

El principal objetivo de esta tesis es desarrollar una interfaz donde el usuario final pueda calibrar los parámetros del algoritmo, determinar la duración generacional del algoritmo y visualizar los resultados usando gráfico de convergencia y estadísticas básicas, entre otras métricas. Esta interfaz es diseñada a través de diagramas UML, además de la implementación en Matlab, una herramienta de software matemático que ofrece un entorno de desarrollo integrado con un lenguaje de programación M.

Índice general

Índice de ilustraciones.....xv
Índice de tablas.....xvi
Capítulo I. Generalidades.....1
1.1 Antecedentes.....1
1.2 Planteamiento del problema.....5
1.2.1 Definición del problema.....5
1.2.2 Delimitación de la investigación.....6
1.2.3 Pregunta de investigación.....7
1.3 Objetivos.....7
1.3.1 Objetivo general.....7
1.3.2 Objetivos específicos.....8
1.4 Justificación.....8
1.5 Metodología utilizada.....9
Capítulo II. Marco teórico.....14
2.1 Marco referencial.....14
2.1.1 BFOA generando un Menú nutricional.....14
2.1.2 Optimización de programación de sistemas de fabricación flexibles utilizando enfoques convencionales y evolutivos.....15
2.1.3 Planificación de estrategias de la ruta de un barco para evitar la colisión usando BFOA.....16
2.1.4 Controlador de orden factorial basado en BFOA para sistemas de control de generación automática de área múltiple con almacenamiento de energía capacitiva.....17
2.1.5 Análisis comparativo de la técnica de doble marca de agua basada en imágenes médicas seguras utilizando BFOA.....18
2.1.6 Reconocimiento de imágenes de huellas dactilares utilizando BFOA.....19

2.1.7 Diagnóstico del cáncer de pulmón mediante la fusión de las máquinas de vectores de soporte y la red neuronal de propagación hacia atrás 20
2.1.8 Algoritmo hibrido con un adaptativo BFOA y DWT-SVD marca de agua con encriptación 21
2.2 Marco conceptual 22
2.2.1 Algoritmos evolutivos 22
2.2.2 Algoritmos de inteligencia colectiva 23
2.2.3 Componentes de un problema de optimización 24
2.3 Marco tecnológico 27
2.3.1 Lenguaje de programación 27
2.3.2 GUI en Matlab 27
2.3.3 Construcción interactiva de una GUI 27
2.3.4 Programación de una GUI 29
2.4 Marco legal 30
Capítulo III. Algoritmos basados en el forrajeo de bacterias 31
3.1 Algoritmos de Optimización basado en el Forrajeo de Bacterias (BFOA) 31
3.2 Algoritmos de Optimización basado en el Forrajeo de Bacterias Modificado (MBFOA). 35
3.3 TS-MBFOA 38
Capítulo IV. Aplicación de la metodología y desarrollo 43
4.1 Modelo de desarrollo 43
4.1.1 Diagrama de clases 43
4.1.2 Diagrama de caso de uso 44
4.1.3 Diagrama de secuencia 46
4.1.4 Diagrama de actividad 47
4.1.5 Diagrama de tiempo 49
4.1.6 Diagrama de despliegue 50
4.2 Desarrollo de la Interfaz Gráfica de Usuario (GUI) 51
4.2.1 Pantalla principal 52
4.2.2 Nueva ejecución 55

4.2.3 Resultados 57
4.2.4 Acerca de 59
Capítulo V. Pruebas y resultados 61
5.1 Problema de prueba 61
5.2 Medidas de rendimiento 62
5.3 Configuración y calibración del algoritmo 63
5.4 Resultados 65
5.5 Convergencia 69
5.6 Prueba no paramétrica *Wilcoxon Signed Rank Test* 70
Capítulo VI. Conclusiones y trabajos futuros 72
6.1 Conclusiones 72
6.2 Trabajos a futuros 73
Bibliografía 74
Glosario 81

Índice de ilustraciones

Ilustración 2.1 Generador de Menús. 15
Ilustración 2.2 GUI correspondiente a secuencias de trabajo usando BFOA. 16
Ilustración 2.3 Interfaz para evitar la colisión de un barco usando BFOA. 17
Ilustración 2.4 Grafica generada en Matlab para funciones de señales de entrada. 18
Ilustración 2.5 GUI propuesta para cargar imagen médica usando BFOA. 19
Ilustración 2.6 GUI reconocedor de huellas dactilares utilizando BFOA. 20
Ilustración 2.7 GUI para la detención de cáncer del pulmón. 21
Ilustración 3.1 Pseudocódigo de BFOA. 34
Ilustración 3.2 Pseudocódigo de MBFOA. 37
Ilustración 3.3 Comportamiento del nado de exploración. 39
Ilustración 3.4 Comportamiento del nado de explotación. 40
Ilustración 3.5 Comportamiento del nado de agrupamiento. 41
Ilustración 3.6 Bacterias después de un nado de exploración, un nado de explotación y un agrupamiento. 41
Ilustración 3.7 Procesos generales de TS-MBFOA. 42
Ilustración 3.8 Pseudocódigo de TS-MBFOA. 42
Ilustración 4.3 Diagrama de secuencia. 46
Ilustración 3.4 Diagrama de actividades. 48
Ilustración 4.5 Diagrama de tiempo. 50
Ilustración 4.7 Pantalla principal del software. 52
Ilustración 4.8 Menú de opciones del software. 53
Ilustración 4.9 Botones de acceso directo del software. 54
Ilustración 4.10 Logo del software. 54
Ilustración 4.11 Interfaz de configuración y calibración del algoritmo. 55
Ilustración 4.12 Mensaje de error. 56
Ilustración 4.13 Barra de progreso. 57
Ilustración 4.14 Interfaz de resultados. 58
Ilustración 4.15 Exportar archivo. 59
Ilustración 4.16 Interfaz de información del software. 60
Ilustración 5.1 Representación del problema de diseño a resolver. 62
Ilustración 5.2 Configuración de la ejecución. 64
Ilustración 5.3 Calibración de parámetros. 65
Ilustración 5.4 Estadísticas de la mejor solución. 66
Ilustración 5.5 Interfaz general de resultados obtenidos de la función de prueba resorte tensión/compresión. 68
Ilustración 5.6 Gráfica de convergencia. 69

Índice de tablas

Tabla 3.1 Descripción de los parámetros del BFOA. 34
Tabla 3.2 Descripción de los parámetros del MBFOA. 37
Tabla 4.1 Descripción de una nueva ejecución. 45
Tabla 4.2 Descripción de una configuración guardada. 45
Tabla 4.3 Descripción de la GUI. 45
Tabla 5.1 Resultados finales de TS-MBFOA de las 30 ejecuciones independientes realizadas en el problema del resorte de tensión/compresión. 67
Tabla 5.2 Comparación del algoritmo TS-MBFOA y MBFOA base a las estadísticas básicas. 68
Tabla 5.3 Resultados utilizados para comparar TS-MBFOA y MBFOA en la prueba *Wilcoxon signed rank test*..... 71

Capítulo I. Generalidades

En este capítulo se destacan las generalidades correspondientes a la evolución del algoritmo TS-MBFOA y sus implementaciones en diferentes áreas, además se plantea el problema con sus delimitaciones, los objetivos que se alcanzaron y la justificación.

1.1 Antecedentes

El computo bio-inspirado ha venido progresando sobre los métodos de computación clásicos, utilizando procedimientos que reproducen ciertas propiedades inspiradas en la naturaleza, como el comportamiento de ciertas especies animales en la búsqueda de alimento o refugio, con el fin de diversificar los resultados obtenidos en la medida que se obtengan mejores resultados (Flórez, Díaz, Gómez, Bautista & Delgado, 2018).

Actualmente, los ingenieros se asisten con herramientas como Diseño Asistido por Computadora (CAD) para diseñar sistemas. Por ejemplo, el diseño de máquinas como automóviles, aviones, aparatos electrodomésticos, retroexcavadoras, cámaras fotográficas automáticas, entre otros más que son considerados como problemas complejos, debido a lo difícil que es resolverlos, ya que involucran muchos objetivos o compromisos que generalmente se encuentran en conflicto (Papalambros & Wilde, 2000). Esto implica tener experiencia y conocimiento de las herramientas computacionales, ya que no son fáciles de configurar.

Actualmente, existen técnicas que buscan soluciones buenas en tiempos razonables conocidas como metaheurísticas. Algunas de estas técnicas emulan o simulan procesos naturales o evolutivos que se agrupan en los llamados algoritmos bio-inspirados (Bosman, 2007), los cuales son parte del cómputo bio-inspirado. Estos algoritmos surgen con la motivación de mejorar máquinas de búsqueda y resolver problemas de optimización a través de la simulación de procesos inteligentes y colaborativos. Para resolver dichos problemas es necesario simular procesos naturales, los algoritmos bio-inspirados engloban un conjunto de algoritmos que permiten esta simulación (Engelbrecht, 2006).

De acuerdo al fenómeno natural en que basan su diseño se clasifican en dos grupos: los Algoritmos Evolutivos (AE) cuyo funcionamiento se basa en emular el proceso de evolución

natural y la supervivencia del más apto (Eiben & Smith, 2003), y los Algoritmos de Inteligencia Colectiva (AIC) los cuales basan su funcionamiento en comportamientos sociales y cooperativos de organismos simples como insectos y aves (Engelbrecht, 2006).

Uno de los algoritmos más novedosos del área de Inteligencia Colectiva es el que emula el comportamiento de forrajeo a nivel de bacterias. Las ideas iníciales de este algoritmo basado en bacterias fueron propuestas por Bremermann (1974) y luego aplicadas por Bremermann y Anderson (1989) en el entrenamiento de una red neuronal. Sin embargo, existen AIC más recientes que aún no han sido estudiados con profundidad como el Algoritmo de Optimización basado en el Forrajeo de Bacterias (BFOA por sus siglas en inglés, *Bacterial Foraging Optimization Algorithm*). A partir de las ideas iníciales de Bremermann, Passino (2002) propone BFOA, en el cual cada bacteria trata de maximizar su energía obtenida por unidad de tiempo empleada en el proceso de forrajeo, donde también evade sustancias nocivas. Más aún, las bacterias se pueden comunicar entre sí mediante la segregación de sustancias. En el BFOA se tienen cuatro procesos principales:

1. Quimiotáxis (nado-giro).
2. Agrupamiento.
3. Reproducción.
4. Eliminación-dispersión.

Las bacterias son soluciones potenciales al problema, y su ubicación representa los valores de las variables de decisión del problema. Las bacterias pueden moverse (generar nuevas soluciones) mediante el ciclo quimiotáxico; se genera además un movimiento mediante la atracción que ejercen soluciones en zonas prometedoras a otras soluciones del espacio de búsqueda, se permite la reproducción de las mejores soluciones y finalmente se eliminan del cúmulo aquellas bacterias localizadas en zonas de baja calidad.

En MBFOA (por sus siglas en inglés *Modified Bacterial Foraging Optimization Algorithm*), un algoritmo basado en BFOA, se tiene un mecanismo para el manejo de las restricciones basado en reglas de factibilidad (Deb, 2000) y una disminución de parámetros respecto a los del BFOA original. Además, el MBFOA se aplicó a un conjunto de problemas de diseño en ingeniería química e ingeniería mecánica obteniendo resultados competitivos. Por otro lado, el MBFOA se

aplicó a la resolución de un problema de diseño mecánico bi-objetivo en presencia de restricciones en (Mezura-Montes, Portilla-Flores & Hernández-Ocaña, 2011).

Las modificaciones realizadas a MBFOA para mejorar su rendimiento en Problemas de Optimización Numérica con Restricciones (PONR*)* se llamó IMBFOA (por sus siglas en inglés *Improved Modified Bacterial Foraging Optimization Algorithm*), la cual implementa un mecanismo de sesgo para crear la población inicial, dos operadores de nado, un operador diferente de dirección aleatoria, tamaños de paso dinámico y un buscador local. Esta propuesta se integra a MBFOA uno de los métodos de programación matemática más exitosos llamado *Sequential Quadratic Programming* (SQP, por sus siglas en inglés) (Powell, 1978) como buscador local con el objetivo de acercar o introducir a las bacterias dentro de la región factible o sacar de óptimos locales a las bacterias y moverlas a otro lugar dentro de la región factible (Hernández-Ocaña, Mezura-Montes & Pozos-Parra, 2016).

La propuesta TS-MBFOA (por sus siglas en inglés *Two-Swim Modified Bacterial Foraging Optimization Algorithm*), intercala dos nados en el proceso quimiotáxico, el primero es el nado original a excepción del tamaño de paso el cual se propone sea aleatorio y el segundo nado incluye el operador de mutación usado en los algoritmos evolutivos, la inclusión de ambos para mejorar la capacidad de exploración y explotación del algoritmo (Hernández-Ocaña, Pozos-Parra, Mezura-Montes, Portilla-Flores, Vega-Alvarado & Calva-Yáñez, 2016). El algoritmo derivado TS-MBFOA normalizado llamado NTS-MBFOA, por sus siglas en inglés, fue propuesto para resolver problemas de optimización numérica con restricciones de diseñó en Ingeniería Mecánica. La propuesta hace uso de una técnica sencilla de normalización que no incluye más parámetros al algoritmo. Esta misma fue probada en los problemas de optimización conocidos como: resorte de tensión/compresión y recipiente de presión (Hernández-Ocaña, Ovando-Bautista, Aquino-De La Cruz & Gaitán-Capetillo, 2018).

Los algoritmos metaheurísticos poseen sus propios parámetros que corresponden a cada uno de sus procesos, dichos parámetros son independientes de las variables del problema a resolver y deben ser calibrados o ajustados de tal forma que permitan un mejor rendimiento del algoritmo, es decir, que el algoritmo encuentre soluciones optimas globales con un costo computacional moderado, no excesivo. El tiempo de ejecución está implícito. Encontrar la calibración perfecta

de los parámetros de un algoritmo es una tarea difícil de completar y aún más si el ajuste de parámetros es directo en las líneas de código del algoritmo sin ayuda de una interfaz gráfica.

BFOA ha sido escasamente utilizado en la solución de problemas de optimización numérica con restricciones, en la mayoría de los casos los usuarios hacen uso del algoritmo a nivel de código. Una interfaz para la calibración de parámetros de algún modo permite que el usuario final realice esta tarea de manera más rápida y con el mínimo esfuerzo.

La evolución de la interfaz gráfica surge en el año 1973 en el centro de investigación Xerox Alto, el objetivo de encontrar un modelo óptimo de interacción humano-computadora, pasa por un proceso de madurez donde se definen sus elementos básicos, para acabar convirtiéndose en un producto de uso estético dentro del software, donde la interfaz más allá de un medio de interacción óptimo, se transforma en un objeto inteligente abierto a los procesos de personalización por parte del usuario.

Con base en las investigaciones, se podría definir la Interfaz Gráfica de Usuario (GUI, por sus siglas en inglés), en el contexto de la interacción humano-computadora, como un artefacto interactivo, que por su diseño y a través de ciertas interfaces humanos, posibilita la interacción de una persona con el sistema informático, haciendo uso de las gramáticas visuales y verbales.

La historia de la interfaz gráfica ha estado marcada en su evolución por dos factores decisivos: la investigación y el negocio. El origen de su nacimiento está en la búsqueda de un método de interacción amigable con las computadoras que superen la interfaz de línea (Expósito, 2006).

El objetivo principal de este trabajo de tesis fue realizar una interfaz para el ajuste de los parámetros del algoritmo basado en el forrajeo de bacterias TS-MBFOA de manera independiente al problema en particular a resolver. La propuesta derivada fue probada en un algoritmo de diseño ingenieril conocido como resorte de tensión/compresión. Los resultados obtenidos se validaron usando estadísticas básicas como media, desviación estándar, mejor y peor resultado.

1.2 Planteamiento del problema

1.2.1 Definición del problema

En áreas como Medicina, Ingeniería, Mecatrónica, Aeronáutica, entre otras, se requieren del uso de herramientas alternativas para solucionar problemas optimización complejos. Los algoritmos bio-inspirados han sido de mucho interés para los investigadores y profesionales al obtener resultados competitivos y favorables al solucionar problemas de optimización global.

Una de las estrategias de inteligencia colectiva es el algoritmo BFOA, que simula el forrajeo de bacterias. Esta propuesta ha sido modificada y aplicada exitosamente para resolver problemas de pronóstico (Majhi & Panda, 2007), reducción de pérdidas de transmisión (Mishra, Redely, Rao & Santosh, 2007) e identificación de sistemas dinámicos no lineales (Majhi & Panda, 2007). También ha sido combinado con otros algoritmos para solucionar problemas multimodales (Biswas, Dasgupta, Das & Abraham, 2007). Recientemente ha sido utilizada para resolver problemas de diseños ingenieril conocido como resorte de tensión/compresión, mecatrónica (Hernández-Ocaña, et al., 2016) y menús nutritivos (Hernández-Ocaña, Chavéz-Bosquez, Hernández-Torruco, Canúl-Reich & Pozos-Parra, 2018) con éxito. Sin embargo, para lograr diferentes resultados óptimos es necesario la calibración o ajuste de parámetros propios del algoritmo y así dar un conjunto de soluciones factibles al usuario final de las cuales tomará la mejor de acuerdo a sus necesidades, facilitando con esto la toma de decisiones.

Para el ajuste de parámetros del algoritmo es necesario interactuar directamente con las líneas de código y se debe tener conocimiento programación; esto implica un consumo de tiempo excesivo para quien no conoce cómo está estructurado el algoritmo. La problemática que se desea resolver en este proyecto es la facilidad del ajuste de parámetros del algoritmo por medio de una interfaz, que permita modificar estos datos de manera rápida y así ahorrar tiempo en la ejecución del algoritmo, lo cual conlleva a la generación de soluciones del problema a resolver. Con base a investigaciones llevadas a cabo sobre la existencia de alguna interfaz para la calibración de BFOA, se encontró que solo existen interfaces diseñadas para resolver un problema en específico tales como: Algoritmo de Optimización del Forrajeo de Bacterias para planificación de Menú (Hernández-Ocaña, et al., 2018), una herramienta Matlab GUI para la optimización de

programación de sistemas de fabricación flexibles utilizando enfoques convencionales y evolutivos (Kumar, Veeranna, Durgaprasad & Sarma, 2013), planificación de estrategias de la ruta de un barco para evitar la colisión usando BFOA (Hongdan, Sheng & Lanyong, 2015), entre otras, la cual son presentadas en capitulo II.

En este proyecto se implementó una interfaz gráfica de usuario que formará parte de un sistema integral el cual permita la calibración de TS-BFOA en problemas de las diferentes áreas.

1.2.2 Delimitación de la investigación

1.2.2.1 Alcances

- La interfaz diseñada contempla el ajuste de los parámetros del algoritmo para su ejecución: número de bacterias, tamaño de paso, factor de escalamiento, ciclo quimiotáxico, bacterias a reproducir, frecuencias del ciclo de reproducción, bacterias a eliminar y el número de evaluaciones.
- La interfaz visualiza el gráfico de convergencia y los resultados del algoritmo en la solución del problema de optimización como: función objetivo, suma de violación de restricciones, tiempo de ejecución y el valor de las variables de decisión.
- La interfaz permite que el usuario final pueda detener y ejecutar el algoritmo.
- EL diseño de diagramas UML permite un mejor diseño y programación de la interfaz.
- El algoritmo encuentra en todos sus experimentos soluciones factibles.
- El algoritmo con la interfaz fue probado en el problema de minimización del resorte de tensión/compresión.
- Los resultados obtenidos fueron evaluados usando medidas de estadísticas básica como son: media, desviación estándar, mejor y peor resultado para conocer la calidad y consistencia del algoritmo, así como la prueba no paramétrica llamada *Wilcoxon Signed Rank Test*.
- Los gráficos de convergencia fueron usados para la interpretación del rendimiento del algoritmo.

1.2.2.2 Limitaciones

- La interfaz diseñada es solo para el algoritmo llamado TS-MBFOA, el cual se utilizó tal cual fue propuesto por sus autores.
- La interfaz elaborada solo contempla el ajuste o calibración de los principales parámetros del algoritmo.
- No se realizó una revisión del estado del arte de los algoritmos bio-inspirados que optimizan el problema del resorte de tensión/compresión.
- No se realizaron estudios sobre sensibilidad de parámetros al algoritmo basado en el forrajeo de bacteria.
- No se realizó estudios del rendimiento en línea del algoritmo, esto es el análisis del comportamiento del algoritmo en una ejecución; en este trabajo solo se estudió el rendimiento del algoritmo a través de los resultados obtenidos visualmente en la gráfica de convergencia.

1.2.3 Pregunta de investigación

Con el objetivo de construir una GUI se plantea la siguiente pregunta de investigación:

¿Cuál es el rendimiento del algoritmo TS-MBFOA al ser desarrollado y ejecutado usando una GUI, al resolver el problema de optimización numérico con restricciones conocido como resorte tensión/compresión?

1.3 Objetivos

1.3.1 Objetivo general

Diseñar una interfaz gráfica que permita el ajuste eficiente de parámetros propios del algoritmo basado en el forrajeo de las bacterias TS-MBFOA en la solución de problemas de optimización global con restricciones.

1.3.2 Objetivos específicos

- Diseñar la interfaz para el módulo de ajuste de parámetros el algoritmo basado en el forrajeo de bacterias.
- Implementar los diagramas UML para comprender la interfaz propuesta.
- Implementar el algoritmo TS-MBFOA para resolver el problema de resorte de tensión/compresión y determinar la combinación de parámetros que permita el mejor desempeño del algoritmo.
- Validar los resultados con las medidas de rendimiento estadístico, prueba no paramétrica, *Wilcoxon Signed Rank Test* y gráfico de convergencia.
- Adaptar el algoritmo TS-MBFOA a la interfaz gráfica.
- Comparar el algoritmo TS-MBFOA con la propuesta MBFOA.

1.4 Justificación

Buscar soluciones de buena calidad en tiempos razonables para problemas de optimización numérica con restricciones y herramientas como la programación matemática, programación asistida por computadora y las metaheurísticas puede volverse una tarea compleja si no se tiene un conocimiento previo de estas herramientas. Las metaheurísticas son algoritmos que favorecen la solución de problemas de optimización de manera rápida dando al usuario un conjunto de soluciones factibles que facilitan la toma de decisiones, algo que la programación matemática y la programación asistida por computadora no permite. Las metaheurísticas son inspiradas en la naturaleza se dividen en dos grandes grupos, en el primero se emulan o simulan procesos naturales o evolutivos que se agrupan en los llamados Algoritmos Evolutivos, y el segundo grupo se encuentran algoritmos que emulan el comportamiento colaborativo de ciertas especies simples e inteligentes, dicho grupo es llamado Algoritmos de Inteligencia Colectiva. Una de las metaheurísticas de Inteligencia Colectiva es el algoritmo basado en el forrajeo de bacterias llamado TS-MBFOA el cual consiste en realizar cuatros proceso; quimiotáxis (nado-giró), agrupamiento, reproducción y eliminación-dispersión, estos procesos son soluciones potenciales a múltiples problemas. Para la calibración de parámetros del algoritmo TS-MBFOA se tiene la necesidad de interactuar directamente con las líneas de código y tener conocimiento

computacional. Sin embargo, para dar solución a esta problemática en este trabajo se desea facilitar la calibración de parámetros del algoritmo por medio de una interfaz gráfica de usuario, que permita modificar estos datos de manera rápida y así ahorrar tiempo en la ejecución del algoritmo. Al ser ejecutado el algoritmo será posible visualizar sus resultados como la función objetivo y gráfico de convergencia, partiendo de estos resultados como soluciones factibles para la solución de problemas de optimización. El algoritmo ya implementado en GUI está desarrollado con el fin de que sea parte de un sistema integral el cual podrá dar posibles soluciones a problemas complejos en múltiples áreas como: Medicina, Ingeniería, Mecatrónica, Aeronáutica, entre otras.

1.5 Metodología utilizada

Según la literatura especializada todo trabajo de investigación se sustenta en dos enfoques principales: el enfoque cuantitativo y el enfoque cualitativo, los cuales de manera conjunta forman un tercer enfoque: el enfoque mixto. Los enfoques principales emplean procesos cuidadosos, metódicos y empíricos en su esfuerzo para generar conocimiento, sin embargo, aunque las aproximaciones cuantitativa y cualitativa comparten esas estrategias generales, cada una tiene sus propias características (Hernández-Sampieri, Fernández-Collado & Baptista, 2010).

Con relación a la implementación de esta tesis se ha utilizado un enfoque cuantitativo debido a que se utilizó medidas estadísticas como mejor, peor, media, mediana y desviación estándar de los resultados obtenidos de la implementación del algoritmo TS-MBFOA con una GUI, además de la prueba no paramétrica llamada *Wilcoxon Signed Rank Test*. Los resultados se han concentrado en tablas y gráficos de convergencia. Dicho algoritmo parte de una población inicial de soluciones aleatorias y en la interfaz se puede contemplar el ajuste de los parámetros del algoritmo para su ejecución, los cuales son: número de bacterias, tamaño de paso, factor de escalamiento, ciclo quimiotáxico, bacterias a reproducir, frecuencias del ciclo de reproducción, bacterias a eliminar y el número de evaluaciones.

Debido a las investigaciones realizadas en este proyecto, las fuentes de investigación se basan en las fuentes secundarias, las cuales se pueden conocer por el nombre de “obras de referencia”,

pues su intención es proporcionar datos puntuales de consulta rápida (Silvestrini & Vargas, 2008).

Algunas fuentes de investigación secundarias que se utilizaron en este proyecto son: tesis de diversos autores, artículos sobre la existencia de interfaces del BFOA, tutoriales acerca del uso y manejo de ciertas herramientas software, entre otras muchas ya referenciadas.

Para la recolección de datos se hizo uso de tablas donde se concentraron los resultados arrojados por las ejecuciones independientes realizadas por el algoritmo al resolver el problema de optimización y así poder ser evaluado el rendimiento y funcionabilidad de la interfaz que permite calibrar los parámetros del algoritmo y visualizar sus resultados obtenidos.

Para el buen desarrollo de la GUI se ha utilizado la programación modular ya que ésta es un paradigma de programación que consiste en dividir un programa en módulos o subprogramas con el fin de hacerlo más legible y manejable (Joyanes-Aguilar, 1990).

En este proyecto de tesis se ha implementado la programación modular, la cual permitió un mejor manejo del algoritmo TS-MBFOA, ya que se buscaba obtener todos los parámetros del algoritmo e implementarlos en la interfaz, que permite la modificación de los mismos de una manera eficiente.

Para lograr la manipulación de estos es necesario usar un lenguaje que permita ejecutarse tanto en el entorno interactivo, como a través de un archivo de script (archivos *.m). Matlab se desarrolla en un lenguaje de programación propio (lenguaje M) este permite hacer operaciones de vectores, matrices, funciones, calculo lambda, y programación orientada a objetos (Goering, 2004).

El éxito de los proyectos de desarrollo de aplicaciones o sistemas se debe a que sirven como enlace entre quien tiene la idea y el desarrollador. El UML (Lenguaje Unificado de Modelado) es un lenguaje gráfico de propósito general y considerada una herramienta que cumple con esta función, ya que le ayuda a capturar la idea de un sistema para comunicarla posteriormente a quien esté involucrado en su proceso de desarrollo; esto se lleva a cabo mediante un conjunto de símbolos y diagramas (Schmuller, 2001).Uno de los principales usos del UML es para visualizar,

especificar, construir y documentar los componentes de un sistema de software (Booch, Rumbaugh & Jacobson, 2004). Los expertos de software crean diagramas de UML para ayudar a los desarrolladores a construir el software. Si es entendido el vocabulario del UML (los elementos representativos de los diagramas y su significado) se puede comprender y especificar con mucha más facilidad un sistema.

El UML está compuesto por diversos elementos gráficos que se combinan para conformar una amplia variedad de diagramas y dictar la manera en que es diseñado un sistema, los diagramas más comunes son: diagrama de clases, diagrama de objetos, diagrama de caso de uso, diagrama de estado, diagrama de secuencia, diagrama de actividad, diagrama de tiempo, diagrama de componentes y diagrama de despliegue. La finalidad de los diagramas es presentar diversas perspectivas de un sistema, a las cuales se les conoce como modelo. Es importante destacar que un modelo UML describe lo que supuestamente hará un sistema, pero no dice cómo implementar dicho sistema (Schmuller, 2001).

En la presente tesis se utilizaron 6 diagramas del modelo UML: Clases, Casos de Uso, Secuencia, Actividad, Tiempo y Despliegue, los cuales se utilizaron para implementarla interfaz de usuario para el ajuste de parámetros del algoritmo TS-MBFOA, descartando 3 diagramas debido a que su implementación no fue requerida para este proyecto. A continuación, se describen brevemente los 6 diagramas utilizados.

Diagrama de clases

El propósito de un diagrama de clase es describir las clases que conforman el modelo de un determinado sistema. Dado el carácter de refinamiento iterativo que caracteriza un desarrollo orientado a objetos, el diagrama de clase va a ser creado y refinado durante las fases de análisis y diseño, estando presente como guía en la implementación del sistema (García-Peñalvo & Pardo-Aguilar, 1998).

Se puede decir que existen tres perspectivas diferentes desde las cuales se pueden utilizar los diagramas de clase:

1. Conceptual: El diagrama de clase representa los conceptos en el dominio del problema que se está estudiando. Este modelo debe crearse con la mayor independencia posible de la implementación final del sistema.

2. Especificación: El diagrama de clase refleja las interfaces de las clases, pero no su implementación. Aquí las clases aparecen más cercanas a los tipos de datos, ya que un tipo representa una interfaz que puede tener muchas implementaciones diferentes.

3. Implementación: Esta vista representa las clases tal cual aparecen en el entorno de implementación.

Diagrama de caso de uso

El caso de uso es un concepto que ayuda a un analista a comprender la forma en que un sistema deberá comportarse. Le ayuda a obtener los requerimientos desde el punto de vista del usuario. Una de las finalidades del proceso de análisis es generar una colección de casos de uso (Schmuller, 2001), que posteriormente se traducirá en diagramas de casos de usos. Los diagramas de caso de uso son una técnica para capturar requisitos o información de cómo un sistema o negocio trabaja, y están compuesto por los casos de uso, los actores que se pueden definir como algo con comportamiento, como una persona (identificada por un rol), sistema informatizado u organización (Larman, 2003), y las relaciones existentes entre ambos.

Diagrama de secuencia

Un diagrama de secuencia se usa para mostrar las comunicaciones dinámicas entre objetos durante la ejecución de una tarea. Este tipo de diagrama muestra el orden temporal en el que los mensajes se envían entre los objetos para lograr dicha tarea. Puede usarse un diagrama de secuencia para mostrar las interacciones en un caso de uso o en un escenario de un sistema de software (Pressman, 2010).

Diagrama de actividad

Los diagramas de actividades son parte de los diagramas de comportamiento UML, que describen la funcionalidad del software en un nivel alto de abstracción. Los diagramas de actividades

describen los flujos de control que son creados, desde los modelos de procesos del negocio hasta los modelos de operación del sistema descritos por elementos tales como: modelos de casos de uso (flujos básicos, subflujos y flujos alternos), clases, operaciones, interfaces, componentes y colaboraciones. Un diagrama de actividades está compuesto por elementos de modelo identificados como nodos de acción (actividad/acción, llamada a actividad externa o subproceso), nodos de control, nodos objeto, flujos de control y flujos de objeto (Tabares, Pineda & Barrera, 2008).

Diagrama de tiempo

Los diagramas de tiempos de UML se usan para mostrar el cambio en el estado o valor de uno o más elementos en el tiempo. Este también puede mostrar la interacción entre los eventos de tiempos, las restricciones de tiempos y la duración que los gobiernan (Sparx-Systems, 2018b).

Diagrama de despliegue

Un Diagrama de Despliegue modela la arquitectura en tiempo de ejecución de un sistema. Esto muestra la configuración de los elementos de hardware (nodos) y muestra cómo los elementos y artefactos del software se trazan en esos nodos (Sparx-Systems, 2018a).

Capítulo II. Marco teórico

En este capítulo se describe los siguientes puntos: marco referencial en el cual se hace mención de los problemas en el que se ha aplicado el algoritmo BFOA y trabajos relacionados donde se usa una interfaz gráfica para su aplicación y funcionamiento, el marco conceptual donde se presenta los conceptos básicos del proyecto, además de incluir el marco tecnológico donde se hace mención de la herramienta para el desarrollo de la GUI y el marco legal donde se muestra el número de licencia utilizado.

2.1 Marco referencial

En la literatura especializada existen propuestas que brindan solución a problemas particulares donde es utilizado el algoritmo BFOA con una implementación en interfaz gráfica. Cada propuesta tiene sus características y ofrecen gráficas sobre las posibles soluciones existente para cada problema. A continuación, es presentado lo referente a las propuestas encontradas con relación a la GUI de BFOA.

2.1.1 BFOA generando un Menú nutricional

Planificación de menús nutritivos utilizando BFOA con algunas adaptaciones menores, en una interfaz gráfica diseñada en lenguaje M y Matlab R2009b (Hernández-Ocaña et al., 2018) donde los autores proponen a una versión del algoritmo llamada TS-MBFOA, en la cual se usan dos nados en el ciclo quimiotáxico. En esta propuesta, el algoritmo es visualizado en una interfaz gráfica donde el usuario final puede ingresar datos que permiten generar sus menús nutricionales y visualizar los resultados. La serie de elementos que integran la interfaz son: etiquetas de textos, botones, editor de textos, una tabla y cuadros de listas. En la Ilustración 2.1 se puede observar la interfaz propuesta.

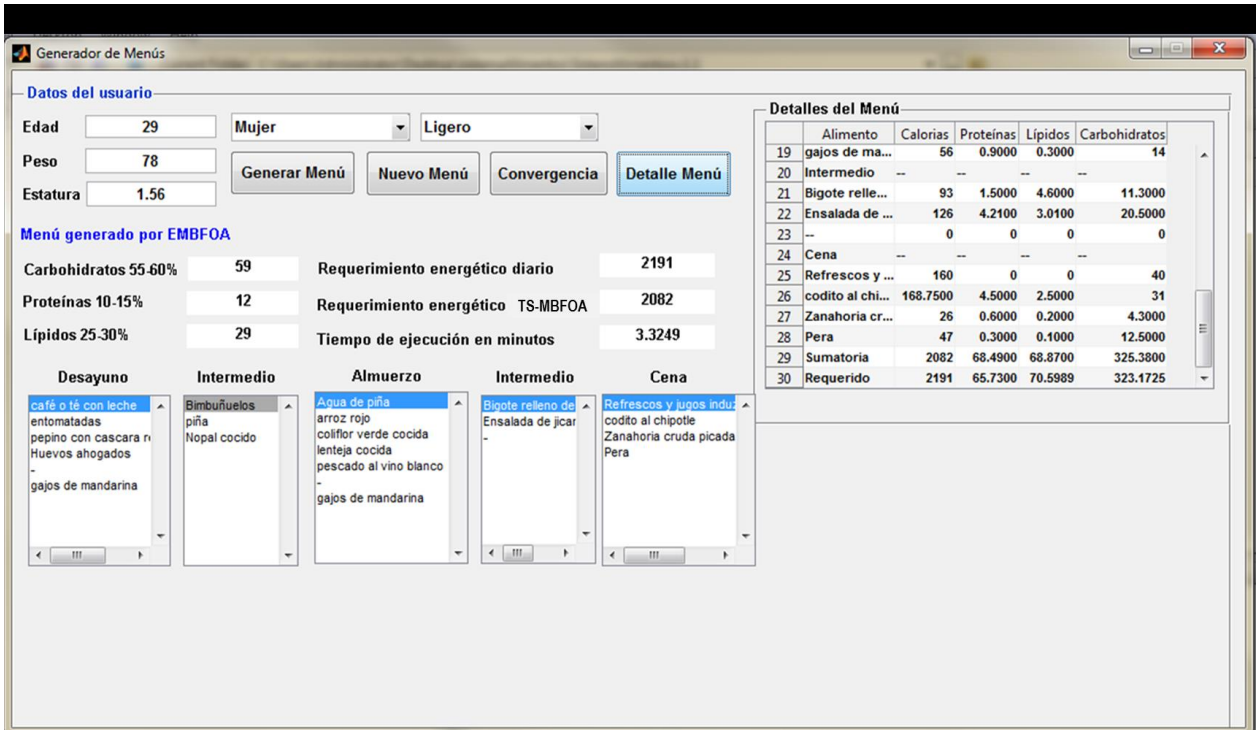

Ilustración 2.1 Generador de Menús.

2.1.2 Optimización de programación de sistemas de fabricación flexibles utilizando enfoques convencionales y evolutivos

En la propuesta los autores han desarrollado procedimientos de optimización basados en tres enfoques: algoritmo genético, evolución diferencial y algoritmo BFOA. Estos se implementan con éxito para resolver los problemas de optimización de programación de Sistemas de Fabricación Flexibles (FMS, por sus siglas en inglés). En la propuesta se ha desarrollado una interfaz basada en Matlab utilizando la versión 7.1, con la finalidad de automatizar el proceso de optimización al proporcionar al usuario una interfaz sencilla y también para proporcionar una herramienta automatizada para la optimización de programación utilizando enfoques convencionales y evolutivos. (Kumar et al., 2013). La interfaz gráfica diseñada por sus autores cuenta con elementos como: etiquetas de textos, menús emergentes (lista desplegable) y cajas de texto. Estos elementos permiten al usuario el ajuste adecuado para la ejecución del software. Cabe destacar que los valores solicitados no son los parámetros del algoritmo

BFOA, sino parámetros del problema a resolver. En la Ilustración 2.2 se muestra la interfaz de secuencias usando BFOA.

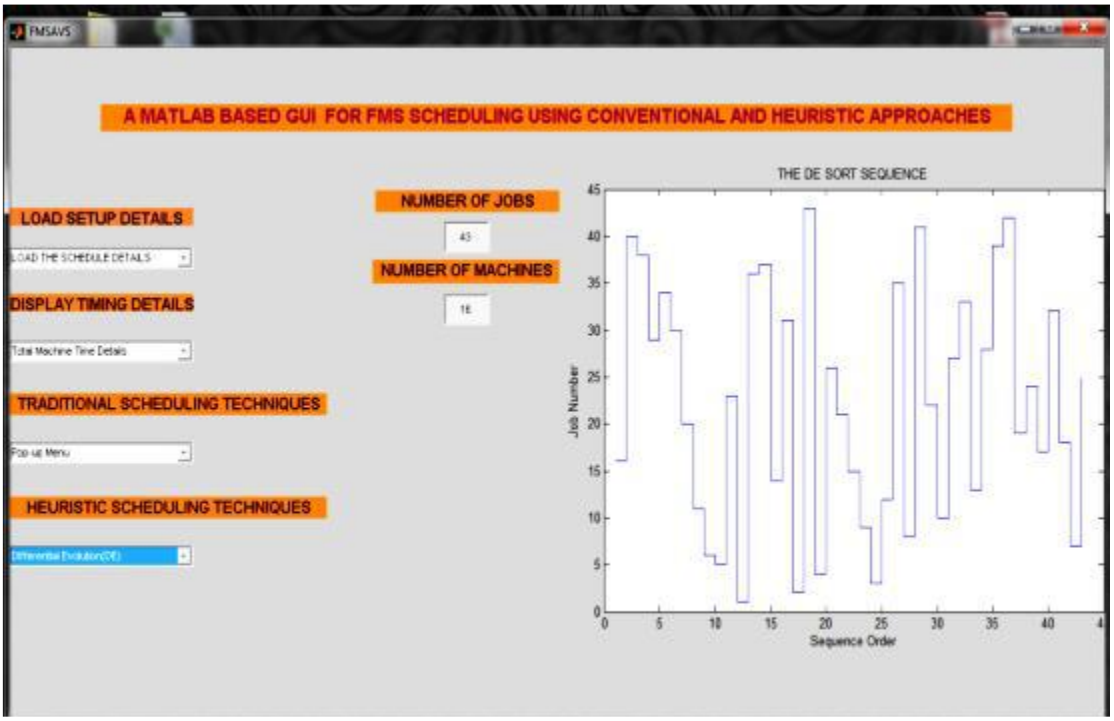

Ilustración 2.2 GUI correspondiente a secuencias de trabajo usando BFOA.

2.1.3 Planificación de estrategias de la ruta de un barco para evitar la colisión usando BFOA

La propuesta consiste en el proceso dinámico de evitar la colisión, la cual se demuestra a través de la interfaz del entorno VC ++ 9.0 basado en VS2010 (Ilustración 2.3), que muestra los parámetros de comparación de evitarla colisión y verifica directa y visualmente la viabilidad y superioridad de aplicar este algoritmo a una nave inteligente para evitar colisiones (Hongdan et al., 2015). La interfaz propuesta cuenta con componentes de uso general como: etiquetas de texto para guiar al usuario, cajas de textos para recibir los valores correspondientes al problema de comparación para evitar la colisión, lista donde se muestra una serie de resultados y una gráfica correspondiente a la colisión.

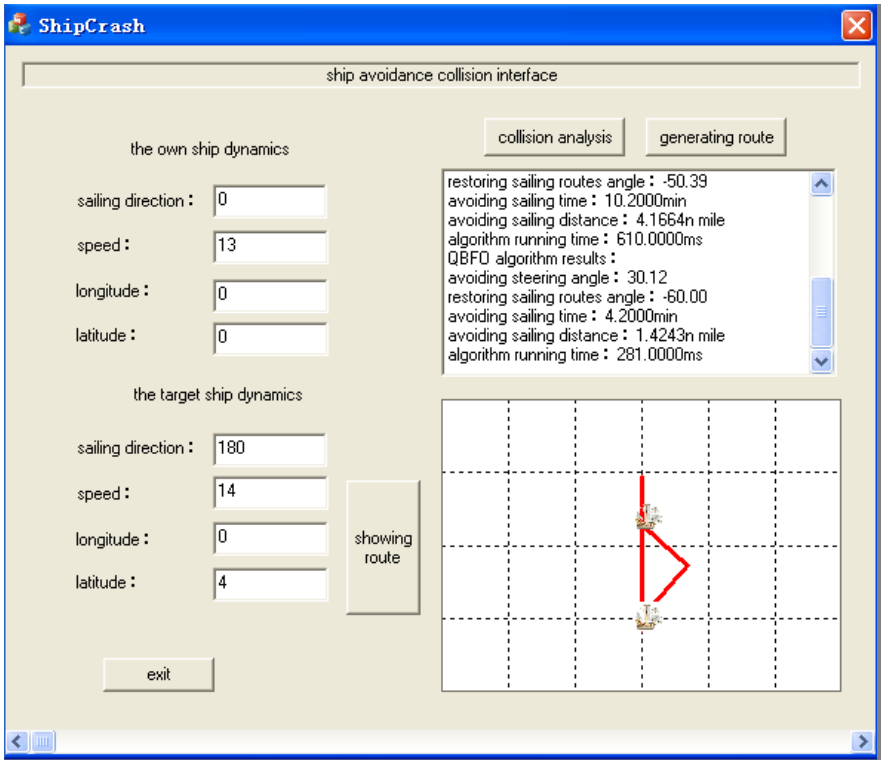

Ilustración 2.3 Interfaz para evitar la colisión de un barco usando BFOA.

2.1.4 Controlador de orden factorial basado en BFOA para sistemas de control de generación automática de área múltiple con almacenamiento de energía capacitiva

La propuesta investiga la eficacia de un controlador de orden fraccional y la configuración de ganancia óptima de los controladores. BFOA busca minimizar una integral del índice de rendimiento de error absoluto ponderado en el tiempo. Las simulaciones del sistema de prueba se realizan con el software Matlab/Simulink y se valida el rendimiento dinámico del controlador (Pappachen & Fathima, 2015). En la Ilustración 2.4, se presenta una gráfica con resultados presentados por los autores, la cual es la única imagen proporcionada por los autores.

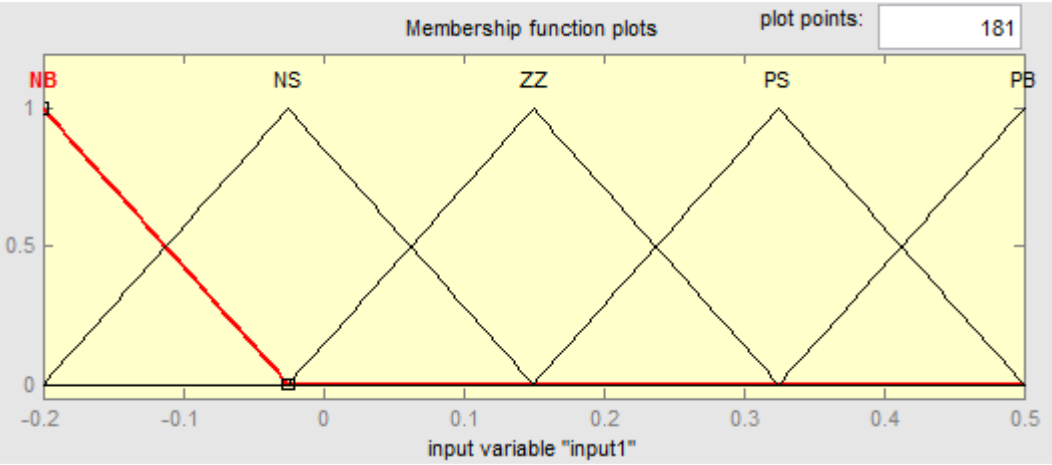

Ilustración 2.4 Grafica generada en Matlab para funciones de señales de entrada.

2.1.5 Análisis comparativo de la técnica de doble marca de agua basada en imágenes médicas seguras utilizando BFOA

La interfaz gráfica propuesta por Reddy & Siddaiah (2016) es diseñada como una herramienta integral capaz de realizar marcas de agua y diferentes análisis según lo requiera el usuario (Ilustración 2.5). El objetivo de esta herramienta es que permita al usuario tener facilidad de operación para cargar la imagen, marcarla con el agua, cifrarla y también recuperar la imagen original cuando sea necesario. La herramienta está codificada utilizando Matlab versión 7.1. La GUI cuenta con elementos gráficos como son: paneles y etiquetas de texto que guían al usuario, botones que permiten al usuario elegir y mostrar la imagen, menús emergentes (lista desplegable) para elegir las posibles configuraciones de la imagen según la necesidad del usuario y axis que visualizan la imagen con sus diferentes capas de configuración. Este sistema no cuenta con una calibración directa del algoritmo BFOA, solo permite al usuario determinar la calibración correspondiente al problema. Cabe mencionar que esta propuesta solo muestra los resultados, ya que no cuenta con un elemento para poder exportar para así poder visualizar en algún software secundario.

Ilustración 2.5 GUI propuesta para cargar imagen médica usando BFOA.

2.1.6 Reconocimiento de imágenes de huellas dactilares utilizando BFOA

Los autores proponen mejorar la tasa de reconocimiento del sistema de verificación de huellas dactilares por medio de una interfaz de usuario que se implementa utilizando Matlab 7.11.0. El diseño de la GUI está preparado en Matlab para un fácil acceso o uso del sistema (Brar & Singh, 2014). El diseño en Matlab se muestra en la Ilustración 2.6. Los elementos principales del software son: etiquetas de texto, radio botones, axis reconocedor de huellas y botones, además de un panel adicional para la visualización de huella como resultado final.

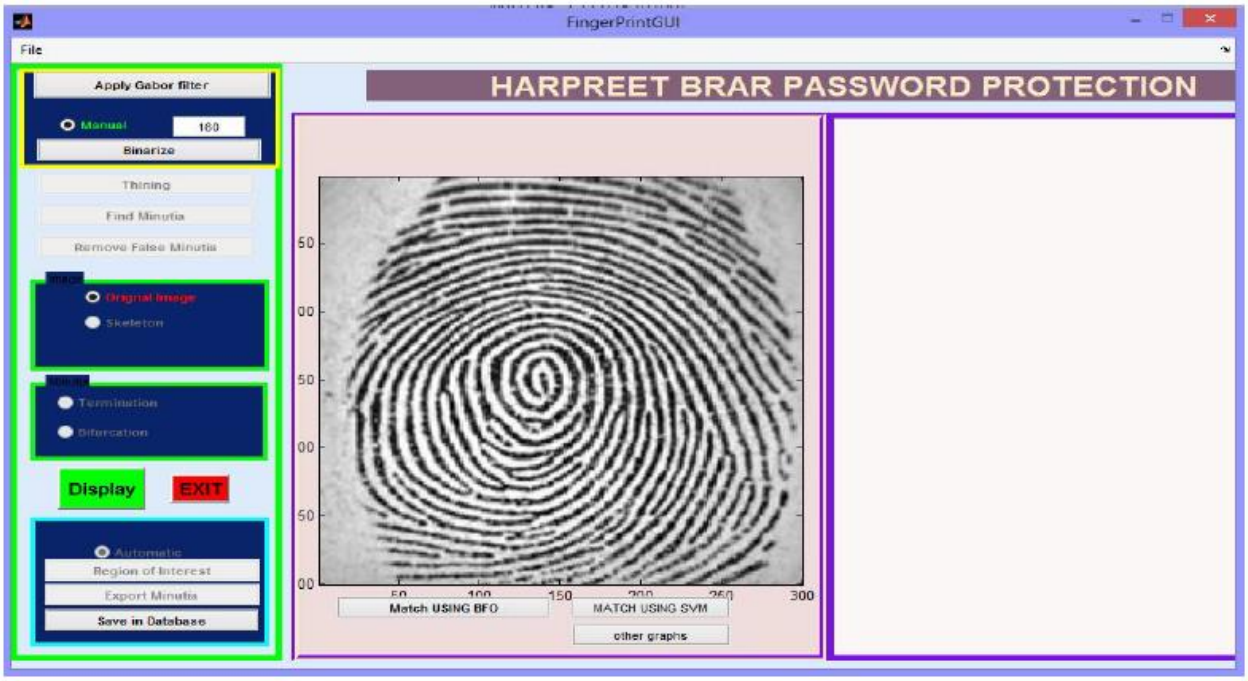

Ilustración 2.6 GUI reconocedor de huellas dactilares utilizando BFOA.

2.1.7 Diagnóstico del cáncer de pulmón mediante la fusión de las máquinas de vectores de soporte y la red neuronal de propagación hacia atrás

Esta propuesta proporciona un modelo de red neuronal y máquinas de soporte vectorial para la detección temprana del cáncer de pulmón. La simulación de rendimiento se lleva a cabo en un entorno Matlab 7.10 (Ilustración 2.7). El Matlab ha incorporado una caja de herramientas de red neuronal y se ha implementado máquinas de soporte vectorial usando dos pasos de entrenamiento y fases de prueba (Kaur & Singh, 2013). Esta propuesta solo cuenta con una serie de botones para poder seleccionar una determinada tarea de la imagen, axis para la visualización del pulmón y un par de lista donde se imprime los resultados correspondientes al problema.

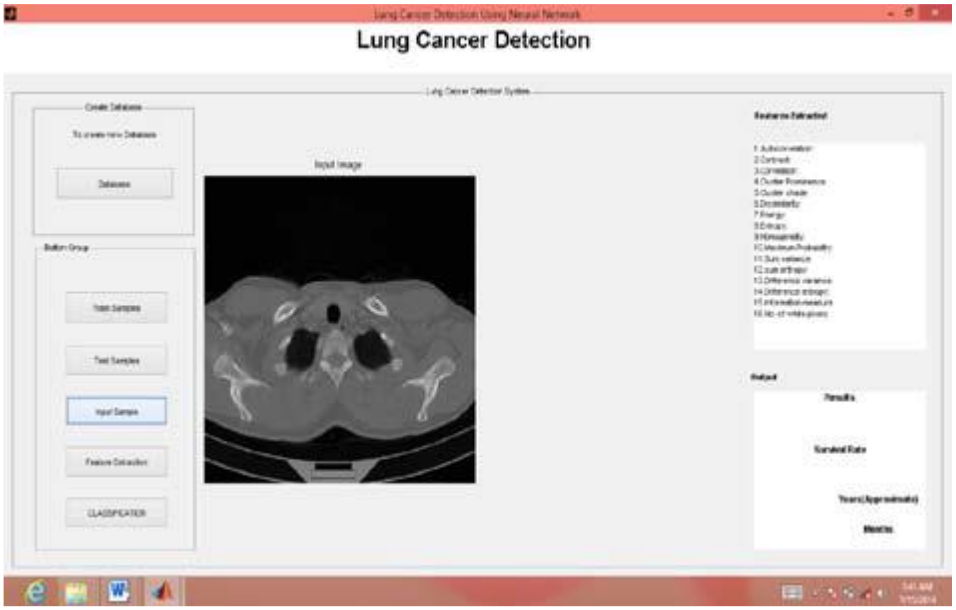

Ilustración 2.7 GUI para la detención de cáncer del pulmón.

2.1.8 Algoritmo hibrido con un adaptativo BFOA y DWT-SVD marca de agua con encriptación

BFOA adaptado y unido al DWT-SVD (*Discrete Wavelet Transform and Singular Value Decomposition*, por sus siglas en inglés) con cifrado donde se propone un enfoque de optimización de objetivos múltiples, además se considera que la imagen médica es la marca de agua incorporada en una imagen natural para evitar la intención de atacar la imagen médica. BFOA se emplea para optimizar la función objetivo para elegir un tipo correcto de wavelet y factor de escala (Venugopal & Siddaiah, 2014). En la propuesta es presentada una GUI que le permite al usuario tener facilidad de operación para cada caso de manipulación sobre la imagen cuando sea necesario. La robustez y la integridad de la marca de agua se prueban midiendo diferentes parámetros de rendimiento y sometiéndolos a varios ataques. En la propuesta se utiliza componentes básicos para el problema a resolver, descartando la calibración de los parámetros propios del algoritmo BFOA y las otras técnicas utilizadas por sus autores.

De acuerdo a las propuestas revisadas que implementan BFOA con ayuda de una interfaz gráfica,

se observa que el software en común de desarrollo es Matlab, los autores hacen uso de la interfaz para ingresar los datos de entrada para el problema a resolver y/o mostrar los resultados en las gráficas que permite generar Matlab. En ninguno de los casos revisados se hace uso de la interfaz para ingresar y/o calibrar los parámetros de entrada del algoritmo. Tampoco se visualiza el rendimiento del algoritmo usando alguna métrica de desempeño o gráficas como la de convergencia. Finalmente, en ninguna de las propuestas es posible exportar los datos generados por el algoritmo.

2.2 Marco conceptual

A continuación, se describen las características de los AEs y se hace mención de algunos AICs, Posteriormente se presenta los conceptos relevantes del proyecto, el cual para su mayor comprensión de ellos mismos se describen a continuación y así poder tener en cuenta que es lo que se busca al ejecutar el algoritmo TS-MBFOA.

2.2.1 Algoritmos evolutivos

Durante las últimas décadas ha habido un creciente interés en algoritmos basados en el principio de la evolución. Los algoritmos evolutivos (AE) son métodos robustos de búsqueda, que permiten tratar problemas de optimización donde el objetivo es encontrar un conjunto de parámetros que minimizan o maximizan una función de adaptación. Entre los AE más conocidos se incluyen los Algoritmos Genéticos (AG), la Programación Genética (PG), las Estrategias Evolutivas (EE), la Programación Evolutiva (PE) y la Evolución Diferencial (ED). Estos modelos basan su funcionamiento en la teoría neodarwinista de la evolución natural y la supervivencia del individuo más fuerte donde la evolución se lleva a cabo a nivel genético (Hernández-Ocaña & Mezura-Montes, 2009).

Los AE son una familia de métodos de optimización, la idea fundamental de estos algoritmos es mantener un conjunto de individuos que representan una posible solución del problema. Estos individuos se mezclan y compiten entre sí, siguiendo el principio de selección natural por el cual sólo los mejor adaptados sobreviven al paso del tiempo. Esto redunda en una evolución hacia soluciones cada vez más aptas (Gutiérrez & Díaz, 2014).

2.2.2 Algoritmos de inteligencia colectiva

La Inteligencia Colectiva (IC) es parte de los algoritmos metaheurísticos inspirados en la naturaleza. La IC descubre y analiza el comportamiento colaborativo y asociativo de animales como una parvada de aves, un conjunto de peces, una colonia de hormigas, de abejas y de bacterias en búsqueda de alimento y/o refugio. Es importante observar, que en los grupos de animales mencionados sobreviven la mayoría de sus integrantes y que todo el grupo sigue por instinto a un líder, dicho elemento es el que tiene la mejor orientación o posición para encontrar alimento o refugio o simplemente para dirigir al grupo de un lugar a otro (Engelbrecht, 2006);(Kennedy & Eberhart, 2001).

Una de las características importantes de estos grupos de animales es que son colaborativos, es decir, se apoyan unos con otros para alcanzar un objetivo y esto se logra a través de la comunicación existente entre ellos. Por ejemplo, para las parvadas de aves, la comunicación se da cuando las aves del grupo identifican al líder, que en su caso es el ave que va a la cabeza de la parvada y deciden seguirlo, ya que su velocidad de vuelo debido a la posición en la que va es rápida o satisfactoria; los peces también tienen su mecanismo de comunicación parecido al de parvada de aves pero es con el fin de nadar rápido; las hormigas son un caso excepcional, se conoce que son muy trabajadoras y que la comunicación para encontrar alimento y dirigirse hacia el camino más corto es a través de la feromona que van depositando al recorrerlo. Las abejas tienen distintos tipos de comunicación, por ejemplo, la danza o revoloteo de alas y contacto de antenas o manos que utilizan para indicar la dirección de vuelo y distancia del lugar donde se encuentra el alimento e indicar la calidad del néctar. Por último, las bacterias, las cuales al hacer la búsqueda de nutrientes utilizan sus químicos atrayentes para comunicarse e indicar qué lugares son promisorios.

Debido al análisis del comportamiento de estos grupos colaborativos, surge la IC, la cual hace referencia a la solución de problemas de comportamiento que surgen de la interacción de los integrantes del grupo; y la IC computacional, se refiere a los algoritmos que emulan dichos comportamientos. Los primeros algoritmos de IC que surgieron fueron PSO (Kennedy & Eberhart, 1995) que emula el comportamiento de parvada de aves, ACO (Dorigo, Maniezzo &

Colorni, 1996) que emula el comportamiento de colonias de hormigas y el algoritmo basado en el forrajeo de las bacterias BFOA (Passino, 2002).

Tomando el interés de la tesis no se describen como tal todos los algoritmos de inteligencia colectiva ya que para este trabajo solo es necesario conocer el BFOA y sus mejoras para la implementación de la interfaz de usuario para el ajuste de parámetros. Debido a la gran popularidad de los algoritmos de inteligencia colectiva se toma un modelo de optimización basado en el forrajeo de bacterias (BFOA) y a partir de este mismo, se utiliza por su potencial para resolver problemas de optimización. En el capítulo III, se describe BFOA y dos de sus mejoras más populares que se han implementado por sus autores como: MBFOA y TS-MBFOA.

2.2.3 Componentes de un problema de optimización

Optimización

La optimización es un proceso que pretende encontrar una solución o conjunto de soluciones que maximice o minimice una cierta medida de calidad, conocida como función objetivo. Dentro de los conceptos más destacados para el término optimización se encuentran: el proceso de buscar la mejor solución posible a un problema, bajo ciertas circunstancias (Sierra, 2006) y el método matemático para determinar los valores de las variables que hacen máximo el rendimiento de un proceso o sistema (RAE, 2008).

Los métodos de optimización son conocidos como técnicas de programación matemática, las cuales son aplicables a problemas de toma de decisiones, así como al establecimiento de las mejores soluciones posibles. Según Deb (1995) los problemas de optimización pueden clasificarse de acuerdo a las siguientes características:

1. Basándose en la existencia de restricciones:
    - Con restricciones.
    - Sin restricciones.
2. Basándose en la naturaleza de las variables de decisión:

- Problemas estáticos o paramétricos.
- Problemas dinámicos o de trayectorias (las variables son función de un parámetro determinado).

3. Basándose en la naturaleza de las ecuaciones involucradas:
   - Problemas lineales.
   - Problemas no lineales.
   - Problemas de programación geométrica.
   - Problemas de programación cuadrática.
4. Basándose en los valores permisibles de las variables de diseño:
   - Problemas de programación entera.
   - Problemas de programación con valores reales.
   - Problema donde se busca un orden óptimo de elementos, (problema de optimización combinatoria).
5. Basándose en la naturaleza determinista de las variables:
   - Problemas estocásticos.
   - Problemas deterministas.
6. Basándose en la separabilidad de las funciones:
   - Problemas separables.
   - Problemas no separables.
7. Basándose en el número de funciones objetivo:
   - Mono-objetivo.
   - Multi-objetivo.

Los problemas de optimización en su forma matemática presentan características especiales como: función objetivo lineal o no lineal, con restricciones de igualdad (lineal o no lineal) o desigualdad (lineal o no lineal), con restricciones activas y diferente número de variables en el problema.

Con base a las investigaciones, Hernández-Ocaña (2009) apunta que existen tres grandes componentes para un problema de optimización, estos son: función objetivo, variable de diseño y restricción de diseño. A continuación se presenta una breve descripción de cada una de los componentes mencionados.

Función objetivo

La función objetivo representa la relación matemática que permite cuantificar el desempeño del sistema u objeto de interés, la cual es determinada después de un profundo análisis del problema. En la función objetivo es donde se evalúa el conjunto de valores del vector de variables de diseño para determinar si éstos son los valores según sea el caso.

Variables de diseño

Las variables de diseño son el conjunto de elementos paramétricos que describen el sistema u objeto de interés, donde cada variable proporciona una información mínima que en conjunto permiten describir el sistema u objeto para ser diseñado de manera óptima. Las variables de diseño las establece el diseñador cuando hace el análisis del sistema u objeto y determina el rango de valores factibles que puede tomar cada una de las variables, esto con el fin de que, al momento de encontrar los valores óptimos, el diseño sea factible de construir. Cuando el diseño del sistema u objeto se lleva a cabo tomando en cuenta el conjunto de variables de diseño, se denomina diseño paramétrico.

Restricciones de diseño

Las restricciones de diseño son el conjunto de limitaciones del sistema que se deben satisfacer para producir un diseño aceptable. Las restricciones de comportamiento o desempeño del sistema son denominadas restricciones funcionales o de comportamiento, mientras que las restricciones

que representan limitaciones físicas tales como disponibilidad, facilidad de fabricación y transpirabilidad son denominadas restricciones geométricas. Las restricciones son estáticas cuando únicamente dependen de la evaluación del conjunto de parámetros del sistema. Cuando se involucran aspectos dinámicos (control o estados del sistema) o cuando se evalúan en un periodo de tiempo (intervalo de tiempo durante el que se optimiza el sistema), las restricciones son dinámicas.

2.3 Marco tecnológico

2.3.1 Lenguaje de programación

Matrix Laboratory, comúnmente conocido como Matlab, es un entorno de programación para el desarrollo de algoritmos, el análisis de datos, la visualización y el cálculo numérico.

Las aplicaciones de Matlab se desarrollan en un lenguaje de programación propio. Este lenguaje es interpretado y puede ejecutarse tanto en el entorno interactivo como a través de un archivo de *script* (archivo*.m). Este lenguaje permite operaciones de vectores y matrices, funciones, cálculo lambda, y programación orientada a objetos (Georing, 2004).

Debido a la necesidad que se tiene de implementar la GUI para la utilización del algoritmo TS-MBFOA se requiere el uso de tecnologías apropiada para los fines que se desea, es por ello que a continuación se describe dicha tecnología y los pasos generales para desarrollar una GUI.

2.3.2 GUI en Matlab

Matlab dispone de un módulo denominado GUIDE (*Graphical User Interface Development Environment*) que permite crear de modo interactivo una GUI. Aún cuando sería posible escribir un archivo .m que contenga todos los comandos para elaborar una GUI, es mucho más fácil utilizar el GUIDE porque permite hacerlo interactivamente.

2.3.3 Construcción interactiva de una GUI

La elaboración de una GUI incluye dos tareas básicas como son:

- Diseño de la GUI: distribución de los componentes de la interfaz.
- Programación de la GUI: programación de cada componente.

Diseño de una GUI se dispone de varias opciones diferentes para acceder a la herramienta GUIDE en Matlab, accediendo a través del menú *File*, del botón *Start* del acceso directo de Matlab *Toolbar* o simplemente introduciendo `>>guide` en la ventana de comandos.

Tras ejecutar cualquiera de las acciones que inician la herramienta aparecerá una ventana de selección denominada GUIDE *Quick Start* que ofrece la posibilidad de escoger entre crear una nueva GUI en blanco o abrir uno existente como ejemplo. Para diseñar el aspecto visual de una GUI, completamente desde cero, se requiere seleccionar la opción *Blank* GUI (*Default*). Inmediatamente aparece la pantalla principal de GUIDE.

En esta ventana se pueden distinguir tres secciones fundamentales:

- Barra de herramientas. Compuesta tanto de las funciones básicas (crear nueva figura, abrir, guardar, etc.) y las funciones necesarias para diseñar una GUI, detalladas a continuación.
    - Alineación de objetos: sobre los objetos seleccionados permite realizar diferentes tipos de alineado.
    - Editor de menús: permite diseñar una estructura de menús desplegables.
    - Editor de orden de tabulación: permite configurar el orden de desplazamiento del foco sobre los objetos de la GUI al pulsar la tecla de tabulación.
    - Editor de barra de herramienta: permite crear una barra de herramientas con las funcionalidades deseadas en la interfaz.
    - Editor de ficheros M: abre el archivo .m donde se encuentra el código fuente de la interfaz.

- Inspector de propiedades: posibilita la modificación de las propiedades de los objetos creados.
- Buscador de objetos: muestra la estructura jerárquica de los objetos.
- Ejecutar: guarda y ejecuta el código fuente asociado a la GUI.

- Área de diseño. Zona donde se ubican los objetos.
- Paleta de componentes. Muestra todos los objetos disponibles para la realización de la interfaz.

GUIDE dispone de una amplia paleta de componentes para realizar la interfaz gráfica. Para situar estos objetos se debe arrastrar desde la paleta hasta el área de diseño. Las propiedades de estos objetos se gestionan a través de la opción *Property Inspector* a la cual se accede al hacer click con el botón derecho sobre el objeto en sí. Se modelan tanto las propiedades físicas (tamaño, tipo de fuente, posición, etc.) como las propiedades relativas a aspectos de programación (*tag*, *callback*, etc.).

2.3.4 Programación de una GUI

Una vez concluido el diseño de la interfaz, GUIDE genera automáticamente un archivo .m que, en resumen, contiene:

- Código de inicialización de la GUI.
- Código para implementar tareas previas a la visualización en pantalla de la GUI.
- Código de los *callbacks* (funciones que responden al evento generado por una acción del usuario).

Cada componente tiene sus propios *callbacks* al igual que propiedades. De este modo, el diseñador simplemente debe codificarlas asignando a cada componente la función deseada (Muragán, 2012).

2.4 Marco legal

Para el buen diseño y desarrollo de la GUI basada en TS-MBFO se tiene la necesidad de usar tecnología adecuada, por ello es necesario adquirir permisos legales. Como anteriormente se ha señalado, se utiliza Matlab para el desarrollo de la GUI y en él, la ejecución de algoritmo TS-MBFOA.

Matlab es una herramienta que permite la construcción de la interfaz que se tiene como objetivo en esta tesis, por lo tanto, para el uso legal se adquiere una licencia con número 40693036 y así poder lograr los planteado sin problemas de diseño limitado en el uso de dicha herramienta.

Capítulo III. Algoritmos basados en el forrajeo de bacterias

En este capítulo es detallado el origen del algoritmo BFOA y las mejoras implementadas por sus autores.

3.1 Algoritmos de Optimización basado en el Forrajeo de Bacterias (BFOA)

El BFOA es un algoritmo que ha sido utilizado para resolver problemas de optimización numérica con y sin restricciones, tanto en funciones de prueba como en problemas particulares modelados como un problema de programación no-lineal. El BFOA, propuesto por Kevin M. Passino en el año 2002, simula el proceso completo de forrajeo de las bacterias E. Coli: quimiotáxis (movimientos de nado-giro), agrupamiento (comunicación entre las bacterias), reproducción (duplicación de las mejores bacterias) y eliminación-dispersión (eliminación de la peor bacteria y creación de una nueva).

El modelado de la búsqueda de alimento o forrajeo de las bacterias mediante el BFOA, se lleva a cabo de la siguiente manera:

i. Inicialmente, las bacterias son distribuidas al azar sobre el mapa de nutrientes.

ii. En la 1ra. generación, las bacterias (E. Coli) se mueven hacia los puntos de concentración de nutrientes (dentro de esta generación ocurre una eliminación y dispersión), algunas bacterias se acercan a puntos de sustancias nocivas, después ocurre la reproducción y posteriormente casi todas las bacterias se colocan en puntos de concentración de nutrientes.

iii. En la 2da. generación, las bacterias han encontrado los puntos de concentración de nutrientes, aunque no todos estos puntos son óptimos, es decir, los más altos de

concentración de nutrientes (las bacterias se enfrentan a sustancias nocivas, las cuales no permiten que las bacterias encuentren el mejor punto, pero estas se agrupan, utilizando la comunicación mediante segregación de atrayentes y realizan la búsqueda juntas). Las bacterias realizan de nuevo el proceso quimiotáxico para posteriormente hacer agrupamiento, reproducción y eliminación-dispersión mostrando.

iv. En la 3ra. generación, las bacterias encuentran el punto más alto de concentración de nutrientes.

v. En la 4ta. generación, las bacterias permanecen en ese punto y la búsqueda de alimento ha terminado; se dispersan nuevamente y buscan una nueva fuente de nutrientes (repetir generaciones).

Durante estas generaciones, las bacterias se enfrentan a varias problemáticas en la búsqueda de sus alimentos, por ejemplo: ambientes nocivos que deben detectar y retroceder en su viaje, hecho que causa la eliminación y dispersión (para ello, estas bacterias se agrupan y son capaces de producir y secretar sustancias químicas que son usadas como mecanismo de señalización y comunicación entre ellas), el tiempo de vida de las bacterias, el recorrido que tienen que hacer para encontrar el punto óptimo de concentración de nutrientes. Usando el modelo biológico que permite encontrar este punto óptimo, tomamos en cuenta estas posibles restricciones y se calcula la suma de las señales enviadas por cada una de las otras bacterias de la colonia, obteniendo así el punto deseado.

Basados en estos pasos, Passino propuso el algoritmo de optimización de bacterias en búsqueda de alimento.

El paso quimiotáxico fue modelado por Passino con la generación de direcciones aleatorias, ver la siguiente ecuación 3.1.

$$\emptyset(i) = \frac{\Delta(i)}{\sqrt{\Delta(i)^T \, \Delta(i)}} \tag{3.1}$$

donde $\Delta(i)^T$ es un vector aleatorio con distribución uniforme generado con elementos dentro de un intervalo: [-1, 1]. Después de esto, cada bacteria $\theta^i\ (j,k,l)$ en su ciclo *j*, en su ciclo de reproducción *k* y en su ciclo de eliminación-dispersión *l* modifica su posición como se indica en la siguiente ecuación:

$$\theta^i(j+1,k,l) = \theta^i(j,k,l) + C(i)\emptyset(i) \tag{3.2}$$

La ecuación 3.1 representa un giro (busca dirección) y la ecuación 3.2 representa un nado con tamaño de paso C_i. El nado será repetido N_s veces si la función objetiva en la nueva posición es mejor que en la previa: $f(\theta^i(j+1,k,l)) < f(\theta^i(j,k,l))$.

El paso de reproducción consiste en ordenar la población de bacterias $\theta^i(j,k,l)\ \forall i, i = 1,\ldots,S_b$ basado en el valor de su función objetivo $f(\theta^i(j,k,l))$ y eliminar aquellas bacterias con el peor valor S_r. La otra mitad será duplicada para mantener un tamaño de población constante.

El paso de eliminación-dispersión consiste en eliminar cada bacteria $\theta^i(j,k,l)\ \forall i, i = 1,\ldots,S_b$ con probabilidad $0 \leq P_{ed} \leq 1$.

El pseudocódigo derivado es presentado en la Ilustración 3.1, donde los parámetros de entrada son: el número de bacterias S_b, límite del paso quimiotáxico N_c, límite del paso de nado N_s, límite del ciclo de reproducción N_{re}, número de bacterias a reproducir N_{sr}, límite del ciclo de eliminación-dispersión N_{ed}, tamaño de paso (C_i) y probabilidad de eliminación-dispersión (P_{ed}).

```
Generar una población inicial aleatoria θ^i(j,k,l) ∀i, i = 1,...,S_b
Evaluar f(θ^i(j,k,l)) ∀i, i = 1,...,S_b
for l=1 to N_ed do
    for k=1 to N_re do
        for j=1 to N_c do
            for i=1 to S_b do
                Realizar el proceso quimiotáxico para la bacteria θ^i(j,k,l), los nados son limitados por N_s
            end
        end
        Realizar el proceso de reproducción ordenando las bacterias y eliminar a las peores bacterias S_b − S_r y
        duplicar a las mejores.
    end
    Realizar el paso de eliminación-dispersión para todas las bacterias θ^i(j,k,l) ∀i, i = 1,...,S_b con probabilidad
    0 ≤ P_ed ≤ 1
end
```

Ilustración 3.1 Pseudocódigo de BFOA.

Los parámetros que utiliza el BFOA son numerosos y los ciclos que intervienen en él hacen que sea un algoritmo complejo, produciendo en primera instancia un número elevado de evaluaciones y por consiguiente un algoritmo lento. Además, la calibración de parámetros puede ser complicado para el usuario. En la Tabla 3.1 se muestran los parámetros utilizados por el BFOA.

Símbolo	Nombre	Descripción
Sb	Bacterias	Número de soluciones en la población (bacterias).
Nc	Paso quimiotáxico	Número de veces que la población de bacterias podrán nadar o girar.
Ns	Nado	Número exacto de límites en que la bacteria podrá nadar.
Nre	Reproducción	Número de veces que la población de bacterias se reproducirá.
Sr	Reproducir de bacterias	Número de bacterias, que se reproducirán de toda la población.
Ned	Eliminación-dispersión	Número de veces en que se aplicara eliminación-dispersión en la población de bacterias.
Ci	Tamaño de paso	Es el tamaño o el paso que la bacteria en proceso podrá avanzar de un punto a otro en su búsqueda, limitado por las dimensiones del problema.
Ped	Probabilidad de eliminación-dispersión	Determina qué bacterias de la población serán eliminadas y dispersadas en la nueva área de búsqueda.

Tabla 3.1 Descripción de los parámetros del BFOA.

3.2 Algoritmos de Optimización basado en el Forrajeo de Bacterias Modificado (MBFOA)

El MBFOA es un algoritmo propuesto por (Hernández-Ocaña & Mezura-Montes, 2009) en el cual se implementan varios mecanismos que permiten al algoritmo resolver problemas de optimización mono-objetivo con restricciones.

Los cuatro mecanismos del MBFOA son los siguientes:

1. Un solo ciclo que incluye el ciclo quimiotáxico, la reproducción y eliminación-dispersión. El ciclo de generaciones es controlado por el número máximo de generaciones (GMAX), en el cual las bacterias realizan su ciclo quimiotáxico. Después de esto se lleva a cabo una sola reproducción y una eliminación-dispersión dentro de un mismo ciclo de generación. En el paso de eliminación-dispersión sólo se elimina a la peor bacteria de la población.

2. Tamaño de paso diferente para cada variable de diseño. El tamaño de paso *C(i)* es definido considerando los limites inferior y superior L_i y U_i para cada variable de decisión x_i, usando la ecuación 3.3.

$$C(i) = R * \left(\frac{\Delta \underset{x_i}{\rightarrow}}{\sqrt{n}}\right) \tag{3.3}$$

donde *C(i)* es el tamaño de paso que, en este caso, ya no está definido por el usuario, $\Delta \underset{x_i}{\rightarrow}$ es la diferencia U_i - L_i, n es el número de variables en el problema de optimización y *R* es el porcentaje del total del tamaño de paso a ser usado, un tamaño de paso inicial pequeño suele ser conveniente en optimización restringida, conclusiones que se reportan en (Hernández-Ocaña & Mezura-Montes, 2009).

En este mismo proceso se asegura que los elementos en $\underset{x}{\rightarrow}$ (variables de diseño) no rebasen sus límites establecidos $L_i \leq x_i \leq U_i, \ \forall i = 1, ..., n$, mediante el siguiente método:

Si $x_i \leq L_i$ entonces:

$$x_i = 2L_i - x_i \qquad (3.4)$$

Si $x_i \leq U_i$ entonces:

$$x_i = 2U_i - x_i \qquad (3.5)$$

3. El manejo de las restricciones del problema se realce usando las reglas de factibilidad de Deb (2000), estas reglas son usadas en el proceso quimiotáxico (específicamente en el nado), reproducción y eliminación-dispersión.

4. Comunicación entre las bacterias y la mejor bacteria de la población de una generación.

   En ciertas iteraciones del ciclo quimiotáxico (a la mitad y al final) se permite que cada bacteria de la población se oriente y nade orientada por la dirección de mejor bacteria de la población (efecto de *swarming*) calculado como se indica en la ecuación 3.6:

$$\theta^i(j+1, G) = \theta^i(j, G) + \beta(\theta^B(G) - \theta^i(j, G)) \qquad (3.6)$$

   donde $\theta^i(j+1, G)$ es la nueva posición de la bacteria *i*, $\theta^B(G)$ es la actual posición de la mejor bacteria generación es hasta *G* y β es un parámetro llamado factor de escalamiento, el cual regula qué tan cerca estará la bacteria *i* de la mejor bacteria θ^B.

   Los pasos restantes del ciclo quimiotáxico son realizados como se indica en la ecuación3.7:

$$\theta^i(j, +1, G) = \theta^i(j, G) + C(i)\emptyset(i) \qquad (3.7)$$

donde la dirección aleatoria de las bacterias es modelada por la ecuación 3.8:

$$\emptyset(i) = \frac{\Delta(i)}{\sqrt{\Delta(i)^T \Delta(i)}} \tag{3.8}$$

$\Delta(i)^T$ es un vector aleatorio generado con elementos dentro de un intervalo [-1, 1]. Los parámetros de MBFOA se describen en la Tabla3.2.

Símbolo	Nombre	Descripción
Sb	Bacterias	Número de soluciones en la población (bacterias).
GMAX	Generaciones	Número de generaciones del algoritmo generado, calculado a partir el número de las evaluaciones permitidas con la formula (Sb/Nc).
Nc	Paso quimiotáxico	Número de veces que la población de bacterias podrán nadar o girar.
Sr	Reproducción	Número de veces que la población de bacterias se reproducirá.
B	Escalamiento	Número de bacterias, que se reproducirán de toda la población.
R	Tamaño de paso	Es el tamaño o el paso que la bacteria en proceso podrá avanzar de un punto a otro en su búsqueda, limitado por las dimensiones del problema.

Tabla 3.2 Descripción de los parámetros del MBFOA.

El pseudocódigo del MBFOA es presentado en la Ilustración 3.2, sus parámetros de entrada son el número de bacterias S_b, límite del paso quimiotáxico N_c, número de bacterias a reproducirse S_r, factor de escalamiento β, porcentaje del tamaño de paso R y el número de generaciones *GMAX*.

```
Crear una población inicial de bacterias θ^i(j,0) ∀i, i = 1,...,S_b
Evaluar f(θ^i(j,0)) ∀i, i = 1,...,S_b
for G=1 to GMAX do
    for i=1 to S_b do
        for j=1 to N_c do
            Realizar el proceso quimiotáxico (nado-giro) para la bacteria θ^i(j,G) usando las reglas de factibilidad y
            las Ec.3.5 y 3.4
        end
    end
    Realizar el proceso de reproducción ordenando las bacterias con base en la técnica de manejo de restricciones,
    eliminar a las peores bacterias S_b − S_r y duplicar a las mejores.
    Realizar el proceso de eliminación-dispersión eliminando a la peor bacteria θ^w(j,G) de la población, basado en las
    reglas de factibilidad y generar una nueva aleatoriamente.
end
```

Ilustración 3.2 Pseudocódigo de MBFOA.

3.3 TS-MBFOA

TS-MBFOA intercala dos nados en el proceso quimiotáxico, el primero es el nado original a excepción del tamaño de paso el cual se propone sea aleatorio y el segundo nado incluye el operador de mutación usado en los algoritmos evolutivos, la inclusión de ambos para mejorar la capacidad de exploración y explotación del algoritmo. Aunque en el estado del arte existen propuestas basados en BFOA que utilizan el operador de mutación dentro del proceso de búsqueda, así como: la propuesta de una hibridación de BFOA con la técnica de optimización de Evolución Diferencial (Biswas, Dasgupta, Das & Abraham, 2007), resolución del problema de formación en el sistema de fabricación celular (Nouri & Hong, 2012), propuesta de un algoritmo genético basado en BFOA (Kushwaha, Bisht & Shah, 2012), entre otras, ninguna de ellas incluye el operador de mutación como mecanismo de nado.

Operadores de nado

En proceso de quimiotáxis, dos nados se intercalan, en cada ciclo solo un nado de explotación o exploración es realizado. El proceso comienza con el nado de explotación (nado clásico). Sin embargo, una bacteria no necesariamente intercalara exploración y explotación en los nados, ya que si la nueva posición de un nado dado, $\theta^i(j,+1,G)$ tiene una mejor aptitud (basado en las reglas de factibilidad) que la posición original $\theta^i(j,G)$, otro nado en la misma dirección se llevará a cabo en el siguiente ciclo. De lo contrario, un nuevo giro será calculado. El proceso se detiene después de N_c intentos.

El nado de exploración usa la mutación entre bacterias y es calculado con la ecuación 3.9.

$$\theta^i(j,+1,G) = \theta^i(j,G) + (\beta - 1)\,(\theta^r{}_1(j,G) - \theta^r{}_2(j,G)) \tag{3.9}$$

donde $\theta^r{}_1(j,G)$ y $\theta^r{}_2(j,G)$ son dos bacterias diferentes seleccionadas aleatoriamente de la población. β es un parámetro definido por el usuario utilizado en el operador de agrupamiento, el cual define la cercanía de la nueva posición de una bacteria con respecto a la posición de la mejor bacteria de la población, en este operador, β - 1 es un parámetro de control positivo para escalar los diferentes vectores en (0,1], es decir, escalas de la zona donde una bacteria puede moverse.

La Ilustración 3.3 muestra el comportamiento de este operador de nado utilizando un espacio de dos variables de decisión, cada uno en dentro del rango [-5, 5].

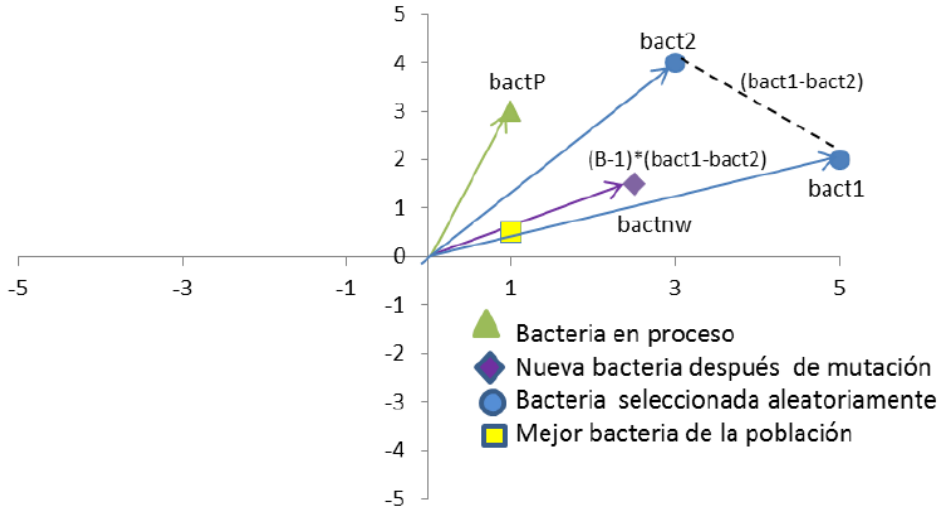

Ilustración 3.3 Comportamiento del nado de exploración.

El nado de explotación es calculado con la ecuación 3.10:

$$\theta^i(j,+1,G) = \theta^i(j,G) + C(i,G)\,\emptyset(i) \tag{3.10}$$

donde $\emptyset(i)$ se calcula con el operador de giro original de BFOA (ecuación 3.8) *y C(i, G)* es el tamaño de paso aleatorio de cada bacteria actualizado con la ecuación 3.11:

$$C(i,G) = R * \theta(\mathrm{i}) \tag{3.11}$$

donde θ(i) es un vector generado de forma aleatoria de tamaño *n* con elementos dentro del rango de cada variable de decisión: $[U_k, L_k]$ $k = 1, \ldots n$ y *R* es un parámetro definido por el usuario para escalar el tamaño de paso, este valor debe ser cercana a cero, por ejemplo. 5.00e-04. El tamaño inicial *C(i, 0)* se genera utilizando θ(i). Este tamaño de paso aleatorio permite que las bacterias se puedan mover en diferentes direcciones dentro del espacio de búsqueda y evita la convergencia prematura como se sugiere en (Kasaiezadeh, Khajepour & Waslander, 2014).

La Ilustración 3.4muestra el comportamiento del nado de explotación, este ejemplo utiliza el mismo espacio de búsqueda de la Ilustración 3.3. En este caso, la bacteria en proceso buscará en un espacio más pequeño cerca de su punto original, la zona marcada con el circulo dibujado con líneas de puntos.

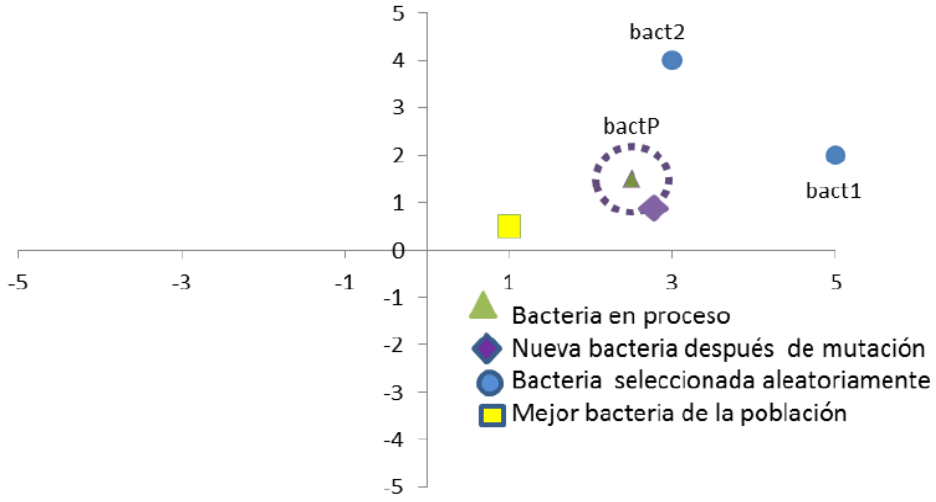

Ilustración 3.4 Comportamiento del nado de explotación.

Es importante remarcar que el nado de exploración (ecuación 3.9) realiza movimientos más grandes debido a la mutación con bacterias seleccionadas aleatoriamente dentro del espacio de búsqueda. El nado de explotación (ecuación 3.10) genera pequeños movimientos utilizando el tamaño de paso aleatorio en el proceso de búsqueda.

En el ciclo medio del proceso quimiotáxico es aplicado el operador de agrupamiento empleando la ecuación 3.6. Un ejemplo de su comportamiento se muestra en la Ilustración 3.5, donde β es un parámetro positivo definido por el usuario entre (1, 2). Sin embargo, en esta propuesta si una solución viola el límite de las variables de decisión, a diferencia de MBFOA, una nueva solución de x_i es generada aleatoriamente entre los límites inferior y superior $L_i \leq x_i \leq U_i$ de las variables de decisión.

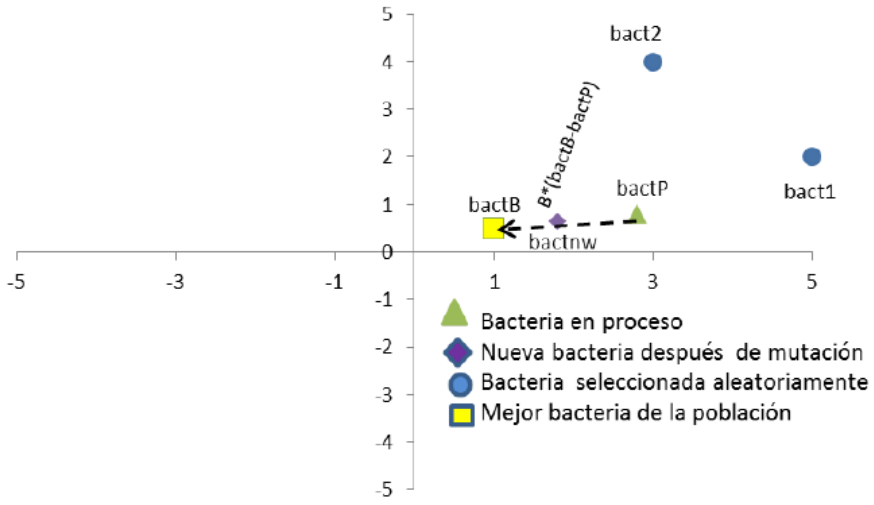

Ilustración 3.5 Comportamiento del nado de agrupamiento.

La Ilustración 3.6 muestra la posición final de las bacterias después de un nado de exploración, un nado de explotación y un agrupamiento. Sin embargo, este comportamiento puede ser diferente debido a que los nados no son secuenciales en el proceso quimiotáxico.

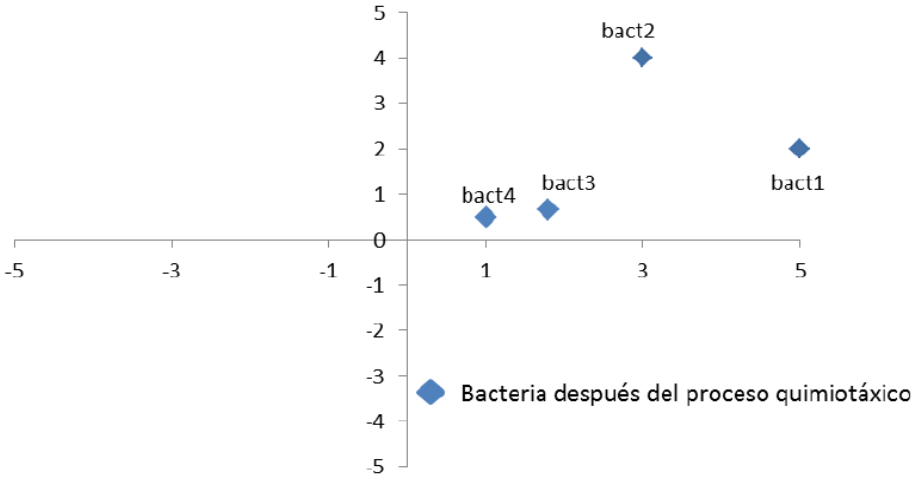

Ilustración 3.6 Bacterias después de un nado de exploración, un nado de explotación y un agrupamiento.

Es importante mencionar que el operador de búsqueda local *SQP* es utilizado solo en la primera y en el ciclo medio de las generaciones del algoritmo. La estructura de TS-MBFOA se muestra en la Ilustración 3.7, el pseudocódigo correspondiente se presenta en la Ilustración 3.8 donde los

parámetros de entrada son: número de bacterias S_b, límite del ciclo quimiotáxico N_c, número de bacterias a reproducir S_r, factor de escalamiento β para el operador de agrupamiento, R factor de escalamiento para el tamaño de paso, frecuencia de la reproducción RepCycle, número de generaciones *GMAX*, frecuencia del buscador local LS_G. Las modificaciones y nuevos mecanismos propuestos son remarcados con texto negritas.

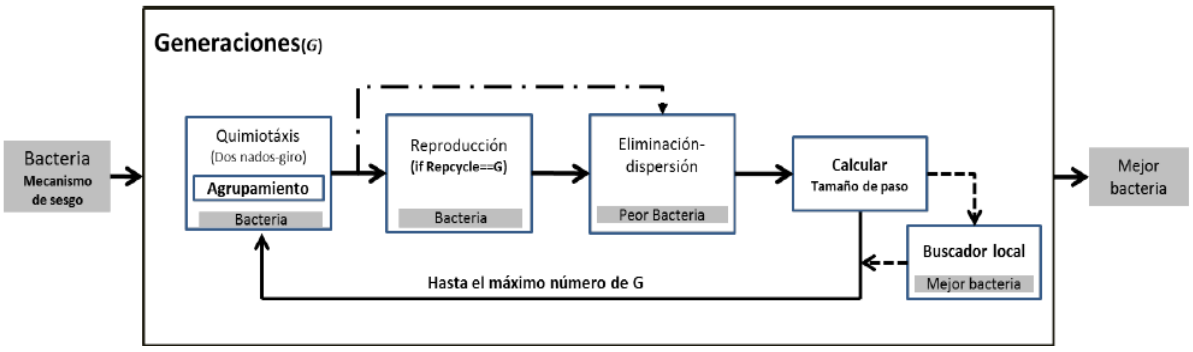

Ilustración 3.7 Procesos generales de TS-MBFOA.

Crear una población inicial de bacterias usando el mecanismo de sesgo $\boldsymbol{\theta^i(j,0)}\ \forall i,\ i = 1,\ldots,S_b$
Evaluar $\boldsymbol{f(\theta^i(j,0))}\ \forall i,\ i = 1,\ldots,S_b$
for *G=1 to* $\boldsymbol{GMAX}$ **do**
  for *i=1 to* $\boldsymbol{S_b}$ **do**
    for *j=1 to* $\boldsymbol{N_c}$ **do**
      En el proceso quimiotáxico intercalar los nados propuestos con las Ecs. 5.1 y 5.2. Aplicar el operador de agrupamiento con la Ec. 3.4 usando β para la bacteria $\boldsymbol{\theta^i(j,G)}$
    end
  end
  if *G mod* $\boldsymbol{RepCycle}$ *== 0* **then**
    Realizar el proceso de reproducción ordenando la población de acuerdo a las reglas de factibilidad y eliminar a $\boldsymbol{S_r}$ peores bacterias y duplicar el resto de bacterias $\boldsymbol{S_b - S_r}$.
  end
  Realizar el proceso de eliminación-dispersión eliminando a la peor bacteria $\boldsymbol{\theta^w(j,G)}$ de la población actual considerando la técnica de manejo de restricciones.
  Actualizar el vector de tamaño de paso usando la Ec. 5.3
  if *G mod* $\boldsymbol{LS_G}$*==0* **then**
    Aplicar SQP a la mejor bacteria de la población. Si la bacteria obtenida es mejor, ésta reemplaza a la mejor bacteria.
  end
end

Ilustración 3.8 Pseudocódigo de TS-MBFOA.

Capítulo IV. Aplicación de la metodología y desarrollo

En este capítulo se muestra la metodología ya implementada en este proyecto, con los diagramas UML y sus diferentes descripciones de cada uno de ellos. Además, se muestra detalladamente la construcción de la GUI, y una breve descripción de sus componentes.

4.1 Modelo de desarrollo

4.1.1 Diagrama de clases

El diagrama de clases representa como trabaja internamente el algoritmo TS-MBFOA, gráficamente. En la Ilustración 3.1 se puede observar que se desglosa desde la clase principal Config y sus funciones a las demás clases, todas dependiendo unas de otras.

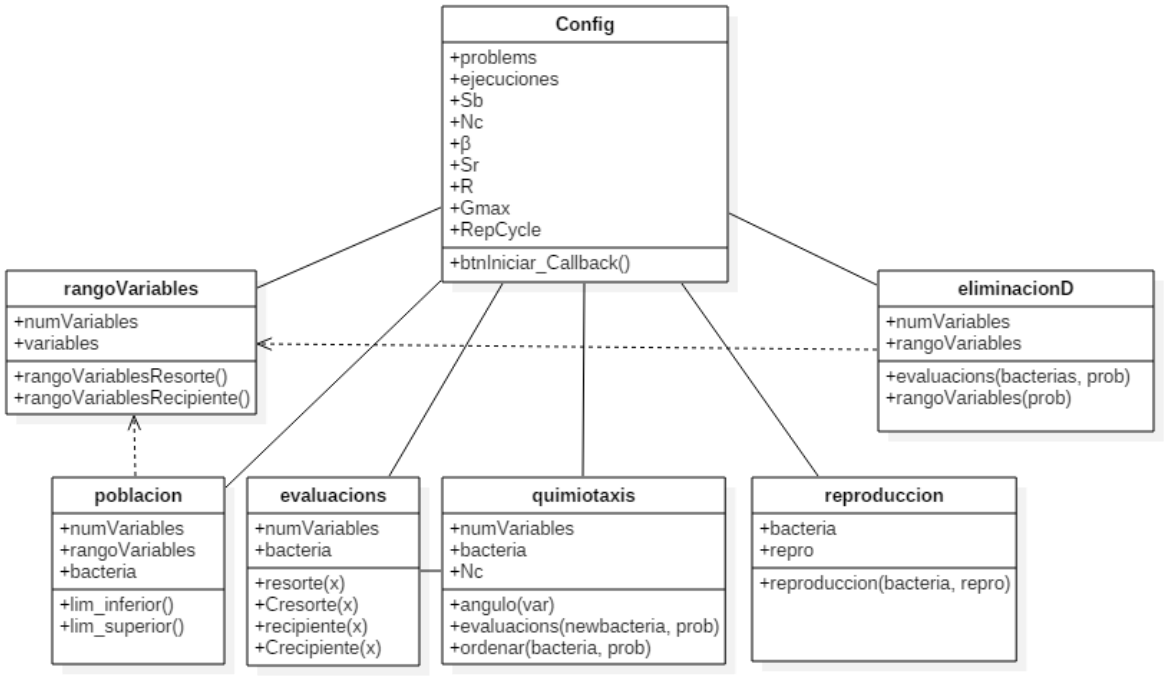

Ilustración 4.1 Diagrama de clases.

Este diagrama se ha implementado para entender cómo el algoritmo funciona, cuales parámetros son los que admite, su proceso quimiotáxico, la reproducción y de donde parte la generación de la población aleatoria.

4.1.2 Diagrama de caso de uso

La GUI desarrollada para el TS-MBFOA tiene un proceso de ejecución favorable para el usuario final, sin embargo, el presente diagrama hace alusión de cómo el software trabaja y se relaciona con el usuario (actor) para la obtención de los resultados a partir de la calibración de los parámetros del algoritmo. En la Ilustración 3.2 se muestra el diagrama de caso de uso del sistema en cual contiene una breve descripción de cada elemento que lo conforma.

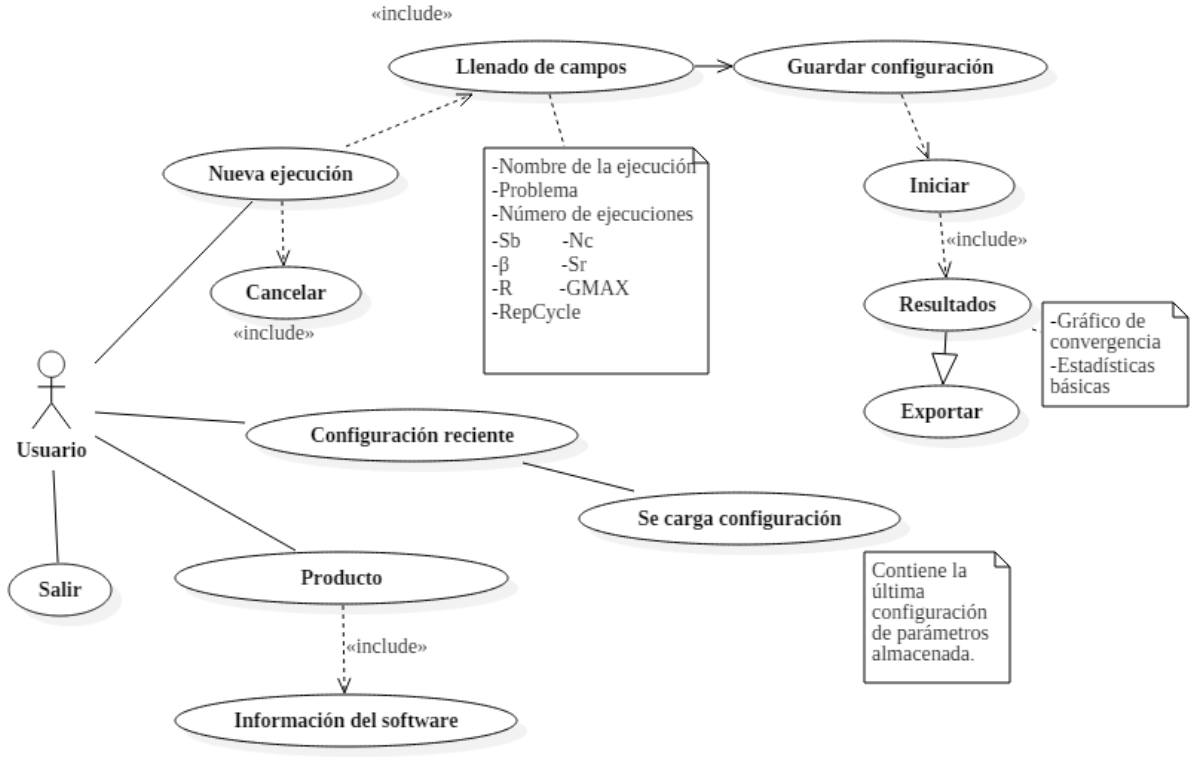

Ilustración 4.2 Diagrama de caso de uso.

Para comprender mejor la representación del diagrama de caso de uso de la Ilustración 4.2, se presenta una breve descripción general de los componentes que interactúan en las Tabla 4.1, Tabla 4.2 y Tabla 4.3.

Nombre	**Nueva ejecución**
Descripción	Ingresa a una nueva ejecución
Actor	Usuario
Precondición	El usuario debe estar dentro de la ventana de nueva ejecución
Flujo normal	El usuario llena los campos necesarios para la ejecución del algoritmo, guarda la configuración, inicia el sistema para calcular los resultados, se obtienen los resultados y se exportan
Flujo alternativo	El usuario cancela la nueva ejecución

Tabla 4.1 Descripción de una nueva ejecución.

Nombre	**Configuración reciente**
Descripción	Se obtiene la configuración reciente de los parámetros
Actor	Usuario
Precondición	El usuario debe entrar al apartado de configuración reciente
Flujo normal	Se carga la última configuración desde los archivos de almacenamiento
Flujo alternativo	El usuario no encuentra la última configuración en los archivos de almacenamiento

Tabla 4.2 Descripción de una configuración guardada.

Nombre	**Producto**
Descripción	Información del software
Actor	Usuario
Precondición	El usuario debe estar dentro del sistema
Flujo normal	El usuario ingresa a la información del software, para conocer los datos del desarrollador, año y fecha en las que fue creado
Flujo alternativo	El usuario sale del sistema

Tabla 4.3 Descripción de la GUI.

4.1.3 Diagrama de secuencia

En el diagrama de secuencia se origina la sucesión que el usuario y software toman para ejecutar y obtener resultados del algoritmo. Dicho diagrama se describe en la Ilustración 4.3.

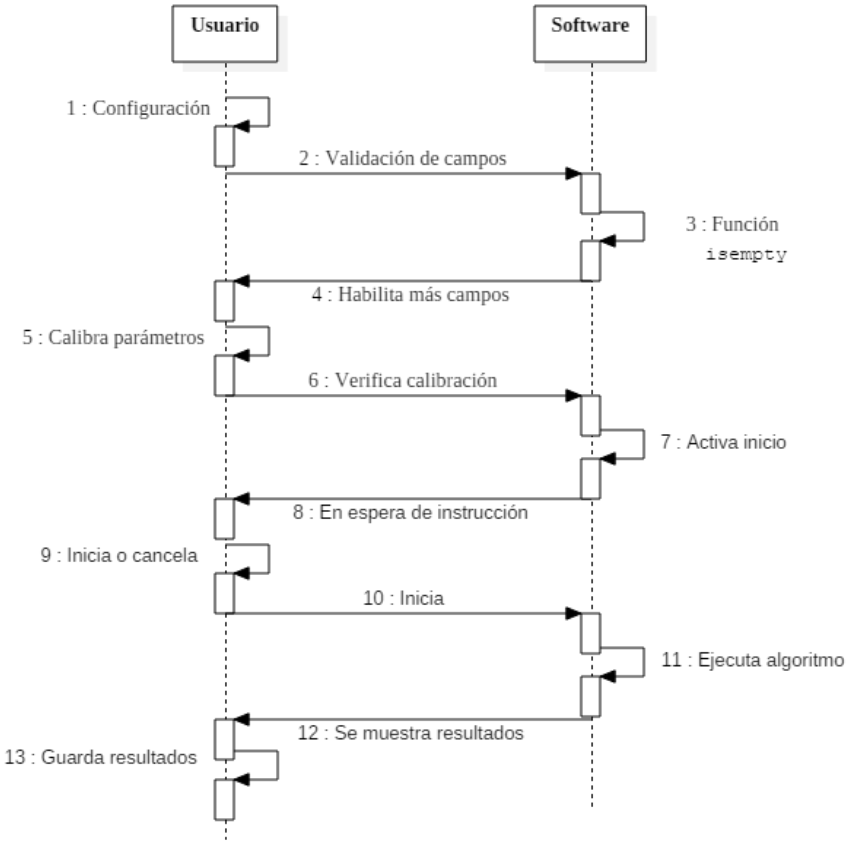

Ilustración 4.3 Diagrama de secuencia.

A continuación, se explican los pasos del diagrama de secuencia:

1. El usuario realiza la configuración.

2. El software valida los campos conforma a la configuración antes realizada.

3. El software ejecuta la función `isempty` para verificar que ningún campo de la configuración este vacío.

4. Una vez ejecutada la función `isempty`, se habilita los campos para ingresar datos.

5. El usuario ingresa la calibración de parámetros según lo que se desee.

6. Una vez ingresado los parámetros para la calibración, el software se encarga de verificar los datos.

7. Una vez el software verifica la calibración activa el inicio.

8. Posteriormente el software queda en espera de instrucciones por el usuario.

9. El usuario inicia o cancela el proceso.

10. El usuario permite la ejecución del proceso.

11. El software recibe indicaciones del usuario para poder iniciar la ejecución del algoritmo.

12. El software muestra los resultados obtenidos durante la ejecución al usuario.

13. El usuario guarda los resultados.

4.1.4 Diagrama de actividad

El diagrama de actividad presentado en la Ilustración 4.4 representa la funcionalidad y describe el flujo de control que tiene el software.

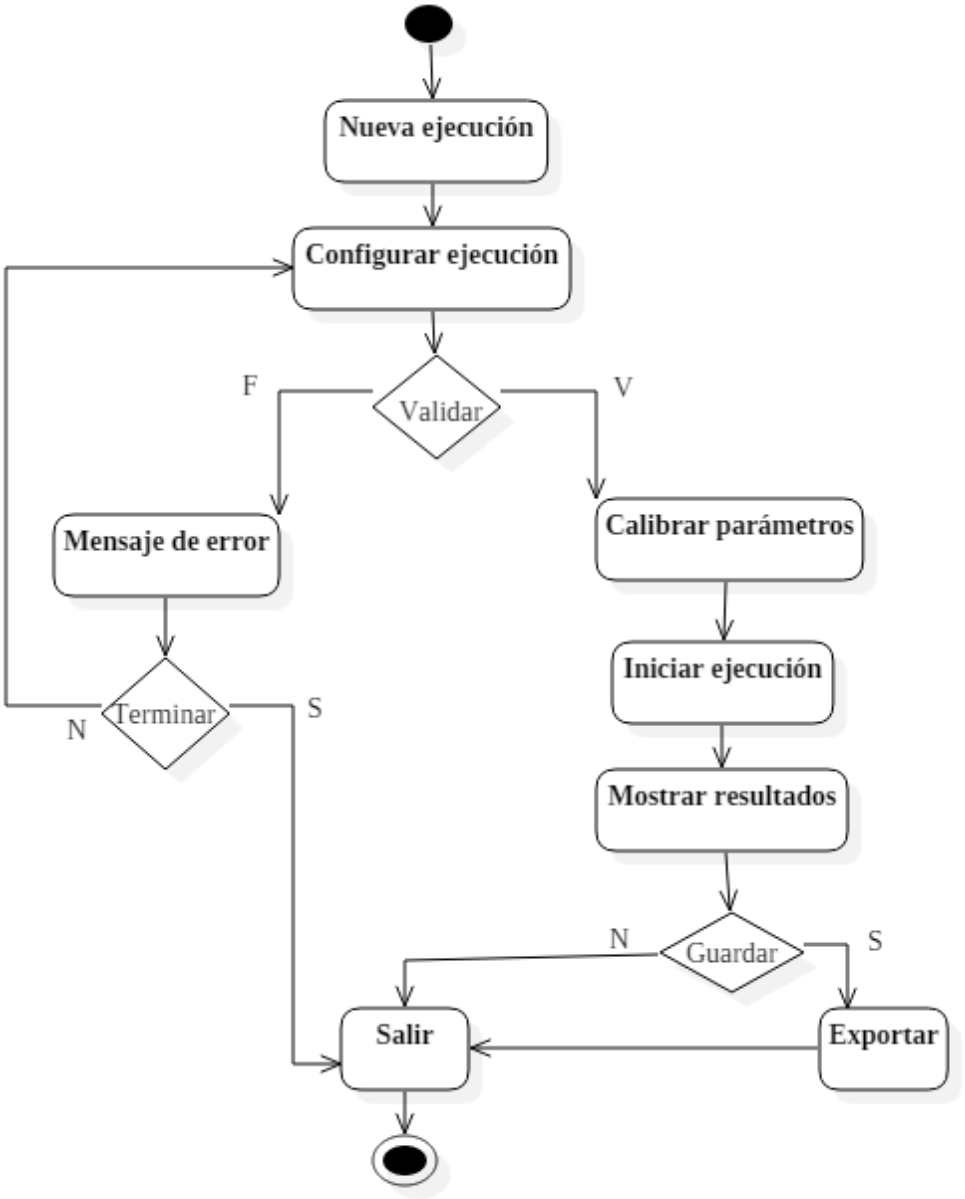

Ilustración 3.4 Diagrama de actividades.

A continuación, se describe la actividad generada en durante la ejecución del TS-MBFOA.

1. Se genera una nueva ejecución.

2. Se configura la ejecución.

3. Una vez obtenida la configuración, se valida si es verdadero continúe en el paso 7, de lo contrario seguir con el paso 4.

4. Se muestra un mensaje de error.

5. Después de mostrar el mensaje se decide si terminar o no el proceso, en caso de no terminar, es necesario regresar al paso 2, de lo contrario seguir con el paso 6.

6. Si se selecciona que, si se desea terminar por el error, se muestra el botón para finalizar el sistema o volver a ejecutar la configuración (paso 2).

7. Se agregan los parámetros para la calibración.

8. Se inicia la ejecución del algoritmo.

9. Se obtienen los resultados de la ejecución anterior.

10. Se toma la decisión de guardar o no guardar. Si no se desea guardar, el software automáticamente muestra el botón salir, una vez seleccionado se termina el proceso, pero si se desea guardar sigue al paso 11.

11. Se exporta los resultados obtenidos, y se da clic al botón salir para terminar el proceso.

4.1.5 Diagrama de tiempo

La Ilustración 4.5 representa la interacción entre los eventos de tiempos y la duración por cada proceso que requiere el algoritmo al ser ejecutado.

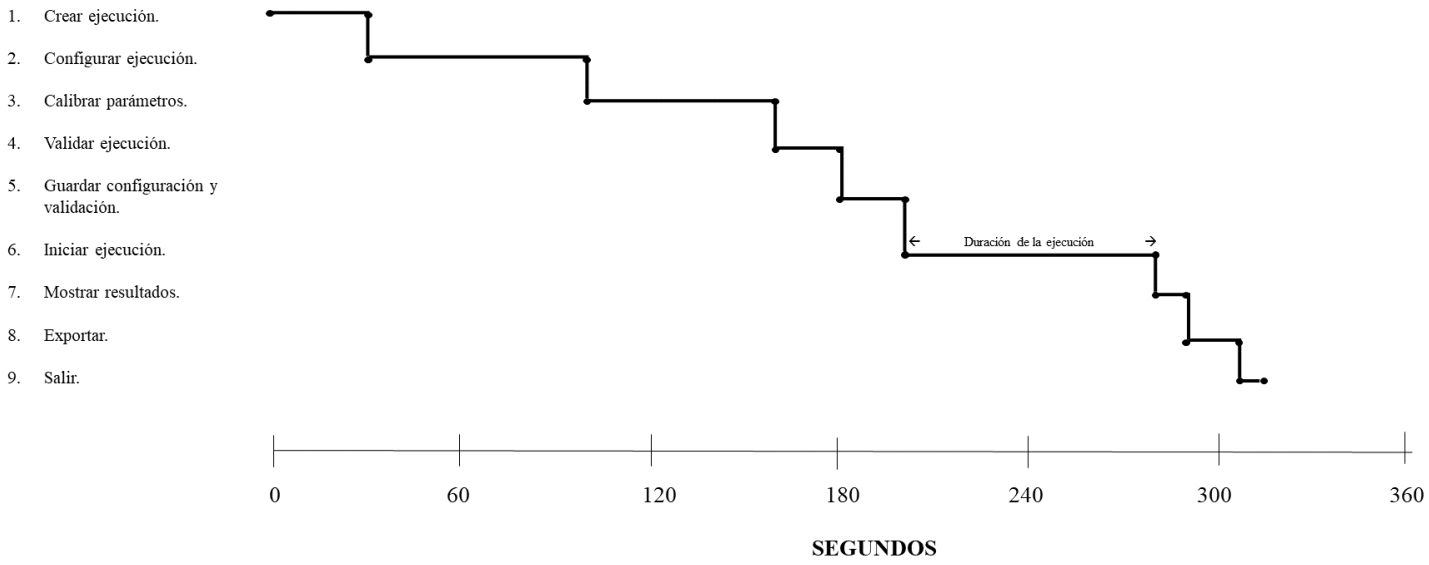

Ilustración 4.5 Diagrama de tiempo.

4.1.6 Diagrama de despliegue

El diagrama de despliegue de la Ilustración 4.6representa más el como el software se ejecuta y el hardware que necesita para ser ejecutado, además de los requerimientos del hardware que se solicita para la ejecución de este.

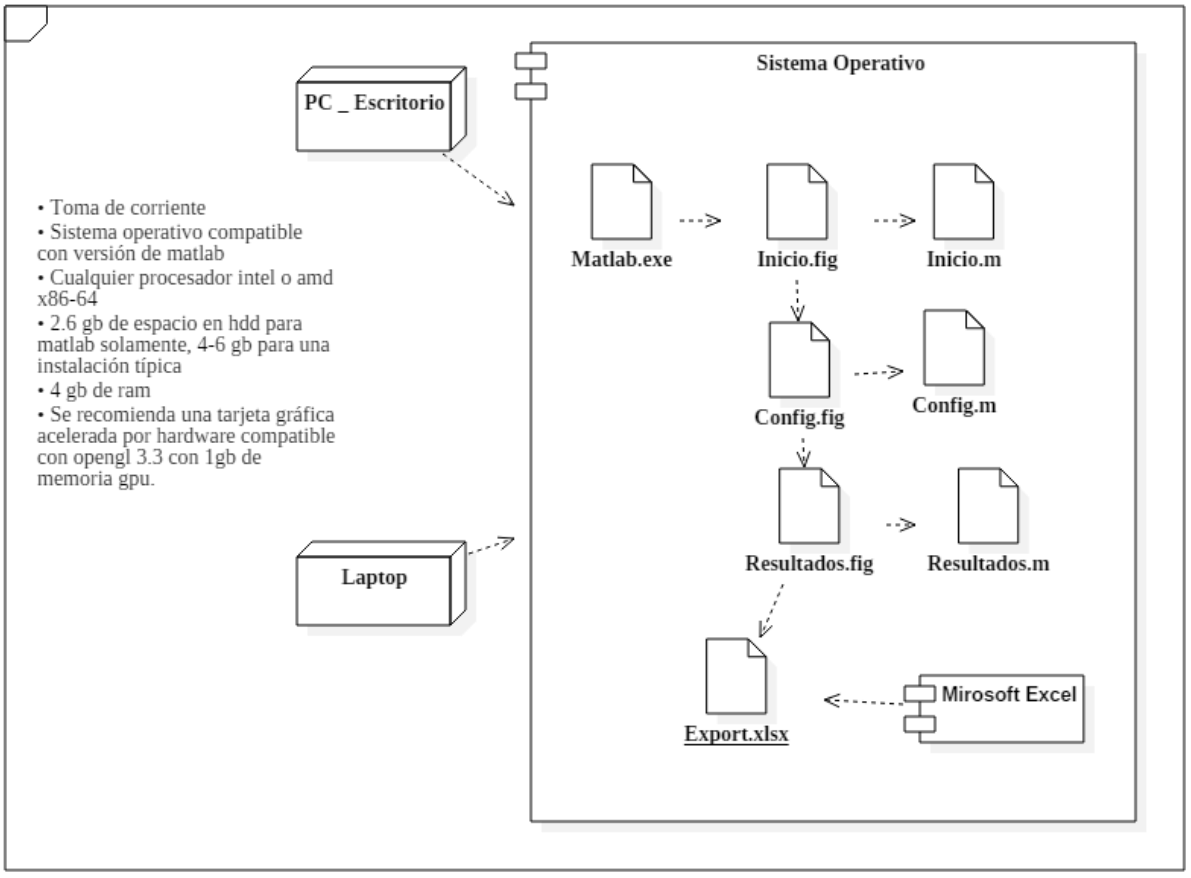

Ilustración 3.6 Diagrama de despliegue.

En el anterior diagrama se muestra los requisitos necesarios de una computadora de escritorio o laptop para la instalación de Matlab, que es la herramienta principal para el desarrollo de la interfaz de este proyecto, ya que el algoritmo se encuentra implantado en lenguaje M.

4.2 Desarrollo de la Interfaz Gráfica de Usuario (GUI)

El software desarrollado para la calibración de parámetros consta de varias interfaces que son presentadas y explicadas brevemente.

4.2.1 Pantalla principal

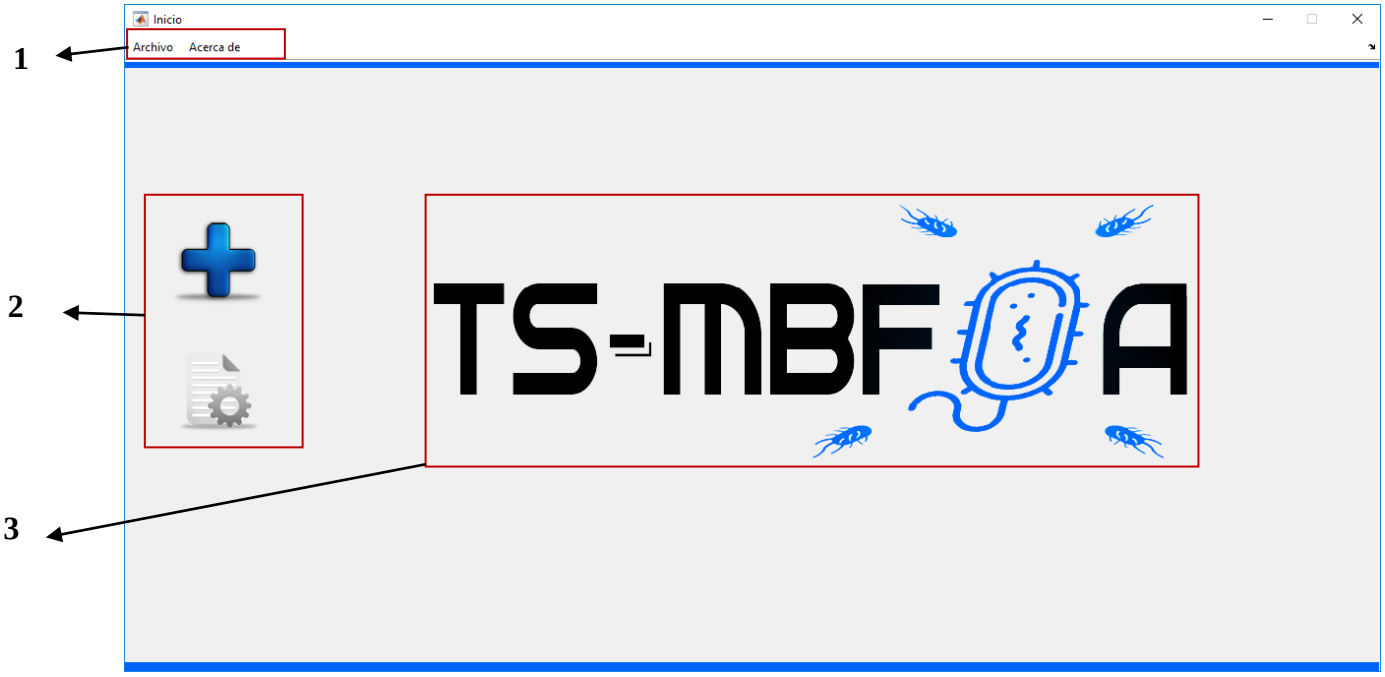

Ilustración 4.7 Pantalla principal del software.

En la pantalla principal (Ilustración 4.7) se puede visualizar varios elementos que la conforman como son: menú de opciones, botones de acceso directo y un logo principal.

1. **Menú de opciones**

El menú de opciones incluye las operaciones que el usuario final va a realizar como hacer una nueva ejecución del algoritmo, cargar la última configuración, salir del software y poder visualizar la información del software (acerca de) esta se muestra en la Ilustración 4.8. Además, que el usuario podrá acceder a ellas por teclado por combinación de teclas.

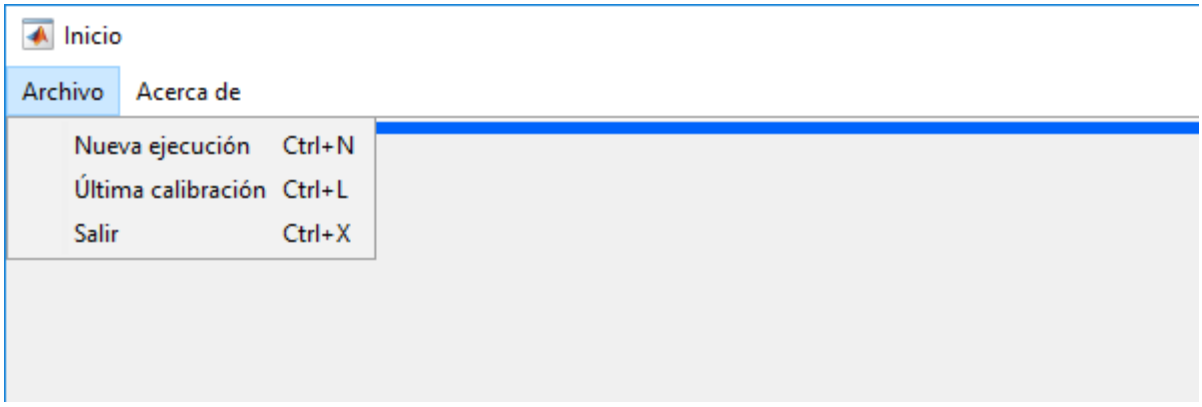

Ilustración 4.8 Menú de opciones del software.

La nueva ejecución permite al usuario elegir y configurar una nueva ejecución con los parámetros que el considere ya que permite la toma de decisiones de el mismo. La ultima ejecución carga la configuración anterior que el usuario calibró esto permitiendo que el usuario tome partida de un cierto punto, recordar con claridad y facilitar la calibración siguiente. La información del software permite al usuario ser informado en cual versión se está utilizando el software, año de desarrollo y los créditos correspondientes para los desarrolladores. Por último, se presenta la opción de salir con la cual culmina la ejecución del software.

2. **Botones de acceso directo**

El botón de acceso directo permite al usuario la forma más rápida de acceder a las opciones principales del software ofreciendo la descripción de lo que hace el botón al posicionar el puntero sobre el elemento (Ilustración 4.9).

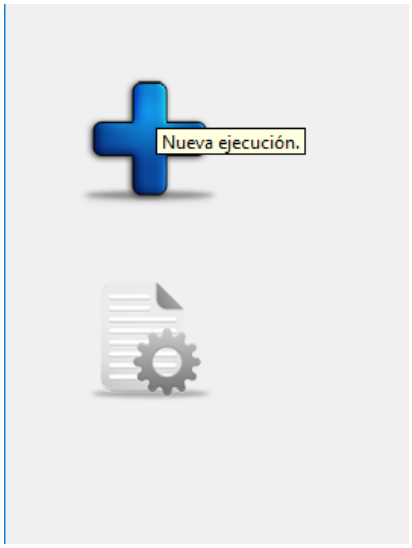

Ilustración 4.9 Botones de acceso directo del software.

Las opciones acceso directo son Nueva ejecución y Ultima calibración, estas fueron mencionadas en el punto número 1.

3. Logo principal

El logo principal es el elemento con el cual el software se distingue (Ilustración 4.10).

Ilustración 4.10 Logo del software.

El logo representa el nombre del algoritmo y se visualiza una cantidad de bacterias en su proceso de agrupamiento. Cabe mencionar que el logo es propiedad del investigador y desarrollador de esta tesis.

4.2.2 Nueva ejecución

La ejecución del algoritmo TS-MBFOA cuenta con la interfaz ya construida la cual en esta se visualiza varios elementos que ayudan al usuario Configurar, Calibrar, Guardar, Iniciar la ejecución y Detener el proceso. Al término de la ejecución se muestra el estado en que culminó el algoritmo y el tiempo transcurrido, además de la visualización de los datos obtenidos.

A continuación, se presenta la interfaz general y su descripción (Ilustración 4.11).

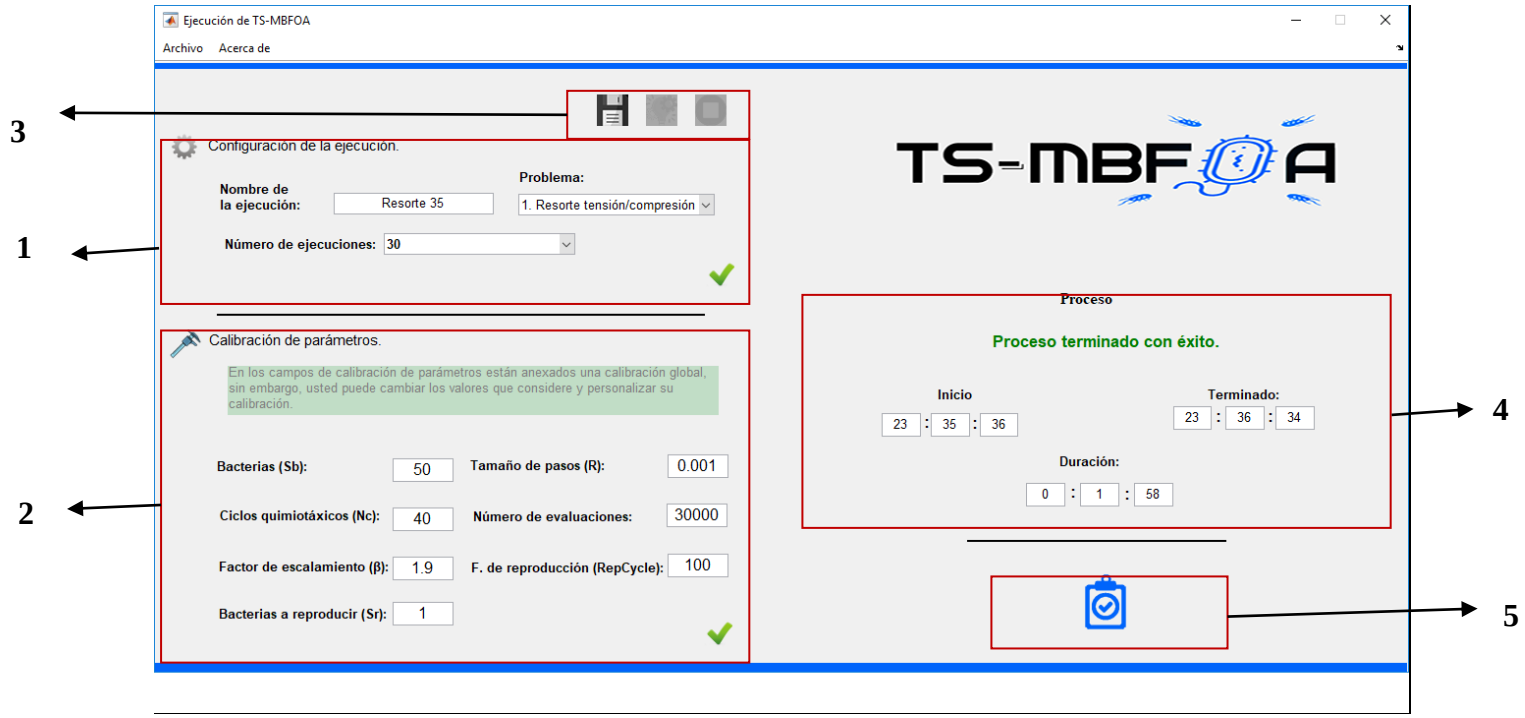

Ilustración 4.11 Interfaz de configuración y calibración del algoritmo.

1. Configuración de la ejecución

En esta sección se agrega el nombre de la ejecución, se selecciona el tipo de problema a resolver y el número de ejecuciones del algoritmo

2. Calibración de parámetros

En el siguiente apartado de esta ventana se mostrarán todos los datos necesarios para la calibración del algoritmo, dependiendo la necesidad del usuario se agregarán ciertos datos los cuales son: Número Bacterias (sb), Tamaño de paso (R), Ciclos quimiotaxicos (Nc), Número de evaluaciones, Factor de escalamiento (β), F. de reproducción (RepCycle), Bacterias a reproducir (Sr). Si el usuario ingresa un valor incorrecto el sistema le hará saber el error, ofreciendo los valores correctos**.** El mensaje de error en caso de ser mostrado es la siguiente Ilustración 4.12.

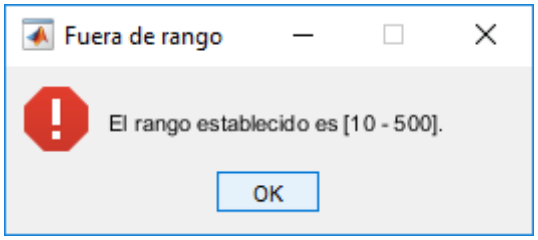

Ilustración 4.12 Mensaje de error.

3. Botones de acción

En la parte superior se observan tres botones el primero de Guardar, para guardar la configuración y calibración de los parámetros, el botón iniciar, para ejecutar el algoritmo el cual mostrará una barra de progreso para saber el estado de ejecución del algoritmo, este se muestra en la Ilustración 4.13y por último el botón Detener, para frenar la ejecución del algoritmo.

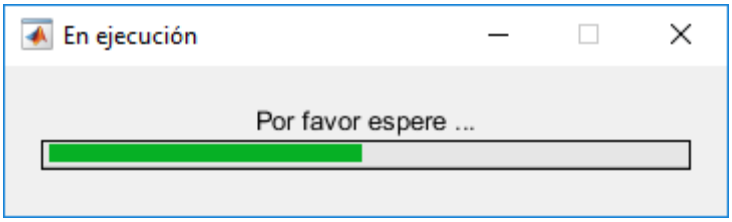

Ilustración 4.13 Barra de progreso.

4. **Estado de la ejecución**

Muestra la duración de ejecución del algoritmo, la hora de inicio y hora a la que termino de ejecutarse el algoritmo.

5. **Botón de resultados**

Este botón se habilita tras haber concluido la ejecución del algoritmo, para posteriormente dirigirnos a otra venta de resultados.

4.2.3 Resultados

Los resultados obtenidos parten de la calibración y configuración de los parámetros y estas son visualizado en la siguiente interfaz la cual se compone de cinco módulos importantes la cuales son: la configuración general, resultados finales, estadísticas, gráfico de convergencia de convergencia y un elemento importante que es un botón para la exportación de resultados a un archivo xlsx. A continuación, se presenta la interfaz general y se describe en breve (Ilustración 4.14).

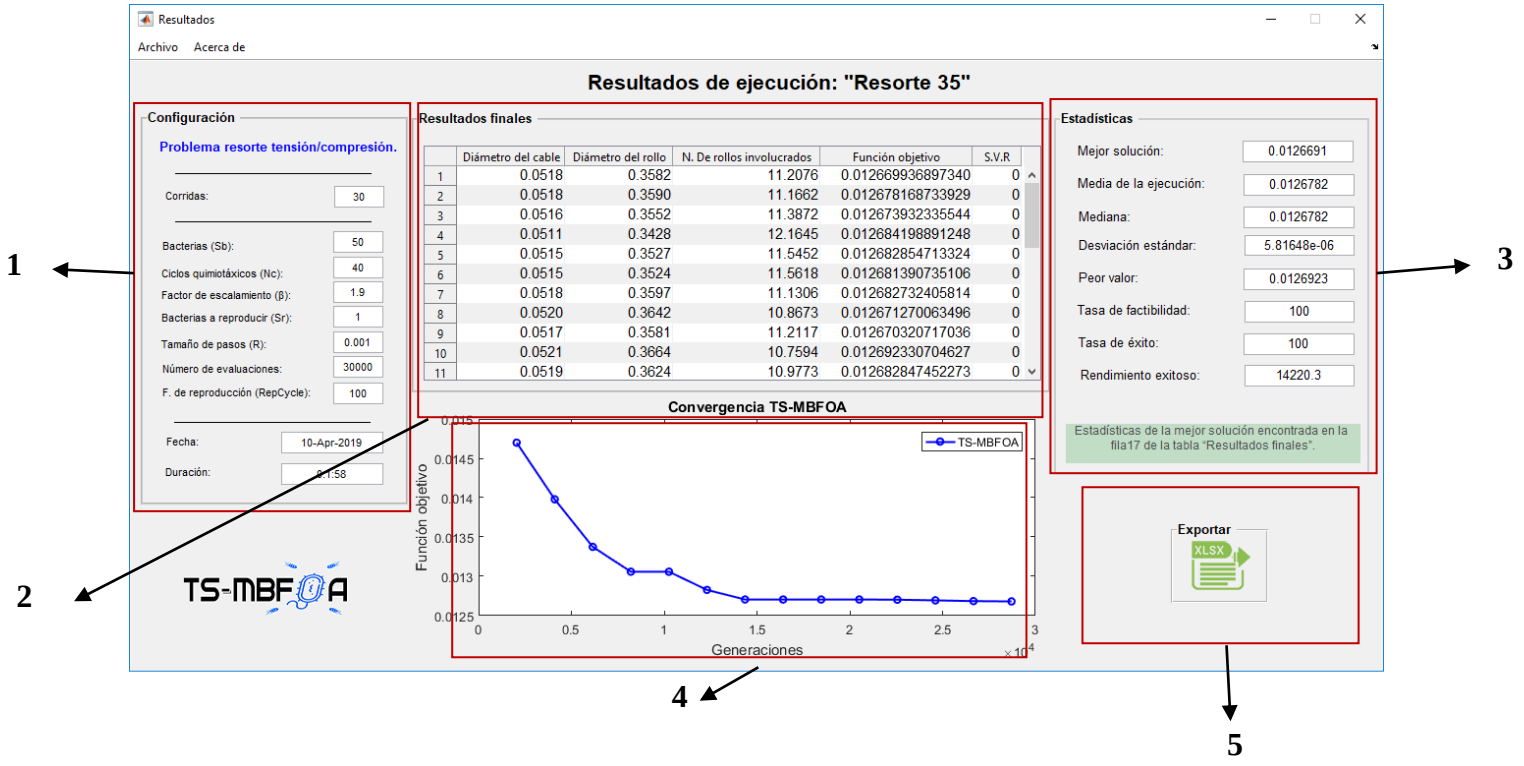

Ilustración 4.14 Interfaz de resultados.

1. Configuración

Se muestra la calibración y configuración agregada en la pantalla anterior de manera estática, para recordar al usuario los datos ingresados.

2. Resultados finales

En esta parte de la interfaz se muestran los resultados finales de las corridas seleccionadas por el usuario, se muestra una tabla la cual contiene los siguientes datos: diámetro de cable, diámetro del rollo, número de rollos involucrados, función objetivo (se describe en el capítulo 4), Suma de Violación de Restricciones (S.V.R.).

3. Estadísticas

Las estadísticas básicas son representadas para la mejor solución. En esta sección en la parte inferior se puede observar un texto resaltado en verde, el cual indica el número de fila de la mejor

solución en la tabla que el algoritmo ha encontrado basándose en: mejor, peor, media, mediana y desviación estándar.

4. Convergencia

En este apartado se muestra gráficamente el comportamiento del algoritmo con respecto a la mejora de la función objetivo durante las generaciones en una ejecución del algoritmo.

5. Botón de exportar

En la última sección se puede observar la opción de exportar una hoja de cálculo de Microsoft Excel (Ilustración 4.15).

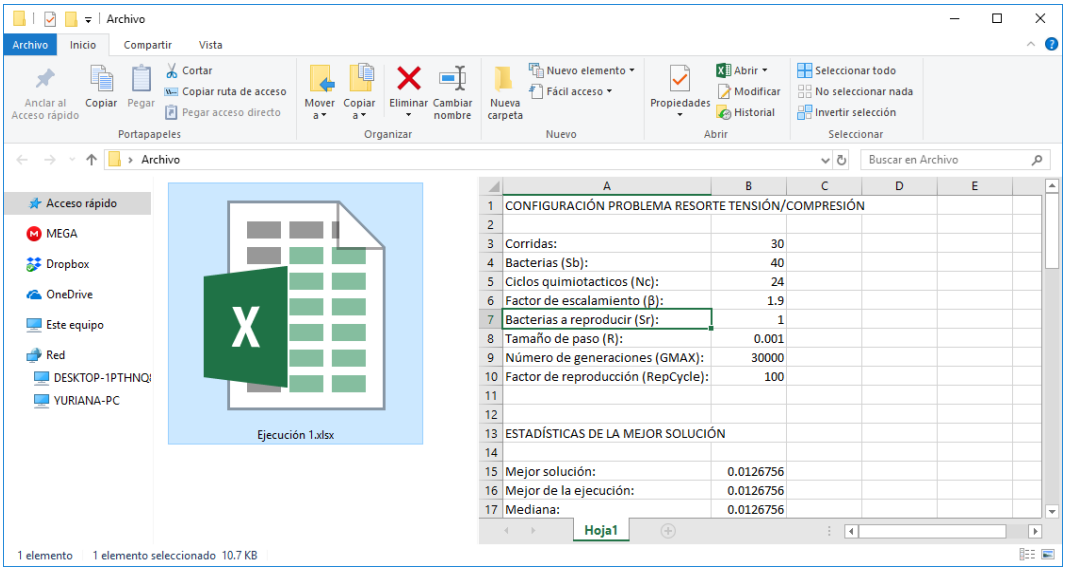

Ilustración 4.15 Exportar archivo.

4.2.4 Acerca de

En esta sección se muestra la información del software de TS-MBFOA, la versión de desarrollo, autor del algoritmo y desarrolladores (Ilustración 4.16).

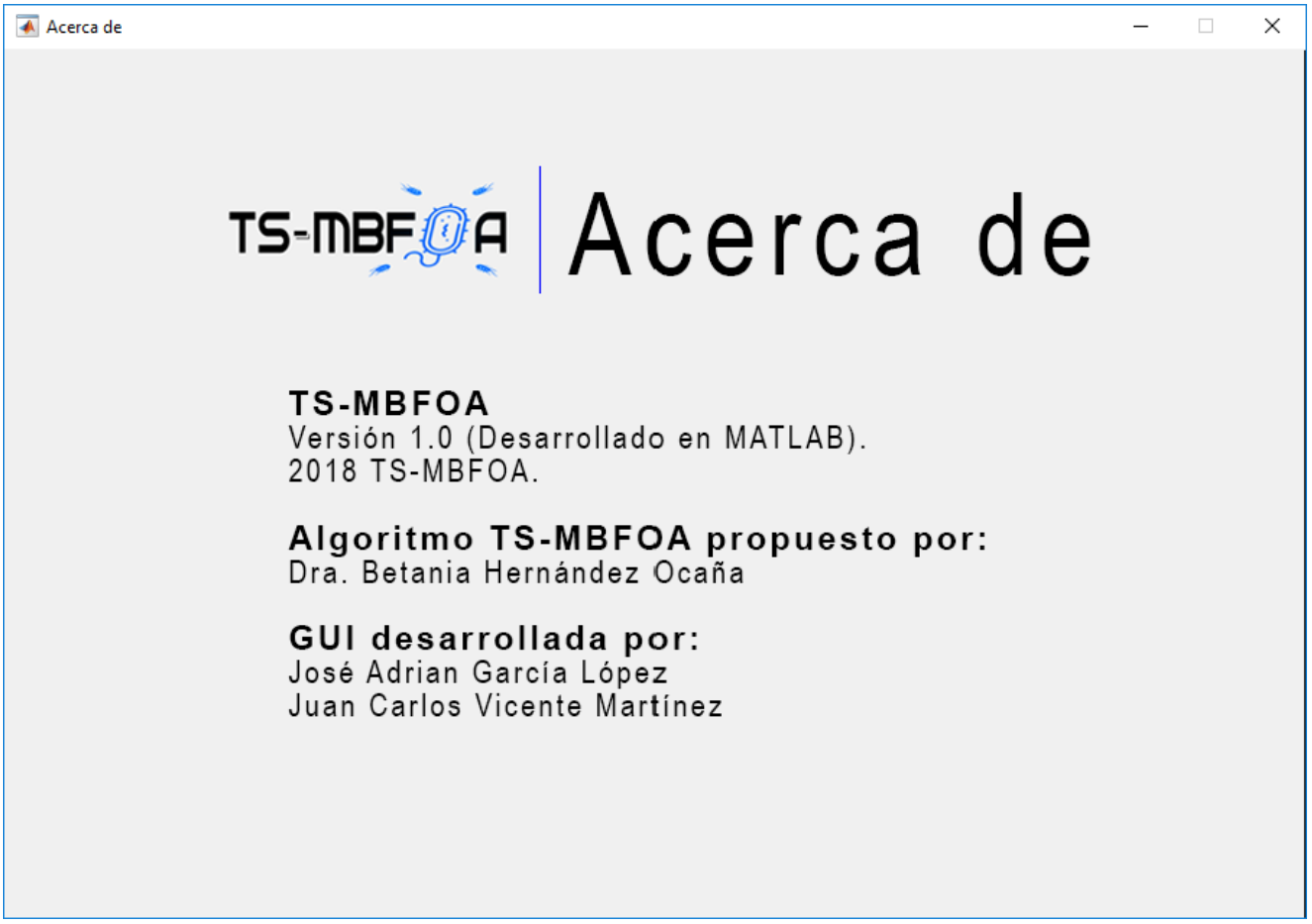

Ilustración 4.16 Interfaz de información del software.

Capítulo V. Pruebas y resultados

En este capítulo se describen el problema de prueba y los resultados obtenidos de los experimentos realizados al problema de diseño en ingeniería mecánica, con el fin de analizar el rendimiento de algoritmo usando una GUI diseñada y programada para el algoritmo TS-MBFOA. Por último, se presentan el gráfico de convergencia y resultados estadísticos para evaluar el rendimiento de TS-MBFOA, además de la prueba no paramétrica *Wilcoxon Signed Rank Test*.

5.1 Problema de prueba

- **Resorte tensión/compresión**

  Se busca minimizar el peso de un resorte de tensión/compresión (Ilustración 5.1). Sujeto a restricciones de desviación mínima, tensión de corte, frecuencia de oleada, límites sobre el diámetro exterior, esto sobre variables de diseño. Formalmente, el problema puede expresarse de la siguiente manera:

  Donde se busca minimizar:

$$(N+2)Dd^2$$

  Sujeto a:

$$g_1(X) = 1 - \left(\frac{D^3N}{71785d^4}\right) \leq 0$$

$$g_2(X) = \left(\frac{4D^2 - dD}{12566(Dd^3 - d^4)}\right) + \left(\frac{1}{5108d^2}\right) - 1 \leq 0$$

$$g_3(X) = 1 - \left(\frac{140.45d}{D^2N}\right) \leq 0$$

$$g_4(X) = \left(\frac{D+d}{1.5}\right) - 1 \leq 0$$

  Donde: $0.05 \leq d \leq 2$

$$0.25 \leq D \leq 1.3$$

$$2 \leq N \leq 15$$

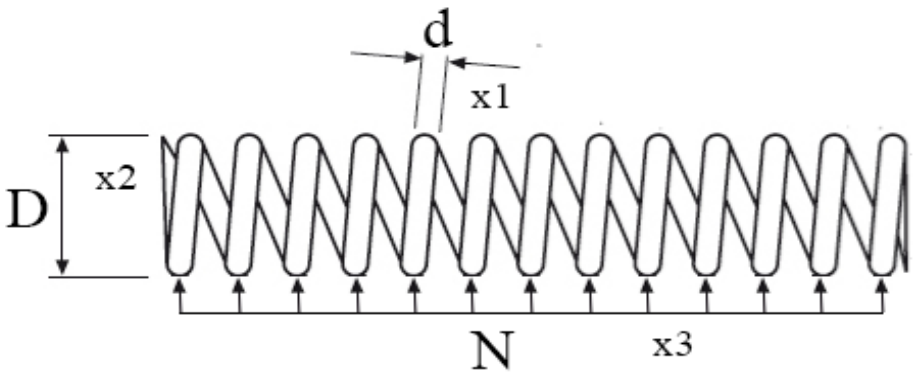

Ilustración 5.1 Representación del problema de diseño a resolver.

5.2 Medidas de rendimiento

Las estadísticas básicas a utilizar son: mejor y peor resultado, el valor de la media, mediana y desviación estándar obtenido del conjunto de 30 ejecuciones independientes realizadas al algoritmo. Otras medidas a utilizar son las siguientes (las primeras cinco son tomadas de Liang, Philip, Mezura-Montes, Clerc, Suganthan, Coello-Coello & Deb, 2006):

- **Ejecución factible:** Una ejecución donde al menos una solución factible es encontrada dentro del MA X_FEs.
- **Ejecución exitosa:** Una ejecución donde al menos una solución factible $\vec{x}$que satisface $f(\vec{x}) - f(\vec{x}*) \leq 0{,}0001$ es encontrada dentro del Max_FEs.
- **Tasa de factibilidad** = (número de ejecuciones factibles) / total de ejecuciones.
- **Tasa de éxito** = (número de ejecuciones exitosas) / total de ejecuciones.
- **Rendimiento exitoso** = media de (evaluaciones por ejecuciones exitosas) × (número total de ejecuciones) / (número de ejecuciones exitosas).
- Para el problema del Resorte tensión/compresión el mejor valor óptimo conocido en el estado del arte es 0.012681.

***Wilcoxon Signed Rank Test*:** Es una prueba no paramétrica de comparación de dos muestras relacionadas. La cual determina que los datos de la población no tienen una distribución normal. Esta prueba debe usarse si las diferencias entre pares de datos no están distribuidas normalmente. Existen dos versiones ligeramente diferentes de la prueba:

1. La *Signed Rank Test* de *Wilcoxon* compara su mediana de muestra con una mediana hipotética.

2. La prueba de *Wilcoxon matched-pairs sign-rank* calcula la diferencia entre cada conjunto de parejas emparejadas, y luego sigue el mismo procedimiento que el test de rango armado para comparar la muestra con alguna mediana.

El término *Wilcoxon* se usa a menudo para cualquiera de las dos pruebas. Esto por lo general no es confuso, ya que deberá ser obvio si los datos coinciden o no. La hipótesis establece que hay diferencias respecto a la tendencia central de las poblaciones y puede ser direccional o no. El contraste se basa en el comportamiento de las diferencias entre las puntuaciones de los elementos de cada par asociado, teniendo en cuenta no solo el signo, sino también la magnitud de la diferencia.

En este trabajo de investigación para el cálculo de WSRT se utilizará una calculadora publica encontrada en la web (Statistics, 2017). El nivel de significación aplicada es del 95%. En todas las tablas, la propuesta derivada es considerada como el segundo algoritmo de comparación.

Las propuestas derivadas de TS-MBFOA han sido codificadas usando lenguaje M y Matlab R2009b y ejecutadas en una PC con un procesador Core 2 Duo 3.5, 4GB de RAM, y 64 bits del Sistema Operativo Windows 7.

5.3 Configuración y calibración del algoritmo

- **Configuración**

Para la ejecución de TS-MBFOA sobre el problema del Resorte tensión/compresión, la configuración de prueba es llamada *Resorte1* y se seleccionó el problema 1 que lleva el mismo nombre. Además, se seleccionó el número de ejecuciones en 30 debido a que es el número común de ejecuciones independientes en el estado del arte. La configuración se muestra en la Ilustración 5.2.

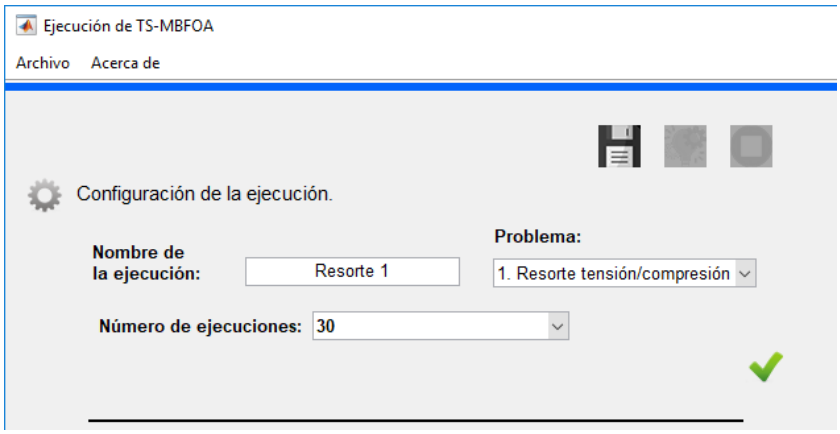

Ilustración 5.2 Configuración de la ejecución.

- **Calibración**

Tomando en cuenta las restricciones que se tienen, los valores asignados para este problema fueron: 50 para las bacterias (tamaño de la población), 40 ciclos quimiotáxico, 1.9 de factor de escalamiento, la bacteria a producir es de 1, tamaño de paso 0.001, 30,000 para el número de evaluaciones y el factor de reproducción de 100. Estos valores fueron ingresados en la parte de calibración de la interfaz como se muestra en la Ilustración 5.3.

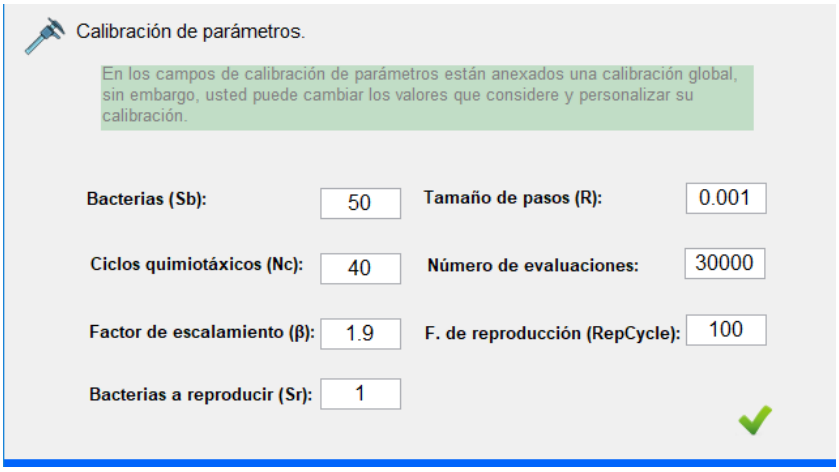

Ilustración 5.3 Calibración de parámetros.

Posteriormente de haber realizado la configuración y calibración de parámetros se guarda e inicia la ejecución del algoritmo.

5.4 Resultados

El algoritmo fue ejecutado 30 veces de manera independientes con los parámetros mostrados en la Ilustración 5.3. La mejor solución encontrada y los valores estadísticos básicos de las 30 ejecuciones independientes son presentados en la Ilustración 5.4, donde la mejor solución fue obtenida en la ejecución número 24 (ver Tabla 5.1). Además, la desviación estándar obtiene un valor muy cercano a cero, lo cual indica que todas las soluciones obtenidas en las 30 ejecuciones, son similares al óptimo global. Algo que nos permite confirmar esta observación es la tasa de factibilidad y tasa de éxito, donde las 30 ejecuciones independientes obtuvieron soluciones factibles e iguales o mejores que el óptimo global conocido en estado del arte. Finalmente, se requieren alrededor de 14,972 evaluaciones para obtener una solución factible y óptima para este problema con este algoritmo.

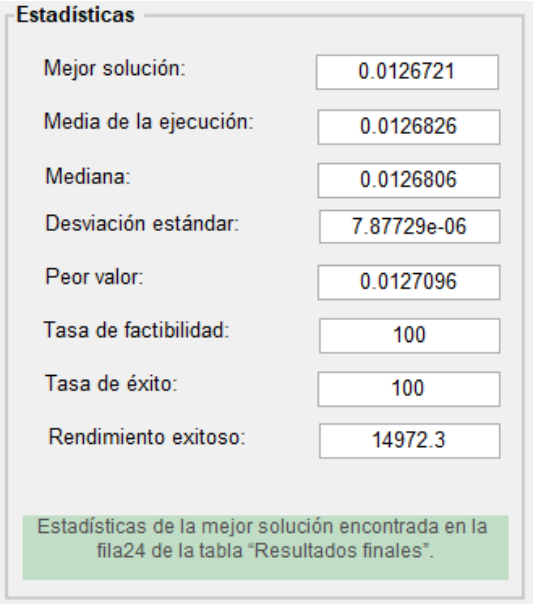

Ilustración 5.4 Estadísticas de la mejor solución.

La GUI presenta los resultados finales en una tabla conformada de 6 columnas; estas son: Diámetros de cable, Diámetros de rollo, Número de rollos involucrados, Función objetivo y S.V.R., las cuales puede verse en la Ilustración 5.5. A continuacion en la Tabla 5.1se presentan todoslos valores de las 30 ejecuciones independientes para cada columna mencionada.Del conjunto de soluciones encontradas por el algoritmo, el usuario final puede escoger una de acuerdo a sus necesidades y tener una mejor toma de decisiones.

Número de ejecución	Diámetro de cable	Diámetro de rollo	N. de rollos involucrados	Función objetivo	S.V.R
1	0.0517296	0.35760766	11.2513631	0.01268077	0
2	0.05115476	0.34379363	12.1005813	0.01268548	0
3	0.05192864	0.36217099	10.9894493	0.01268581	0
4	0.05208554	0.36627054	10.7605879	0.01267964	0
5	0.05181574	0.35959295	11.1305614	0.01267704	0
6	0.05116694	0.34418532	12.0703601	0.01267875	0
7	0.05225321	0.37036344	10.5372263	0.01267814	0
8	0.05248889	0.37626452	10.2439467	0.01269257	0
9	0.0525995	0.37898763	10.1037827	0.01269139	0
10	0.0522845	0.37107354	10.4989813	0.01267887	0
11	0.05194046	0.36270468	10.9534709	0.01267509	0
12	0.05166998	0.35606428	11.3423223	0.01268342	0
13	0.05189252	0.36155071	11.033371	0.01268924	0
14	0.05176469	0.35840028	11.197365	0.01267427	0
15	0.05095166	0.33897416	12.4225714	0.01269188	0
16	0.05207011	0.36594727	10.7791399	0.01267935	0
17	0.05204645	0.36525051	10.8106126	0.01267485	0
18	0.05095945	0.33939919	12.3827643	0.01267659	0
19	0.05229909	0.37139601	10.4838346	0.01268158	0
20	0.0517248	0.35737567	11.2648301	0.01268307	0
21	0.05201921	0.36445733	10.8597567	0.01268256	0
22	0.05148744	0.3518832	11.5975487	0.01268416	0
23	0.05128776	0.3471252	11.8874392	0.01268048	0
24	0.0520203	0.36471615	10.8394948	0.0126721	0
25	0.05195909	0.36288709	10.9571452	0.01269416	0
26	0.05143139	0.35037591	11.6775959	0.01267653	0
27	0.05149649	0.35201666	11.5760046	0.01267332	0
28	0.05155732	0.35331879	11.5009603	0.01267979	0
29	0.05215049	0.36788874	10.6819386	0.01268875	0
30	0.05227878	0.3705618	10.549316	0.01270959	0

Tabla 5.1 Resultados finales de TS-MBFOA de las 30 ejecuciones independientes realizadas en el problema del resorte de tensión/compresión.

En la Ilustración 5.5 se presenta las ejecuciones independientes realizadas por TS-MBFOA al problema del Resorte. En esta interfaz el usuario final puede observar todos los resultados que el algoritmo genera como: tabla de resultados finales, estadísticas básicas, grafica de convergencia y la opción de exportar en excel estos resultados.

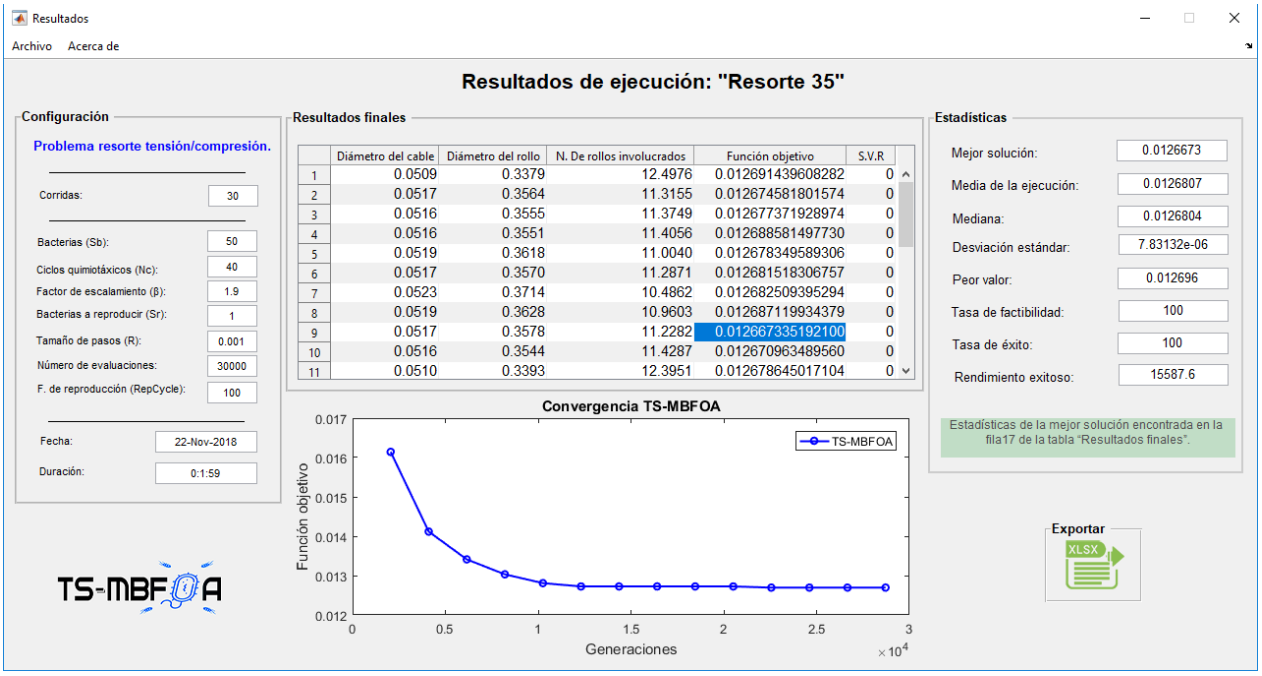

Ilustración 5.5 Interfaz general de resultados obtenidos de la función de prueba resorte tensión/compresión.

Criterios	TS-MBFOA	MBFOA
Mejor	0.01268077	0.01630937
Media	0.012682641	0.01524111
Mediana	0.01267979	0.0148695
Desviación estándar	7.87728E-06	0.00125107
Peor	0.01270959	0.01897264

Tabla 5.2 Comparación del algoritmo TS-MBFOA y MBFOA base a las estadísticas básicas.

En la Tabla 5.2 se presenta un comparativo de los resultados de TS-MBFOA y MBFOA con estadísticas básicas, donde los resultados de ambos algoritmos son generados con los mismos

parámetros, los cuales se pueden observar en la Ilustración 5.3. De acuerdo a los valores de la tabla, TS-MBFOA es el algoritmo que mejor resultados obtiene en cada una de las estadísticas al resolver este problema de prueba.

5.5 Convergencia

La convergencia de un algoritmo ocurre cuando una solución se conserva sin variaciones hasta el final de una ejecución. Esta solución puede ser: factible o no factible, un óptimo local o global. En la Ilustración 5.6 se muestra la convergencia de TS-MBFOA en el problema del resorte con los datos de la ejecución número 15, la cual es la mediana de las 30 ejecuciones. Para este problema de prueba, TS-MBFOA antes de las 10,000 generaciones tiene una convergencia rápida progresiva y continua, después de estas generaciones el algoritmo se mantiene sin variación, es decir convergen en un óptimo local debido a que el resultado de esta ejecución tiene un valor de 0.01269188.

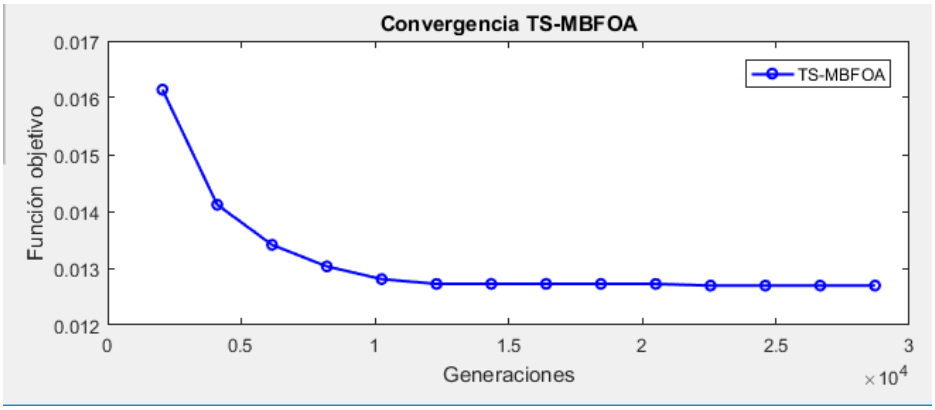

Ilustración 5.6 Gráfica de convergencia.

La Ilustración 5.6 presenta la convergencia de la ejecución número 15, donde las bacterias obtienen un buen resultado que hacen constante durante la ejecución del algoritmo; en promedio, la convergencia en una solución de buena calidad para el problema de diseño en ingeniería se obtiene entre la generación 10,000 a la 15,000 aproximadamente, de acuerdo a la Ilustración 5.6 y al resultado de la métrica llamada Rendimiento exitoso.

5.6 Prueba no paramétrica *Wilcoxon Signed Rank Test*

En la Tabla 5.3 se muestra el mejor resultado obtenido de cada una de las 30 ejecuciones independientes realizadas por MBFOA y TS-MBFOA. Estos datos son usados por la prueba WSRT con el objeto de conocer si existe diferencia significativa entre ambos grupos de valores.

La prueba WSRT fue configurada con un nivel de significancia del 0.05 y una hipótesis de 2 colas obteniendo un valor de z de -4.7821 y un valor de p menor a 0.00001, lo cual indica una diferencia significativa entre el conjunto de resultados de TS-MBFOA y MBFOA.

Número de ejecución	TS-MBFOA	MBFOA
1	0.01268077	0.01630937
2	0.01268548	0.01365103
3	0.01268581	0.01504151
4	0.01267964	0.01393995
5	0.01267704	0.01502329
6	0.01267875	0.01419695
7	0.01267814	0.01452018
8	0.01269257	0.01652729
9	0.01269139	0.01674499
10	0.01267887	0.01616278
11	0.01267509	0.0148695
12	0.01268342	0.0151396
13	0.01268924	0.01600057
14	0.01267427	0.01498494
15	0.01269188	0.01339251
16	0.01267935	0.01593207
17	0.01267485	0.01429101
18	0.01267659	0.01675205
19	0.01268158	0.01897264
20	0.01268307	0.0147699
21	0.01268256	0.01474724
22	0.01268416	0.01395306
23	0.01268048	0.01699236
24	0.0126721	0.01425301
25	0.01269416	0.01466751
26	0.01267653	0.01499838
27	0.01267332	0.01687673
28	0.01267979	0.01372426
29	0.01268875	0.01485669
30	0.01270959	0.01494201

Tabla 5.3 Resultados utilizados para comparar TS-MBFOA y MBFOA en la prueba *Wilcoxon signed rank test*.

Capítulo VI. Conclusiones y trabajos futuros

Para dar finalidad a esta tesis en este capítulo se presentan la conclusión del desarrollo del proyecto y los trabajos futuros que se esperan realizar.

6.1 Conclusiones

Debido a la necesidad del ajuste de parámetros del algoritmo TS-MBFOA por medio de una interfaz, que permita modificar sus parámetros propios de manera rápida y así ahorrar tiempo en la implementación del algoritmo, se ha desarrollado una GUI y se ha adaptado al algoritmo tal cual fue propuesto por sus autores. Esta interfaz tiene el propósito de permitir la calibración y configuración de los parámetros, mostrar las estadísticas básicas como: mejor y peor solución encontrada, media, desviación estándar, tasa de factibilidad, entre otras estadísticas, además, mostrar resultados en todas sus ejecuciones independientes, gráfica de convergencia y tiempo de ejecución Finalmente, el usuario puede exportar los resultados a una hoja de cálculo de Microsoft Office Excel.

La interfaz gráfica fue desarrollada en Matlab y para su diseño se generaron diferentes diagramas UML: Diagrama de clases, Diagrama de caso de usos, Diagrama de secuencia, Diagrama de actividad, Diagrama de tiempo y Diagrama de despliegue. TS-MBFOA fue adaptado para resolver un problema de ingeniería mecánica conocido como resorte de tensión/compresión. Para la prueba de la interfaz el algoritmo fue ejecutado en 30 ejecuciones independientes con el problema del resorte y los resultados fueron comparados con una versión anterior llamada MBFOA, donde los resultados de TS-MBFOA superaron a los resultados de MBFOA de acuerdo a las estadísticas básicas y la prueba no-paramétrica de *Wilcoxon Signed Rank Test*.

La GUI desarrollada tiene un rendimiento exitoso al ser ejecutada, por lo tanto, se obtienen beneficios como: permite ahorrar tiempo de calibración, ejecución, visualización e interpretación de resultados facilitando la toma de decisiones del usuario final.

6.2 Trabajos a futuros

El producto final de este trabajo es la GUI para el ajuste de parámetros del algoritmo, por lo tanto, este proyecto tiene extensiones, la cual son las siguientes:

1. Implementar un sistema integral (*framework*) donde se pueda dar posibles soluciones a muchos problemas de optimización permitiendo al usuario ingresar la función objetivo de su interés y así hacer que el algoritmo se pueda adaptar a cualquier función.

2. Hacer una migración de este proyecto a un lenguaje de programación que sea *open source*.

Bibliografía

Biswas, A., Dasgupta, S., Das, S. & Abraham, A. (2007). A Synergy of Differential Evolution and Bacterial Foraging Optimization for global optimization. *Neural Network World, 17*(6), 607-626.

Booch, G., Rumbaugh, J. & Jacobson, I. (2004). *Unified Modeling Language Reference Manual, The (2ª edición).* Addison-Wesley, Pearson Educación Superior.

Bosman, P. A. (2007). *Learning and Anticipation in Online Dynamic Optimization*, Vol. 51 of *Studies in Computational Intelligence,* Chapter Part I Optimum Tracking in Dynamic Environments, páginas. 129-152. Springer.

Brar, H. S., & Singh, V. P. (2014). *Fingerprint Image Recognition Using Bacterial Foraging Optimization Algorithm (BFOA)* (Tesis doctoral). Computer science and engineering department Thapar University, Patiala.

Bremermann, H. J. (1974). Chemotaxis and optimization. *J. Franklin Inst*, *297*(5), 397-404.

Bremermann, H. J. & Anderson, R. (1989). An alternative to backpropagation: A simple rule of synaptic modification for neural net training and memory, Technical Report PAM-483, Center for Pure and Applied Mathematics, University of California, Berkeley, California, USA.

Deb, K. (1995). *Optimization for Engineering Design*. Prentice-Hall, 1ra. Edición.

Deb, K. (2000). An Efficient Constraint Handling Method for Genetic Algorithms. *Computer Methods in Applied Mechanics and Engineering*, *186*(2-4), 311-338.

Dorigo, M., Maniezzo, V. & Colorni, A. (1996). The Ant System: Optimization by a Colony of Cooperating Agents. *IEEE Transactions of Systems, Man and Cybernetics-Part B, 26*(1), 29-41.

Eiben, A. & Smith, J. E. (2003). *Introduction to Evolutionary Computing (Volumen 53, página 18).* Natural Computing Series. Berling: Springer-Verlag.

Engelbrecht, A. (2006). *Fundamentals of Computational Swarm Intelligence.* Hoboken: John Wiley & Sons, Ltd.

Expósito, C. M. (2006). Interfaz gráfica de usuario: Aproximación semiótica y cognitiva. *Diseño gráfico y comunicación visual*, 1.

Flórez, E., Díaz, N., Gómez, W., Bautista, L. & Delgado, D. (2018). Evaluación de algoritmos bio-inspirados para la solución del problema de planificación de trabajos. I+D *Revista de Investigaciones*, 11(1), 142-155.

García-Peñalvo, F. J. & Pardo-Aguilar, C. (1998). GRIAL repository. Universidad de Burgos. Diagramas de clase en UML 1.1. Recuperado de: https://repositorio.grial.eu/bitstream/grial/353/1/DClase.pdf

Goering, R. (2004). Matlab edges closer to electronic design automation world. *Electronic Engineering Times*, (1341), 4-5.

Gutiérrez, J. A. G. & Díaz, A. M. H. (2014). Análisis e implementación de algoritmos evolutivos para la optimización de simulaciones en ingeniería civil (draft). *arXiv preprint arXiv*:*1401.5054.*

Hernández-Sampieri, R., Fernández-Collado, C. & Baptista-Lucio, P. (2010). *Metodología de la investigación.* Volumen 3. México: McGraw-Hill.

Hernández-Ocaña, B. (2009). *Adaptación del forrajeo de bacterias para resolver problemas de optimización con restricciones* (Tesis de pregrado). Universidad Juárez Autónoma de Tabasco, Cunduacán.

Hernández-Ocaña, B. & Mezura-Montes, E. (2009). Modified bacterial foraging optimization for engineering design, Editor C. H. Dagli & et al., *Proceedings of the Artificial Neural Networks in Engineering Conference (ANNIE'2009)*, Volumen 19 of *Intelligent Engineering Systems Through Artificial Neural Networks*, páginas 357-364, St. Louis, MO, USA. ASME Press.

Hernández-Ocaña, B., Mezura-Montes, E. & Pozos-Parra, P. (2016). Improved modified bacterial foraging optimization algorithm to solve constrained numerical optimization problems. *Applied Mathematics and Information Sciences*, *10*(2), 607-622.

Hernández-Ocaña, B., Pozos-Parra, P., Mezura-Montes, E., Portilla-Flores, E. A., Vega-Alvarado, E., & Calva-Yáñez, M. B. (2016). Two-swim operators in the modified bacterial foraging algorithm for the optimal synthesis of four-bar mechanisms. *Computational Intelligence and Neuroscience, 2016*, 17.

Hernández-Ocaña, B., Chavéz-Bosquez, O. A., Hernández-Torruco, J., Canúl-Reich, J. & Pozos-Parra, P. (2018). Bacterial Foraging Optimization Algorithm for Menu Planning. *IEEE Access*, *6*, 8619-8629.

Hernández-Ocaña, B., Ovando-Bautista, O., Aquino-De La Cruz, R. L. & Gaitán-Capetillo, J. D. (2018). *Tecnologías De Información En La Sociedad Contemporánea*, capitulo Dirección de Difusión y Divulgación Científica y Tecnológica. Optimización numérica con

restricciones usando el algoritmo basado en el forrajeo de bacterias normalizado, pagina 10, Dirección de Difusión y Divulgación Científica y Tecnológica.

Hongdan, L., Sheng, L. & Lanyong, Z. (2015). Ship collision avoidance path planning strategy based on quantum bacterial foraging algorithm. *2nd International Conference on Electrical, Computer Engineering and Electronics, 2015,* 612-621.

Joyanes-Aguilar, L. (1990). *Problemas de metodología de la programación*. Madrid, España: MaGraw-Hill Interamericana. Páginas 500.

Kasaiezadeh, A., Khajepour, A. & Waslander, S. L. (2014). Spiral bacterial foraging optimization method: Algorithm, evaluation and convergence analysis. *Engineering Optimization, 46*(4), 439-464.

Kaur, G. & Singh, H. (2013). An Intelligent System for Lung Cancer Diagnosis Using Fusion of Support Vector Machines and Back Propagation Neural Network. *International Journal of Science and Research* (IJSR), ISSN, páginas 2319-7064.

Kennedy, J. & Eberhart, R. C. (2001). *Swarm Intelligence*. Morgan Kaufmann Publishers, Electronic Data System, Inc. Páginas 512.

Kennedy, J. & Eberhart, R. C. (1995). Particle swarm optimization, in *Proceedings of IEEE International Conference on Neural Networks, Piscataway*, volumen 4, páginas 1942-1948.

Kumar, A., Veeranna, V., Durgaprasad, B. & Sarma, B. (2013). A MATLAB GUI tool for optimization of FMS scheduling using conventional and evolutionary approach. *International Journal of Current Engineering and Technology, 3*(5), 1739-1744.

Kushwaha, N., Bisht, V. S. & Shah, G. (2012). Genetic algorithm based bacterial foraging approach for optimization. In *IJCA Proceedings on National Conference on Future spects of Artificial intelligence in Industrial Automation*, número 2, páginas 11-14.

Larman, C. (2003). *UML y Patrones*. Pearson Educación en Madrid. Prentice Hall. 2da. Edición. Páginas 624.

Liang, J.J., Philip R., T., Mezura-Montes, E., Clerc, M., Suganthan, P.N., Coello-Coello, C.A. & Deb, K. (2006). Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization. *Journal of Applied Mechanics, 41*(8), 8-31.

Majhi, B. & Panda, G. (2007). Bacteria foraging based identification of nonlinear dynamic system. In *2007 IEEE Congress on Evolutionary Computation,* páginas 1636-1641. IEEE.

Mezura-Montes, E., Portilla-Flores, E. A. & Hernández-Ocaña, B. (2011). Optimization of a mechanical design problem with the modified bacterial foraging algorithm. In *XVII Congreso Argentino de Ciencias de la Computación*. Página 171-180.

Mishra, S., Redely, G. D., Rao, P. & Santosh, K. (2007). Implementation of new evolutionary techniques for transmission loss reduction. In *IEEE Congress on Evolutionary Computation*. Páginas 2331-2336. IEEE.

Muragán, C. I. D. (2012). Mejora de la interfaz del simulador de voz vox. Proyecto Fin de Carrera / Trabajo Fin de Grado, E.U.I.T. Telecomunicación (UPM), Madrid.

Nouri, H. & Hong, T. S. (2012). A bacterial foraging algorithm based cell information considering operation time. *Journal Manufacturing Systems, 31*(3), 326-336.

Papalambros, P. Y. & Wilde, D. J. (2000). *Principles of optimal design: Modeling and Computation*. Cambridge University Press. 2da. Edición.

Pappachen, A., & Fathima, A. P. (2015). BFOA based FOPID controller for multi area AGC system with capacitive energy storage. *International Journal on Electrical Engineering and Informatics*, *7*(3), 429-442.

Passino, K. M. (2002). Biomimicry of bacterial foraging for distributed optimization and control. *IEEE Control Systems Magazine*, *22*(3), 52-67.

Powell, M. J. D. (1978). Algorithms for Nonlinear Constraints that use Lagrangian Functions. *Mathematical Programming*, *14*(1), 224-248.

Pressman, R. (2010). Ingeniería del software (un enfoque práctico) (séptima edición, volumen 1). *España: Mc-GrawHill. Sitio Agrícola.*

RAE. (2008). Definición de optimización. Diccionario de la real Academia Española. Recuperado de: http://raes.es/

Reddy, C. V. & Siddaiah, P. (2016). Comparative Analysis of Dual Secure Based Medical Image Watermarking Technique to Increase Security of Watermark Data Using BFOA. *International Journal of Computer and Communication Engineering*, *5*(6), 381-397.

Sierra, M. M. R. (2006). Use of Coevolution and Fitness Inheritance for Multi-Objective Particle Swarm Optimization (Tesis doctoral), Centro de Investigación y de Estudios Avanzados Del Instituto Politécnico Nacional, Ciudad de México.

Schmuller, J. (2001). We think in the UML language: Programmers bookcase. Prentice Hall. Pearson Educación Latinoamerica.

Silvestrini, M. & Vargas, J. (2008). Fuentes de información primarias, secundarias y terciarias. Recuperado de: http://ponce.inter.edu/cai/manuales/FUENTESPRIMARIA.pdf.

Sparx-Systems (2018a). Diagrama de despliegue UML 2. Recuperado de: http://www.sparxsystems.com.ar/resources/tutorial/uml2_deploymentdiagram.html/

Sparx-Systems (2018b). Diagrama de tiempo UML 2. Recuperado de: http://www.sparxsystems.com.ar/resources/tutorial/uml2_timingdiagram.htm/

Statistics, S. S. (2017). Wilcoxon signed-rank test calculator. Recuperado de: http://www.socscistatistics.com/tests/signedranks/default2.aspx

Tabares, M. S., Pineda, J. D. & Barrera, A. F. (2008). Un patrón de interacción entre diagramas de actividades UML y sistemas workflow. *Revista EIA*, (10), 105-120.

Venugopal, R., C.H. & Siddaiah, P. (2014). Bacterial foraging optimization algorithm (BFOA) optimized adaptive hybrid DWT-SVD watermarking with encryption. *International Journal of Applied Engineering Research. 9*(24), 30911-30933.

Glosario

A

ACO: Optimización de Colonias de Hormigas.

AE: Algoritmos Evolutivos.

AG: Algoritmos Genéticos.

AIC: Algoritmos de Inteligencia Colectiva.

B

BFOA: Algoritmo de Optimización basado en el Forrajeo de Bacterias.

C

CAD: Diseño Asistido por Computadoras.

E

ED: Evolución Diferencial.

EE: Estrategias Evolutivas.

EPS: Script Post Encapsulado.

G

GUI: Interfaz Gráfica de Usuario.

GUIDE: Entorno de desarrollo gráfico de la interfaz de usuario.

I

IMBFOA: Algoritmo mejorado de optimización de forraje bacteriano modificado.

J

JPEG: Grupo Conjunto de Expertos en Fotografía.

M

MATLAB: Laboratorio de Matrices.

MBFOA: Algoritmo de Optimización basado en el Forrajeo de Bacterias Modificado.

N

NTS-MBFOA: Dos nados normalizado - Algoritmo de Optimización basado en el Forrajeo de Bacterias.

P

PE: Programación Evolutiva.

PG: Programación Genética.

PNG: Gráficos de red portátiles.

PONR: Problemas de Optimización Numérica con Restricciones.

PSO: Optimización por Enjambre de Partículas.

S

SQP: Programación cuadrática secuencial.

T

TS-MBFOA: Doble nado - Algoritmo de Optimización basado en el Forrajeo de Bacterias.

U

UML: Lenguaje Unificado de Modelado.