Tecnológico Nacional de México

Instituto Tecnológico de Veracruz

Implementación de Búsqueda Local en el Algoritmo de Colonia Artificial de Abejas para Resolver Problemas de Optimización con Restricciones

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

Ingeniero en Sistemas Computacionales

PRESENTA:

Diego de Jesús Isla López

Asesores:

Dr. Efrén Mezura Montes

Dr. Rafael Rivera López

2018

Dedicado a
mi familia

Resumen

Las aportaciones de la Inteligencia Artificial en el área de la optimización han resultado en el desarrollo de diversos métodos (metaheurísticas) inspirados en procesos presentes en la naturaleza que han demostrado alcanzar resultados favorables cuando se utilizan para resolver diversos problemas de optimización. Las metaheurísticas basadas en procesos de inteligencia colectiva son de las más estudiadas ya que por sus características son sencillas de implementar además de presentar un buen desempeño al abordar problemas de optimización. Existen metaheurísticas que han reportado resultados favorables tanto en competencias como en problemas reales, sin embargo algunas de ellas utilizan métodos de búsqueda que emplean cálculo de derivadas, lo que reduce la aplicación de estos métodos ya que las funciones deben ser continuas, además del aumento en el costo computacional de su implementación.

En este trabajo de tesis se presenta una variación al algoritmo de colonia artificial de abejas modificado (M-ABC) presentado por Cetina y Mezura añadiendo un operador de búsqueda local utilizando el método de direcciones conjugadas de Powell, el cual no utiliza cálculo de derivadas, para intentar obtener resultados similares al algoritmo de Evolución Diferencial con restricciones ε y mutación basada en gradiente (εDEag) presentado por Takahama y Sakai.

Este trabajo se divide en cinco capítulos descritos a continuación:

- En el capítulo 1 se presenta la descripción del proyecto de investigación, la justificación y el alcance del mismo.

- En el capítulo 2 se presentan las bases teóricas del proyecto, abordando los temás de optimización numérica y metaheurísticas inspiradas en la naturaleza.

- En el capítulo 3 se describe el algoritmo ABC original y las diferentes variantes existentes, haciendo énfasis en la modificación propuesta por Cetina y Mezura.

- El capítulo 4 describe la modificación propuesta en este trabajo de tesis al algoritmo de Cetina y Mezura.

- En el capítulo 5 se presenta la implementación en código del algoritmo propuesto, los problemas de prueba a utilizar, el mecanismo de calibración de parámetros utilizado y los resultados obtenidos.

Finalmente, se presentan las conclusiones obtenidas a partir del análisis de los resultados y las propuestas para trabajos futuros.

Índice general

Página

Índice de figuras

Página

Índice de tablas

Página

Índice de algoritmos

Página

Agradecimientos

Navegamos, oh diversos
Amigos míos, conmigo ya en la popa
Sois la fastuosa vanguardia que corta
El fluir de rayos e inviernos

Stéphane Mallarmé

Agradezco a mis padres y familiares por siempre apoyarme y estar conmigo en todo momento.

A mis amigos por la compañía, los regaños, los consejos, las aventuras, las risas, las lágrimas y la confianza.

A JARO por su aprecio y sus palabras de aliento para terminar esta etapa y comenzar las que vienen.

Al Dr. Efrén Mezura Montes por darme la oportunidad de trabajar en este proyecto y abrirme las puertas para realizar mis residencias en LANIA.

Al Dr. Rafael Rivera López por su tiempo, su apoyo incondicional, sus enseñanzas y su disposición a asesorarme en este proyecto.

Capítulo 1

Marco Metodológico

1.1. Antecedentes

En el área de la Investigación de Operaciones (IO) se tratan problemas tanto de ingeniería como de otras áreas que requieren ser optimizados de acuerdo a sus necesidades. Desde la aparición del método simplex en 1947, muchos problemas de la industria y otras áreas se han resuelto utilizando métodos de optimización. Optimización significa, a grandes rasgos, obtener la mejor solución posible a un problema determinado, dicho problema deberá estar representado normalmente por un modelo matemático. De acuerdo a las características del problema los modelos pueden ser lineales o no lineales, con restricciones o sin ellas, así como continuos o discretos y, según las necesidades del problema, el modelo puede ser de minimización o maximización. Aún con los avances en la tecnología y el desarrollo de los métodos de optimización, existen en la actualidad muchos problemas en los cuales no existen algoritmos que los resuelvan eficientemente.

Los métodos de optimización existentes en la literatura especializada pueden ser utilizados para resolver problemas de maximización y minimización y son la base de muchas aplicaciones prácticas. Por otro lado, con el ánimo de resolver problemas que carecen de algoritmos eficientes, se han desarrolado métodos que se basan en la experiencia y en el conocimiento del problema. Estos algoritmos son conocidos como “no tradicionales”, o

heurísticas, y son diseñados para problemas específicos. El estudio moderno de las heurísticas comienza desde las publicaciones del matemático George Pólya [Pol45] en la década de 1940, quien sentó las bases del pensamiento para diseñar nuevos métodos de solución a problemas. Las heurísticas logran encontrar una solución viable en un tiempo menor al que tardaría un método tradicional, aunque no dan garantía de encontrar la mejor solución. A pesar de esto, la solución que ofrecen las heurísticas puede ser muy similar a la óptima, lo que hace que las heurísticas sean un camino viable a seguir al momento de atacar diversos problemas.

El área de Inteligencia Artificial (IA) ha sido de gran ayuda en el campo de la optimización mediante el desarrollo de metaheurísticas (heurísticas más generales y que pueden abarcar más tipos de problemas) que ocupan enfoques poco ortodoxos, como es el caso de los algoritmos inspirados en la naturaleza. Estos algoritmos emulan fenómenos que se encuentran en el medio ambiente, como el comportamiento de las aves, los mecanismos de evolución de las especies, o la forma en que las moléculas se reacomodan para mantener estados de energía estable. El desarrollo de algoritmos inspirados en la naturaleza (bio-inspirados) ha ganado auge en épocas recientes, cada día estudiándose más fenómenos que puedan ser adaptados. Los algoritmos bio-inspirados se pueden agrupar en aquellos basados en la teoría evolutiva de Darwin, denominados algoritmos evolutivos, y los inspirados en el comportamiento de animales sociales (enjambres, o *swarm* en inglés), llamados algoritmos de inteligencia colectiva. En sus dos vertientes, estos algoritmos han demostrado tener un desempeño favorable en la resolución de problemas complejos, aunado a que son fáciles de implementar en código y adaptar a problemas diversos. En la literatura suele denominarse a estos algoritmos como de "caja negra" (*black box*), ya que para fines prácticos el funcionamiento interno del algoritmo no es lo importante, sino los resultados que obtiene al recibir diversos tipos de problemas como entrada.

Los algoritmos evolutivos se basan en la teoría de evolución de las especies de Darwin y la teoría genética de Mendel (selección natural y mutación genética), que en conjunto se conoce como *neodarwinismo*. Uno de los más populares es el algoritmo genético (GA, por sus siglas en inglés), pero ha sido criticado por la gran cantidad de parámetros de ajuste que utiliza. En la actualidad una variante del GA es el algoritmo de Evolución Diferencial,

propuesto en [SP95] por Storn y Price en 2005, pues ha demostrado tener un desempeño muy bueno para diferentes tipos de problemas, además de ser muy rápido en comparación a otros algoritmos evolutivos.

Por otro lado, los algoritmos de inteligencia colectiva surgen de la motivación de resolver problemas de optimización mediante la simulación de procesos inteligentes y colaborativos. Los primeros algoritmos de este tipo fueron el algoritmo de optimización por cúmulos de partículas (PSO) propuesto por Eberhart y Kennedy en [KE95] y el algoritmo de optimización por colonia de hormigas (ACO) propuesto por Dorigo en [Dor92]. Otro enfoque es el de las colonias de abejas, que en la actualidad son de los más estudiados. Existen varios algoritmos que se basan en el comportamiento de las abejas, en particular, el algoritmo de la colonia artificial de abejas (ABC) propuesto por Karaboga [Kar05] en 2005 simula la manera en que las abejas obtienen alimento.

Los problemas de optimización numérica con restricciones (CNOP, por sus siglas en inglés) son muy estudiados ya que el análisis de problemas del mundo real frecuentemente presentan restricciones en las variables propias del problema. Muchas de las metaheurísticas que se han desarrollando en años recientes han sido enfocadas en resolver problemas de este tipo, pero representan una complejidad computacional muy grande al momento de ser implementadas en código. Si bien los teoremas *no free lunch* (NFL) [WM97] indican que es imposible generar un método que pueda tener un desempeño favorable para todos los problemas de optimización, esto es, que todos los métodos que se desarrollen tendrán un desempeño equivalente en la mayoría de los casos, se ha buscado crear algoritmos que permitan manejar de manera adecuada diferentes tipos de problemas para proponer diversos tipos de modificaciones que logren tener un buen desempeño en tareas más específicas. Entre estas tareas más específicas se encuentran los CNOPs.

Muchos de los algoritmos bio-inspirados se han implementado para problemas sin restricciones y existen varios estudios donde se aplican y comparan diferentes metaheurísticas usando estos tipos de problemas. En particular, son de interés los algoritmos que simulan el comportamiento de una o varias abejas ya sea para buscar su alimento o para reproducirse. Dentro de los diferentes enfoques basados en abejas, en este trabajo se utiliza el

algoritmo ABC. Para el caso de las metaheurísticas, se han implementado varios enfoques para introducir el manejo de restricciones, en particular el algoritmo ABC modificado (M-ABC) propuesto por Cetina y Mezura en [CM12] el cual introduce mecanismos de selección novedosos y el algoritmo de Evolución Diferencial con restricciones ε y mutación basada en gradiente (εDEag) propuesto por Takahama y Sakai[TS10].

1.2. Planteamiento del problema

El algoritmo ABC original fue diseñado para trabajar con problemas de optimización numérica sin restricciones, lo que lo hace limitado al tratar de resolver problemas en espacios de búsqueda restringidos. Por otra parte, M-ABC supera esta limitante, mas existe un rezago en los resultados comparados a otros algoritmos en el estado del arte actual. De los enfoques propuestos, los algoritmos que utilizan métodos de gradiente son los que han mostrado mejores resultados en su desempeño, sin embargo estos métodos son difíciles de implementar en código y representan una complejidad computacional grande. Una alternativa para evitar el cálculo del gradiente de una función es utilizar los enfoques tradicionales que iteran sin usar derivadas, como el método de Rosenbrock [Ros60]. En particular, es conocido que el método de Direcciones Conjugadas de Powell [Pow64] es el que mejor desempeño ha presentado al resolver problemas con restricciones, por lo que bien se podría utilizar para considerar las restricciones de un problema. Es por esto que se propone usar un método de búsqueda local más simple, como es el método de Direcciones Conjugadas de Powell el cual no utiliza derivadas, para buscar lograr un desempeño similar a los algoritmos que implementan métodos de gradiente sin las desventajas que estos implican. Por lo anterior, se puede presentar el siguiente planteamiento:

¿Es factible utilizar el método de Direcciones Conjugadas de Powell como mecanismo de búsqueda local dentro del algoritmo M-ABC para manejar restricciones y obtener resultados comparables con los enfoques existentes?

1.3. Objetivos

El presente trabajo de tesis tiene los siguientes objetivos general y específicos:

1.3.1. Objetivo general

Implementar una estrategia de búsqueda local basada en el método de Direcciones Conjugadas de Powell dentro del algoritmo M-ABC que permita mejorar los resultados obtenidos al resolver problemas de optimización numérica con restricciones.

1.3.2. Objetivos específicos

- Analizar las estrategias de optimización que no utilizan derivadas.
- Implementar el algoritmo de direcciones conjugadas dentro de M-ABC.
- Hacer un análisis comparativo del comportamiento de M-ABC con la estrategia de búsqueda local contra los resultados obtenido por el algoritmo εDEag.

1.4. Justificación

Este proyecto basa su pertinencia desde el punto de vista científico en la ausencia de estudios como el propuesto en este proyecto en la literatura especializada. Por otro lado, desde el punto de vista de las aplicaciones derivadas se observa la necesidad de metaheurísticas cada vez más competitivas para resolver problemas complejos del mundo real que en la gran mayoría de los casos requieren en sus modelos el planteamiento de restricciones.

Comparar el desempeño del algoritmo desarrollado en este proyecto contra el método εDEag tiene su base en que este algoritmo ha demostrado tener un desempeño muy bueno en problemas de optimización con restricciones usando métodos de búsqueda de gradiente junto con manejo de restricciones con valor ε.

1.5. Hipótesis

Es posible acercarse al desempeño obtenido por un algoritmo del estado del arte que utiliza búsqueda basada en gradiente mediante la combinación del algoritmo M-ABC con el método de Direcciones Conjugadas de Powell al resolver problemas de optimización con restricciones.

1.6. Alcances y limitaciones

1.6.1. Alcances

El algoritmo a implementarse será probado únicamente en funciones de prueba encontradas en la literatura especializada y la resolución de problemas reales quedará para trabajo futuro. Solo se desarrollará una comparación empírica de los resultados.

Se implementará una modificación del algoritmo M-ABC utilizando algoritmos de búsqueda local basado en el método de Direcciones Conjugadas de Powell.

Se compararán los resultados de esta modificación contra los publicados por M-ABC y εDEag.

1.6.2. Limitaciones

Las limitaciones del proyecto residen en que es probable que el algoritmo de este proyecto no supere al desempeño obtenido por el algoritmo de Takahama, pero una buena aproximación a los resultados será suficiente.

Solo se desarrollará el método propuesto y los resultados de los otros métodos se obtendrán de la literatura. No se aplicará a problemas reales.

Capítulo 2

Marco Teórico

2.1. Optimización

De acuerdo a [Gor12] la optimización se define como el proceso llevado a cabo para encontrar los valores o características que optimicen algún otro valor que depende de ellos. El propósito de la optimización es encontrar o identificar la mejor solución posible, entre todas las soluciones potenciales, para un problema dado, en términos de uno o varios criterios de efectividad o desempeño.

La optimización está presente en todos los campos de la ciencia y la tecnología, la mayoría de los problemas se encuentran sujetos a un conjunto de restricciones que pueden representarse en términos lineales o no lineales, lo que complica el proceso de búsqueda generando una gran cantidad de clases de problemas. Por esta razón es que no existe solo un método de optimización sino que es posible encontrar un gran número de ellos en la literatura especializada. Desafortunadamente existen problemas donde los algoritmos que emergen de un análisis matemático resultan poco eficientes ya que utilizan demasiado tiempo y/o recursos (computacionalmente hablando) para encontrar la solución óptima del problema. El estudio de los algoritmos alternativos de optimización está enfocado a la resolución de problemas muy complejos que tomaría mucho tiempo o sería imposible realizar por medio de técnicas convencionales. Estos algoritmos se implementan para buscar,

en un conjunto de soluciones posibles para el problema en cuestión, aquella que satisfaga de mejor manera una serie de condiciones planteadas.

2.1.1. El problema de optimización

Un problema de optimización, como se muestra en [NW99], puede representarse de la siguiente manera:

$$\begin{aligned} &\min f(x) \\ &\text{sujeto a} \quad h(x) = 0 \\ &\qquad\qquad\ \ g(x) \geq 0 \end{aligned}$$

Donde:

- x es el vector de variables, también llamadas incógnitas o parámetros.
- f es la función objetivo.
- $h(x)$ es el conjunto de restricciones de igualdad.
- $g(x)$ es el conjunto de restricciones de desigualdad.

Para fines de este trabajo de tesis el enfoque de optimización con el que se desarrollan los problemas es el de minimización.

2.1.2. Optimización numérica

La optimización numérica se encarga de estudiar y proponer algoritmos para resolver problemas representados de forma matemática donde se utiliza una función objetivo de la cual se busca maximizar o minimizar el valor de ésta, dependiendo del problema. Al proceso de identificar la función objetivo, así como las variables y restricciones de un problema se conoce como “modelado del problema”. La construcción de un modelo apropiado es el primer paso en el proceso de optimización. Si el modelo es muy simple no proveerá

una visión muy útil del problema, pero si es muy complejo puede llegar a ser muy difícil de resolver [NW99]. En la fig. 2.1 se muestra la representación gráfica de un problema de optimización numérica; es importante indicar que el ámbito de los posibles valores de las variables en los problemas de optimización numérica es continua, a diferencia de los problemas de optimización combinatoria.

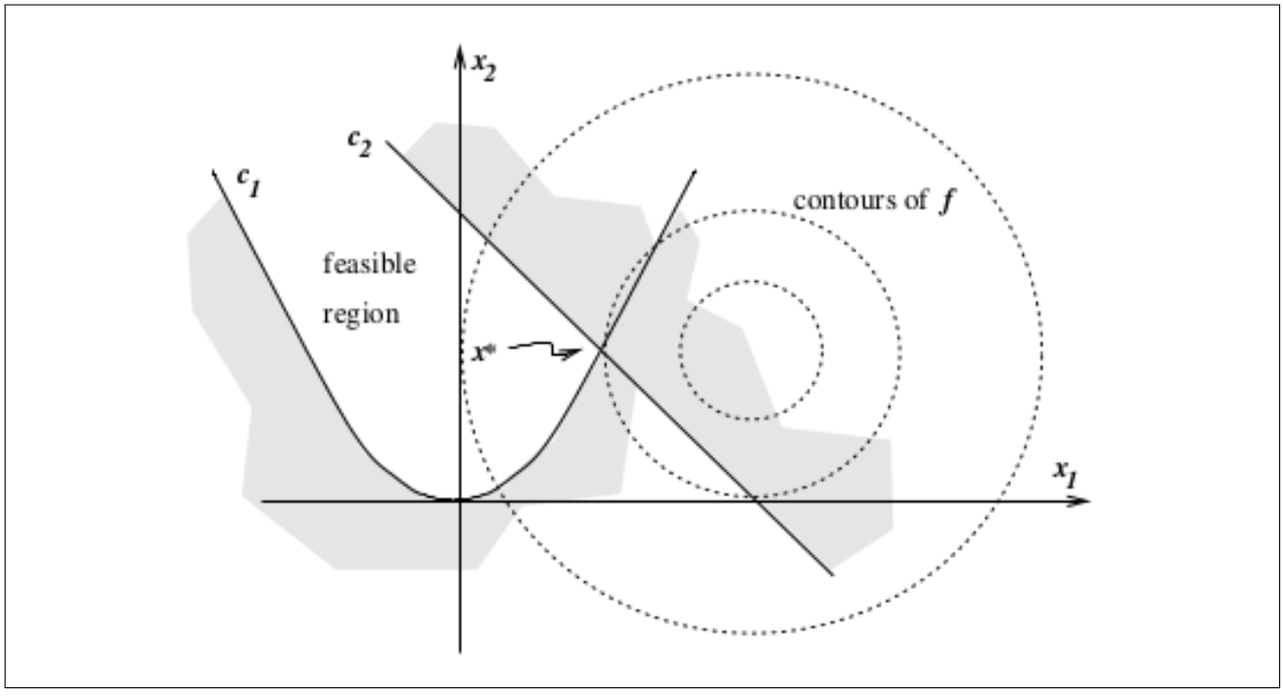

Figura 2.1: Representación gráfica de un problema de optimización con restricciones. La curva $c1$ y $c2$ representan las restricciones del problema y las líneas punteadas indican diferentes valores obtenidos para la función objetivo f, donde x^* representa el valor óptimo.

Optimización numérica sin restricciones

Los problemas de optimización sin restricciones pueden encontrarse en muchas aplicaciones prácticas. Si hay restricciones naturales en las variables, por lo general éstas son descartadas y se supone que no tienen efecto en la solución óptima. Este tipo de problemas también pueden resultar de la reformulación de problemas con restricciones, las cuales se reemplazan por términos de penalización en la función objetivo.

Optimización numérica con restricciones

Los problemas de optimización con restricciones surgen de modelos que incluyen restricciones explícitas en las variables. Estas restricciones pueden ser límites simples como $0 \leq x_1 \leq 100$, restricciones lineales más generales como $\sum_i x_i \leq 1$, o desigualdades no lineales que representen relaciones complejas entre las variables [NW99]. No existe una manera única de formular restricciones en todos los problemas, por lo que el número de restricciones a utilizar depende del usuario [Deb04].

Hay dos tipos de de restricciones que emergen de la mayoría de los modelos: igualdad y desigualdad. Las restricciones de desigualdad indican que las relaciones funcionales entre las variables de diseño son mayores, menores o iguales a un valor dado. La mayoría de las restricciones encontradas en problemas de ingeniería son de este tipo. Las restricciones de igualdad indican que las relaciones funcionales deberán coincidir exactamente con un valor dado. Estas restricciones, por lo general, son más difíciles de manejar y, por lo tanto, deben evitarse siempre que sea posible. Si las relaciones de igualdad son más simples, puede ser posible reducir el número de variables de diseño usando restricciones de igualdad. En tal caso, las restricciones de igualdad reducen la complejidad del problema.

2.1.3. Optimización combinatoria

La optimización combinatoria se encarga de problemas donde se conoce el espacio de soluciones (generalmente grande). Lawler [Law76] define la optimización combinatoria como el "estudio matemático del ordenamiento, agrupamiento y selección de objetos discretos, por lo general finitos en número". El objetivo es explorar el espacio de búsqueda y reducirlo para obtener un conjunto mejor de soluciones. Entre los problemas más conocidos de este tipo están los problemas de calendarización, el problema del agente viajero (TSP, por sus siglas en inglés), el problema de las N reinas, etc. En la fig. 2.2 se observa un ejemplo del problema TSP.

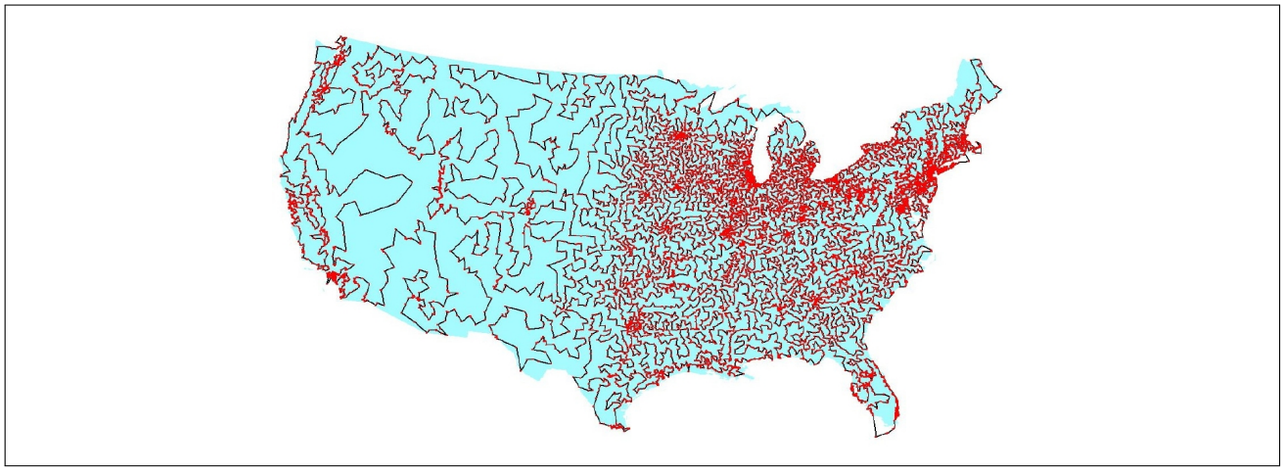

Figura 2.2: Ejemplo de TSP utilizando las 13,509 ciudades de Estados Unidos con población de 500 habitantes o más.

2.1.4. Convexidad

La convexidad es una propiedad fundamental en la teoría de optimización ya que muchos métodos de búsqueda de una solución óptima basan su desempeño en las propiedades convexas del problema. El término convexo se puede aplicar tanto a conjuntos como a funciones.

Un conjunto N es convexo si cualquier punto $x \in N$ puede ser representado como una combinación lineal de otros dos puntos cualesquiera que también pertenezcan a N. Geométricamente significa que un conjunto es convexo si la línea que conecta dos puntos dentro de N se encuentra dentro de dicha región fig. 2.3. Por otro lado, una función $f(x)$ definida en N es convexa si, para cualquier par de puntos x y y en N, se cumple:

$$F(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y)$$

tal que

$$0 \leq \lambda \leq 1$$

El significado geométrico de esta relación indica que la gráfica de la función está por debajo de la línea recta que une x con y.

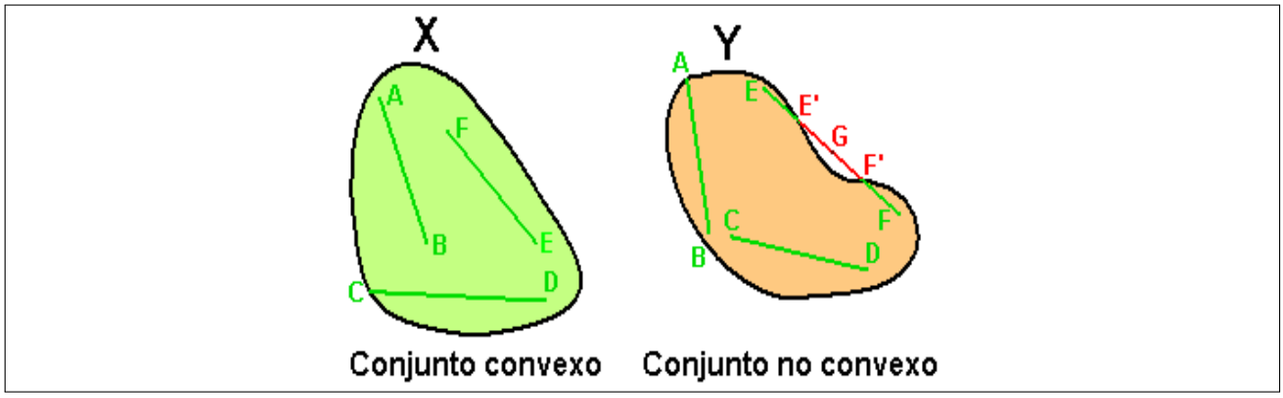

Figura 2.3: Ejemplos de convexidad

2.1.5. Tipos de soluciones

En general las soluciones de un problema se pueden agrupar de dos formas: óptimo local y óptimo global. En la fig. 2.4 se ejemplifican ambos tipos de soluciones.

- **Óptimo global:** es un punto x^* donde se cumple que $f(x^*) \leq f(x)$ para todas las x.
- **Óptimo local:** es un punto x^* donde se cumple que $f(x^*) \leq f(x)$ para toda x en la vecindad N del problema. [NW99].

En la figura fig. 2.4 se ejemplifican ambos tipos de soluciones.

2.1.6. Condiciones de optimalidad

Como resultado del estudio de los problemas de optimización se conocen condiciones que indican si una solución es óptima. Como se indicó anteriormente, se establece que los problemas de optimización pueden ser **sin restricciones** o **con restricciones** [RRR06]. Debido a esta clasificación, se pueden definir las condiciones en las que una solución es óptima, de acuerdo a la presencia o no de restricciones.

- **Problemas sin restricciones:** Si $f(x)$ es continua y doblemente diferenciable, su gradiente y su hessiano permiten determinar si el punto es un mínimo local. Estas condiciones utilizan el concepto de **dirección factible**. Una dirección factible es el vector

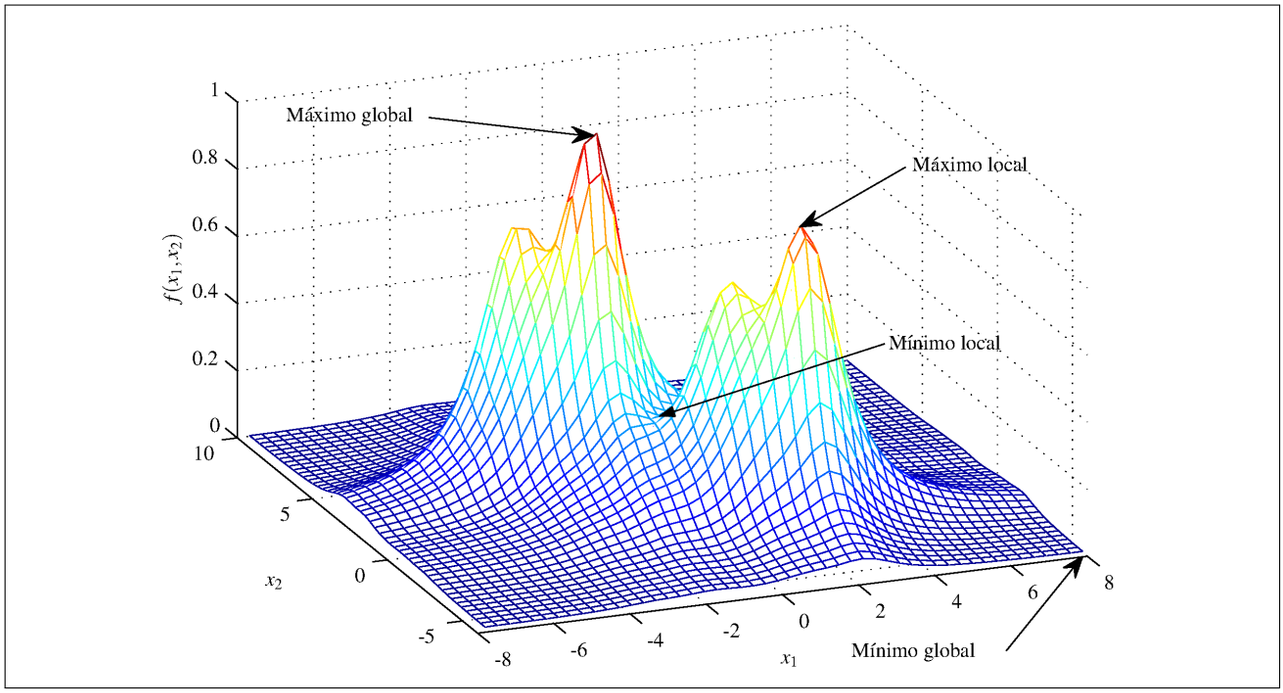

Figura 2.4: Tipos de soluciones óptimas

unitario d que, asociado a un punto $x \in N$ produce otro punto $x + \alpha d \in N$ para un $\alpha > 0$ [1]. Una dirección de descenso factible es aquella que cumple:

$$f(x + d) < f(x)$$

- **Problemas con restricciones:** En la mayoría de los problemas de la vida real que se pretenden solucionar mediante métodos de optimización, existen restricciones para las diferentes variables que se utilizan. La presencia de estas restricciones reduce la región de búsqueda. Si bien esto puede sonar a que se simplifica la tarea de encontrar una solución óptima, la realidad es que se vuelve más complicada. Existen dos tipos de restricciones:

  - **Restricciones de igualdad:** Estas restricciones son las que se igualan a un valor específico, que por lo general es 0. Formalmente se definen de esta manera:

$$h(x_1, \ldots, x_N) = 0$$

[1] La dirección factible de x a x^* se define como: $d = \frac{x - x^*}{||x - x^*||}$

- **Restricciones de desigualdad:** Estas restricciones imponen una condición de desigualdad a un valor determinado, el cual también suele ser 0. Formalmente se definen de la siguiente manera:

$$g(x_1, ..., x_N) \leq 0$$

Condiciones de optimalidad para problemas sin restricciones

Para problemas sin restricciones existen las siguientes condiciones de optimalidad:

1. Condición necesaria de primer orden:

$$\nabla f(x^*) = 0$$

2. Condición necesaria de segundo orden:

$$d^T \nabla^2 f(x) \geq 0$$

Condiciones de optimalidad para problemas con restricciones

Estas condiciones se conocen como condiciones de Karush-Kuhn-Tucker. Kuhn y Tucker extendieron el uso de multiplicadores de Lagrange para desarrollar criterios de optimalidad en problemas con restricciones de igualdad, para que pudiera manejar también restricciones de desigualdad [BSS06].

1. Condición necesaria de primer orden:

$$\nabla_x L(x^*, \lambda^*, \mu^*) = \nabla f(x^*) - \lambda^{*^T} \nabla h(x^*) - \mu^T \nabla g(x^*) = 0$$

2. Condición necesaria de segundo orden:

$$d^T \nabla_x^2 L(x^*, \lambda^*, \mu^*) d \geq 0$$

donde: $\nabla_x L$ es el lagrangiano de la función y x y μ son multiplicadores de Lagrange.

2.2. Técnicas para resolver problemas de optimización

El área encargada de abordar los problemas de optimización es conocida como Investigación de Operaciones (IO). Los métodos dentro de esta área son planteados a manera de algoritmos, los cuales han sido implementados como programas de computadora debido a la complejidad y número de iteraciones que requieren los mismos.

Por lo general se comienza con una solución que es el estimado inicial, para después generar una secuencia de estimaciones mejoradas hasta llegar a una solución. La estrategia para moverse de una iteración a la siguiente es lo que distingue a un algoritmo de otro. La mayoría de las estrategias utilizan los valores de la función objetivo, las restricciones y, en algunos casos, los valores de la primera y segunda derivadas de estas funciones. Algunos algoritmos recogen información de iteraciones previas, mientras otros usan solamente información local del punto actual. De acuerdo a [NW99], independientemente de estas especificaciones, los algoritmos deben tener las siguientes características:

- **Solidez:** Deberán tener un buen desempeño en una amplia variedad de problemas de su clase, para todas las opciones razonables de las variables iniciales.
- **Eficiencia:** No deben requerir mucho tiempo computacional ni almacenamiento.
- **Exactitud:** Deberán ser capaces de identificar una solución con precisión, sin ser demasiado sensibles a errores en los datos o errores aritméticos cuando el algoritmo se implementa en una computadora.

Estos objetivos pueden tener conflictos entre sí. Por ejemplo, un método que converge de manera rápida para problemas no lineales puede requerir mucho espacio de almacenamiento en problemas con un espacio de búsqueda muy amplio. Por otro lado, un método sólido puede presentar un desempeño lento debido a su complejidad.

2.2.1. Clasificación de los métodos de optimización

De acuerdo a sus características, como se menciona en [Deb04], los algoritmos de optimización se clasifican de la siguiente manera:

- **Métodos tradicionales:** Se les conoce también con el nombre de "métodos numéricos", cuya implementación se basa en un modelo matemático:
  - Métodos para optimización sin restricciones:
    - Métodos de una sola variable: Métodos de delimitación, métodos de eliminación de regiones, métodos de estimación de punto, métodos basados en gradiente.
    - Métodos multivariable: Método de búsqueda Directa, métodos basados en gradiente.
  - Métodos para optimización con restricciones: métodos de transformación, búsqueda directa para problemas con restricciones, técnicas de búsqueda lineales, métodos basados en gradiente.
  - Métodos especializados: programación entera, programación geométrica.
- **Métodos no tradicionales:** También llamados "heurísticas", son técnicas las cuales no garantizan encontrar la mejor solución a un problema dado, pero obtienen buenas aproximaciones a problemas complejos en tiempos razonables. Se pueden clasificar de la siguiente manera, de acuerdo a [Tal09]:
  - Métodos basados en solución una sola solución: recocido simulado, búsqueda tabú, entre otros.
  - Métodos basados en población: algoritmos evolutivos, algoritmos de inteligencia colectiva.

2.2.2. Métodos tradicionales

Los métodos tradicionales son algoritmos derivados de análisis matemáticos puros, por lo que su efectividad está probada. Sin embargo, estos métodos suelen ser poco eficientes a la hora de trabajar con problemas reales complejos.

Métodos de una sola variable

Los métodos de variable simple están diseñados para trabajar con funciones de una sola variable. Estos métodos son utilizados como parte de los procedimientos de diversos métodos multivariable. Los métodos de variable simple están divididos en cuatro tipos:

- **Métodos de delimitación:** En estos métodos el mínimo de una función se encuentra en dos fases: primero, se encuentran los límites inferior y superior del valor mínimo mediante una técnica de fuerza bruta. Después se usa un método más sofisticado para buscar dentro de estos límites y encontrar la solución óptima con la precisión deseada. Dentro de este tipo de métodos se encuentran el método de búsqueda exhaustiva y el método Bounding Phase [Deb04].
- **Métodos de eliminación de regiones:** Estos métodos son utilizados en conjunto con los métodos de delimitación. Una vez que se ha encontrado el rango donde se encuentra el punto mínimo, se necesita un método más sofisticado para mejorar la exactitud de la solución. Dentro de este tipo de algoritmos se encuentran el método de división de intervalos, el método Fibonacci y el método de búsqueda por sección áurea. La regla fundamental para los métodos de eliminación de regiones [Deb04] es: sean dos puntos x_1 y x_2, los cuales se encuentran en el intervalo (a, b) y satisfacen $x_1 < x_2$. Para minimización de funciones unimodales [2], se puede concluir lo siguiente:
  - Si $f(x_1) > f(x_2)$, entonces el mínimo no se encuentra en (a, x_1).
  - Si $f(x_1) < f(x_2)$, entonces el mínimo no se encuentra en (x_2, b).

[2] Función que solo tiene un valor óptimo.

- Si $f(x_1) = f(x_2)$, entonces el mínimo no se encuentra en (a, x_1) ni (x_2, b).

- **Métodos de estimación de punto:** Este tipo de métodos involucra el uso de muestras pequeñas de una población para poder estimar un resultado a gran escala. Los métodos más representativos de esta clase son el método de máxima verosimilitud y el método de momentos.

- **Métodos basados en gradiente:** Estos métodos, aunque más utilizados en problemas de muchas variables, también pueden utilizarse en casos de variable única. Trabajan cambiando de dirección dependiendo del aumento o disminución del gradiente en la función.

Métodos multivariable

Estos métodos están diseñados para trabajar con más de una variable (o dimensiones), por lo que son adecuados para atacar la mayoría de los problemas presentados en la vida real. Se agrupan en métodos de búsqueda directa y basados en gradiente.

Métodos de búsqueda directa: Estos métodos realizan la búsqueda de soluciones de manera lineal, esto es, sin utilizar cálculo de derivadas, lo que los hace fáciles de implementar pero la calidad de los resultados puede ser inferior a los métodos que utilizan dicho tipo de cálculos. Dentro de este tipo de algoritmos se encuentran el método Nelder-Mead [NM65] (fig. 2.5) y el método Rosenbrock. El método de direcciones conjugadas de Powell es uno de los más utilizados ya que ha demostrado tener un desempeño favorable en la resolución de problemas complejos.

Métodos basados en gradiente: Este tipo de métodos utilizan cálculo de derivadas, por lo que implementarlos en código se vuelve una tarea compleja, a la par de que la complejidad computacional es muy grande. Sin embargo, este tipo de métodos ha demostrado tener buena eficiencia, logrando obtener resultados de buena calidad en menos iteraciones que los métodos de búsqueda directa. Dentro de esta clasificación se encuentran el método de gradiente conjugado de Powell (fig. 2.6), el método de descenso máximo, y el método de Gauss-Newton, entre otros.

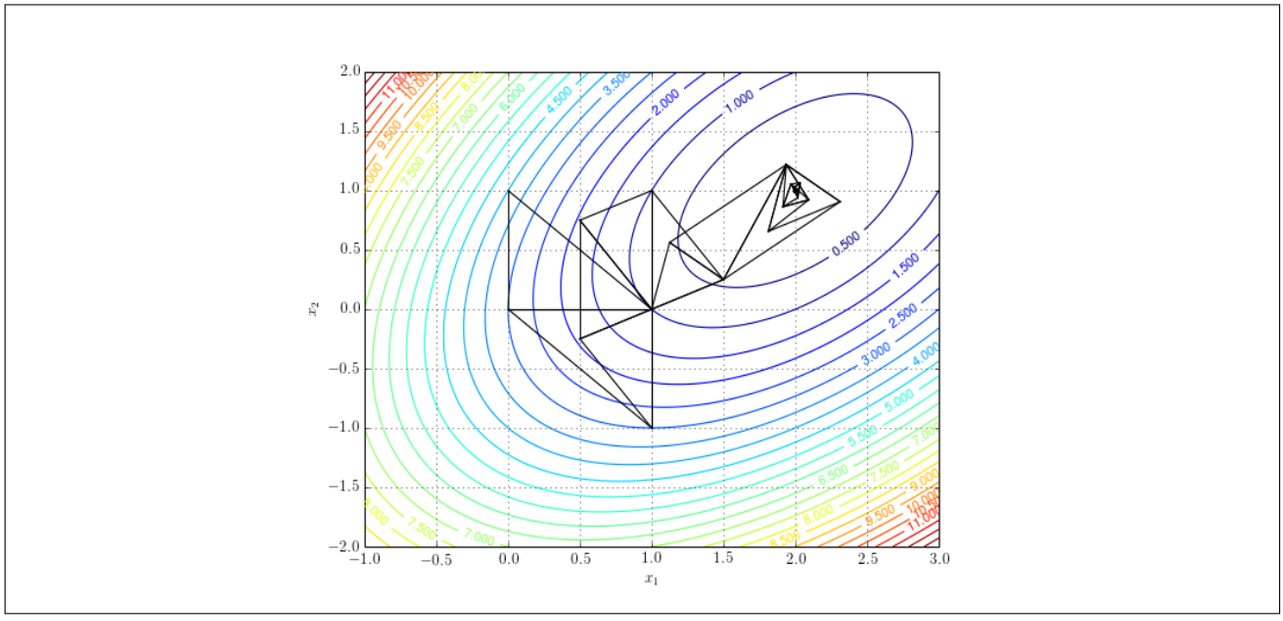

Figura 2.5: Método de Nelder-Mead

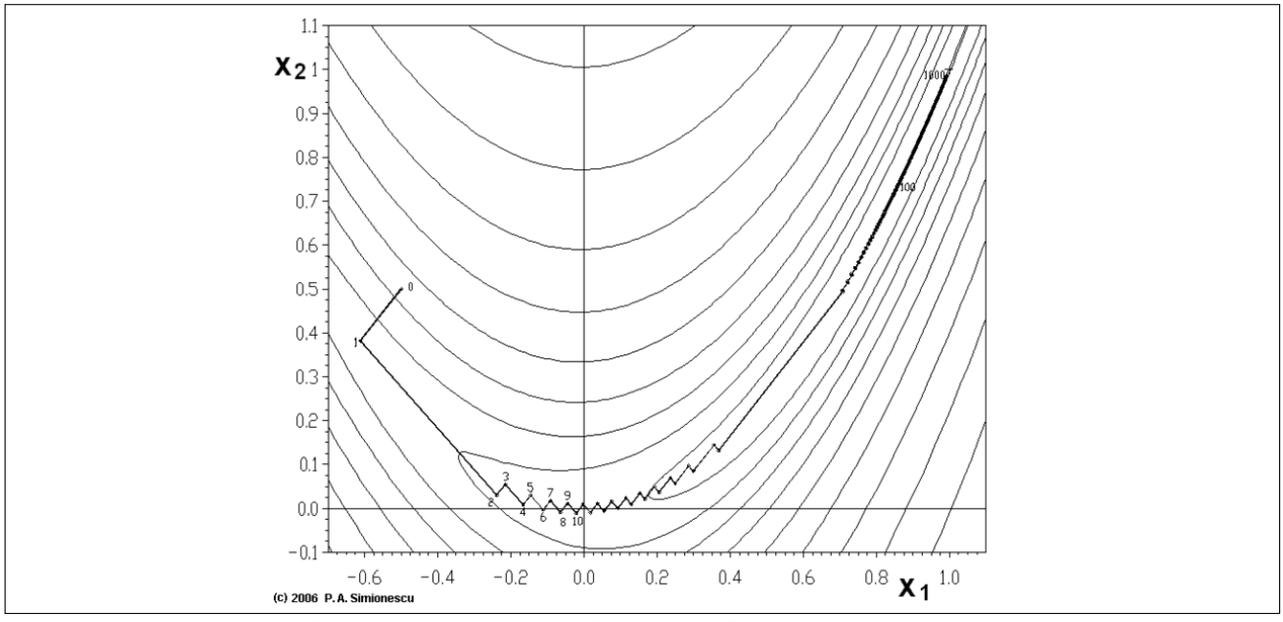

Figura 2.6: Método de Gradiente Conjugado de Powell

Métodos para optimización con restricciones

Al trabajar con problemas que presentan restricciones, los métodos que no contemplan estas reglas resultan insuficientes. Es por esto que se han diseñado diversos algoritmos y técnicas capaces de trabajar con dichas condiciones. La presencia de restricciones limita el espacio de búsqueda pero, al mismo tiempo, dificulta el encontrar la solución óptima ya que se pierden algunos de los criterios de optimalidad (como que el gradiente es nulo en el óptimo). Dentro de los métodos tradicionales para problemas con restricciones se encuentran el método de Multiplicadores de Lagrange y el método de penalización de restricciones.

2.2.3. Métodos no tradicionales (heurísticas)

Se les llama heurísticas a los métodos de solución de problemas que se usan como alternativa a los métodos numéricos tradicionales. Con frecuencia las heurísticas están basadas en métodos tradicionales, aunque la implementación de estos no es ortodoxa. Las heurísticas no garantizan encontrar la solución factible a un problema determinado; a pesar de esto, sí son capaces de hallar una solución muy cercana a la factible en un tiempo mucho menor al que un método tradicional tardaría. Por esta razón el campo de estudio de las heurísticas ha ganado popularidad y existen diversas investigaciones que han arrojado resultados interesantes. Una heurística se basa en las características del problema a resolver. Por otro lado, existen heurísticas generales, denominadas metaheurísticas, que por sus características se pueden aplicar a diversos tipos de problemas. Martí en [Mar03] cataloga a las heurísticas de la siguiente manera:

- **Métodos de descomposición:** El problema original se descompone en subproblemas más sencillos de resolver, teniendo en cuenta, aunque sea de manera general, que ambos pertenecen al mismo problema.
- **Métodos inductivos:** La idea de estos métodos es generalizar de versiones pequeñas o más sencillas al caso completo. Propiedades o técnicas identificadas en estos casos más fáciles de analizar pueden ser aplicadas al problema completo.

- **Métodos de reducción:** Consiste en identificar propiedades que se cumplen mayoritariamente por las buenas soluciones e introducirlas como restricciones del problema. El objeto es restringir el espacio de soluciones simplificando el problema. El riesgo obvio es dejar fuera las soluciones óptimas del problema original.

- **Métodos constructivos:** Consisten en construir literalmente paso a paso una solución del problema. Usualmente son métodos deterministas y suelen estar basados en la mejor elección en cada iteración. Estos métodos han sido muy utilizados en problemas clásicos como el TSP.

- **Métodos de búsqueda local:** A diferencia de los métodos anteriores, los procedimientos de búsqueda o mejora local comienzan con una solución del problema y la mejoran progresivamente. El procedimiento realiza en cada paso un movimiento de una solución a otra con mejor valor. El método finaliza cuando, para una solución, no existe ninguna solución accesible que la mejore.

2.3. Metaheurísticas

Si las heurísticas son métodos para resolver un problema determinado, las metaheurísticas sirven para resolver problemas de manera más general. Las metaheurísticas están diseñadas para atacar problemas complejos de optimización donde las heurísticas y métodos clásicos para optimización han fracasado en ser efectivas y eficientes. Osman y Laporte [OL96] las definen como un proceso de generación iterativo que guía a una heurística subordinada combinando inteligentemente diferentes conceptos para explorar y explotar el espacio de búsqueda, utilizándose estrategias de aprendizaje para estructurar la información en pos de encontrar soluciones cercanas al valor óptimo.

2.3.1. Criterios utilizados en metaheurísticas

Según lo señalado en [Tal09], en el diseño de una metaheurística deben tomarse en cuenta dos criterios contradictorios entre sí: la exploración del espacio de búsqueda (diversi-

ficación) y la explotación de las mejores soluciones halladas (intensificación). Las regiones prometedoras en el espacio de búsqueda se determinan por el número de soluciones buenas obtenidas. En intensificación, las regiones prometedoras se exploran con mayor exhaustividad con la esperanza de encontrar mejores soluciones. En diversificación, las regiones no exploradas deben ser visitadas para asegurarse de que todas las regiones del espacio de búsqueda han sido recorridas de igual manera y que la búsqueda no está confinada a un número reducido de regiones.

2.3.2. Principales conceptos usados en metaheurísticas

Existen dos aspectos de diseño relacionados a todas las metaheurísticas: la representación de las soluciones manejadas por los algoritmos y la definición de la función objetivo que guiará la búsqueda.

Representación de soluciones

Este es un aspecto fundamental en el desarrollo de metaheurísticas. La codificación de las soluciones tiene un rol importante en la eficiencia y efectividad de cualquier metaheurística. Ésta debe ser adecuada y relevante al problema de optimización a atacar. Además, la eficiencia de una representación también se relaciona con los operadores de búsqueda aplicados en esta representación (vecindad, recombinación, etc). Una representación debe tener las siguientes características:

- **Integridad:** Todas las soluciones asociadas con el problema deben ser representadas.
- **Conectividad:** Debe existir un camino de búsqueda entre cualquier par de soluciones. Cualquier solución, especialmente el óptimo global, debe ser alcanzada.
- **Eficiencia:** La representación debe ser fácil de manipular para los operadores de búsqueda.

Función objetivo

La función objetivo asocia un valor real a cada solución en el espacio de búsqueda que describa la calidad o la aptitud de la solución. Además, guía la búsqueda hacia soluciones viables.

2.3.3. Tipos de metaheurísticas

Talbi en [Tal09] señala que las metaheurísticas se pueden agrupar en metaheurísticas basadas en solución única o en metaheurísticas basadas en población. Las metaheurísticas basadas en poblaciones utilizan un conjunto de soluciones para construir nuevas soluciones, a diferencia de las de solución única que solo presentan una solución.

Las metaheurísticas basadas en población se pueden agrupar en Cómputo Evolutivo y en algoritmos de Inteligencia Colectiva. Los algoritmos de Cómputo Evolutivo utilizan mecanismos de selección, cruzamiento y mutación en la búsqueda del óptimo y los algoritmos de Inteligencia Colectiva utilizan un componente de inteligencia social para mejorar sus soluciones.

Métaheurísticas basadas en solución única

Siguiendo lo presentado por Talbi[Tal09], estas metaheurísticas (también conocidas como S metaheurísticas) pueden verse como "caminatas" a través de una vecindad o trayectorias sobre el espacio de búsqueda del problema. Estas trayectorias se generan mediante procedimientos iterativos que se mueven de una solución a otra en el espacio de búsqueda. Dentro de este grupo se encuentran, entre otros, la búsqueda local, el método de recocido simulado y la búsqueda tabú.

Búsqueda local (LS): El método de búsqueda local es tal vez el método más antiguo y simple de metaheurística. Iniciando en una solución inicial, este método reemplaza la solución actual por alguna solución vecina que mejore la función objetivo en cada iteración. La búsqueda se detiene cuando todos los vecinos candidatos son peores que la solución actual, significando que se alcanzó un óptimo local. Pueden aplicarse diversas estrategias

en la selección del mejor vecino:

- **Descenso máximo:** En esta estrategia se selecciona siempre el mejor vecino. Se evalúa el vecindario de manera determinista, por lo que la exploración del vecindario es exhaustiva y todos los movimientos posibles se prueban para poder seleccionar el mejor vecino. Este tipo de exploración puede consumir mucho tiempo en vecindarios grandes.
- **Primer mejora:** Consiste en elegir el primer vecino que mejore la solución actual. Esto involucra una evaluación parcial del vecindario. En el peor de los casos se realiza una evaluación completa del vecindario.
- **Selección aleatoria:** Se selecciona un vecino al azar como mejora de la solución actual.

Recocido simulado (SA): El algoritmo de recocido simulado (SA, por sus siglas en inglés) fue propuesto por Kirkpatrick *et al.* en 1983 [KGV83] y es uno de los métodos más representativos de las metaheurísticas, ya que resultó ser muy eficiente en la resolución de problemas de optimización combinatoria, por lo que también se ha utilizado en problemas continuos [Tal09].

El algoritmo está basado en los principios de mecánica estadística donde el proceso de recocido requiere un calentamiento y sucesivamente un enfriamiento lento de una sustancia para eventualmente obtener una estructura cristalina fuerte.

Búsqueda tabú (TS): Fue propuesto por Glover y Laguna en [GL99]. Se comporta como un algoritmo de búsqueda local de descenso máximo, pero acepta soluciones que no mejoran la solución actual para escapar de los óptimos locales cuando todos los vecinos son soluciones que no mejoran a la actual. Por lo general, se explora todo el vecindario de manera determinista. Al igual que búsqueda local, cuando se encuentra un mejor vecino, se reemplaza la solución actual. Cuando se llega a un óptimo local, la búsqueda continúa seleccionando un candidato peor que la solución actual. La mejor solución del vecindario se selecciona como la mejor solución actual, incluso si no la mejora.

Para evitar ciclos, el algoritmo descarta vecinos que ya han sido visitados previamente. Memoriza la trayectoria de búsqueda reciente, aplicando la llamada "lista tabú", donde guarda dichas trayectorias y soluciones ya visitadas.

Métaheurísticas basadas en cómputo evolutivo

Los algoritmos evolutivos están basados en la teoría moderna de la evolución de las especies, tambien llamada "neo-darwinismo" ya que combina la teoría de selección natural de Darwin con la teoría genética de Mendel.

Algortimo genético (GA): El algoritmo genético, desarrollado por Holland en la década de 1960, busca sobre un conjunto de "cromosomas" que representan soluciones candidatas a un problema (en algunos casos la solución consiste de un conjunto de cromosomas). Este algoritmo utiliza los conceptos de población, cruzamiento, mutación y selección natural [Mit96], los cuales funcionan como operadores dentro del algoritmo:

- **Población**: Es el conjunto de individuos de una especie.
- **Cruzamiento**: Es el resultado de combinar dos conjuntos diferentes de cromosomas.
- **Mutación**: Este operador cambia de manera aleatoria uno o más elementos de la cadena de cromosomas. Por lo general se utiliza determinando una probabilidad muy baja.
- **Selección:** Se refiere a la elección de los cromosomas a ser utilizados para el cruzamiento.

Algoritmos meméticos (MA): Los algoritmos meméticos aplican un proceso de búsqueda local para mejorar a los individuos (mejorar su aptitud) [SEW07]. Estos métodos están inspirados por modelos de adaptación en sistemas naturales que combinan la adaptación evolutiva de poblaciones de individuos con el aprendizaje individual dentro de una generación. Además, estos algoritmos se basan en el concepto de "meme" presentado por Dawkins en [Daw76] que representa una unidad de evolución cultural que puede representar una mejora local.

Desde el punto de vista de la optimización, los algoritmos meméticos son algoritmos evolutivos híbridos que combinan búsqueda global y local usando un algoritmo evolutivo para realizar la exploración del espacio de búsqueda y un mecanismo de búsqueda local para explotar los resultados hallados.

Evolución diferencial (DE): El algoritmo de evolución diferencial (DE, por sus siglas en inglés) es uno de los más populares hoy en día ya que es fácil de implementar, además de que ha demostrado tener un desempeño satisfactorio al ser aplicado a diversos problemas. Fue propuesto por Storn y Price en [SP95] y hoy en día cuenta con diversas variantes que han extendido su funcionalidad.

Este algoritmo mantiene una población de soluciones candidatas que se recombinan con la diferencia de otras soluciones seleccionadas al azar para generar nuevos individuos. Las soluciones candidatas iniciales son generadas al azar y en cada iteración del algoritmo se evalúan las soluciones obtenidas y se reemplazan las que hayan sido superadas.

Algoritmos de inteligencia colectiva

Se define inteligencia colectiva como la conducta que emerge de la interacción de diversas especies animales sociales trabajando bajo muy pocas reglas. Estas especies forman grupos en los cuáles no existe un líder (contrario a otras especies donde el macho o hembra alfa son los que dirigen la manada), además de que los miembros del grupo no están conscientes de estar llevando a cabo un proceso más complejo. La interacción entre los miembros del grupo es lo que hace posible que se lleve a cabo el comportamiento colectivo que es el objeto de estudio. Lalbakhsh y Fesharaki presentan en [LF08] las siguientes características como ventajas de los sistemas de inteligencia colectiva:

- **Flexibilidad:** El grupo se puede adaptar a un entorno cambiante.
- **Robustez:** Aun cuando algún individuo falla, el grupo puede llevar a cabo su tarea.
- **Auto-organización:** Las actividades no son controladas central ni localmente. Ya que el comportamiento del grupo emerge de las interacciones entre los individuos, el grupo trabaja bien en situaciones complejas e impredecibles.

Al grupo de individuos (agentes) se le denomina colonia, que se comunican entre ellos (directa o indirectamiente) actuando en su medio ambiente local. Para modelar el comportamiento de una colonia se necesitan de los siguientes principios ([Hu12]):

- **Principio de proximidad:** Las unidades básicas de la colonia deben ser capaces de hacer cálculos simples relacionados a su entorno. En este ámbito un cálculo se refiere a una respuesta conductual a una variación en el medio y a las interacciones entre individuos.
- **Principio de calidad:** Una colonia debe ser capaz de responder a factores de calidad tales como alimento y seguridad.
- **Principio de respuesta diversa:** Los recursos no deben estar concentrados en una región angosta. La distribución debe estar diseñada de tal manera que cada agente pueda estar protegido ante cambios en el entorno.
- **Principio de estabilidad y adaptabilidad:** Se espera que las colonias se adapten a variaciones en el entorno sin cambiar de modos (estados) rápidamente, ya que esto representa un gasto de energía.

El objetivo de los modelos computacionales de inteligencia colectiva es representar las conductas básicas de los individuos y la interacción local con el medio ambiente e individuos vecinos, de manera que se pueden obtener conductas más complejas que se puedan utilizar para resolver problemas de optimización. El proceso principal en el cuál están basados estos algoritmos es en la búsqueda de alimento por estas especies.

Optimización por colonia de hormigas (ACO): En el algoritmo de optimización por colonia de hormigas (ACO), propuesto por Dorigo en [Dor92], se emula la manera en la que las hormigas recogen comida: al caminar de la fuente de comida al nido y viceversa, depositan en el suelo una sustancia llamada feromona, trazando un camino con esta sustancia. Las hormigas huelen el rastro de feromonas, y al escoger su camino, suelen elegir aquél que tenga el rastro más fuerte. El rastro de feromonas permite a las hormigas encontrar el camino de vuelta a la fuente de comida. También puede ser usado por otras hormigas para

encontrar la ubicación de otras fuentes de comida encontradas por otras hormigas del nido [DDC99]. Este algoritmo ha sido utilizado para problemas de optimización combinatoria, especialmente con el TSP.

Optimización por cúmulo de partículas (PSO): En el algoritmo de Optimización por Colonia de Partículas (PSO), propuesto por Eberhart y Kennedy en [KE95], se recrea la conducta presentada por parvadas de aves, donde el miembro que divisa una fuente de alimento se vuelve el que dirige al resto del grupo. Esta conducta está influenciada por la habilidad de los individuos para oler la comida y por la memoria producto de la experiencia de cada uno. En cada iteración, cada individuo define su posición y velocidad utilizando dos elementos: la posición actual de la partícula, la mejor posición conocida por la partícula y la mejor posición conocida por el colectivo.

Colmenas de abejas: Estos algoritmos basados en colmenas o colonias de abejas son de los más utilizados dentro de los algoritmos de inteligencia colectiva debido a la facilidad de su implementación y a los buenos resultados que reportan. El enfoque más popular es el basado en la recolección de alimento (forrajeo), aunque también existen otros como el de abeja reina. En el enfoque de forrajeo se utilizan diferentes tipos de abejas (empleadas, exploradoras, observadoras) para encontrar y evaluar fuentes de alimento (soluciones). Dentro de este tipo de algoritmos se encuentra el descrito por Karaboga en [Kar05], el cual es base del algoritmo presentado en este trabajo de tesis.

2.4. Manejo de restricciones en metaheurísticas

Tratar con restricciones es otro punto importante para el diseño eficiente de una metaheurística. Las restricciones pueden ser de cualquier tipo: lineales o no lineales, igualdad o desigualdad. Por lo tanto existen estrategias para el manejo de éstas, que se pueden clasificar en estrategias de rechazo, penalización, reparación, decodificación y preservación. Gracias a estas estrategias, solo se mantienen las soluciones factibles durante la búsqueda.

2.4.1. Clasificación de Coello

De acuerdo a Coello [Coe02], un aspecto usualmente pasado por alto al utilizar algoritmos evolutivos es que estos son por lo general pensados para procedimientos donde no existen restricciones, por lo que es necesario idear maneras de incorporar restricciones a la función objetivo. Se puede clasificar a los métodos de manejo de restricciones de la siguiente manera:

- **Uso de funciones de penalización:** La manera más común de manejo de restricciones en algoritmos evolutivos es el uso de penalizaciones. La estrategia básica es definir la aptitud de un individuo i extendiendo el dominio de la función objetivo $f(x)$ usando

  $$aptitud_i(x) = f_i(x) \pm Q_i$$

  donde Q_i representa ya sea una penalización por un individuo no factible i o el costo de reparar dicho individuo (hacerlo factible).

- **Mantener una población factible mediante representaciones especiales y operadores genéricos:** Los decodificadores fueron uno de los métodos de manejo de restricciones más competitivos en los inicios de esta área. Se basaron en la idea de mapear la región factible del espacio de búsqueda en un espacio más sencillo de realizar un muestreo donde un algoritmo pueda tener un mejor desempeño.

  Se han desarrollado esquemas de representación especiales para atacar problemas particularmente difíciles donde la representación binaria convencional de los algoritmos genéticos no sea adecuada. El objetivo de este cambio de representación es simplificar la forma del espacio de búsqueda mientras los operadores especiales se utilizan para mantener la factibilidad de una cierta solución en todo momento.

- **Algoritmos de reparación:** En muchos problemas de optimización combinatoria es relativamente fácil "reparar" un individuo no factible (volverlo factible). Dicha versión reparada puede ser utilizada ya sea solo para fines de evaluación o puede reemplazar al individuo original en la población.

- **Separación de objetivos y restricciones:** Contrarias a la idea de combinar la función objetivo y los valores de las restricciones en un valor único como se hace en las funciones de penalización, existen técnicas de manejo de restricciones que trabajan de manera que se mantengan estos valores independientes uno del otro.

- **Métodos híbridos:** En esta categoría se agrupan a los algoritmos evolutivos que se combinan con otros métodos para manejar restriccciones, como con el manejo de multiplicadores de Lagrange y también con lógica difusa, así como aquellos que se combinan con otras metaheurísrticas como colonias de hormigas y sistemas inmunes.

2.4.2. Clasificación actualizada por Mezura y Coello

En [MC11] Mezura presenta una clasificación actualizada de métodos de manejo de restricciones para algoritmos bioinspirados que incluye técnicas que han tenido un impacto significativo en el estudio y desarrollo de dichos algoritmos.

- **Reglas de factibilidad:** Un ejemplo de este tipo de mecanismo de manejo de restricciones son las condiciones de Deb descritas en [Deb00]:

  - De entre dos soluciones factibles, se elige la que tenga el menor valor de función objetivo.
  - Entre una solución factible y una no factible, se elige la factible.
  - Entre dos funciones no factibles, se elige la que tenga menor suma de violación de restricciones.

- **Ranking estocástico:** Este mecanismo, propuesto por Runarsson y Yao [RY00], fue diseñado para lidiar con los inconvenientes de las funciones de penalización definiendo un parámetro P_f que controla el criterio utilizado para comparar soluciones no factibles (basado en la suma de violación de restricciones o en el valor de la función objetivo).

- **Método de restricción ε:** Propuesto por Takahama *et al.* [TSI05], transforma a los CNOP en problemas no restringidos. Este mecanismo es utilizado por Mezura *et al.* [MDC10] dentro de su versión del algoritmo ABC.

- **Funciones de penalización innovadoras:** A pesar de que en la literatura están documentadas las diversas desventajas de utilizar funciones de penalización, en la actualidad aún siguen siendo utilizadas de maneras innovadoras siendo capaces de llegar a buenos resultados.

- **Operadores especializados innovadores:** En este grupo se consideran diferentes tipos de operadores como los definidos para manejar los límites del espacio de búsqueda (operadores de frontera) y aquellos que combinan soluciones factibles con no factibles, entre otros.

- **Uso de conceptos multi-objetivo:** Son aquellos enfoques que combinan mas de un método para el manejo de restricciones con el ánimo de resolver un problema.

2.5. Comentarios Finales

En este capítulo se presentó la base teórica de este trabajo de tesis, la cual toma prestados conceptos provenientes de diversas áreas del conocimiento, principalmente la Biología. Esto demuestra el alcance que puede tener el área de la Inteligencia Artificial y la Optimización al proponer nuevos métodos de resolución de problemas.

De entre todos los tipos de algoritmos presentados en este capítulo, para este trabajo de tesis se eligió un algoritmo basado en colonia de abejas, específicamente el desarrollado por Cetina [Cet09] para ser modificado con un mecanismo de búsqueda local.

En el siguiente capítulo se describirá con detalle la forma en la que se implementan los algoritmos basados en abejas.

Capítulo 3

Algoritmos inspirados en abejas

3.1. Introducción

Entre las especies de insectos que exhiben un proceso de inteligencia colectiva, las abejas han demostrado tener una estructura social muy fuerte que es la clave de su supervivencia, ya que al haber abejas que desempeñan diferentes funciones logran satisfacer sus necesidades a través de un proceso muy eficaz. Ésto, aunado a al hecho de que es relativamente fácil modelar este proceso y representarlo mediante un algoritmo, ha hecho de los algoritmos basados en abejas un tema de estudio que ha ganado popularidad.

Las abejas son insectos sociales nativos de Europa y África, se alimentan de néctar y utilizan el polen como fuente de proteínas para sus larvas. Una colonia de abejas por lo general consiste en una reina, de cero a muchas abejas zángano (dependiendo de la estación) y entre 10,000 y 60,000 abejas obreras [HAM06]. Según lo establecido en [Nat11], en el mundo existen aproximadamente 20,000 especies de abejas, dentro de las cuales solamente entre el cinco y diez por ciento son sociales. La abeja mielera es uno de los tipos de abejas más utilizados en estudios de comportamiento, debido a su forma de vida social, la cual requiere de coordinación entre todos los individuos de la comunidad; esto a su vez implica comunicación, establecimiento de jerarquías y división de trabajo. La división de trabajo dentro de una colonia de abejas es consecuencia de cambios fisiológicos relacionados con

la edad de las obreras y con la variación genética entre ellas que hace que realicen diferentes tareas. Dentro de las características de los insectos sociales se pueden mencionar la existencia de varias generaciones simultáneamente, el cuidado cooperativo de las larvas, la existencia de castas estériles (individuos que no se reproducen dentro de las colmenas) y su comportamiento altruista.

3.2. Comportamiento social de las abejas

La interacción social entre las abejas es la base del algoritmo presentado en este trabajo de tesis, siendo ésta el mecanismo a simular para la resolución de problemas complejos.

3.2.1. Tipos de abejas

Karaboga y Akay en [KA09b] indican que existen tres tipos de abejas en una colonia: zánganos, obreros y abeja reina. En la fig. 3.1 se muestran gráficamente.

- **Abeja reina:** Es la única hembra capaz de poner huevos, por lo que es la madre de toda la colonia. Usualmente, la reina copula solo una vez en su vida y fertiliza por uno o dos años más, usando el esperma obtenido en el apareamiento. Después de haber consumido el esperma, produce huevos no fertilizados y se elige a alguna de las hijas como sucesora.
- **Zánganos:** Los zánganos son las abejas macho, padres de la colonia. No viven más de seis meses y llegan a existir cientos de estas abejas en la colonia, principalmente en verano. Los zánganos mueren una vez que se han apareado con la abeja reina.
- **Obreros:** Las abejas obreras recolectan y almacenan alimento, eliminan basura y abejas muertas, ventilan y protegen la colmena. Construyen las celdas de cera donde la abeja reina deposita sus huevos y alimenta a las larvas. Las tareas de una abeja obrera se basan en su edad y las necesidades de la colonia. Viven alrededor de seis meses en épocas veraniegas y de cuatro a nueve meses en épocas invernales.

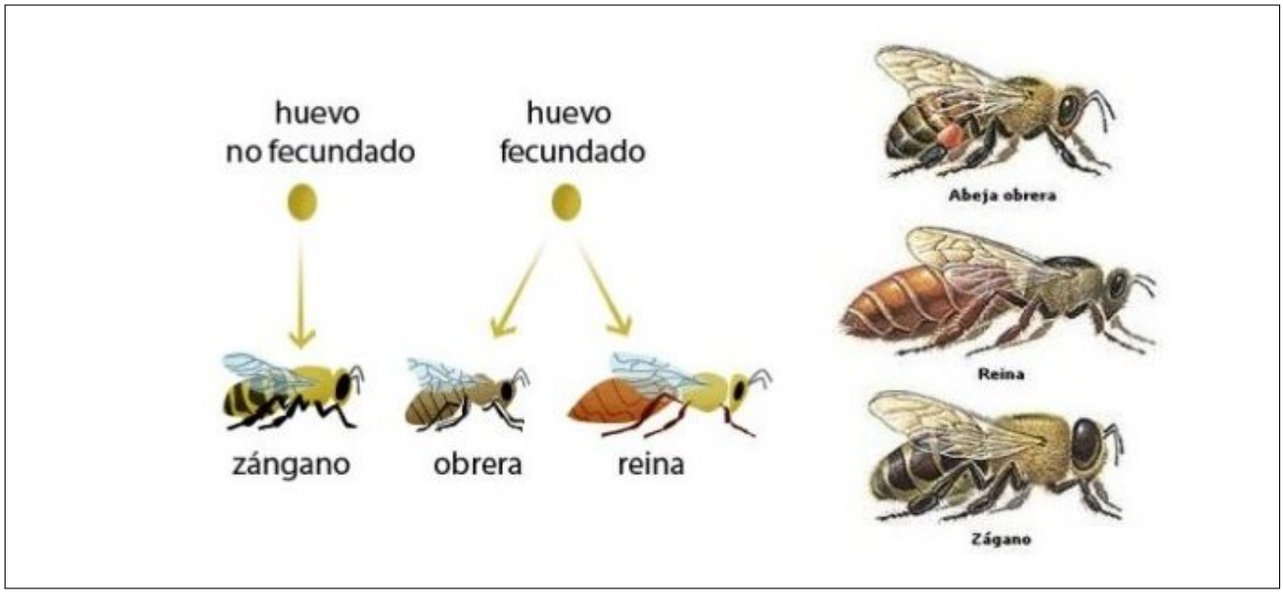

Figura 3.1: Tipos de abejas en una colmena

Una colonia de abejas puede fundarse de dos maneras: en la "fundación independiente", la colonia inicia con dos o más hembras reproductivas que construyen el nido, ponen los huevos y alimentan a las larvas. El primer grupo de camadas se crían solas hasta que toman responsabilidad del trabajo de la colonia. Subsecuentemente, se realiza la división de labores: la reina se especializa en poner huevos y las obreras en el cuidado de las larvas. Otro método de fundación es el llamado "enjambre", en el cual una nueva colonia es fundada por una o más reinas junto a un grupo de obreras de la colonia original.

3.2.2. Comportamiento de las abejas

De acuerdo a Xing y Gao en [XG14] el comportamiento de una abeja puede resumirse en forrajeo (recolección de alimento), casamiento y comunicación.

- **Forrajeo:** Este proceso incluye dos modos: hallazgo de una fuente de alimento y abandono de la fuente. Las abejas que encuentran una fuente la indican a las otras mediante una danza. Las abejas visitan las fuentes de alimento hasta que se encuentre una mejor.
- **Casamiento:** Se refiere al apareamiento entre las abejas. Los zánganos fecundan a la reina, que es la única capaz de depositar huevos. Esto sucede cuando la reina empren-

de un vuelo alejada de la colmena, alrededor de 40 metros. Los zánganos se aparean con ella en el aire y mueren poco después.

- **Comunicación:** Las abejas utilizan una serie de movimientos (danzas) para compartir información (indicaciones acerca de fuentes de alimento) y persuadir a las otras abejas a que sigan sus direcciones.

3.2.3. Modelo de comportamiento para búsqueda de alimento

Las abejas poseen un lenguaje extremadamente preciso basado en danza. Después de haber encontrado alimento y regresado a la colmena, la abeja exploradora informa a las otras la distancia, dirección, cantidad y calidad del alimento encontrado. Gracias a su percepción táctil, visual y olfatoria, las otras abejas reciben esta información. Hay dos tipos de danzas: la danza giratoria que indica que la fuente de alimento es muy cercana; la danza en bucle, que dibuja un "8", indica distancia (fig. 3.2) y dirección de la fuente de alimento, tomando como referencia al Sol.

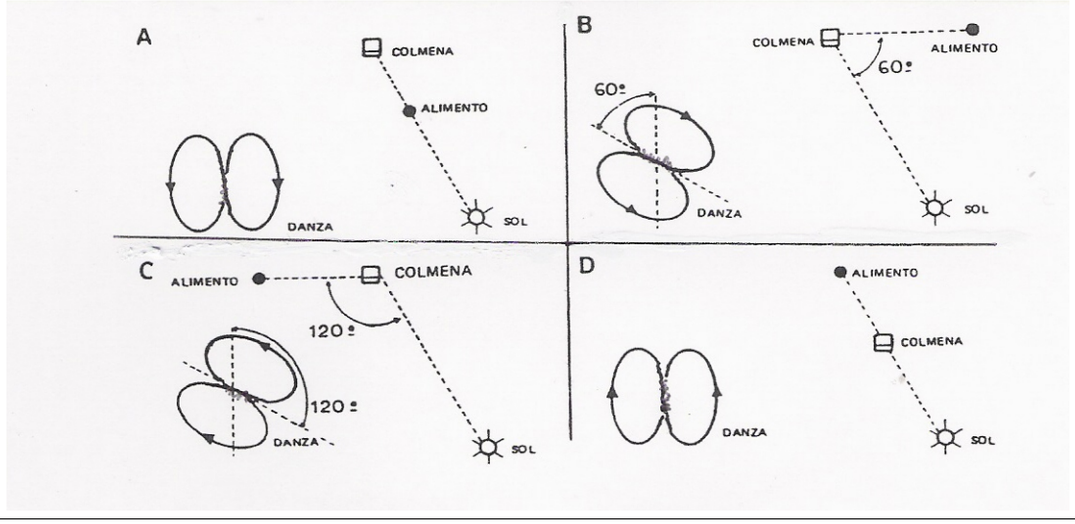

Figura 3.2: Danzas de comunicación de las abejas

Karaboga y Akay en [KA09a] describen el modelo de comportamiento de búsqueda de alimento desarrollado por Tereshko y Loengaror en [TL05]. Este modelo consiste en tres componentes esenciales: fuentes de alimento, recolectores empleados y recolectores desempleados, además de describir dos modelos principales del comportamiento de la colonia: asignación a una fuente de alimento y el abandono de la misma.

Tereshko explica estos compononentes de la siguiente manera:

- **Fuentes de alimento:** Para seleccionar una fuente de alimento, una abeja recolectora evalúa diversas propiedades relacionadas con la fuente de alimento tales como su proximidad a la colmena, riqueza de la energía, sabor del néctar y la dificultad de extraer la energía.
- **Recolectores empleados:** Un recolector empleado está asignado a una fuente de alimento específica la cual se encuentra explotando en determinado momento. El recolector contiene información acerca de su fuente de alimento y la comparte con las otras abejas que están esperando en la colmena.
- **Recolectores desempleados:** Un recolector que busca una fuente donde trabajar es llamado desempleado. Puede ser tanto un explorador que busca el entorno de manera aleatoria o un observador que intenta encontrar una fuente de alimento de acuerdo a la información que ha recibido de alguna abeja empleada.

El intercambio de información entre abejas es el suceso más importante en la formación del conocimiento colectivo. Al examinar la colmena entera es posible distinguir algunas partes que son comunes para todas las colmenas. La parte más importante de la colmena con respecto al intercambio de información es el área de danza. La comunicación relacionada con la calidad de las fuentes de alimento ocurre en este lugar. A la danza relacionada se le llama bamboleo. Al estar toda la información de las diferentes fuentes de alimento en la zona de danza, existe una mayor probabilidad de que una abeja observadora elija una fuente rentable comparando la información de todas las que existen. Cuando la abeja elige una fuente de alimento, vuela hasta a ella memorizando su ubicación para inmediatamente explotarla. A su regreso a la colmena para depositar el néctar recolectado, tiene tres opciones:

- Abandonar la fuente de alimento.
- Realizar el bamboleo y reclutar más abejas a esa fuente.
- Continuar explotando la fuente por sí sola.

Es importante señalar que no todas las abejas inician el forrajeo de manera simultánea. Se ha confirmado con experimentos que nuevas abejas inician la recolección a

una tasa proporcional a la diferencia entre el total de abejas y el número de abejas recolectando en el momento presente.

3.3. Modelos de simulación del comportamiento de abejas

En el ámbito de los algoritmos inspirados en colonias de abejas, Koc en [Koc10] propone la siguiente clasificación: algoritmos basados en búsqueda de alimento, basados en casamiento y basados en Abeja Reina. Por otro lado, Xing y Gao [XG14] indican que las metaheurísticas basadas en abejas se pueden agrupar en algoritmos basados en búsqueda de alimento y en algoritmos basados en reproducción.

3.3.1. Algoritmos basados en búsqueda de alimento (forrajeo)

Los algoritmos de abejas basados en forrajeo conforman la clasificación más popular de estos métodos. El algoritmo M-ABC, que es la base de este trabajo de tesis, pertenece a este rubro.

Sistema de abejas (BS)

Este paradigma fue el primero en su clase, presentado por Sato y Hagiwara en [SH97] y posteriormente por Lucic y Teodorovic en [LT01], quienes fueron los primeros en utilizar este algoritmo en problemas de optimización combinatoria como el TSP y otros problemas de transporte.

Este enfoque dice estar inspirado básicamente en un comportamiento de “encontrar una fuente y reclutar a otros en ella”. Sin embargo, esta idea fue implementada como un algoritmo genético híbrido. En el algoritmo algunos de los cromosomas se consideraban superiores y se trataba de generar soluciones alrededor de ellos usando múltiples poblaciones. Además, el algoritmo usa operadores como la cruza concentrada y métodos pseudo-simplex [Koc10].

Lucic y Teodorovic presentaron en [LT01] otra versión del algoritmo, el cual fue uno de los primeros intentos en desarrollar un algoritmo inspirado directamente en abejas. El algoritmo fue desarrollado para problemas combinatorios y se aplicó a TSP. En la lógica del algoritmo, la colmena está ubicada en una posición aleatoria del espacio de búsqueda y se mejora siguiendo una selección probabilística similar a la utilizada en el algoritmo ACO.

Algoritmo de abejas virtuales (VBA)

Fue propuesto por Yang en 2005 [Yan05] donde la manera en la que maneja la función objetivo es similar a la de los algoritmos genéticos, con los cuales fue comparado y logró obtener mejores resultados en problemas de una y dos variables.

El algoritmo inicia desplegando una tropa de abejas virtuales en el espacio de búsqueda para realizar una búsqueda aleatoria. De acuerdo a Koc en [Koc10] los pasos principales del algoritmo VBA para la optimización de funciones son:

1. Crear una población de abejas. Cada abeja está asociada a un banco de memoria con varias cadenas.
2. Codificar los objetivos del problema y convertirlas en alimento virtual.
3. Definir un criterio de comunicación de la dirección y distancia en forma de una función.
4. Actualizar la posición de los individuos.
5. Después de cierto tiempo, el sitio con mayor visitas de abejas corresponde a la mejor estimación.
6. Decodificar los resultados para obtener la solución del problema.

Algoritmo de búsqueda de alimento de abejas (BF)

Este algoritmo fue presentado por Lemmens *et al.* en [LKAdJS07] donde se maneja una serie de estados de los agentes (abejas) para la toma de decisiones, simulando el compor-

tamiento de las abejas para la recolección de alimento. Las abejas reciben información de otras abejas que previamente han encontrado fuentes de alimento y deciden si utilizar esa información o salir a explorar en busca de una nueva fuente de alimento.

Se utilizan dos estrategias esenciales en el algoritmo: reclutamiento y navegación. La estrategia de reclutamiento se utiliza para distribuir información relacionada con las fuentes de néctar a los miembros de la colonia. La estrategia de navegación sirve para mejorar la eficiencia al explorar una región desconocida a través de una "integración de caminos" que se usa por las abejas para viajar de vuelta a la colmena mientras se mueven entre fuentes de néctar apartadas. Koc [Koc10] señala que la estructura general del algoritmo es similar a la del algoritmo ACO.

Sistema de abejas con feromonas de inhibición (BSP)

Este algoritmo fue presentado por Lemmens *et al.* en [LdJTN07], posterior a su publicación de BF. Debido a que el algoritmo BF utilizaba sólo direcciones lineales para realizar las búsquedas, se optó por diseñar este algoritmo híbrido, el cual utiliza "feromonas de inhibición" (mecanismo tomado del algoritmo ACO), que afectan la trayectoria de las abejas en su búsqueda de fuentes de alimento.

El primer procedimiento propuesto es una manera simple de mejorar la capacidad de evadir obstáculos que ayuda al agente cuando sigue su vector de "integración de caminos". Cuando se topa con un obstáculo, el agente selecciona una dirección aleatoria y recorre el perímetro de obstáculo hasta que puede continuar sobre el vector. El segundo procedimiento mejora la capacidad de aprendizaje del algoritmo ya que los agentes pueden depositar feromonas de inhibición en un punto dado. Los agentes que siguen el vector de "integración de caminos" se benefician con este procedimiento al encontrar mejores soluciones en el espacio de búsqueda.

Colonia artificial de abejas (ABC)

Este algoritmo fue presentado por Karaboga en [Kar05] donde describe un modelo diseñado para resolver problemas de optimización numérica mediante dos conductas principales: el reclutamiento de abejas en una fuente de comida y el abandono de dichas fuentes. En el diseño de este algoritmo se supone que cada cada fuente de alimento corresponde a una posible solución y la cantidad de néctar de la fuente corresponde a la calidad de la solución.

El algoritmo consta de dos grupos de abejas: empleadas y desempleadas. Las abejas empleadas se encuentran trabajando en una fuente de alimento en particular y las desempleadas son exploradoras u observadoras. Las abejas exploradoras buscan nuevas y mejores fuentes de alimento que puedan reemplazar a las ya conocidas; las observadoras, por otro lado, esperan a que las exploradoras les comuniquen sus hallazgos mediante una danza para poder decidir si explotar dichas fuentes de alimento o no. En la figura fig. 3.3 se muestra una representación gráfica de este proceso.

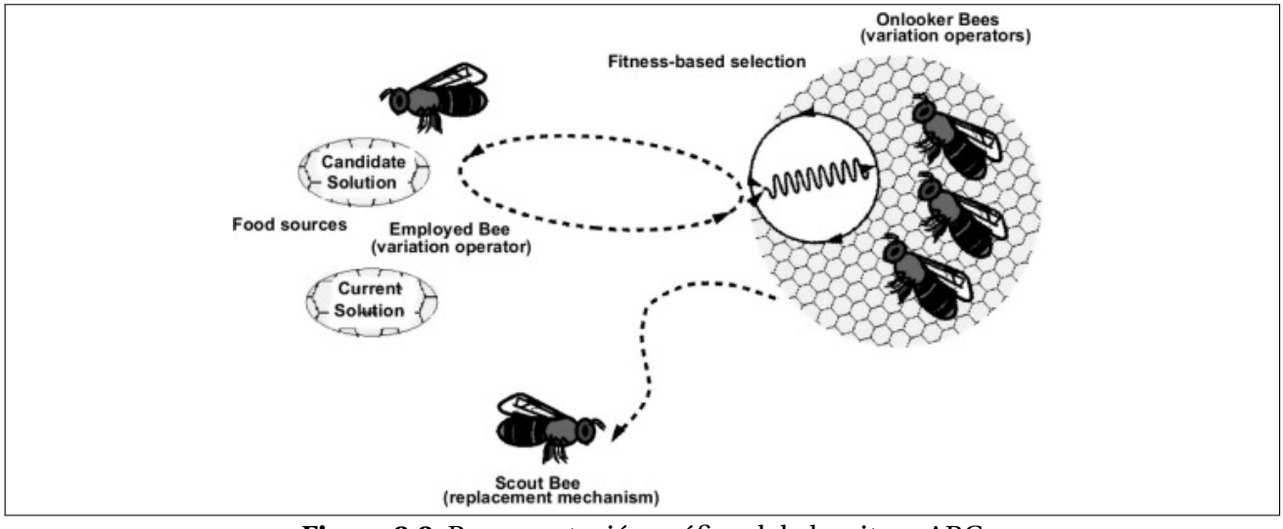

Figura 3.3: Representación gráfica del algoritmo ABC

Una de las ventajas de este algoritmo es el número reducido de parámetros que necesita: número de soluciones (fuentes de alimento), el número de abejas empleadas y observadoras, el número total de ciclos (iteraciones) del algoritmo y el número de ciclos que una solución puede permanecer sin ser mejorada antes de ser reemplazada.

Algoritmo de optimización por colonia de abejas (BCO)

Esta metaheurística fue propuesta por Teodorovic *et al.* en 2006 [TLMDO06] para resolver problemas de optimización combinatoria. De manera similar al algoritmo ACO, crea las soluciones de manera constructiva aunque la diferencia reside en que BCO hace esto de manera parcial. En cada etapa del algoritmo las abejas crean una solución parcial volando sobre los nodos en un "recorrido hacia delante". En la etapa del "recorrido hacia atrás" todas las abejas son enviadas de vuelta a la colmena y comparten información de las soluciones parciales encontradas para decidir si abandonar alguna de estas soluciones, expandirla con la información de otras abejas o reclutar más abejas para regresar a dicha solución parcial.

Algoritmo BCO para TSP

Tomando como base el algoritmo de Teodorovic, Wong *et al.* [WLC08] añadieron mecanismos de búsqueda local para resolver TSP. En esta adaptación las abejas pueden recordar cuántas abejas han visitado una solución; además, las abejas muestran la ruta factible en su totalidad en lugar de recorridos parciales y la colmena está situada de manera equidistante a todas las fuentes de alimento. Durante la construcción de las soluciones, las abejas se basan en la factibilidad del arco y la distancia entre fuentes de alimento.

Algoritmo de colmena de abejas (BH)

Este algoritmo fue propuesto por Wedde *et al.* [WFZ04] como un algoritmo de enrutamiento para redes fijas, basándose en los mecanismos de comunicación de las abejas. Las abejas en este algoritmo visitan regiones de la red (zonas de recolección de alimento) y la información que obtienen se utiliza para actualizar las tablas de enrutamiento de la red.

Algoritmo BeeAdHoc

Wedde *et al.* [WFP+05] presentaron este algoritmo con el propósito de manejar redes móviles de manera eficiente en cuestiones de energía. Está basado en el algoritmo BeeHive,

aunque en este algoritmo se incluyen nuevos tipos de agentes: empacadores (reciben y almacenan paquetes de datos de la capa de transporte), exploradores (descubren nuevas rutas), recolectores (reciben y transportan paquetes de datos) y enjambres (ayudan a manejar protocolos de transporte poco confiables). De manera similar a BH, las abejas se usan en redes de conmutación de paquetes para encontrar rutas confiables entre nodos actualizando la tabla de rutas. Se usan dos tipos de agentes: agentes de corta distancia, que diseminan información de las rutas y agentes de larga distancia que viajan a todos los nodos de la red [Koc10].

Algoritmo de colmena de abejas artificiales (ABH)

Muñoz *et al.* en [MLC09] presentan este algoritmo como un modelo en donde cada abeja es representada como un individuo cuyo comportamiento se regula mediante una estructura de control. La conducta de una abeja se determina por la información interna y externa disponible y su estado de motivación, de acuerdo a un conjunto de reglas específicas. Las reglas son las mismas para todas las abejas, pero la conducta cambia debido al lugar en el espacio que ocupan las abejas.

Algoritmo de optimización de enjambre de abejas (BSO)

El algoritmo BSO, presentado por Akbari *et al.* en [AMZ10], contiene tres tipos de abejas: recolectoras, observadoras y exploradoras, que vuelan en un espacio de búsqueda para encontrar la solución óptima. La factibilidad de una abeja representa la calidad de la fuente de alimento encontrada por la abeja en cuestión; además, el algoritmo emplea un proceso estocástico para hallar la solución óptima.

Algoritmo de abejas (BA)

En el algoritmo BA, presentado por Pham y Castellani en [PC09], cada punto en el espacio de búsqueda representa una fuente de alimento. Las abejas exploradoras generan soluciones de manera aleatoria y, mediante la función de factibilidad, reportan la calidad de las

fuentes visitadas. De esta manera se localizan las soluciones más prometedoras para explorar su vecindad y encontrar un óptimo global de la función objetivo.

3.3.2. Algoritmos basados en casamiento de abejas

Los algoritmos basados en casamiento de abejas simulan el comportamiento de dichos insectos al momento de aparearse.

Algoritmo de optimización por casamiento de abejas (MBO)

Este algoritmo, propuesto por Abbas en [Abb01], simula la evolución de las abejas en diversas etapas. Comienza con una colonia solitaria (una reina sin colonia) y escala hasta tener una colmena completa con una o dos reinas. El algoritmo está basado en el recocido simulado, al cual se asemeja en varios puntos.

Optimización por apareamiento de abeja (HBMO)

De acuerdo a lo presentado por Haddad *et al.* en [HAM06], el vuelo nupcial se considera un conjunto de transiciones donde la reina se mueve entre una serie de estados a una velocidad determinada y copula con los zánganos que encuentra en cada estado.

3.3.3. Algoritmos basados en el comportamiento de la abeja reina

Algoritmo de evolución por abeja reina (QBE)

El algoritmo de Evolución por Abeja Reina (QBE) fue propuesto por Jung [Jun03] para mejorar las capacidades de los algoritmos evolutivos. Está basado en el papel de la abeja reina en el proceso de reproducción. En el algoritmo, como el individuo más apto de la generación, la abeja reina se cruza con otras abejas seleccionadas como padres por un algoritmo de selección. Esto aumenta la probabilidad de que exista una convergencia prematura en los resultados, por lo que se propone un proceso de mutación intensiva para solucionar

este problema.

Cruzamiento de abejas (Bee Crossover)

Kaci propone en [Kar04] un operador de cruzamiento inspirado en el apareamiento de las abejas. Este método selecciona un cromosoma específico presente en la población como abeja reina. Mientras que la abeja reina seleccionada es uno de los padres en el cruzamiento, todos los cromosomas restantes tienen la oportunidad de ser el siguiente padre en cada generación. Con este propósito se definen tres métodos de cruzamiento: en el primer método el cromosoma con mejor aptitud se elige como abeja reina y se vuelve un padre fijo para el cruzamiento en la generación actual. El segundo método maneja el cromosoma con menor aptitud. Finalmente, la abeja reina es reemplazada secuencialmente en cada generación.

3.4. Algoritmos basados en abejas para problemas con restricciones

Los anteriores algoritmos basados en abejas tienen la desventaja de no trabajar en problemas con restricciones. Existen diversas estrategias para atacar este tipo de problemas con los algoritmos mencionados, que se explican a continuación. Utilizando la clasificación propuesta por Mezura y Coello [MC11], se pueden agrupar los diferentes métodos basados en abejas para manejo de restricciones en:

- **Reglas de factibilidad**

  Debido a la popularidad de este tipo de manejo de restricciones, dada su facilidad de implementación, existen diversos algoritmos que hacen uso de estas reglas en conjunto con algoritmos basados en abejas. Karaboga y Basturk [KB07] y Karaboga y Akay [KA11] modificaron una selección voraz basada en el valor de la función objetivo utilizando reglas de factibilidad con el propósito de adaptar el algoritmo ABC para resolver CNOPs. Mezura y Cetina en [MC09] extendieron el paradigma de Kara-

boga usando un operador especial diseñado para localizar soluciones cercanas a la mejor solución factible en el método ABC con comportamiento modificado de abejas exploradoras (SM-ABC). Además, Mezura y Velez en [MV10] proponen una modificación denominada ABC elitista (E-ABC) donde se mejoraron dos operadores y se añadió un operador de búsqueda directa local.

Existen en la literatura otros métodos que proponen el uso de reglas de factiblidad para el manejo de restricciones, como el método ABC restringido simple (SC-ABC) de Brajevic [BTS10], el ABC restringido modificado (MO-ABC) descrito por Subotic en [Sub11], el ABC guiado (G-ABC) propuesto por Tuba *et al.* en [TBS11], el ABC modificado (M-ABC) propuesto por Cetina y Mezura en [CM12], así como la modificación propuesta por Babeizadeh y Ahmad en [BA14] y el algoritmo basado en cruzamiento de Brajevic [Bra15].

- **Método de restricción ε**

  Mezura *et al.* [MDC10] utilizaron el método de restricción ε dentro del algoritmo ABC con operadores de vuelo (SF-ABC). Además se consideró un mecanismo dinámico para reducir la tolerancia para restricciones de igualdad.

- **Funciones de penalización**

  Haddad y Mariño en [HM07] proponen el uso de funciones de penalización en el algoritmo HBMO aplicado al control de tuberías de agua.

- **Operadores especiales**

  En el algoritmo ABC genéticamente inspirado (GI-ABC) descrito por Bacanin y Tuba [BT12] se ocupan operadores de uso común en algoritmos genéticos para mejorar la creación de soluciones candidatas.

- **Conceptos multiobjetivo** [HZGN15]

  Li y Tin en [LY14] proponen una versión de ABC autoadaptable (SACABC) que combina dos técnicas de manejo de restricciones para mejorar sus resultados, proponiendo el uso de técnicas derivadas de Evolución Diferencial. Por otro lado, Huo *et al.* [HZGN15] describen el método ABC multiobjetivo guiado por soluciones élites

(EMOABC) donde aplican un ordenamiento de soluciones no dominadas para medir la calidad de las soluciones y seleccionar las mejores. Además, usando una estrategia de generación de soluciones guiadas por las soluciones élites permite examinar eficientemente la vecindad de las soluciones candidatas.

3.5. Comentarios finales

Dentro de los métodos de inteligencia colectiva los enfoques inspirados en abejas demuestran ser unos de los más populares, con diversos algoritmos existentes en la literatura para cada una de las clasificaciones (tipos de comportamiento), siendo la de recolección de alimento (forrajeo) la más explorada. Este tipo de comportamiento, al estar inspirado en la habilidad de las abejas de explorar sus alrededores y compartir sus hallazgos con las abejas presentes en la colmena, promete recorrer una buena parte del espacio de búsqueda en pocas iteraciones. Si bien el algoritmo ABC original tiene un buen desempeño al explorar el espacio de búsqueda, su capacidad de explotación es limitada, por lo que existen diversas versiones modificadas de este algoritmo donde se proponen mecanismos para aumentar las capacidades de explotación además de generar nuevas soluciones factibles a partir de las ya conocidas, de esta manera explorando regiones del espacio de búsquedas que hayan podido ser pasadas por alto. Entre estos algoritmos modificados se encuentra el M-ABC de Mezura y Cetina [CM12] que sirve como punto de partida para la realización de este trabajo de tesis.

Capítulo 4

Descripción del Proyecto

4.1. Introducción

Los algoritmos de optimización basados en inteligencia colectiva encuentran soluciones mediante métodos de prueba y error colaborativos. El aprendizaje entre pares dentro de colonias sociales es la fuerza principal detrás del desarrollo de diversos algoritmos de optimización de alta eficiencia [BSJ13]. Jayanth *et al.* [JKK15] indican que el algoritmo ABC está basado en la conducta de las abejas al momento de encontrar fuentes de alimento sin el beneficio de la información visual. El intercambio de información de las abejas es un conocimiento integrado sobre qué caminos elegir y la calidad de las fuentes de alimento compartido mediante una danza. Las abejas calculan su fuente de alimento utilizando una selección probabilística. Bansal *et al.* [BSJ13] indican que ABC consiste en una población de soluciones potenciales, mismas que representan fuentes de alimento de las abejas. La aptitud se determina en términos de la calidad (cantidad de néctar) de la fuente de alimento. ABC es un proceso iterativo donde existen dos procesos fundamentales que se derivan de la evolución de la población: el proceso de variación, que hace posible explorar diferentes áreas del espacio de búsqueda, y el proceso de selección, que asegura la explotación de las fuentes encontradas. Los procesos de ABC constan de cuatro fases: inicialización, creación de abejas empleadas, selección de abejas observadoras y reemplazo de abejas exploradoras.

La modificación presentada a M-ABC en este proyecto sigue la sugerencia de Mezura *et al.* [MCH10] de agregar un método de búsqueda local en el algoritmo para mejorar los resultados obtenidos. El método de búsqueda local elegido es el de Direcciones Conjugadas de Powell, el cual a su vez utiliza los métodos Bounding Phase y Búsqueda por Sección Áurea para generar los vectores de dirección que utiliza en su funcionamiento.

4.2. Algoritmo ABC

A continuación se describe de manera detallada el funcionamiento del algoritmo ABC indicando los principales elementos así como sus pasos.

4.2.1. Elementos de ABC

El algoritmo ABC presenta los siguientes elementos:

- **Parámetros del algoritmo:** El parámetro SN es el número de soluciones (fuentes de alimento), y además es el número de abejas empleadas y de abejas observadores; MCN es el número total de ciclos (iteraciones) del algoritmo; $limit$ es el número de ciclos que una solución no mejorada puede ser guardada antes de ser reemplazada. El tamaño de la colonia es de $2 * SN$.
- **Representación de soluciones:** Las posibles soluciones del problema representan fuentes de alimento. Éstas a su vez se representan por vectores de D dimensiones, donde D es el número de variables del problema. Una solución x_i representa la i-ésima solución en el ciclo g del proceso de búsqueda de ABC. Cada una de las variables en la solución está asociada a un rango ($L_i \leq x_i \leq U_i$), el cual debe ser considerado cuando se generan las soluciones iniciales de manera aleatoria.
- **Mecanismo de selección:** El mecanismo de selección de la mejor solución se lleva a cabo a través de la comunicación entre abejas empleadas y observadoras, para lograr que aquellas fuentes de mejor calidad sean visitadas más frecuentemente.

- **Operadores de variación (abejas):** En ABC las abejas son vistas como operadores de variación ya que cuando una abeja llega a una fuente de alimento, genera una nueva solución candidata $v_{i,j}$ usando la fórmula $v_{i,j}^{g} = x_{i,j}^{g} + \phi_j * (x_{i,j}^{g} - x_{k,j}^{g})$, donde $x_{i,g}$ representa la solución en la que la abeja se encuentra en ese momento, $x_{k,g}$ es una solución existente escogida de manera aleatoria, g es el número de la iteración actual y ϕ es un número real aleatorio en el rango [-1, 1]. En la fig. 4.1 se muestra una descripción gráfica de este proceso.

- **Mecanismo de reemplazo:** Este mecanismo se lleva a cabo mediante las abejas exploradoras. Estas renuevan aquellas fuentes de alimento que no han sido mejoradas por un número determinado de ciclos generando nuevas fuentes de alimento de manera totalmente aleatoria con distribución uniforme.

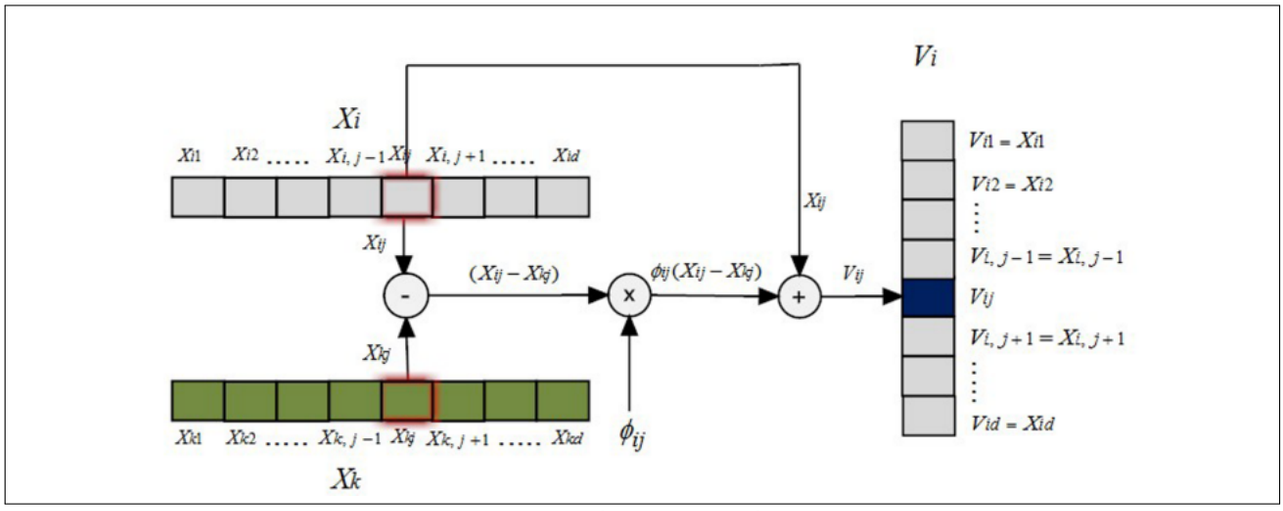

Figura 4.1: Creación de nuevas soluciones en ABC

4.2.2. Pasos del algoritmo ABC

El algoritmo 1 representa los pasos de ABC, los cuales se describen a continuación:

Se comienza generando SN soluciones iniciales de manera aleatoria. Después de ser generadas, se evalúan para determinar su aptitud. A continuación se inicia un ciclo de MCN repeticiones que incluye las fases de abejas empleadas, desempleadas y exploradoras.

En la **fase de abejas empleadas** se crean soluciones candidatas a partir de las ya existentes.

Si la candidata es mejor a la original (de acuerdo a las reglas de Deb), ésta se reemplaza.

En la **fase de abejas desempleadas** se realiza un torneo binario entre dos soluciones al azar, donde la que resulte elegida se utilizará para generar una nueva solución candidata que será evaluada con las reglas de Deb.

En la fase de **abejas exploradoras** se buscan aquellas soluciones que han sido abandonadas (no han sido mejoradas en la condición impuesta por el valor $limit$) y son reemplazadas con soluciones generadas a partir de la mejor solución hasta el momento.

En el **paso final** se compara la mejor solución del ciclo contra la mejor solución encontrada hasta el momento, siendo ésta reemplazada en el caso de no ser mejor que la encontrada en la presente iteración.

Algoritmo 1: Algoritmo ABC

```
1  Iniciar la población de soluciones x_i^0, i = 1, ..., SN
2  Evaluar cada x_i^0, i = 1, ..., SN                                   // Paso 1
3  g = 1
4  repetir
5      para i = 1,SN hacer
6          Generar v_i^g con x_i^{g-1} usando ??                        // Abejas empleadas
7          Evaluar v_i^g
8          si v_i^g es mejor que x_i^{g-1} entonces
9              x_i^g = v_i^g
10         en otro caso
11             x_i^g = x_i^{g-1}
12         fin
13     fin
14     para i = 1,SN hacer
15         Escoger, según la aptitud, una fuente de alimento x_l^g
16         Generar v_l^g con x_l^g usando ??                            // Abejas desempleadas
17         Evaluar v_l^g
18         si v_l^g es mejor que x_l^g entonces
19             x_l^g = v_l^g
20         fin
21     fin
22     Generar nuevas fuentes de alimento de manera aleatoria para aquellas que hayan pasado el
       límite.                                                          // Abejas exploradoras
23     Guardar la mejor solución hasta el momento.
24     g = g + 1
25 hasta que g = MCN;
```

4.3. Algoritmo ABC modificado (M-ABC)

La versión modificada de ABC propuesta por Mezura y Cetina en [CM12] presenta una adaptación de ABC de manera que sea capaz de trabajar con problemas de optimización numérica con restricciones. El enfoque para trabajar en un espacio de búsqueda con restricciones se centró en la selección voraz entre fuentes de alimento, implementando un nuevo parámetro MR (tasa de modificación) en el rango [0,1]:

$$v_{i,j}^{g} = \begin{cases} x_{i,j}^{g} + \phi_j * (x_{i,j}^{g} - x_{k,j}^{g}) & \text{si } rand(0,1) < MR \\ x_{i,j}^{g} & \text{de otro modo} \end{cases} \tag{4.1}$$

4.3.1. Elementos adicionados por M-ABC

En M-ABC se agregaron cuatro mecanismos más a ABC, los cuáles se explican a continuación:

- **Selección por torneo:** Este mecanismo incluye el uso de las reglas de factibilidad de Deb.
- **Tolerancia dinámica para restricciones de igualdad:** Al ser difíciles de calcular, las restricciones de igualdad se reformulan como restricciones de desigualdad introduciendo el valor de tolerancia ε:

$$|h_j(x)| - \varepsilon \leq 0 \tag{4.2}$$

  Una manera de manejar el valor de ε es mantenerlo fijo durante la duración de la búsqueda. Por otro lado, usar un mecanismo para variar el valor de este término puede llevar a mejores resultados. Esta última es la opción utilizada para M-ABC por medio de un parámetro de control dinámico, definido de la siguiente manera:

$$\varepsilon(g+1) = \frac{\varepsilon(g)}{dec} \tag{4.3}$$

donde g es el ciclo actual y dec es la tasa de decremento de cada ciclo ($dec > 1$). El objetivo es iniciar con una región factible más amplia que la original, lo que hace que las restricciones de igualdad puedan satisfacerse más fácilmente. A medida que avanzan los ciclos, la tolerancia se reduce para disminuir la violación de restricciones. En la fig. 4.2 se aprecia una representación gráfica de este proceso.

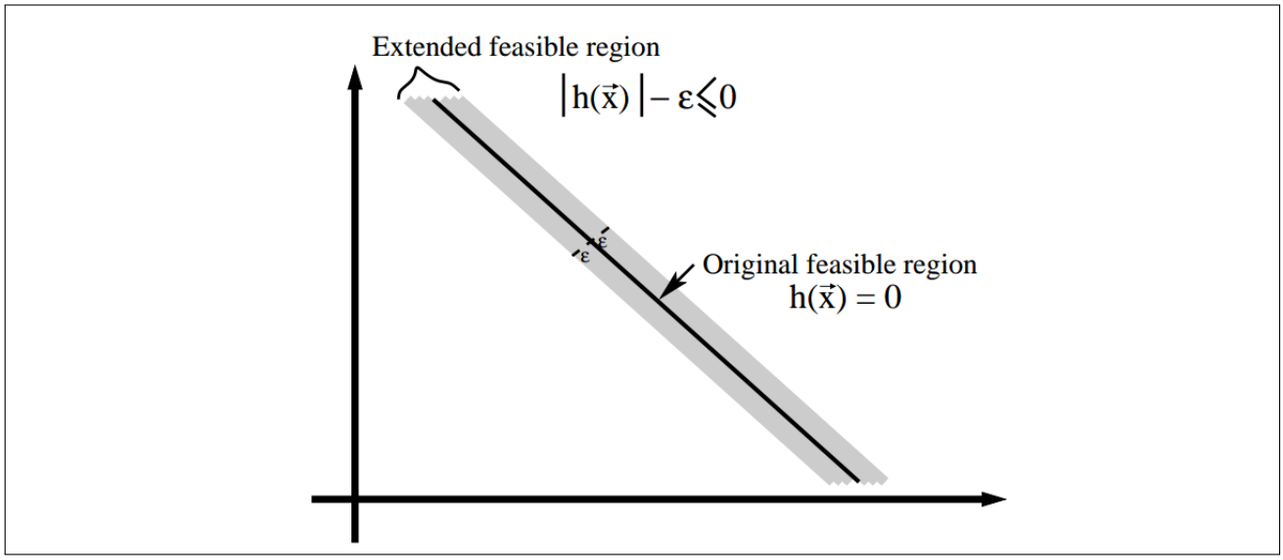

Figura 4.2: Ejemplo de una restricción de igualdad convertida en una de desigualdad

- **Operador de vuelo inteligente:** Basados en el hecho de que en un problema con restricciones la región factible es muy pequeña con respecto al espacio de búsqueda, es complicado generar soluciones de manera aleatoria. Por lo tanto, se adaptó un mecanismo originalmente propuesto para PSO en [LC07] de esta manera: se genera una nueva solución v_i^g por la abeja exploradora con ayuda de la fuente a ser reemplazada x_i^g, que se utiliza como base para crear una dirección de búsqueda definida por la mejor solución en la población actual x_B^g y una solución escogida aleatoriamente x_k^g:

$$v_{i,j}^g = x_{i,j}^g + \phi * (x_{k,j}^g - x_{i,j}^g) + (1-\phi)(x_{B,j}^g - x_{i,j}^g) \tag{4.4}$$

- **Manejo de restricciones en los límites:** El manejo de operadores de variación puede generar valores fuera del espacio de búsqueda definido por los límites inferior y superior de las variables del problema. Por lo tanto, se añadió un mecanismo para reparar estos valores a todas las abejas, descrito de la siguiente manera:

$$v_{i,j} = \begin{cases} 2 * L_j - v^g_{i,j} & \text{si } v^g_{i,j} < L_j \\ 2 * U_j - v^g_{i,j} & \text{si } v^g_{i,j} > U_j \\ v^g_{i,j} & \text{de otro modo} \end{cases} \tag{4.5}$$

- **Cálculo del valor $limit$:** Se implementó una nueva manera de calcular el valor $limit$, debido a que en el algoritmo ABC este parámetro tiende a variar significativamente de problema a problema. Siendo así, se propuso que el nuevo valor fuera determinado por el número total de ciclos del algoritmo y el tamaño de la población: $limit = MCN/(2 * SN)$.

4.3.2. Pasos del algoritmo M-ABC

Los pasos del algoritmo en M-ABC son muy parecidos a los del algoritmo ABC original, solo variando en la implementación de los mecanismos extras detallados anteriormente, como se observa en el algoritmo algoritmo 2.

4.4. Algoritmo M-ABC con búsqueda local basada en el método de Powell

La modificación propuesta en este trabajo de tesis al algoritmo M-ABC, que llevará por nombre MABC-CD, implementa un operador de búsqueda local mediante el método de Direcciones Conjugadas de Powell, el cual es un método de búsqueda lineal y ha demostrado tener un desempeño favorable en su funcionamiento.

Algoritmo 2: Algoritmo M-ABC

Iniciar la población de soluciones x_i^0, $i = 1, ..., SN$;
Evaluar la población; // Paso 1
$g = 1$ **si** *Existen restricciones de igualdad* **entonces** // Paso 2
  Inicializar $\varepsilon = 1.0$;
fin
repetir
  si *Existen restricciones de igualdad* **entonces**
    Evaluar la población con εg
  fin
  para $i = 1, SN$ **hacer**
    Generar v_i^g con x_i^{g-1} usando eq. (4.1)
    Evaluar v_i^g
    si v_i^g *es mejor que* x_i^{g-1} *(basado en las reglas de Deb)* **entonces**
      $x_i^g = v_i^g$
    en otro caso
      $x_i^g = x_i^{g-1}$
    fin
  fin
  // Selección por torneo
  para $i = 1, SN$ **hacer**
    Escoger una fuente de alimento x_l^g según las reglas de Deb.
    Generar v_l^g con x_l^g usando eq. (4.1)
    Evaluar v_l^g
    si v_l^g *es mejor que* x_l^{g-1} *(basado en las reglas de Deb)* **entonces**
      $x_l^g = v_l^g$
  fin
  Generar nuevas soliciones (eq. (4.4)) para aquellas soluciones que hayan llegado al límite.
  // Vuelo inteligente
  Guardar la mejor solución hasta el momento.
  $g = ciclos + 1$;
  si *Existen restricciones de igualdad* **entonces**
    Actualizar $\varepsilon(g)$ usando eq. (4.3)
  fin
hasta que $g = MCN$;

4.4.1. Elementos del algoritmo

Los elementos del algoritmo MABC-CD son muy similares a los del algoritmo M-ABC variando en la implementación de la búsqueda local:

- **Parámetros:** Los parámetros en MABC-CD son los mismos que en ABC y M-ABC.
- **Operador de búsqueda local:** El método de direcciones conjugadas de Powell [Pow64] es el mecanismo elegido como operador de búsqueda local en el algoritmo propuesto. Al aplicar este método es posible mejorar la solución encontrada por el operador de vuelo inteligente. El método de Powell se apoya en dos métodos de búsqueda lineal para lograr su cometido, uno de delimitación y otro de eliminación de regiones. De esta manera, se eligió el método Bounding Phase (algoritmo 3) y el método de búsqueda por sección áurea (algoritmo 4) respectivamente, adecuados con las reglas de factibilidad de Deb para mantener la consistencia de los resultados.

  El método de Direcciones Conjugadas de Powell (mostrando en el algoritmo 5) es de los más populares y que ha generado mejores resultados en diversos problemas de optimización para ingeniería. Como se describe en [Deb04], hace uso de una memoria de soluciones previas para crear nuevas direcciones de búsqueda. Está probado que puede converger en funciones cuadráticas, además de que se han podido resolver funciones no cuadráticas de manera exitosa utilizando este método.

  Este método cuenta con la propiedad de subespacios paralelos, la cual estipula que, teniendo dos puntos x_1 y x_2 y una dirección d, al hacer dos búsquedas unidireccionales desde cada punto en esa dirección, se crearán dos puntos y_1 y y_2. En el caso de una función cuadrática, se dice que el mínimo de la función se encuentra en la línea que une estos dos últimos puntos. El vector $(y_2 - y_1)$ forma una dirección conjugada con el vector original d.

  El método *Bounding Phase*, presentado en el algoritmo algoritmo 3, es un método de delimitación que garantiza delimitar el mínimo de una función unimodal [1]. Inicia con una estimación inicial para a partir de ella encontrar una dirección de búsqueda

[1] Que presenta sólo un valor óptimo

basada en dos o más evaluaciones en los alrededores de ésta. Después, se utiliza una estrategia de búsqueda exponencial para alcanzar el valor óptimo. Por otro lado, el algoritmo de búsqueda por Sección Áurea realiza el proceso de eliminación de regiones desplazándose en relación a la proporción áurea. A pesar de que no es el algoritmo de búsqueda más eficiente, funciona de manera satisfactoria con funciones complejas unimodales, además de que puede ser adaptado para trabajar con funciones no unimodales. En la fig. 4.3 se muestra una representación gráfica de este proceso. El proceso de eliminación de regiones se lleva a cabo mediante la regla descrita en [Deb04]. De esta manera se puede proceder a descartar regiones de búsqueda para lograr llegar al valor óptimo de manera eficiente. El algoritmo de búsqueda por sección áurea funciona tomando el espacio de búsqueda (a, b) de manera unitaria (0,1), y así obteniendo dos puntos a una distancia τ de cada extremo del espacio de búsqueda para que al cabo de cada iteración la región eliminada sea $1 - \tau$.

Algoritmo 3: Algoritmo Bounding Phase

```
1  Escoger una estimación inicial x^(0) y un incremento Δ ;                    // Paso 1
2  Iniciar k = 0;
3  si f(x^0 − |Δx|) ≥ f(x^0 + |Δx|) entonces                                   // Paso 2
4  |  Volver a Δ positivo
5  si no, si f(x^0 − |Δx|) ≤ f(x^0 + |Δx|) entonces
6  |  Volver a Δ negativo
7  en otro caso
8  |  Volver al paso 1;
9  fin
10 Asignar x^(k+1) = x^(k) + 2^k Δ ;                                           // Paso 3
11 si f(x^(k+1)) < f(x^(k)) entonces                                           // Paso 4
12 |  k = k + 1;
13 |  Ir al paso 3;
14 en otro caso
15 |  El mínimo está en el intervalo (x^(k−1), x^(k+1));
16 fin
```

4.4.2. Pasos del algoritmo

Los pasos del algoritmo (mostrados en el algoritmo 6) son relativamente los mismos con respecto a M-ABC. Después de la fase de abejas exploradoras, entra en funcionamiento el método de Powell a partir de la mejor solución encontrada hasta el momento.

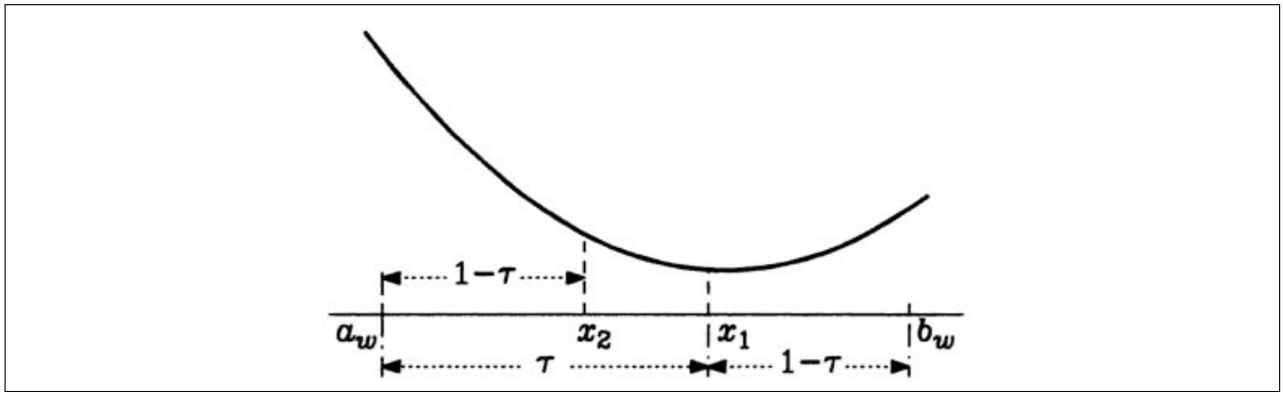

Figura 4.3: Representación gráfica del método de búsqueda por sección áurea

Algoritmo 4: Algoritmo de Búsqueda por Sección Áurea

Escoger un límite inferior a y un límite superior b
Escoger un valor ε como tolerancia
Normalizar x usando $w = (x-a)/(b-a)$
$a_w = 0$
$b_w = 0$
$L_w = 1$
$k = 1$
$w_1 = a_w + (0.618)L_w$
$w_2 = b_w - (0.618)L_w$
Calcular $f(w_1)$ o $f(w_2)$ dependiendo de cuál no haya sido evaluada
Usar las reglas de eliminación de regiones según corresponda
Actualizar a_w y b_w con el nuevo rango obtenido
si $|L_w| < \varepsilon$ *lo suficientemente pequeño* **entonces**
    break
en otro caso
    $k = k + 1$
fin

Algoritmo 5: Direcciones Conjugadas de Powell

Escoger un punto inicial x^0 y un conjunto N de direcciones linearmente independientes.
Minimizar a lo largo de N usando el punto mínimo previo para iniciar la siguiente búsqueda. Iniciar con la dirección de búsqueda s^1 y terminar con s^N. Después, realizar otra búsqueda unidireccional en s^1.
Formar una nueva dirección conjugada d usando la propiedad de subespacios paralelos.
si $||d||$ *es pequeña o las direcciones son linearmente dependientes* **entonces**
    teminar
en otro caso
    Reemplazar $s^j = s^{j-1}$ para toda $j = N, N-1, ..., 2$
    $s^1 = \frac{d}{||d||}$ y regresar al paso 2.
fin

Algoritmo 6: Algoritmo MABC-CD

Inicializar la población de soluciones $x_{i,0}$, $i = 1, ..., SN$;
Evaluar cada solución incial. $g = 1$;
si *Existen restricciones de igualdad* **entonces**
    Iniciar $\varepsilon(g)$
fin
mientras $g \leq MCN$ **hacer**
    si *existen restricciones de desigualdad* **entonces**
        Evaluar cada x_i^0, $i = 1, ..., SN$ con $\varepsilon(g)$
    fin
    para $i = 1 \rightarrow SN$ **hacer**
        Producir nuevas soluciones $v_{i,g}$ para las abejas empleadas usando Ec. 1 y evaluarlas;
        si v_i^g *es mejor que* x_i^{g-1} *(selección por torneo)* **entonces**
            $x_i^g = v_i^g$
        en otro caso
            $x_i^g = x_i^{g-1}$
        fin
    fin
    para $i = 1 \rightarrow SN$ **hacer**
        Seleccionar las soluciones según su aptitud (selección por torneo). Producir nuevas soluciones $v_{i,g}$ para las abejas observadoras usando Ec. 1 y evaluarlas. **si** v_i^g *es mejor que* x_i^{g-1} *(selección por torneo)* **entonces**
            $x_i^g = v_i^g$
        fin
    fin
    Aplicar el operador de vuelo inteligente con las abejas exploradoras a las soluciones que han pasado del límite de generaciones;
    Aplicar el método de Powell;
    Almacenar la mejor solución hasta el momento.
    $g = g + 1$;
    si *existen restricciones de igualdad* **entonces**
        Actualizar $\varepsilon(g)$
    fin
fin

4.5. Comparación con el algoritmo εDEag

Como se ha mencionado con anterioridad, el algoritmo MABC-CD será comparado en sus resultados con el algoritmo de Evolución Diferencial con mutación basada en gradientes y restricciones ε (εDEag) pesentado por Takahama y Sakai en [TS10]. El método de restricciones ε se utiliza para convertir restricciones de igualdad en restricciones de desigualdad tomando en cuenta la suma de violación de restricciones y la comparación a nivel ε, que se describe a continuación [TS10]:

La violación de restricciones puede ser dada por por el máximo de todas las restricciones o la suma de todas las restricciones:

$$\phi(x) = \text{máx}\{\text{máx}_j\{0,\ g_j(x)\},\ \text{máx}_j\ |h_j(x)|\}$$
$$\phi(x) = \sum_j \|\text{máx}\{0,\ g_j(x)\}\|^p + \sum_j \|h_j(x)\|^p$$

donde p es un número positivo.

Este algoritmo integra los siguientes aspectos [TS10]:

- Se incluye un registro para mantener la diversidad de los individuos. Este registro es similar al usado en [ZS09].
- Se sigue un modelo para la generación de descendientes para nivelar la velocidad con la que se llega a una solución óptima: cuando un padre genera un hijo y este no es mejor al padre, el padre puede generar un hijo nuevo.
- Se adopta el método de restricciones ε con selección automática del parámetro de control ε de manera que se pueda mejorar la usabilidad del algoritmo
- Se adopta el proceso de mutación basada en gradiente para resolver problemas con restricciones difíciles.

4.5.1. Pasos del algoritmo εDEag

A continuación se describe a detalle el algoritmo (algoritmo 7) [TS10]:

1. **Se inicializa el registro:** Los individuos $A = \{x^i,\ i = 1, 2, ..., M\}$ se generan de manera aleatoria y forman el registro.

2. **Se inicializa el nivel ε:** Se da el nivel inicial de ε con la función de control $\varepsilon(0)$.

3. **Se inicializa la población:** Los mejores N individuos se seleccionan del registro A y forman una población $P = \{x^i\}$. El rango de los individuos se define por las comparaciones de nivel ε.

4. **Condición de término:** Si el número de evaluaciones de la función excede el número máximo de evaluaciones FE_{max}, el algoritmo termina.

5. **Operaciones de Evolución Diferencial:** Cada individuo x^i se selecciona como padre. Si todos los individuos son seleccionados, ir al paso 7. Se aplica la operación DE/rand/1/exp y se genera un nuevo hijo x^{hijo}. Un factor fijo de escala F_0 y una tasa fija de cruzamiento CR_0 se usan con una probabilidad de 0.95, y el factor de escala F (generado aleatoriamente) y la tasa de cruzamiento CR_0 se usan con probabilidad de 0.05. Si el nuevo individo es mejor que el padre, basándose en la comparación de nivel ε, el padre x_i se reemplaza por el vector x^{hijo} y continúa en el paso 6. En εDEag se adopta un modelo de generaciones continuas [2]. De otra manera, se aplica la misma operación al padre.

6. **Mutación basada en gradiente:** Si x^{hijo} no es factible, o $\phi(x^{hijo}) > 0$, el hijo se modifica mediante mutación basada en gradiente con probabilidad P_g hasta que el número de cambios llega a R_g o x^{hijo} se vuelve factible. Regresar al paso 5 y se selecciona al siguiente individuo como padre.

7. **Control del nivel ε:** El nivel ε se actualiza mediante la función $\varepsilon(t)$ y regresar al paso 4.

[2] Modelo donde individuos de diferentes generaciones se reproducen entre sí.

Algoritmo 7: Algoritmo εDEag

$F = F_0$; $CR = CR_0$;
$A = M$;
$FE = M$;
$\varepsilon = \varepsilon(0)$;
P = Mejores N individuos x^i usando $<_\varepsilon$;
$A = A - P$;
para $t = 1; FE \leq FE_{max}; t$++ **hacer**
    $F = F_0$
    si $t > 0.95T_C$ y $t < T_C$ **entonces**
        Modificar ε y F
    si $u(0, 1) < 0.05$ **entonces**
        $F = 1 + |rand_G(0, 0.05)|$, F se trunca a 1.1
    para $i = 1; i \leq N; i$++ **hacer**
        para $k = 1; k \leq 2; k$++ **hacer**
            x^{p1} = selección aleatoria de P
            x^{p2} = selección aleatoria de P
            si $u(0, 1) < 0.05$ **entonces**
                x^{p3} = selección aleatoria de P
            en otro caso
                x^{p3} = selección aleatoria de $P \cup A$
            $x' = x^{p1} + (x^{p2} - x^{p3})$
            x^{hijo} = vector de prueba generado de x^i y x' **si** $t\ \%n == 0$ y $u(0, 1) < P_g$ **entonces**
                para $h = 1; h \leq R_g$ y $\phi(x^{hijo} > 0); h$++ **hacer**
                    Aplicar mutación basada en gradiente a x^{hijo}
                    $FE = FE + n + 1$
                fin
            en otro caso
                $FE = FE + 1$
            fin
            si $((f(x^{hijo}), \phi(x^{hijo}) <_\varepsilon (f(x^i), \phi(x^i)))$ **entonces**
                $x^i = x^{hijo}$
                break
            en otro caso
                x^A = selección aleatoria de A
                $A = A + x^{hijo} - x^A$
            fin
        fin
    fin
    $\varepsilon = \varepsilon(t)$
fin

4.6. Comentarios Finales

En este capítulo se detalló el algoritmo diseñado para este trabajo de tesis, el cual añade el método de Direcciones Conjugadas de Powell como mecanismo de búsqueda local, por lo que se nombra al algoritmo como MABC-CD. Cabe señalar que el método de Powell no es un método diseñado para la resolución de problemas con restricciones, por lo que es posible que su desempeño no sea satisfactorio y sea necesario recurrir a mecanismos extra. Sin embargo, este método ha demostrado tener un desempeño favorable en la resolución de problemas complejos, por lo que ponerlo a prueba en el manejo de otros tipos de problemas es una tarea que vale la pena llevar a cabo.

Capítulo 5

Experimentación y análisis de resultados

5.1. Introducción

El estudio experimental del algoritmo propuesto en esta tesis se realizó utilizando el conjunto de problemas del CEC 2010 [MS10]. Se utilizaron herramientas para calibración de parámetros (SPOT, irace), las cuales se describen más adelante. Asimismo, se utilizaron pruebas estadísticas para comparar los resultados del algoritmo presentado en esta tesis contra los obtenidos por los algoritmos M-ABC y εDEag.

5.2. Selección de problemas

Se eligieron los problemas del CEC 2010 ya que el algoritmo εDEag fue probado en dicho concurso y los resultados obtenidos por este son la base para determinar el rendimiento de MABC-CD. Es un conjunto de 18 problemas que se evalúan para 10 y 30 dimensiones (10D y 30D, respectivamente), utilizando un conjunto establecido de datos. Los problemas se describen a continuación:

C01

$$\text{mín } f(x) = -\left| \frac{\sum_{i=1}^{D} \cos^4(z_i) - 2\prod_{i=1}^{D} \cos^2(z_i)}{\sqrt{\sum_{i=1}^{D} iz_i^2}} \right| \qquad \text{z} = x - o$$

$$g_1(x) = 0.75 - \prod_{i=1}^{D} z_i \leq 0$$

$$g_2(x) = \sum_{i=1}^{D} z_i - 7.5D \leq 0$$

$$x \in [0, 10]^D$$

C02

$$\text{mín } f(x) = \text{máx}(z) \qquad \text{z} = x - o,\ y = z - 0.5$$

$$g_1(x) = 10 - \frac{1}{D}\sum_{i=1}^{D}[z_i^2 - 10\cos(2\pi z_i) + 10] \leq 0$$

$$g_2(x) = \frac{1}{D}\sum_{i=1}^{D}[z_i^2 - 10\cos(2\pi z_i)] + 10] - 15 \leq 0$$

$$h(x) = \frac{1}{D}\sum_{i=1}^{D}[y_i^2 - 10\cos(2\pi y_i) + 10] - 20 = 0$$

$$x \in [-5.12, 5.12]^D$$

C03

$$\text{mín } f(x) = \sum_{i=1}^{D-1}(100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2) \qquad \text{z} = x - o$$

$$h(x) = \sum_{i=1}^{D-1}(z_i - z_{i+1})^2 = 0$$

$$x \in [-1000, 1000]^D$$

C04

$$\text{mín } f(x) = \text{máx}(z) \qquad \text{z} = x - o$$

$$h_1(x) = \frac{1}{D}\sum_{i=1}^{D}(z_i \cos(\sqrt{|z_i|}) = 0$$

$$h_2(x) = \sum_{i=1}^{(D/2)-1}(z_i - z_{i+1})^2 = 0$$

$$h_3(x) = \sum_{i=(D/2)+1}^{D-1}(z_i^2 - z_{i+1})^2 = 0$$

$$h_4(x) = \sum_{i=1}^{D} z = 0$$

$$x \in [-50, 50]^D$$

C05

$$\text{mín } f(x) = \text{máx}(z) \qquad \text{z} = x - o$$

$$h_1(x) = \frac{1}{D}\sum_{i=1}^{D}(-z_i \sin(\sqrt{|z_i|}) = 0$$

$$h_2(x) = \frac{1}{D}\sum_{i=1}^{D}(-z_i \cos(0.5\sqrt{|z_i|}) = 0$$

$$x \in [-600, 600]^D$$

C06

$$\text{mín } f(x) = \text{máx}(z)$$

$$\text{z} = x - o,\ y = (x + 483.6106156535 - o)M - 483.6106156535$$

$$h_1(x) = \frac{1}{D}\sum_{i=1}^{D}(-y\sin(\sqrt{|y_i|})) = 0$$

$$h_2(x) = \frac{1}{D}\sum_{i=1}^{D}(-y\cos(\sqrt{|y_i|})) = 0$$

$$x \in [-600, 600]^D$$

C07

$$\text{mín } f(x) = \sum i = 1^{D-1}(100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2)$$

$$\text{z} = x + 1 - o,\ y = x - o$$

$$g(x) = 0.5 - \exp(-0.1\sqrt{\frac{1}{D}\sum_{i=1}^{D} y_i^2}) - 3\exp(\frac{1}{D}\sum_{i=1}^{D}\cos(0.1y)) + \exp(1) \leq 0$$

$$x \in [-140, 140]^D$$

C08

$$\text{mín } f(x) = \sum_{i=1}^{D-1}(100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2)$$

$$\text{z} = x + 1 - o,\ y = (x - o)M$$

$$g(x) = 0.5 - \exp(-0.1\sqrt{\frac{1}{D}\sum_{i=1}^{D} y_i^2}) - 3\exp(\frac{1}{D}\sum_{i=1}^{D}\cos(0.1y)) + \exp(1) \leq 0$$

$x \in [-140, 140]^D$

C09

$$\text{mín } f(x) = \sum_{i=1}^{D-1}(100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2)$$

$$\text{z} = x + 1 - o,\ y = x - o$$

$$h(x) = \sum_{i=1}^{D}(y\sin(\sqrt{|y_i|})) = 0$$

$$x \in [-500, 500]^D$$

C10

$$\text{mín } f(x) = \sum_{i=1}^{D-1}(100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2)$$

$$\text{z} = x + 1 - o,\ y = (x - o)M$$

$$h(x) = \sum_{i=1}^{D}(y\sin(\sqrt{|y_i|})) = 0$$

$$x \in [-500, 500]^D$$

C11

$$\begin{aligned}
\text{mín } f(x) =& \frac{1}{D}\sum_{i=1}^{D}(-z_i \cos(2\sqrt{|z_i|})) \\
& \text{z} = (x-o)M,\ y = x+1-o \\
& h(x) = \sum_{i}^{D-1}(100(y_i^2 - y_{i+i})^2 + (y_i - 1)^2) \\
& x \in [-100, 100]^D
\end{aligned}$$

C12

$$\begin{aligned}
\text{mín } f(x) =& \sum_{i=1}^{D}(z_i \sin(\sqrt{|z_i|})) \qquad \text{z} = x - o \\
& h(x) = \sum_{i=1}^{D-1}(z_i^2 - z_{i+i})^2 \\
& g(x) = \sum_{i=1}^{D}(z - 100\cos(0.1z) + 10) \leq 0 \\
& x \in [-1000, 1000]^D
\end{aligned}$$

C13

$$\begin{aligned}
\text{mín } f(x) =& \frac{1}{D}\sum_{i=1}^{D}(-z \sin(\sqrt{|z_i|})) \qquad \text{z} = x - o \\
& g_1(x) = -50 + \frac{1}{100D}\sum_{i}^{D} z_i^2 \leq 0 \\
& g_2(x) = \frac{50}{D}\sum_{i=1}^{D}\sin(\frac{1}{50}\pi z) \leq 0
\end{aligned}$$

$$g_3(x) = 75 - 50(\sum_{i=1}^{D} \frac{z_i^2}{4000} - \prod_{i=1}^{D} \cos(\frac{z_i}{\sqrt{i}}) + 1) \leq 0$$

$$x \in [-500, 500]^D$$

C14

$$\text{mín } f(x) = \sum_{i=1}^{D-1} (100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2)$$

$$\text{z} = x + 1 - o,\ y = x - o$$

$$g_1(x) = \sum_{i=1}^{D} (-y_i \cos(\sqrt{|y_i|})) - D \leq 0$$

$$g_2(x) = \sum_{i=1}^{D} (y_i \cos(\sqrt{|y_i|})) - D \leq 0$$

$$g_3(x) = \sum_{i=1}^{D} (y_i \sin(\sqrt{|y_i|})) - 10D \leq 0$$

$$x \in [-1000, 1000]^D$$

C15

$$\text{mín } f(x) = \sum_{i=1}^{D-1} (100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2)$$

$$\text{z} = x + 1 - o,\ y = (x - o)M$$

$$g_1(x) = \sum_{i=1}^{D} (-y_i \cos(\sqrt{|y_i|})) - D \leq 0$$

$$g_2(x) = \sum_{i=1}^{D} (y_i \cos(\sqrt{|y_i|})) - D \leq 0$$

$$g_3(x) = \sum_{i=1}^{D}(y_i \sin(\sqrt{|y_i|})) - 10D \leq 0$$

$$x \in [-1000, 1000]^D$$

C16

$$\text{mín } f(x) = \sum_{i=1}^{D} \frac{z_i^2}{4000} - \prod_{i=1}^{D} \cos(\frac{z_i}{\sqrt{i}}) + 1$$

$$\text{z} = x - o$$

$$g_1(x) = \sum_{i=1}^{D}[z_i^2 - 100\cos(\pi z_i) + 10] \leq 0$$

$$g_2(x) = \prod_{i=1}^{D} z_i \leq 0$$

$$h_1(x) = \sum_{i=1}^{D}(z_i \sin(\sqrt{|z_i|})) = 0$$

$$h_2(x) = \sum_{i=1}^{D}(-z_i \sin(\sqrt{|z_i|})) = 0$$

$$x \in [-10, 10]^D$$

C17

$$\text{mín } f(x) = \sum_{i=1}^{D-1}(z_i - z_{i+1})^2 \qquad \text{z} = x - o$$

$$g_1(x) = \prod_{i=1}^{D} z_i \leq 0$$

$$g_2(x) = \sum_{i=1}^{D} z_i - 7.5D \leq 0$$

$$h(x) = \sum_{i=1}^{D}(z_i \sin(4\sqrt{|z_i|})) = 0$$

$$x \in [-10, 10]^D$$

C18

$$\text{mín } f(x) = \sum_{i=1}^{D-1}(z_i - z_{i+1})^2 \qquad z = x - o$$

$$g(x) = \frac{1}{D}\sum_{i=1}^{D}(-z_i \sin(\sqrt{|z_i|})) \leq 0$$

$$h(x) = \frac{1}{D}\sum_{i=1}^{D}(z_i \sin(\sqrt{|z_i|})) \leq 0$$

$$x \in [-50, 50]^D$$

Problema	Rango de Búsqueda	Tipo de objetivo	Número de restricciones		Región factible ρ	
			E	*I*	*10D*	*30D*
C01	$[0, 10]^D$	No Separable [a]	0	2 No separables	0.997689	1.000000
C02	$[-5.12, 5.12]^D$	Separable [b]	1, separable	2 No separables	0.000000	0.000000
C03	$[-1000, 1000]^D$	No Separable	1 Separables	0	0.000000	0.000000
C04	$[-50, 50]^D$	Separable	2 No Separables, 2 separables	0	0.000000	0.000000
C05	$[-600, 600]^D$	Separable	2 separables	0	0.000000	0.000000
C06	$[-600, 600]^D$	Separable	2 rotadas	0	0.000000	0.000000
C07	$[-140, 140]^D$	No Separable	0	1 separable	0.505123	0.503725
C08	$[-140, 140]^D$	No Separable	0	1 rotada	0.379512	0.375278
C09	$[-500, 500]^D$	No Separable	1 separable	0	0.000000	0.000000
C10	$[-500, 500]^D$	No Separable	1 rotada	0	0.000000	0.000000
C11	$[-100, 100]^D$	Rotada	1 no separable	0	0.000000	0.000000
C12	$[-1000, 1000]^D$	Separable	1 no separable	1 separable	0.000000	0.000000
C13	$[-500, 500]^D$	Separable	0	2 separables, 1 no separable	0.000000	0.000000
C14	$[-1000, 1000]^D$	No Separable	0	3 separables	0.003112	0.006123
C15	$[-1000, 1000]^D$	No Separable	0	3 rotadas	0.003210	0.006023
C16	$[-10, 10]^D$	No Separable	2 separables	1 separable, 2 No separable	0.000000	0.000000
C17	$[-10, 10]^D$	No Separable	1 separable	2 no separables	0.000000	0.000000
C18	$[-50, 50]^D$	No Separable	1 separable	1 separable	0.000010	0.000000

Tabla 5.1: Características de los problemas de prueba

[a] Función donde alguno de sus parámetros x_i no son independientes.
[b] Función de muchas variables que puede ser reformulada como la suma de funciones de una sola variable

En la tabla 5.1 se muestran las características de los problemas de prueba. Entre los problemas se encuentran algunos de naturaleza separable y no separable. Se dice que una función de n variables es separable si puede ser reformulada como la suma de n funciones de solo una variable. Si una función $f(x)$ es separable, sus parámetros x_i se llaman independientes. Una función no separable $f(x)$ es llamada m-no separable si m parámetros son no separables. Si ninguno de sus parámetros es independiente, se dice que la función es totalmente no separable.

5.3. Calibración de parámetros

Para fines de uniformidad y apego a los términos del concurso CEC 2010 para la evaluación de algoritmos, se decidió calibrar los parámetros utilizados por el algoritmo y así tener un conjunto único de valores de estos que serían utilizados para todos los problemas a evaluar.

La biblioteca **SPOT** para el lenguaje estadístico R[1] realiza una serie de evaluaciones en un algoritmo dado, aunque su limitante reside en que sólo obtiene los valores óptimos de los parámetros para un problema en particular. El paquete SPOT fue utilizado en un principio para calibrar los parámetros del programa, no obstante eventualmente se decidió optar por el paquete **irace**[2]. El paquete irace permite encontrar configuraciones de parámetros para diferentes instancias de un problema.

Si bien el algoritmo MABC-CD se probó con un conjunto de 18 problemas, para poder lograr la calibración con irace se utilizó un parámetro adicional, que indica el problema a utilizar en la ejecución del algoritmo. Además, se normalizó la salida de los datos (que es la información que irace usa para realizar la calibración) utilizando el valor de la diferencia entre el resultado obtenido por εDEag y MABC-CD.

La configuración encontrada por irace y que fue utilizada para la ejecución del código se muestra en la tabla 5.2.

[1] disponible en http://cran.r-project.org/web/packages/SPOT/index.html
[2] disponible en http://cran.r-project.org/web/packages/irace/index.html

Parámetro	Valor
SN	19
$limit$	307
MR	0.415642
dec	1.375443

Tabla 5.2: Configuración de parámetros para MABC-CD

5.4. Implementación del algoritmo

El algoritmo se implementó utilizando el lenguaje Java versión 7. Las clases que se utilizaron para el funcionamiento del algoritmo son:

- **MABC-CD:** Implementación del algoritmo principal
- **LocalSearchTest:** Implementación del algoritmo de Powell, junto con Bounding Phase y Búsqueda por Sección Áurea
- **MVector:** Clase con funciones para realizar operaciones con vectores
- **Benchmarks:** Implementación de todos los problemas (C01 a C18) a evaluar.

5.5. Criterios de comparación

Para fines de comparación de los resultados del algoritmo, se decidió en primer lugar comparar contra el algoritmo M-ABC original y después contra el algoritmo ϵDEag. Los aspectos en los que se compara el algoritmo son:

- Mejor resultado
- Peor resultado
- Media

- Desviación Estandar

Los valores de la media y desviación estandar indicarán la solidez de los resultados y la capacidad del algoritmo para encontrar soluciones factibles.

5.6. Descripción de las pruebas

Las pruebas se realizaron una vez que se tuvo el conjunto de parámetros óptimos proporcionados por irace. Cabe mencionar que se realizaron diversas ejecuciones de la calibración.

Para cada problema, se ejecutó el algoritmo 30 veces. De acuerdo a los lineamientos del CEC 2010, se registran los resultados del algoritmo de la siguiente manera:

- Para 10 dimensiones (10D):
    1. Después de 20,000 llamadas a la función.
    2. Después de 100,000 llamadas a la función.
    3. Después de 200,000 llamadas a la función.
- Para 30 dimensiones (30D):
    1. Después de 60,000 llamadas a la función.
    2. Después de 300,000 llamadas a la función.
    3. Después de 600,000 llamadas a la función.

5.7. Resultados

A continuación se presentan las tablas de resultados obtenidos por el algoritmo MABC-CD, comparándolos con los algoritmos M-ABC y εDEag. En las tablas se muestra el valor promedio (media) obtenido por cada problema en las diferentes marcas especificadas por los

lineamientos del concurso. Asimismo se muestra el ranking para cada algoritmo, siendo 1 el valor para el mejor resultado y 3 para el peor. Los valores indicados como NA representan que no se pudo encontrar una solución factible.

Problema	**MABC**	**MABC-CD**	**εDEag**	R_{mabc}	$R_{mabc-cd}$	R_{edeag}
C01	-0.739726	-0.742386	-0.746199	3	2	1
C02	3.492609	3.537670	-1.827410	2	3	1
C03	13,231,430,000,000	13,854,000,000,000	88.597930	2	3	1
C04	4.475471	4.019270	24.17510	3	2	1
C05	410.98790	407.8030	-449.2350	3	2	1
C06	479.23730	420.6260	-504.0510	3	2	1
C07	11.974160	21.643130	6.973119	2	3	1
C08	163.777900	412.684000	39.398580	2	3	1
C09	8696190000000	9429670000000	832.205600	2	3	1
C10	10,528,360,000,000	10,162,500,000,000	1473.15	2	3	1
C11	-0.380279	-12.1160	0.062741	3	2	1
C12	8.654658	-611.930	-879.6310	3	2	1
C13	-66.5150	-59.4970	-6.35490	1	2	3
C14	4233900000000.000000	104130000000000.000000	220.900000	3	2	1
C15	91407000000000.000000	92444000000000.000000	935.190000	2	3	1
C16	1.040000	1.040000	0.512380	2	2	1
C17	511.000000	288.000000	0.592800	3	2	1
C18	12400.000000	9250.000000	11.485000	3	2	1
			Promedio	2.44444	2.33333	1.11111

Tabla 5.3: Resultados para 10D (20,000 evaluaciones)

5.8. Análisis

5.8.1. Análisis estadístico

El uso de análisis estadístico mediante pruebas no parametrizadas para comparar los resultados obtenidos de dos o más algoritmos en un mismo conjunto de problemas es una técnica que ha ganado popularidad en los últimos años. El uso de estas herramientas estadísticas es importante al reportar hallazgos válidos e imparciales de una investigación.

En el ámbito de los algoritmos de inteligencia colectiva, existen dos pruebas que son las más populares para comparar el desempeño entre algoritmos: la prueba de Friedman y la prueba de Bergmann-Hommel.

Problema	MABC	MABC-CD	εDEag	R_{mabc}	$R_{mabc-cd}$	R_{edeag}
c01	-0.74389	-0.74310	-0.747040	2	3	1
C02	3.5324480	3.668850	-2.258060	2	3	1
C03	12,888,580,000,000	20,950,000,000,000	0.00000000000016840370	2	3	1
C04	NA	NA	0.00000000000003291390	2	3	1
C05	NA	NA	0.0000390144	3	2	1
C06	NA	NA	-578.651	3	2	1
C07	3.553624	5.34443	0.00000000000000013226	2	3	1
C08	73.22267	22.3297	6.727555	3	2	1
C09	13,615,520,000,000	6,768,760,000,000	0.00000000000000562574	3	2	1
C10	11,098,990,000,000	8,704,640,000,000	0.00000000000016754800	3	2	1
C11	NA	NA	0.06274059	2	3	1
C12	6.908797	-653.58	-879.631	3	2	1
C13	-66.8090	-66.344	-67.476	3	2	1
C14	4931300	480,200,000,000	0.00000000000006144900	3	2	1
C15	31,930,000,000,000	45,440,000,000,000	0.179930	2	3	1
C16	1.04	1.04	0.51238	2.5	2.5	1
C17	404	288	0.5928	3	2	1
C18	11900	12200	0.00000000000199740000	2	3	1
			Promedio	2.416667	2.472222	1.111111

Tabla 5.4: Resultados para 10D (100,000 evaluaciones)

Problema	MABC	MABC-CD	εDEag	R_{mabc}	$R_{mabc-cd}$	R_{edeag}
c01	-0.743894	-0.74311	-0.74704	2	3	1
C02	3.00692	3.60083	-2.25887	2	3	1
C03	1,287,605,000,0000	11,569,500,000,000	0.00000	3	2	1
C04	NA	NA	-0.00000991845	2	3	1
C05	NA	NA	-483.61	3	2	1
C06	NA	NA	-578.652	3	2	1
C07	1.812159	3.76247	0.000000	2	3	1
C08	73.17343	3.53335	6.727528	3	2	1
C09	7,414,562,000,000	5,990,780,000,000	0.000000	3	2	1
C10	3.002471	9,394,770,000,000	0.000000	2	3	1
C11	NA	NA	-0.00152271	2	3	1
C12	5.974666	-456.44 -336.734	3	1	2	
C13	-66.8	-68.057	-68.429	3	2	1
C14	4931300	1559700000	0.00000	3	2	1
C15	17,701,000,000,000	26,359,000,000,000	0.1799	2	3	1
C16	1.04	1.04	0.37021	2.5	2.5	1
C17	434	610	0.12496	2	3	1
C18	9700	10800	0.00000000000000000097	2	3	1
			Promedio	2.361111	2.305556	1.333333

Tabla 5.5: Resultados para 10D (200,000 evaluaciones)

Problema	MABC	MABC-CD	εDEag	R_{mabc}	$R_{mabc-cd}$	R_{edeag}
c01	-0.7743368	-0.78227	-0.70496	2	1	3
C02	4.20451	4.20451	-2.14878	2.5	2.5	1
C03	NA	279,253,000,000,000	56,724,800	3	2	1
C04	12.0881	37.687	18.9792	3	2	1
C05	NA	NA	-379.851	3	2	1
C06	NA	NA	-504.051	3	2	1
C07	95.7603	1662.23	3886.22	1	3	2
C08	1549.88	136144	6.16037	2	3	1
C09	NA	NA	10.7676	3	2	1
C10	NA	NA	4639140	3	2	1
C11	NA	NA	0.204744	2	3	1
C12	NA	-2284.1	-374.338	3	1	2
C13	-61.0979	-40.2945	-2.07793	1	2	3
C14	7,521,840,000,000	75,418,300,000,000	17,489,700	2	3	1
C15	266,401,000,000,000	247,732,000,000,000	82,420,900	3	2	1
C16	1.1507	1.14641	0.225134	3	2	1
C17	1459.83	1001.27	7.53478	3	2	1
C18	33953.7	36359.4	87.5457	2	3	1
			Promedio	2.361111	2.083333	1.555556

Tabla 5.6: Resultados para 30D (60,000 evaluaciones)

Problema	MABC	MABC-CD	εDEag	R_{mabc}	$R_{mabc-cd}$	R_{edeag}
C01	-0.778774036	-0.81402	-0.81738	3	2	1
C02	4.20451	4.20451	-2.14878	2.5	2.5	1
C03	NA	27.9253	30.0961	3	2	1
C04	NA	37.687	0.362055	3	2	1
C05	NA	NA	-448.375	2	3	1
C06	NA	NA	-578.651	3	2	1
C07	38.1727	53.5885	2.36724	2	3	1
C08	1549.88	136144	6.16037	2	3	1
C09	NA	19,660,000,000,000	10.7676	3	2	1
C10	59,869,900,000,000	31,110,300,000,000	33.2618	3	2	1
C11	NA	-3.96453	0.00131151	3	1	2
C12	NA	-2284.1	-461.82	3	1	2
C13	-63.5875	-51.6734	-58.132	1	3	2
C14	833,509	12,931,100,000,000	7.32786	2	3	1
C15	13.4007	190943000000000	21.6193	2	3	1
C16	1.14651	1.13759	0.00000000000000011768	2	3	1
C17	1459.83	1631.72	6.32649	2	3	1
C18	33953.7	36359.4	87.5457	2	3	1
			Promedio	2.361111	2.361111	1.277778

Tabla 5.7: Resultados para 30D (300,000 evaluaciones)

Problema	MABC	MABC-CD	εDEag	R_{mabc}	$R_{mabc-cd}$	R_{edeag}
C01	-0.7787795	-0.81474	-0.82087	3	2	1
C02	3.92151	3.92151	-2.15142	2.5	2.5	1
C03	NA	279,253,000,000,000	28.8379	3	2	1
C04	NA	37.687	0.00816297	3	2	1
C05	NA	NA	-449.546	3	2	1
C06	NA	NA	-578.652	3	2	1
C07	3.743	35.69060	0.00000000000000260363	2	3	1
C08	1244.22	44.4089	0.00000000000007831460	3	2	1
C09	NA	19660000000000	10.7214	3	2	1
C10	49,728,600,000,000	25,006,800,000,000	33.2618	3	2	1
C11	NA	-3.96453	-0.000286388	3	2	1
c12	NA	-913.301	356.2330	3	1	2
C13	-63.5877	-60.6272	-65.3531	2	3	1
C14	833455	1433740000000	0.00000000000030894100	3	2	1
C15	81484400000000	159557000000000	21.6038	3	2	1
C16	1.133890	1.138870	0.00000000	2.5	2.5	1
C17	1459.830	1631.720	6.326490	2	3	1
C18	29217.90	33811.400	87.54570	2	3	1
			Promedio	2.416667	2.416667	1.166667

Tabla 5.8: Resultados para 30D (600,000 evaluaciones)

El test de Friedman trabaja asignando rankings r_{ij} a los resultados obtenidos por cada algoritmo j en cada problema i. Estos rankings se asignan de forma ascendente. El test de Friedman no identifica las diferencias existentes entre el mejor algoritmo encontrado y el resto, se limita a detectar la existencia o no de diferencias en todo el conjunto de resultados [DGMH12]. Si el test encuentra diferencias (basadas en un p-valor obtenido a partir del ranking promedio), es necesario aplicar un procedimiento post-hoc de identificación de diferencias. En este caso se utiliza la prueba de Bergmann-Hommel como procedimiento post-hoc.

5.8.2. Descripción de los resultados

Al ejecutar la prueba de Bergmann-Hommel con los resultados obtenidos por la prueba de Friedman es posible ver que entre los algoritmos M-ABC y MABC-CD no existe diferencia estadística puesto que el p-valor encontrado es menor a 0.05. Al comparar ambos algoritmos contra el algoritmo εDEag, se encuentra que existe diferencia estadística, como se aprecia en la tabla 5.9, tabla 5.10, tabla 5.11, tabla 5.12, tabla 5.13, y tabla 5.14.

Comparación	No ajustado	Bergmann-Hommel
M-ABC vs MABC-CD	$8.68E-1$	0.8676323348
M-ABC vs εDEAG	$8.98E-5$	**0.0001331828**
MABC-CD vs εDEAG	$4.4E-5$	**0.0001331828**

Tabla 5.9: Resultados de la prueba Bergmann-Hommel para 10D en 20,000 evaluaciones

Comparación	No ajustado	Bergmann-Hommel
M-ABC vs MABC-CD	$8.676323e-01$	0.8676323348
M-ABC vs εDEAG	$8.978175e-05$	**0.0001331828**
MABC-CD vs εDEAG	$4.439427e-05$	**0.0001331828**

Tabla 5.10: Resultados de la prueba Bergmann-Hommel para 10D en 100,000 evaluaciones

Comparación	No ajustado	Bergmann-Hommel
M-ABC vs MABC-CD	0.867632335	0.867632335
M-ABC vs εDEAG	0.002046957	**0.006140872**
MABC-CD vs εDEAG	0.003537936	**0.006140872**

Tabla 5.11: Resultados de la prueba Bergmann-Hommel para 10D en 200,000 evaluaciones

Comparación	No ajustado	Bergmann-Hommel
M-ABC vs MABC-CD	0.4046568	0.4046568
M-ABC vs εDEAG	0.01566335	**0.04699006**
MABC-CD vs εDEAG	0.11334551	**0.11334551**

Tabla 5.12: Resultados de la prueba Bergmann-Hommel para 30D en 60,000 evaluaciones

Comparación	No ajustado	Bergmann-Hommel
M-ABC vs MABC-CD	1.00	1.00
M-ABC vs εDEAG	0.00115405	**0.00346215**
MABC-CD vs εDEAG	0.00115405	**0.00346215**

Tabla 5.13: Resultados de la prueba Bergmann-Hommel para 30D en 300,000 evaluaciones

Comparación	No ajustado	Bergmann-Hommel
M-ABC vs MABC-CD	1.00	1.00
M-ABC vs εDEAG	0.0001768346	**0.0005305037**
MABC-CD vs εDEAG	0.0001768346	**0.0005305037**

Tabla 5.14: Resultados de la prueba Bergmann-Hommel para 30D en 600,000 evaluaciones

5.9. Comentarios finales

Como fue posible apreciar en los resultados arrojados por las pruebas de Friedman y Bergmann-Hommel, los algoritmos M-ABC y MABC-CD tienen un desempeño muy similar con los problemas del CEC 2010. Al compararse estos algoritmos contra el algoritmo εDEag se observa que el valor arrojado por la prueba de Bergmann-Hommel es prácticamente idéntico, confirmando lo anterior. Cabe mencionar que el algoritmo MABC-CD logró superar al algoritmo M-ABC en varias de las pruebas de 10D y en una prueba de 30D (C12), teniendo un desempeño casi idéntico en las demás pruebas de 30D.

Conclusiones

En este trabajo de tesis se propuso la utilización del método de direcciones conjugadas de Powell como operador de búsqueda local en el algoritmo M-ABC con el objetivo de mejorar los resultados obtenidos por dicho método, utilizando como funciones de prueba las presentadas en el CEC 2010, las cuales son 18 diferentes y se evalúan utilizando 10 y 30 variables, esto para ser comparados con los resultados obtenidos por el algoritmo εDEag, el cual utiliza funciones basadas en gradiente, lo cual hace que su implementación en código y su ejecución impliquen una gran complejidad computacional. Al analizar los resultados se puede concluir que este método es capaz de llegar a resultados comparables con los del εDEag en algunos de los problemas, sin embargo en otros tantos no es capaz de llegar a resultados satisfactorios. Teniendo en cuenta las limitantes de tiempo a las que estuvo sujeto este proyecto no fue posible realizar un análisis a fondo para determinar si dicho rezago en el desempeño tiene que ver con la naturaleza de los problemas de prueba (funciones separables y no separables).

A lo largo de este trabajo de tesis es posible apreciar la importancia que tiene el desarrollo de algoritmos de optimización y el gran impacto que tiene en esta área la investigación y la creación de métodos basados en procesos provenientes de la naturaleza puesto que éstos han demostrado tener un gran desempeño en la resolución de problemas variados y poder ser adaptados a problemas específicos sin necesidad de modificar radicalmente su estructura. Entre estos problemas específicos se encuentran aquéllos en la categoría NP-Completo (que no pueden resolverse en tiempo polinómico) los cuales son ampliamente estudiados y tienen gran importancia dentro del área de la computación ya que existe la teoría que establece que de ser posible resolver alguno de ellos con un método particular,

se habrá encontrado la solución a todos los problemas de esta categoría además de aquellos que pueden resolverse en tiempo polinómico (categoría P), llevando a la resolución de diversas problemáticas de importancia general para la sociedad.

Como trabajo futuro para este proyecto se propone realizar un análisis más profundo del desempeño de este algoritmo con problemas de prueba de tipo separable y no separable, además de revisar la literatura existente para buscar otro método de búsqueda local que pueda resultar más adecuado.

Bibliografía

[Abb01] H. A. Abbass. Mbo: marriage in honey bees optimization-a haplometrosis polygynous swarming approach. In *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546)*, volume 1, pages 207–214 vol. 1, 2001. 43

[AMZ10] R. Akbari, A. Mohammadi, and K. Ziarati. A novel bee swarm optimization algorithm for numerical function optimization. *Communications in Nonlinear Science and Numerical Simulation*, 15(10):3142–3155, 2010. 42

[BA14] S. Babaeizadeh and R. Ahmad. A modified artificial bee colony algorithm for constrained optimization problems. *Journal of Convergence Information Technology*, 9(6):151, 2014. 45

[Bra15] I. Brajevic. Crossover-based artificial bee colony algorithm for constrained optimization problems. *Neural Computing and Applications*, 26(7):1587–1601, 2015. 45

[BSJ13] J. Bansal, H. Sharma, and S. Jadon. Artificial bee colony algorithm: a survey. *International Journal of Advanced Intelligence Paradigms*, 5(1-2):123–159, 2013. 47

[BSS06] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty. *Nonlinear Programming: Theory and Algorithms*. Wiley-Interscience, 2006. 14

[BT12] N. Bacanin and M. Tuba. Artificial bee colony (abc) algorithm for constrained optimization improved with genetic operators. *Studies in Informatics and Control*, 2012. 45

[BTS10] I. Brajevic, M. Tuba, and M. Subotic. Improved artificial bee colony algorithm for constrained problems. In *Proceedings of the 11th WSEAS international conference on nural networks and 11th WSEAS international conference on evolutionary computing and 11th WSEAS international conference on Fuzzy systems*, pages 185–190. World Scientific and Engineering Academy and Society (WSEAS), 2010. 45

[Cet09] O. Cetina. Una adaptación del comportamiento de la abeja exploradora en el algoritmo de la colonia artificial de abejas para resolver problemas de optimización con restricciones. Master's thesis, Laboratorio Nacional de Informática Avanzada, Xalapa, Mexico, 2009. 31

[CM12] O. Cetina and E. Mezura. Empirical analysis of a modified artificial bee colony for constrained numerical optimization. *Applied Mathematics and Computation*, 2012. 4, 45, 46, 51

[Coe02] C.A. Coello. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Computer methods in applied mechanics and engineering*, 191(11):1245–1287, 2002. 29

[Daw76] R. Dawkins. *The Selfish Gene*. Oxford University Press, 1976. 25

[DDC99] M. Dorigo and G. Di Caro. Ant algorithms for discrete optimization. *Artificial Life*, 1999. 28

[Deb00] K. Deb. An efficient constraint handling method for genetic algorithms. *Computher methods in applied mechanics and engineering*, 2000. 30

[Deb04] K. Deb. *Optimization for Engineering Design*. PHI Learning, 2004. 10, 16, 17, 55, 56

[DGMH12] J. Derrac, S. Garcıa, D. Molina, and F. Herrera. Un tutorial sobre el uso de test estadısticos no paramétricos en comparaciones múltiples de metaheurısticas y algoritmos evolutivos. In *Memorias del VIII Congreso Español Sobre Metauristicas, Algoritmos Evolutivos y Bioinspirados*, pages 455–462. UCLM, 2012. 78

[Dor92] M. Dorigo. *Optimization, Learning and Natural Algorithms*. PhD thesis, Politecnico di Milano, 1992. 3, 27

[GL99] F. Glover and M. Laguna. Tabu search. In D Du and P. Pardalos, editors, *Handbook of Combinatorial Optimization*, pages 2093–2229. Springer US, 1999. 24

[Gor12] L. A. Gordían. Implementación de un algoritmo multioperador basado en evolución diferencial para optimización numérica con restricciones. Master's thesis, Laboratorio Nacional de Informática Avanzada, Xalapa, Mexico, 2012. 7

[HAM06] O.B. Haddad, A. Afshar, and M. A. Mariño. Honey-bees mating optimization (hbmo) algorithm: A new heuristic approach for water resources optimization. *Water Resources Management*, 2006. 32, 43

[HM07] O. Haddad and M. Mariño. Dynamic penalty function as a strategy in solving water resources combinatorial optimization problems with honey-bee mating optimization (hbmo) algorithm. *Journal of Hydroinformatics*, 9(3):233–250, 2007. 45

[Hu12] Y. Hu. Swarm intelligence. 2012. Disponible en http://guava.physics.uiuc.edu/~nigel/courses/569/Essays_Fall2012/Files/Hu.pdf. 27

[HZGN15] Y. Huo, Y. Zhuang, J. Gu, and S. Ni. Elite-guided multi-objective artificial bee colony algorithm. *Appl. Soft Comput.*, 32(C):199–210, July 2015. 45

[JKK15] J. Jayanth, S. Koliwad, and T. Kumar. Classification of remote sensed data using artificial bee colony algorithm. *The Egyptian Journal of Remote Sensing and Space Science*, 2015. 47

[Jun03] S. Jung. Queen-bee evolution for genetic algorithms. *Electronics letters*, 39(6):575–576, 2003. 43

[KA09a] D. Karaboga and B. Akay. A comparative study of artificial bee colony algorithm. *Applied Mathematics and Computation*, 214(1):108–132, 2009. 35

[KA09b] D. Karaboga and B. Akay. A survey: algorithms simulating bee swarm intelligence. *Artificial Intelligence Review*, 31(1-4):61–85, 2009. 33

[KA11] D. Karaboga and B. Akay. A modified artificial bee colony (abc) algorithm for constrained optimization problems. *Applied Soft Computing*, 11(3):3021–3031, 2011. 44

[Kar04] A. Karcı. *Imitation of Bee Reproduction as a Crossover Operator in Genetic Algorithms*, pages 1015–1016. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004. 44

[Kar05] D. Karaboga. An idea based on honey bee swarm for numerical optimization. Technical report, Erciyes University, 2005. 3, 28, 40

[KB07] D. Karaboga and B. Basturk. Artificial bee colony (abc) optimization algorithm for solving constrained optimization problems. In *Foundations of Fuzzy Logic and Soft Computing*, pages 789–798. Springer, 2007. 44

[KE95] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942–1948 vol.4, Nov 1995. 3, 28

[KGV83] S. Kirkpatrick, C. D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *Science*, 1983. 24

[Koc10] E. Koc. *The Bees Algorithm: Theory, Improvements and Applications*. PhD thesis, University of Wales, Cardiff, 2010. 37, 38, 39, 42

[Law76] E. L. Lawler. *Combinatorial Optimization: Networks and Matroids*. University of California at Berkley, 1976. 10

[LC07] H. Lu and W. Chen. Self-adaptive velocity particle swarm optimization for solving constrained optimization problems. *Journal of Global Optimization*, 41(3):427–445, 2007. 52

[LdJTN07] N. Lemmens, S. de Jong, K. Tuyls, and Now. N.é. Bee system with inhibition pheromones. In *Proceedings of NiSIS 2007*, 2007. B-Paper (Originally published at ECCS 2007). 39

[LF08] P. Lalbakhsh and M.N. Fesharaki. Basic concepts and anatomy of swarm intelligence ande its roles in today and future network centric environments. In *Proceedings of the International MultiConference of Engineers and Computer Scientists*, 2008. 26

[LKAdJS07] N. Lemmens, Tuyls K., Nowe A., and de Jong S. Bee behaviour in multi-agent systems: a bee foraging algorithm. In *Proceedings of the 7th ALAMAS Symposium*, pages 126–138, 2007. 38

[LT01] P. Lucic and T. Teodorovic. Bee system: Modeling combinatorial optimization transportation engineering problems by swarm intelligence. *Preprints of the Tristan IV Triennial Symposium on Transportation Analysis*, 2001. 37, 38

[LY14] X. Li and M. Yin. Self-adaptive constrained artificial bee colony for constrained numerical optimization. *Neural Computing and Applications*, 24(3-4):723–734, 2014. 45

[Mar03] R. Martı. Procedimientos metaheurısticos en optimización combinatoria. *Matemátiques, Universidad de Valencia*, 1(1):3–62, 2003. 20

[MC09] E. Mezura and O. Cetina. Exploring promising regions of the search space with the scout bee in the artificial bee colony for constrained optimization. In *Intelligent Engineering Systems through Artificial Neural Networks*. ASME Press, 2009. 44

[MC11] E. Mezura and C.A. Coello. Constraint-handling in nature-inspired numerical optimization: past, present and future. *Swarm and Evolutionary Computation*, 1(4):173–194, 2011. 30, 44

[MCH10] E. Mezura, O. Cetina, and B. Hernández. *Nuevas Heuristicas Inspiradas en la Naturaleza para Optimizacion Numerica*. IPN, 2010. 48

[MDC10] E. Mezura, M. Damián, and O. Cetina. Smart flight and dynamic tolerances in the artificial bee colony for constrained optimization. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1–8. IEEE, 2010. 31, 45

[Mit96] M. Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, 1996. 25

[MLC09] M. Muñoz, J. López, and E. Caicedo. An artificial beehive algorithm for continuous optimization. *International Journal of Intelligent Systems*, 24(11):1080–1093, 2009. 42

[MS10] R. Mallipeddi and P.N. Suganthan. Problem definitions and evaluation criteria for the cec 2010 competition on constrained real-parameter optimization. Technical report, Nanyang Technological University, 2010. 63

[MV10] E. Mezura and R.E. Velez. Elitist artificial bee colony for constrained real-parameter optimization. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1–8. IEEE, 2010. 45

[Nat11] G. Nates. Genética del comportamiento: Abejas como modelo. *Acta Biológica Colombiana*, 16(3):213–229, 2011. 32

[NM65] J. A. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7(4):308–313, 1965. 18

[NW99] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, 1999. 8, 9, 10, 12, 15

[OL96] I. H. Osman and G. Laporte. Metaheuristics: A bibliography. *Annals of Operational Research*, 1996. 21

[PC09] D. Pham and M. Castellani. The bees algorithm: modelling foraging behaviour to solve continuous optimization problems. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 223(12):2919–2938, 2009. 42

[Pol45] G. Polya. *How to Solve It.* Stanford University, 1945. 2

[Pow64] M.J.D. Powell. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The computer journal*, 7(2):155–162, 1964. 4, 55

[Ros60] H. H. Rosenbrock. An automatic method for finding the greatest or least value of a function. *The Computer Journal*, 3(3):175–184, 1960. 4

[RRR06] A. Ravindran, K. M. Ragsdell, and G. V. Reklaitis. *Engineering Optimization: Methods and Applications.* Wiley, 2006. 12

[RY00] T. Runarsson and X. Yao. Stochastic ranking for constrained evolutionary optimization. *IEEE Transactions on Evolutionary Computation*, 4:284–294, 2000. 30

[SEW07] A. Saremi, T.Y. Elmekkawy, and G.G. Wang. Tuning the parameters of a memetic algorithm to solve vehicle routing problem with backhauls using design of experiments. *International Journal of Operations Research*, 2007. 25

[SH97] T. Sato and M. Hagiwara. Bee system: finding solution by a concentrated search. In *Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on*, pages 3954–3959 vol.4, 1997. 37

[SP95] R. Storn and K. Price. Differential evolution - a simple and efficient adaptive scheme for global optimization over continous spaces. Technical report, International Computer Science Institute, 1995. 3, 26

[Sub11] M. Subotic. Artificial bee colony algorithm with multiple onlookers for constrained optimization problems. In *Proc. of the European Computing Conference*, pages 251–256, 2011. 45

[Tal09] E. Talbi. *Metaheuristics: From Design to Implementation.* Wiley, 2009. 16, 21, 23, 24

[TBS11] M. Tuba, N. Bacanin, and N. Stanarevic. Guided artificial bee colony algorithm. In *Proceedings of the 5th European Conference on European Computing Conference*, ECC'11, pages 398–403, Stevens Point, Wisconsin, USA, 2011. World Scientific and Engineering Academy and Society (WSEAS). 45

[TL05] V. Tereshko and A. Loengarov. Collective decision-making in honey bee foraging dynamics. *Computing and Information Systems Journal, ISSN 1352-9404*, 2005. 35

[TLMDO06] D. Teodorovic, P. Lucic, G. Markovic, and M. Dell' Orco. Bee colony optimization: Principles and applications. In *Neural Network Applications in Electrical Engineering, 2006. NEUREL 2006. 8th Seminar on*, Sept 2006. 41

[TS10] T. Takahama and S. Sakai. Constrained optimization by the ϵ constrained differential evolution with an archive and gradient-based mutation. In *CEC IEEE*, 2010. 4, 59, 60

[TSI05] T. Takahama, S. Sakai, and N. Iwane. Constrained optimization by the $\varepsilon\mathbb{0}$ constrained hybrid algorithm of particle swarm optimization and genetic algorithm. In S. Zhang and R. Jarvis, editors, *AI 2005: Advances in Artificial Intelligence*, volume 3809 of *Lecture Notes in Computer Science*, pages 389–400. Springer Berlin Heidelberg, 2005. 31

[WFP+05] H.F. Wedde, M. Farooq, T. Pannenbaecker, B. Vogel, C. Mueller, J. Meth, and R. Jeruschkat. Beeadhoc: An energy efficient routing algorithm for mobile ad hoc networks inspired by bee behavior. In *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation*, GECCO '05, pages 153–160, New York, NY, USA, 2005. ACM. 41

[WFZ04] H-F. Wedde, M. Farooq, and Y. Zhang. Beehive: An efficient fault-tolerant routing algorithm inspired by honey bee behavior. In M. Dorigo, M. Birattari, C. Blum, L. Gambardella, F. Mondada, and T. Stützle, editors, *Ant Colony Optimization and Swarm Intelligence*, volume 3172 of *Lecture Notes in Computer Science*, pages 83–94. Springer Berlin Heidelberg, 2004. 41

[WLC08] L.P. Wong, M. Y. H. Low, and C. S. Chong. A bee colony optimization algorithm for traveling salesman problem. In *2008 Second Asia International Conference on Modelling x00026; Simulation (AMS)*, pages 818–823, May 2008. 41

[WM97] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Optimization*, 1997. 3

[XG14] B. Xing and W. Gao. *Innovative computational intelligence: a rough guide to 134 clever algorithms*. Springer, 2014. 34, 37

[Yan05] X. Yang. Engineering optimizations via nature-inspired virtual bee algorithms. In *Proceedings of the First International Work-conference on the Interplay Between Natural and Artificial Computation Conference on Artificial Intelligence and Knowledge Engineering Applications: A Bioinspired Approach - Volume Part II*, IWINAC'05. Springer-Verlag, 2005. 38

[ZS09] J. Zhang and A. Sanderson. Jade: Adaptive differential evolution with optional external archive. *Trans. Evol. Comp*, 13(5):945–958, October 2009. 59