UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Diseño y desarrollo de un método heurístico basado en un sistema socio-cultural de creatividad para la resolución de problemas de optimización no lineales y diseño de zonas electorales

Dr. Román Anselmo Mora Gutiérrez

Dr. Javier Ramírez Rodríguez
Asesor

UNIVERIDAD NACIONAL AUTÓNOMA DE MÉXICO

PROGRAMA DE MAESTRÍA Y DOCTORADO EN INGENIERÍA

FACULTAD DE INGENIERÍA

DISEÑO Y DESARROLLO DE UN MÉTODO HEURÍSTICO

BASADO EN UN SISTEMA SOCIO-CULTURAL DE CREATIVIDAD

PARA LA RESOLUCI´ON DE PROBLEMAS DE OPTIMIZACIÓN

CONTINUOS NO LINEALES y DISE˜NO DE ZONAS ELECTORALES

T E S IS

QUE PARA OPTAR POR EL GRADO DE:
DOCTOR EN INGENIERÍA
SISTEMAS-INVESTIGACIÓN DE OPERACIONES

PRESENTA:

M. EN I. ROMAN ANSELMO MORA GUTIÉRREZ

TUTOR:

DR. JAVIER RAMÍREZ RODRÍGUEZ

UNIVERSIDAD AUTÓNOMA METROPOLITANA

UNIDAD AZCAPOTZALCO

DEPARTAMENTO DE SISTEMAS

México DF febrero de 2013

Jurado Asignado

Presidente	Dra. Angélica del Rocío Lozano Cuevas
Vocal	Dra. Idalia Flores de la Mota
Secretario	Dr. Javier Ramírez Rodríguez
Suplente	Dr. Miguel Ángel Gutiérrez Andrade
Suplente	Dr. Carlos Gersheson Gracía

Lugar donde se realizó la tesis:

Facultad de Ingeniería, UNAM.

TUTOR DE TESIS:

Dr. JAVIER RAMÍREZ RODRÍGUEZ

FIRMA

Dedico esta Tesis :

A mis padres.

A mis hemanos, sobrinos, cuñada y cuñados.

A mis abuelos.

A toda mi famila.

A mis amigos .

"... La música es así, remedio de los males,
inagotable fuente a escanciar cada día;
sosiego de palacios, templanza de arrabales,
y placidez del alma, armonizante guía. ..."

Fragmento del poema Música de Marilina Rébora

Agradecimientos

A la Universidad Nacional Autónoma de México (UNAM); al programa de posgrado en ingeniería y al Consejo Nacional de Ciencia y Tecnología (CONACyT); por darme la oportunidad de seguir formándome como profesionista.

A los miembros de mi comité tutorial Dra. Idalia Flores de la Mota; Dra. Angélica del Rocío Lozano Cuevas; Dr. Carlos Gershenson García; Dr. Miguel Ángel Gutiérrez Andrade y Dr. Javier Ramírez Rodríguez por todo el apoyo, orientación, consejos, recomendaciones, ayuda y confianza que me brindaron durante mi trabajo doctoral.

Al Dr. Javier Ramírez-Rodríguez; Dr. Eric Alfredo Rincón-García; Dr. Antonin Ponsich; Dra. Ma. Elena Larraga y Lic. Denisse Ventre por creer en este proyecto; por su ayuda permanente e incondicional, pero sobre todo por su paciencia y amistad.

A los profesores del posgrado en ingeniería de la UNAM y a los profesores del departamento de Sistemas de la UAM-Azcapotzalco por toda la ayuda brindada y el conocimiento compartido.

A mis padres y abuelos por su amor, consejos, paciencia y comprensión que desde niño me brindaron; por guiar mi camino y estar junto a mi en todo momento. A mis hermanos, sobrinos, primos y tíos, por su compañia, palabras de aliento y en especial por ser siempre mí apoyo.

A todos mis amigos por cada instante compartido; en especial a Emilio, Mariela, Lidia, Toño y a mis compañeros de EMCI.

A mis alumnos.

Les doy las gracias

Atentamente:

Roman A. Mora Gutiérrez

Resumen

En este trabajo, se presenta una nueva metaheurística cultural, la cual imita proceso de composición musical dentro de un sistema de creatividad (socio-cultural y personal), por lo que se le llamó Método de Composición Musical o MMC.

El método propuesto se utilizó para resolver instancias referenciales de problemas de optimización. Los problemas de prueba utilizados en este trabajo fueron problema de programación no lineal (no restringida y limitada) y problema de zonificación electoral.

Los resultados numéricos, muestran que el MMC posee un buen comportamiento para resolver casos de estos problemas. Además, el análisis estadístico de los resultados mostró cuáles son las ventajas y desventajas de la metaheurística propuesta. En otras palabras, el MMC es una metaheurística competitiva para resolver las instancias de estos tipos de problemas de optimización.

Palabras clave: Metaheurísticas; algoritmos sociales; sistema socio-cultural de creatividad; sistema personal de creatividad ; composición musical

Abstract

In this paper, we presented a new cultural metaheuristic, which mimics process of music composition within of a creativity system (socio-cultural and personal), so we have it called Method of Musical Composition or MMC.

Our method was used to solve a set reference instances of optimization problems. Test problems, in this paper, were non-linear problem (unconstrained and constrained) and districting problem.

Numerical results, that our algorithm was able to achieve, show to MMC has a good behavior to solve instances of these problems. Also, statistical analysis on results has shown which are advantages and disadvantages of our metaheuristic versus other metaheuristics. In other words the MMC is a competitive metaheuristic to solve instances of these kinds optimization problems.

Keywords: Optimization, Metaheuristics, Social algorithms, Socio-cultural system of creativity and Personal system of creativity and Musical composition

Índice general

Índice de tablas

Índice de figuras

Introducción

El conocimiento es dinámico; por ende, ningún concepto es absoluto e incuestionable; por el contrario, todo los conceptos siempre son incompletos, y pueden ser mejorados e incluso cambiados. A lo largo del tiempo, cualquier producto del razonamiento cambiará, con base en el desarrollo tecnológico, social y económico.

La labor de un investigador debe ser la creación, búsqueda, innovación y desarrollo de conocimientos, métodos y técnicas más adecuados para hacer frente a los problemas propios de cada época.

Hoy en día, los profesionistas y los tomadores de decisiones se enfrentan con problemas de creciente complejidad, los cuales emergen en diversas áreas del conocimiento; como: biocomputación, logística, procesamiento de imágenes, lingüística, diseño de sistemas mecánicos, entre otros. Un mecanismo mediante el cual estos problemas pueden ser abordados y resueltos es la optimización.

La optimización es un área del conocimiento que surgió durante la segunda guerra mundial, la cual se integra por: teorías, técnicas y algoritmos necesarios al plantear un problema; elegir los objetivos de un sistema; sintetizar un sistema; analizar las alternativas de un sistema y seleccionar la mejor alternativa posible [99]. A continuación, se formaliza el concepto de optimización.

Definición 1 *La **optimización** es el área de conocimiento concerniente al desarrollo, analisis y perfeccionamiento de métodos y técnicas para la resolución de problemas matemáticos que posean la forma siguiente [180]:*

$$\text{máx(mín)} f(x)$$

sujeto a:

$$x \in F \subseteq S$$

$$S \subseteq \mathbb{R}^n$$

donde: x denota el conjunto de las n variables de decisión del modelo; S es el espacio de búsqueda, el cual corresponde al conjunto de soluciones o configuraciones posibles [173]; f(x) es la función

objetivo que es un instrumento para evaluar a cada solución candidata; F es el conjunto de soluciones factibles, el cual es un subconjunto de S cuyos elementos satisfacen una serie de restricciones impuestas por el problema [173]; $\mathbb{R}$ *conjunto de números reales.*

Además, en la optimización se incluyen los procedimientos implicados en la la caracterización y análisis de las soluciones encontradas.

Existe una gran cantidad de situaciones en los sistemas sociales y naturales, que involucran el uso de problemas de optimización, como son: diseño de redes de trasporte y rutas de envío; programación y asignación de personal a estaciones de trabajo; la predicción de la estructura de las proteínas; análisis filogenético, entre otros.

Debe distinguirse entre un **problema de optimización** y una **instancia de un problema de optimización**. De manera informal, una instancia es una colección de datos e información sobre una situación de interés expresados como un modelo de programación matemática; del cual, se generará una solución particular. En contraste, un problema es una generalización de una colección de instancias. Algunas definiciones de estos conceptos son:

Definición 2 *Una* ***instancia de un problema de optimización*** *es representada por (F,f),donde: F es el conjunto de soluciones factibles y f es la función objetivo [179].En el caso de minimización, el mapeo f:*$F \to \mathbb{R}$ *involucra buscar una solución* $x^\star \in F$*, tal que:* $f(x^\star) \leq f(x)$ *para toda* $x \in F$*. Mientras que en el caso de maximización, el mapeo f:*$F \to \mathbb{R}$ *involucra buscar una solución* $x^\star \in F$*,tal que* $f(x^\star) \geq f(x)$ *para toda* $x \in F$*.*

Definición 3 *Un* ***problema de optimización*** *es un conjunto* $\boldsymbol{I}$ *de instancias de un problema de optimización [179]*

La tarea esencial de las herramientas de optimización consiste en localizar un punto meta (aquel que ofrece el mejor resultado con base al criterio de decisión) dentro del espacio de soluciones [139]. Por ende, solucionar una instancia de optimización es encontrar una configuración de valores de las variables dentro de las restricciones especificadas y bajo las condiciones no controladas relevantes [2], que genere el mejor valor de acuerdo con el criterio de decisión. A dicha configuración se le denomina como **solución optima** (x^*). No existe un algoritmo universal para la optimización, sino,

numerosos procedimientos cada uno de los cuales se ha desarrollado para abordar una clase específica de problemas; por ende, al elegir un procedimiento para resolver alguna instancia se deben considerar las características e información disponible sobre el problema que se desea resolver.

Los procedimientos de optimización se clasifican en: a) **métodos exactos** (también conocidos como métodos analíticos) y b) **métodos heurísticos** de optimización. Los **métodos exactos** son aquellos procedimientos que analizan todo el espacio de búsqueda (de manera explícita o implícita), además de garantizar el encontrar la solución óptima de la instancia. En contraste los **métodos heurísticos** de optimización son aquellos procedimientos que encuentran soluciones de alta calidad con un costo computacional razonable. Los métodos heurísticos se pueden dividir en las siguientes categorías: b.1) algoritmos aproximados son aquellos que localizan una solución dentro un radio de aproximación $p(n)$ de la solución optima; b.2) heurísticas son aquellos procedimientos para resolver un problema de optimización bien definido mediante una aproximación intuitiva, en la que se utiliza el conocimiento sobre un problema de manera inteligente para obtener una buena solución; y b.3) las metaheurísticas (tambíen llamadas heurísticas modernas) utilizan mecanismos ingeniosos de diversificación e intensificación para la exploración del espacio de búsqueda [210].

Los métodos heurísticos surgen por la necesidad de resolver problemas con un alto grado de complejidad (NP-completos), ya que para resolver cualquier instancia de esta clase de problemas no se cuenta con un procedimiento de optimización clásica; o bien, su implementación es muy costosa (tiempo y/o recursos computacionales). Por lo tanto, en las últimas décadas se han generado y adaptado un gran número de técnicas heurísticas, como son: recocido simulado (SA) [119], búsqueda tabú (TS) [87, 88], algoritmos genéticos (AG) [104], algoritmos culturales, optimización por nube de partículas (PSO), optimización por colonia de hormigas (ACO) [60], búsqueda armónica (HS) [80], entre otros. Cabe mencionar, que cada uno de estos procedimientos presenta ventajas y desventajas frente a cualquier otro al ser utilizados para resolver alguna instancia de optimización.

Crear, modificar y adaptar un algoritmo heurístico para resolver un subconjunto de instancias de optimización son acciones que obedecen a la ambición científica de alcanzar mejores resultados que los obtenidos hasta el momento por los procedimientos disponibles para dicho subconjunto de casos de optimización. Debe considerarse, al realizar alguno de los procesos anteriores, que cualquier

mejora en el rendimiento de una heurística al resolver un subconjunto de problemas es exactamente igual a la disminución del rendimiento de dicha heurística sobre otro subconjunto de problemas [249].

La creación de un nuevo método heurístico es un proceso creativo, en donde se conjuntan conocimiento, ideas y experiencia de un individuo (o grupo de individuos) a fin de construir, introducir, adecuar, ordenar una serie de conjeturas, conceptos y estrategias; los cuales, se estructuran dentro de una técnica capaz de solucionar algún subconjunto de instancias de optimización obteniendo buenos resultados.

En la presente investigación se desarrolla un nuevo procedimiento metaheurístico basado en un sistema de creatividad socio-cultural que imita el proceso de composición musical, al cual se le ha llamado "Método de composición musical"(MMC por las siglas en inglés de *method of musical composition*). Lo cual implicó generar, adaptar y modificar: conceptos, ideas, preceptos y deducciones que vinculan el conocimiento sobre creatividad, sistemas multi-agente, composición musical, optimización y métodos heurísticos.

Esta investigación surgió tras el análisis de mejoras posibles al algoritmo híbrido entre SA y HS, desarrollado en la investigación de maestría [158]. El objetivo de la presente investigación es: *crear un método heurístico que emula el proceso de composición musical dentro de un sistema de creatividad socio-cultural.*

0.1. Justificación y alcances

La premisa inicial es: ¿Sí el conjunto de algoritmos heurísticos y metaheurísticos actuales son todos los procedimientos que se pueden desarrollar para resolver los problemas de optimización? Antes de responder esta interrogante se deben mencionar los teoremas "Nada Es Gratis (No Free Lunch o NFL)", así como sus implicaciones dentro de la optimización; pues estas ideas son la base de la diversidad de métodos que han surgido en la optimización recientemente.

Teorema 1 *Para algún par de algoritmos* a_1 *y* a_2

$$\sum_f P(d_m^y \mid f, m, a_1) = \sum_f P(d_m^y \mid f, m, a_2)$$

donde: f denota el espacio de todos los problemas posibles, a_1 y a_2 denotan los algoritmos que se comparan, m es el número de iteraciones asignado para la comparación suponiendo una no re visitación, es decir, que el número de evaluaciones de la función objetivo es dividido por el tamaño de la población, d_m^y es el conjunto ordenado de los valores de f , $P(d_m^y \mid f, m, a_1)$ es la medida de actuación del algoritmo a_1 evaluando m veces f [249].

En el teorema previo se define $P(d_m^y \mid f, m, a)$ como la probabilidad condicional de obtener una muestra particular d_m^y bajo las condiciones consideradas. La comparación entre la suma de todas las posibles f de $P(d_m^y \mid f, m, a_1)$ con respecto a todas las posibles f de $P(d_m^y \mid f, m, a_2)$, demuestra que $P(d_m^y \mid f, m, a)$ es independiente de a calculando sobre el promedio de todos los problemas posibles. En esencia este teorema demuestra que: "si todos los elementos del espacio de problemas posibles f son igualmente probables, entonces la probabilidad de observar alguna secuencia arbitraria de m en el curso de la búsqueda no depende del algoritmo de búsqueda".

Teorema 2 *Sea D_m el espacio de todas la muestras de tamaño m, es decir $d_m \in D_m$. Dados $d_m^y, D_m^y, m > 1, f$ y los algoritmos a_1 y a_2 entonces:*

$$\sum_T P(d_m^y \mid f, T, m, a_1) = \sum_T P(d_m^y \mid f, T, m, a_2)$$

y

$$\sum_T P(D_m^y \mid f, T, m, a_1) = \sum_T P(D_m^y \mid f, T, m, a_2)$$

donde: f, T denota el espacio de todos los problemas dinámicos posibles.

Si un algoritmos de búsqueda a_1 supera a otro a_2, para cierta clase de funciones dinámicas de costo; entonces, a_2 supera a a_1, en otra clase de funciones dinámicas, sobre el conjunto de todas las funciones dinámicas. En este teorema establece una sutíl dependencia de las funciones de costo en relación al tiempo [249].

Los teoremas de NFL originalmente se desarrollaron para ser aplicados dentro del aprendizaje automático supervisado de máquinas; sin embargo, después las ideas implicadas en dichos teoremas se ampliaron, lo que permitió utilizar lo enunciado por los teoremas NFL para diseñar, analizar y explicar a los algoritmos de búsqueda y optimización.

Teorema 3 *Todos los algoritmos de búsqueda u optimización, si son evaluados sin remplazo sobre el espacio de todos los problemas posibles presentarán el mismo rendimiento. Por ende, para cualquier algoritmo de optimización o búsqueda a_i, cualquier mejora en el rendimiento sobre alguna clase de problemas será proporcional a la reducción del rendimiento en otra clase de problemas.*

La premisa anterior demuestra que no existe un algoritmo perfecto que resuelva bien todos los problemas; ya que, para cualquier algoritmo cualquier mejora en su utilidad para una clase de problemas, conlleva un empeoramiento en la utilidad de dicho algoritmo en alguna otra clase de problemas.

Teniendo en cuenta las proposiciones previas es posible concluir que para un conjunto de instancias de optimización es posible diseñar un algoritmo específico que genere los mejores resultados sobre dicho conglomerado y solamente se justifica utilizar un algoritmo general, sin incorporar ningún conocimiento, cuando no exista otra alternativa.

Para lograr esto, se debe incorporar en el diseño, en la construcción, en la modificación y en la adaptación en a, el conocimiento disponible sobre el conjunto de problemas a resolver. En caso contrario, es tan probable que a trabaje peor que la búsqueda aleatoria, como que trabaje mejor sobre el conjunto de instancias de interés.

Ahora bien, tras analizar la información antes mencionada es posible responder la pregunta inicial de esta sección con un NO, pues con base en el teorema NFL se puede generar un algoritmo de alto rendimiento para cada una de las instancias de los problemas de optimización. De hecho, para un gran número de instancias de optimización han sido creados y adaptados varios métodos heurísticos en los últimos años, e.g:

- Búsqueda de armonía (HS) [80]. En este artículo fue presentado el algoritmo general HS, el cual se adaptó y modificó para resolver tres instancias de problemas de optimización, las cuales fueron: una instancia de 20 ciudades del problema del agente viajero, el problema de Braken y McCormick que es una instancia del Problema de Programación No Lineal (PLN) restricto del tipo minimización con dos variables y el diseño de redes hidráulicas para la ciudad de Hanói, Vietnam.

- Optimización con enjambre de partículas (PSO) de aprendizaje global para la optimización

global de funciones multimodales introducido en [132].

- Algoritmo evolutivo que utiliza mapeo de homomorfismo para la optimización de problemas PLN restrictos [123]
- Algoritmo de sociedad y civilización [196]

En esta investigación doctoral se presenta una nueva metaheurística capaz de resolver algunas instancias de las siguientes clases de problemas de optimización: A) No lineales irrestrictos B) No lineales restrictos y C) Diseño de zonas. Esta metaheurística emula el proceso composición músical en un sistema socio-cultural de creatividad, por lo cual se le ha denominado "Método de Composición Musical" (MMC). Cabe mencionar que Z. W. Geem fue el primero en vincular los conceptos de metaheurísticas, programación matemática y música [77] al crear el método HS, el cual emula un proceso artístico dentro de un sistema personal de creatividad .

0.2. Objetivos de investigación

El objetivo general es "Diseñar, desarrollar y proponer un método heurístico capaz de resolver algunas instancias de las siguientes clases de problemas de optimización: A) No lineales irrestrictos B) No lineales restrictos y C) Diseño de zonas electorales. Dicho método imitará el proceso de composición musical dentro de un sistema socio-cultural de creatividad". Además se considerarán a las siguientes premisas como elementos fundamentales para alcanzar el objetivo plateado: i) manejo adecuado de las características de la instancia a resolver (tipo de problema, codificación de soluciones, tipo de restricciones); ii) el manejo adecuado del costo computacional implicado, ya que se desea que éste no sea excesivamente elevado.

Objetivos particulares:

1 Investigar, analizar y comprender la información disponible sobre:

[1.1] Características, definiciones, métodos de solución y problemas tipo (benchmark) de las siguientes clases de problemas de optimización: a) PLN sin restricciones B) PLN con restricciones y C) diseño de zonas.

[1.2] Sistemas creativos y el proceso de composición musical

[1.3] Sociedades artificiales y creatividad artificial.

[1.4] Metaheurísticas basadas en sistemas sociales y en sistemas creativos.

2 Concebir, adaptar y desarrollar las ideas y los conceptos necesarios para la construcción del algoritmo básico MMC.

3 Modificar y ajustar el algoritmo básico MMC, a fin de construir una versión eficiente para la solución de problemas de la clase PLN sin restricciones.

4 Resolver un conjunto de problemas tipo PLN sin restricciones, posteriormente analizar los resultados obtenidos y compararlos con las soluciones conseguidas por otras metaheurísticas.

5 Modificar y ajustar el algoritmo básico MMC, a fin de construir una versión eficiente para la solución problemas de la clase PLN con restricciones.

6 Resolver un conjunto de funciones benchmark PLN con restricciones, posteriormente analizar los resultados obtenidos y compararlos con las soluciones conseguidas por otras metaheurísticas.

7 Modificar y ajustar el algoritmo básico MMC, a fin de construir una versión eficiente para la solución del problema de diseño de zonas.

8 Resolver un conjunto de instancias del problema de diseño de zonas, posteriormente analizar los resultados obtenidos y compararlos con las soluciones conseguidas por otras metaheurísticas.

0.3. Metodología de investigación y resultados obtenidos

La generación y adaptación de una alternativa para la solución de algún dilema, implica un proceso creativo, donde se conjuntan, ajustan y desarrollan los conocimientos, ideas y experiencias de un individuo (o grupo de individuos) con el fin de estructurar una estrategia que resuelva el caso de interés.

Se utilizó como base el procedimiento propuesto por Wallas [228] [227] para la generación de algún procedimiento para solución de problemas, el cual involucra las siguientes fases: a) **preparación**: que consiste en la recolección y análisis de información sobre el problema a resolver y los métodos de solución; b) **incubación**: que involucra el proceso mental de conexión de conceptos; es decir, en esta fase se genera el *constructo*[1] sobre el problema a resolver; c) **inspiración**: que implica la "creación del proceso creativo". En otras palabras, la fase de inspiración es la unión de conceptos disjuntos a fin de hallar una táctica original para resolver el problema y d) **verificación**: es la aplicación de pruebas prácticas que permitan corroborar la validez de la táctica desarrollada.

El proceso de Wallas es recursivo, y se busca alcanzar un equilibrio entre el grado de cercanía de la solución obtenida, con respecto a los recursos utilizados para su generación. Para su ejecución se requiere de un conjunto de problemas (no triviales) que sirva como marco conceptual, los cuales deben reunir las siguientes características: A) Permitan una amplia forma de ser planteados, B) cuenten con una gran variedad de procedimientos para solucionarlos y C) opcionalmente, cuenten con un conjunto de elementos que permita la comparación entre los procedimientos de solución. Dentro de la investigación de operaciones los problemas que reúnen estas características son los del tipo NP-Duros, pues ellos han sido objeto de una extensa investigación científica.

Una analogía es una guía (a veces borrosa) dentro del proceso de diseño y desarrollo de algún método metaheurístico, la cual, facilita la concatenación, organización y sistematización de ideas y conceptos. Si bien utilizar una analogía no es indispensable, recurrir a ella favorece una visión distinta para resolver un problema, pues permite la asociación, la integración la ampliación y la innovación de conocimiento.

Para el diseño y el desarrollo del algoritmo MMC se utilizaron las similitudes entre los procesos de composición musical y de optimización. La estructura básica del algoritmo propuesto se muestra en el Algoritmo 16.

Los pasos del algoritmo MMC son clasificados dentro de las siguientes fases: a) Inicializar el proceso de optimización (incluye desde ingresar la información al algoritmo hasta el paso 4); b)intercambió de información entre los agentes (incluye a los pasos 6 y 7); c)generar una nueva melodía para cada uno de los agentes (incluye a los pasos 9 y 10); d)actualizar la obra de arte de cada

[1] Un constructor es un modo de construir o interpretar el mundo [188].

Algoritmo 1: Algoritmo MMC básico

Input: MMC Parámetros del método e información sobre la instancia a resolver

Output: Las mejores melodías generadas por los compositores

```
1  Crear una sociedad artificial con reglas de interacción entre los agentes.
2  for cada uno de los agentes de la sociedad do
3  |  Generar aleatoriamente una obra musical (Para esta actividad se considera la información
   |  sobre la instancia a resolver)
4  end
5  repeat
6  |  Actualizar la sociedad artificial.
7  |  Intercambiar información entre agentes.
8  |  for cada uno de los agentes de la sociedad do
9  |  |  Actualizar la matriz de conocimiento.
10 |  |  Generar y evaluar una nueva melodía (x_{⋆,new})
11 |  |  if x_{⋆,new} es mejor que la peor melodía (x_{x-worst}) en la obra artística del individuo
   |  |  then
12 |  |  |  Remplazar a x_{x-worst} por x_{⋆,new} en la obra de arte.
13 |  |  end
14 |  end
15 |  Construir el conjunto con las mejores soluciones (melodías).
16 until Satisfacer el criterio de paro;
```

uno de los agentes (incluye desde el paso 11 hasta el paso 13); e)construir el conjunto de soluciones (paso 15); y f)repetir hasta que se satisfaga el criterio de paro (incluye desde el paso 5 hasta el paso 16).

Parte I

Conceptos básicos

Capítulo 1

Fundamentos de Optimización

El presente capítulo tiene la finalidad de exponer y analizar algunos conceptos básicos sobre optimización; examinar las propiedades esenciales de los problemas de optimización e instancias.

Éste se encuentra dividido en dos secciones: 1.1) la optimización y 1.2) diseño de algoritmos. La información vertida en este capítulo será utilizada como base a lo largo del presente trabajo.

1.1. La optimización

La optimización es una idea fundamental en diversas disciplinas del conocimiento, como son: investigación de operaciones, administración, finanzas, telecomunicaciones... la cual se utiliza en el diseño, el análisis y toma de decisiones en sistemas[1]. Algunas definiciones del término son:

1. Luenberger precisa que la optimización es uno de los principios básicos del análisis de problemas[2] complejos de decisión, y su proceso consiste en la asignación de valores a un conjunto de variables interrelacionadas, centrando la atención en un mecanismo diseñado para cuantificar la calidad de la decisión [138].

2. Hall expresa que la optimización es lograr la mejor armonía entre el sistema y sus integrantes;

[1] Ackoff define a un sistema como un conjunto de elementos interrelacionados[3], en contraste Winston, como una organización de componentes interdependientes que trabajan juntos para alcanzar un objetivo [248].

[2] Un problema es una diferencia, desviación o un desequilibrio entre el estado real e ideal de un sistema, además de ser lo suficientemente importante para justificar su resolución.

y su proceso comprende desde el planteamiento de un problema, hasta el análisis y selección de la mejor alternativa [99].

3. La optimización es seleccionar de un conjunto de alternativas posibles a la mejor de ellas, con base en algún criterio de decisión [194].

4. Optimización es obtener la mejor solución posible de una actividad o un proceso, a través del uso adecuado de información y conocimientos disponibles.

5. La optimización (también denominada "programación matemática") es una parte de la investigación de operaciones[3], la cual trata de resolver problemas de decisión en los que se deben determinar las acciones que optimicen un determinado objetivo, pero satisfaciendo ciertas limitaciones en los recursos disponibles [215].

6. La "programación matemática" es una potente técnica de modelado usada en el proceso de toma de decisiones [26].

Basado en lo anterior, el término "optimización" se puede entender como el conjunto de conocimiento, principios, teorías, técnicas, herramientas útiles y necesarias para resolver problemas de programación matemática.

De manera general, resolver un problema es un proceso racional que involucra desde identificar el problema de interés hasta la elección y ejecución de alguna acción a fin de eliminarlo o reducirlo. Este proceso debe ser sistemático y guiado por el conocimiento disponible sobre el sistema.

Un "problema de optimización" puede ser expresado como encontrar el valor de unas variables de decisión para las que una determinada función [4] objetivo (o varias funciones objetivo) alcanza su valor máximo o mínimo, de acuerdo con las características del problema. En ocasiones el valor de las variables de decisión está sujeto a un conjunto de restricciones [142].

[3]La investigación de operaciones es una rama de las matemáticas aplicadas, que consistente en el uso del enfoque científico en la toma de decisiones con el objeto de mejorar el diseño u operación de un sistema [248]. Ackoff la define como la aplicación por grupos interdisciplinarios del método científico a problemas relacionados con el control de las organizaciones o sistemas a fin de que se produzcan soluciones que mejor sirvan a los objetivos de toda la organización [2].

[4]Una función $f : D \subseteq \mathbb{R}^n \to \mathbb{R}^m$ es cualquier criterio que a cada punto $x \in D$ le asigna un único punto $f(x) \in R^m$. El conjunto D se llama dominio de la función y $f(x)$ es la imagen de x por f.

Un "modelo" es una representación o abstracción selectiva (cuantitativa o cualitativa) de las características de un sistema. Todo problema de optimización debe ser formulado a través de un "modelo matemático"; ya que estos modelos describen de modo conciso y sin ambigüedad las relaciones o condiciones del problema a resolver por medio del lenguaje y estructuras matemáticas; lo cual permite emplear técnicas matemáticas y computacionales de alto poder, para analizar y resolver dicho problema. En el anexo A , se analiza el proceso de modelación de los problemas de optimización.

1.1.1. Propiedades de los problemas de optimización

1.1.1.1. Topológicas

La **topología** es una rama de las matemáticas, que se ocupa del estudio de aquellas propiedades de los cuerpos geométricos que permanecen invariantes; cuando dichos cuerpos son género neutro, de modo tal que, no aparezcan nuevos puntos dentro del cuerpo o se hagan coincidir puntos diferentes.Las transformaciones permitidas presuponen la existencia de una correspondencia biunívoca entre los puntos de la figura original y los de la transformada.

El estudio y análisis de la características y propiedades topológicas del espacio generado por algún problema de optimización es fundamental para la programación matemática; ya que a través de este análisis es posible determinar y utilizar, o bien diseñar y desarrollar un método eficiente en resolución de dicho problema. En el anexo B se abordan los conceptos básicos de topología; a continuación, se abordan sus implicaciones en la optimización.

A)Espacios métricos, normas y vectoriales

Un espacio métrico es un tipo especial de espacio topológico denotado por el par (X, d) (o simplemente por M), donde X es un conjunto y d es una métrica.

Definición 4 *Dado un conjunto $X \neq \varnothing$ una métrica o distancia sobre X es una función $d : X \times X \to \mathbb{R}$, si satisface:*

- *propiedad identidad: $a, b \in X$ $d(a, b) = 0$ si y sólo si $a = b$*
- *desigualdad triangular: $d(a, b) + d(a, c) = d(b, c)$ para todo $a, b, c \in X$*

- *simetría* $d(a,b) = d(b,a)$ *para cada* $a, b \in X$

[122, 211]

En el ejemplo 1.1 se muestra un espacio métrico

Ejemplo 1.1

El espacio métrico $(\mathbb{R}, d)$ *, donde la función distancia es* $d(x_1, x_2) = |x_1 - x_2|$ *para cualquier par* $x_1, x_2 \in \mathbb{R}$ *se llama la recta real.*

En el anexo C se muestran algunas de las funciones distancia más utilizadas. Toda métrica permite definir de manera natural un espacio topológico formado por las uniones arbitrarias de bolas de centro r y radio d:

$$B(x, \epsilon) \tag{1.1.1}$$

Si una topología U es inducida por una métrica d se dice que U y d son compatibles. El espacio topológico inducido por una métrica se denomina "espacio metrizable"; el cual se forma por las uniones arbitrarias de bolas de centro x, radio ϵ y se denota por $B(x, \epsilon)$.

Lema 1 *Todo espacio métrico es un espacio topológico.*

Nótese que es posible definir más de una métrica sobre un espacio, por ejemplo: en el espacio $\mathbb{R}^n$ se puede utilizar la métrica Euclidiana, la métrica Manhattan, o bien la distancia de Chebychev.

Definición 5 *Sea* (X, d) *un espacio métrico,* $x \in X$ *y* $\epsilon > 0$*. El conjunto* $B(x, \epsilon) = \{y \in X : d(x, y) < \epsilon\}$ *se llama bola abierta centrada en* x *con radio* ϵ*. El conjunto* $B[x, \epsilon] = \{y \in X : d(x, y) \leq \epsilon\}$ *se llama la bola cerrada centrada en* x *con radio* ϵ *y al conjunto* $S(x, \epsilon) = \{y | d(x, y) = \epsilon\}$ *se le denomina circunferencia de radio* ϵ *y centro* x*.*

En el conjunto $\mathbb{R}$, la bola abierta centrada en un numero real x con radio ϵ es un intervalo abierto $(x - \epsilon, x + \epsilon)$. En la Figura 1.1, se muestran bolas abiertas de radio 1 centradas en el origen por las métricas ρ_1, ρ_2 y ρ_3 definidas sobre $\mathbb{R}^2$.

Definición 6 *Sea* $X \subset M$ *tal que para todo* $x \in X$ *existe una bola abierta que contiene a* x*; entonces se dice que* X *es un subconjunto abierto de* M*.*

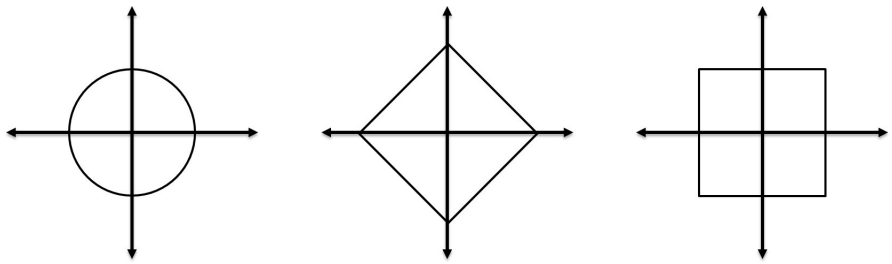

Figura 1.1: Bolas abiertas generadas por tres métricas distintas.

Definición 7 *Sea $X^c \subset M$; se dice que: X es un subconjunto cerrado si su complemento X^c es un subconjunto abierto de M.*

Teorema 4 *Sea (X, d) un espacio métrico y sea U el conjunto de todas las bolas abiertas en X (ver definición 6), entonces $((X), U)$ es el espacio topológico inducido por el espacio métrico.*

Sin embargo, existen espacios topológicos que no pueden ser inducidos por ningún espacio métrico [122].

Proposición 1 *Dado (X, d) un espacio métrico, $x_1, x_2 \in X$ y $\epsilon_1, \epsilon_2 \in \mathbb{R}^+$. Si existe $x_3 \in B(x_1, \epsilon_1) \cap B(x_2, \epsilon_2)$, entonces existe un $\epsilon_3 > 0$ tal que $B(x_3, \epsilon_3) \subset B(x_1, \epsilon_1) \cap B(x_2, \epsilon_2)$ (ver Figura 1.2)*

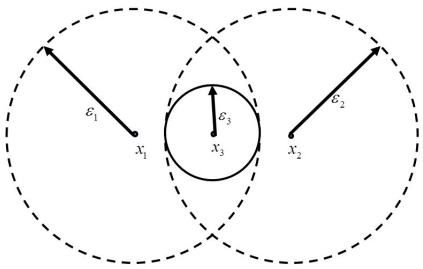

Figura 1.2: $B(x_3, \epsilon_3) \subset B(x_1, \epsilon_1) \cap B(x_2, \epsilon_2)$

En los espacios métricos es posible utilizar el concepto de "cercanía"; como por ejemplo: dado un punto de referencia x_3 y los puntos x_1 y x_2, es posible determinar si la distancia entre x_3 y x_1 es menor o no, a la distancia entre x_3 y x_2; si $d(x_3, x_1) < d(x_3, x_2)$, entonces se dice: el punto x_1 es más

cercano a x_3 en comparación al punto x_2. Decir que un punto está tan cerca de otro como queramos, significa que los puntos están a una distancia menor que un número positivo fijado con anterioridad.

Definición 8 *Sea X un espacio métrico, $x \in X$. Un subconjunto V de X es una **vecindad** de x, si existe $\epsilon > 0$, tal que $B(x, \epsilon) \subset V$ y se denota con $V(x)$ al conjunto de todas las vecindades del punto x (ver Figura 1.3)*

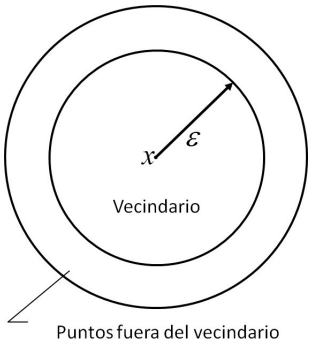

Figura 1.3: Idea de vecindario

Definición 9 *Un espacio vectorial sobre $\mathbb{K}$ es un conjunto $X \neq \varnothing$, dotado con las aplicaciones:*

$$+ : X \times X \to X \qquad : \mathbb{K} \times X \to X$$
$$(x_1, x_2) \mapsto x_1 + x_2 \quad (\alpha, x) \mapsto \alpha x$$

tal que para todos $x_1, x_2, x_3 \in X$ y todo $\alpha_1, \alpha_2 \in \mathbb{K}$ se cumplen los siguientes axiomas:

$x_1 + (x_2 + x_3) = (x_1 + x_2) + x_3$

$x_1 + x_2 = x_2 + x_1$

Existe un elemento en X denominado elemento neutro tal que $O + x_1 = x_1$

Para cada elemento $x \in X$ existe un elemento $-x \in X$ tal que $x + (-x) = O$

$\alpha_1(x_1 + x_2) = \alpha_1 x_1 + \alpha_1 x_2$

$(\alpha_1 + \alpha_2)x_1 = \alpha_1 x_1 + \alpha_2 x_1$

$(\alpha_1 \alpha_2)x_1 = \alpha_1(\alpha_2 x_1)$

$x_1 = 1x_1$

Sea U una topología sobre un espacio vectorial X tal que:

- Cada punto de X es un punto cerrado.
- Las operaciones de espacio vectorial son continuas respecto de X.

Entonces, se dice que U es una **topología vectorial** sobre X , y que X es un espacio vectorial topológico; el cual se denota con (X, U), y es normable si existe una norma sobre X tal que la métrica d inducida por $\|\|$ es compatible con U.

Definición 10 *Dado un espacio vectorial X. Una norma para X es una función $\|\| : X \to \mathbb{R}$ que a cada vector $x \in X$ se le asocia el número real $\|x\|$ con las siguientes propiedades para $x_1, x_2 \in X$ y $\lambda \in \mathbb{R}$*

$$\|x\| \geq 0$$
$$\|x\| = 0 \Leftrightarrow x \text{ es el vector cero}$$
$$\|\lambda x\| = |\lambda|\|x\|$$
$$\|x_1 + x_2\| \leq \|x_1\| + \|x_2\|$$

Un espacio normado es un par $(X, \|\|)$ donde X es un espacio vectorial sobre $\mathbb{K}$ y $\|\|$ es una norma sobre X.

Lema 2 *Todas las topologías en $\mathbb{R}^n$ generadas por normas definen los mismos abiertos.*

B) Conjuntos Compactos, Conexos, Convexos y No-convexos

Un espacio compacto puede ser entendido como: un espacio con propiedades similares a las de un conjunto finito. A continuación, se formaliza el concepto de espacios compactos.

Definición 11 *Se dice que X es compacto si para todo cubrimiento por conjuntos abiertos de $X(\bigcup_{i\in I} O_i \supseteq X)$ existe un subcubrimiento finito.*

Definición 12 *Un conjunto X es compacto, si y sólo si, toda sucesión de elementos de X tiene una subsucesión convergente.*

Generalmente, se dice que un **conjunto X es convexo** si cualquier par de puntos en X se unen por un segmento lineal l, el cual esta totalmente incluido en X; en contraste, se dice que X **es no-convexo** si $l \notin X$. En otras palabras, X es convexo si la combinación lineal de cualquier par de puntos en X queda contenida en X. De otra manera, X es un conjunto no convexo [59]. En la Figura 1.4, se muestran algunos conjuntos convexos y no-convexos en $\mathbb{R}^2$. A continuación, se

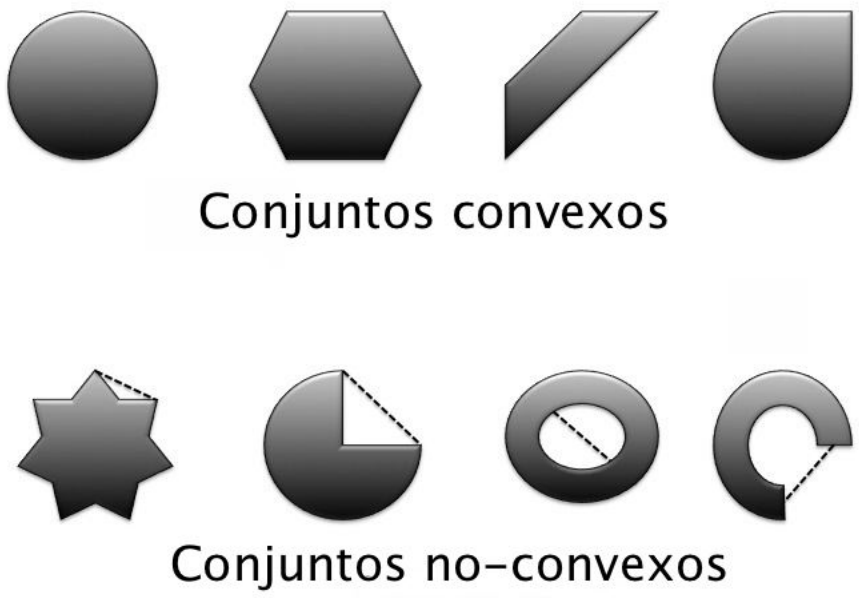

Figura 1.4: Conjuntos Convexos y No-convexos

define formalmente conjunto convexo, así como otras ideas relacionadas que serán utilizadas en los siguientes capítulos.

Definición 13 *Un conjunto $\mathcal{F} \subseteq \mathbb{R}^n$ es un convexo si para todo $x_i, x_j \in \mathcal{F}$ y $\lambda \in [0,1]$ se satisface que:*

$$\lambda xi + (1-\lambda)x_j \in \mathcal{F} \tag{1.1.2}$$

[59, 121].

Una ecuación equivalente es: dado un par de puntos cualquiera x_i, x_j el conjunto de puntos que

satisface el segmento rectilíneo que los une también esta $\mathcal{F}$

$$f(\lambda xi + (1-\lambda)x_j) \leq \lambda f(xi) + (1-\lambda)f(x_j) \tag{1.1.3}$$

[34]

Definición 14 *Un elemento* $x' \in$ *es una combinación lineal convexa de* $x_1, x_2, \ldots, x_k \in \mathbb{R}^n$ *si se satisface el siguiente sistema de ecuaciones lineales:*

$$\begin{gathered} x' = \sum_{i=1}^{k} \lambda_i x_i \\ \sum_{i=1}^{k} \lambda_i = 1 \\ \lambda_i \geq 0 \text{ para todo } i = 1, 2, \ldots, k \end{gathered} \tag{1.1.4}$$

[59, 121].

Topológicamente, un conjunto conexo es un subconjunto de un espacio topológico (X, U) tal que no puede ser descrito como la unión disjunta de dos conjuntos abiertos de la topología U. Es decir, el conjunto X es conexo, si y sólo si, los únicos conjuntos de U abiertos y cerrados a la vez son los conjuntos triviales.

Definición 15 *Un conjunto* X *es conexo, si y sólo si, cada vez que* $X \subset A \cup B$ *siendo* A *y* B *conjuntos abiertos y* $A \cap B = \varnothing$ *entonces* $A = X$ *y* $B = \varnothing$ *o bien* $A = \varnothing$ *y* $B = X$.

Proposición 2 *Los conjuntos convexos gozan de las siguientes propiedades [179, 71, 155]:*

- *El* $\varnothing$ *es un conjunto convexo.*
- *Un punto es un conjunto convexo.*
- *Un segmento de línea recta es conjunto convexo pues satisface la siguiente condición* $x \in \mathbb{R}^n / x = \lambda x_i + (1-\lambda)x_j; x_i, x_2 \in \mathbb{R}^n; \lambda \in [0, 1]$.
- *El conjunto* $\mathbb{R}^n$ *es conjunto convexo.*

- *La intersección finita o infinita de conjuntos convexos es un conjunto convexo.*
- *La unión de conjuntos convexos en general, no tiene que ser un conjunto convexo.*
- *La combinación lineal de conjuntos convexos es un conjunto convexo.*
- *Si A y B son conjuntos convexos y $k \in \mathbb{R}$ entonces los siguientes conjuntos también son convexos :*
  - $A \times B = \{\bar{x} \in \mathbb{R}^{2n} / \bar{x} = (\bar{a}, \bar{b}), \bar{a} \in A, \bar{b} \in B\}$.
  - $A + B = \{\bar{x} \in \mathbb{R}^{2n} / \bar{x} = (\bar{a}, \bar{b}), \bar{a} \in A, \bar{b} \in B\}$.
  - $k * A = \{\bar{x} \in \mathbb{R}^{n} / \bar{x} = k * \bar{a}, \bar{a} \in A\}$.

En los teoremas de Heine-Borel y selección de Blaschke se combinan los conceptos de *conjuntos compactos y conexos.*

Teorema 5 (Teorema de Heine-Borel) *Si un conjunto $X \subset \mathbb{R}^n$ tiene alguna de las siguientes propiedades, entonces poseerá también las otras dos:*

- *X es cerrado y conexo.*
- *X es conexo*
- *Todo subconjunto infinito de X tiene un punto de acumulación en la frontera de X.*

Teorema 6 (Selección de Blaschke) *De cualquier sucesión uniformemente acotada de conjuntos convexos compactos se puede extraer una subsucesión convergente a un conjunto convexo y compacto.*

Definición 16 *Un hiperplano $h(X)$ es un subespacio de $\mathbb{R}^n$, el cual, satisface la siguiente ecuación [193]:*

$$h(X) = \sum_{i=1}^{n} a_i x_i = b$$

donde: $h(X) \in \mathbb{R}^n$, $a_i \in \mathbb{R}^n$, $a_i \neq 0$, $b \in \mathbb{R}$.

Teorema 7 *Un hiperplano es un conjunto convexo [191]*

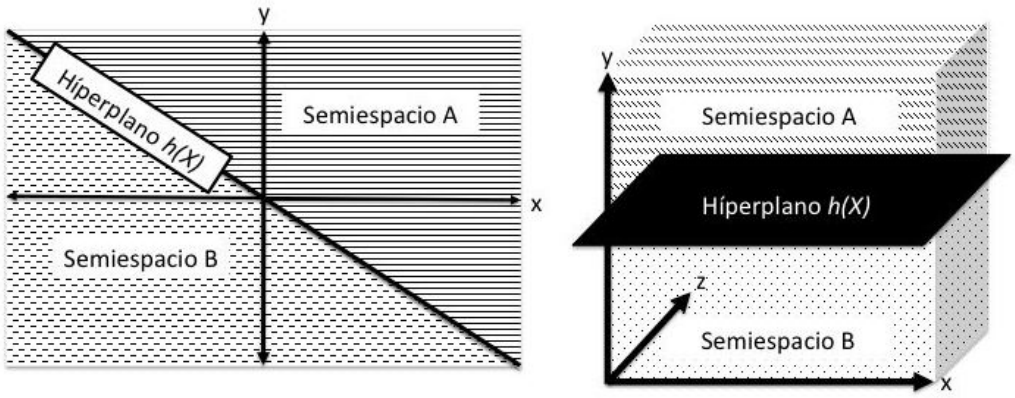

Figura 1.5: Ejemplos de hiperplano y semiespacios

Un hiperplano divide a $\mathbb{R}^n$ en dos regiones denominadas **semiespacios**. En la Figura 1.5, se ejemplifican las ideas de hiperplano y semiespacio en $\mathbb{R}^2$ y en $\mathbb{R}^3$. S

Definición 17 *Un semiespacio $\mathcal{F}$ es un conjunto de punto, el cual satisface [14]:*

$$\begin{aligned} \mathcal{F} &= \left\{x \in \mathbb{R}^n : a^T x \leq b\right\} \\ o\ bien: \mathcal{F} &= \left\{x \in \mathbb{R}^n : a^T x \geq b\right\} \end{aligned} \tag{1.1.5}$$

Si un semiespacio $\mathcal{F}$ satisface la ecuación 1.1.5 como una desigualdad estricta, entonces se dice $\mathcal{F}$ es un **espacio abierto**. En otro caso se dice que $\mathcal{F}$ es un **espacio cerrado**.

Teorema 8 *Un semiespacio es un conjunto convexo [191].*

La envoltura convexa o casco convexo de X _ representada como $conv(X)$_ es el conjunto que contiene a todas las combinaciones lineales convexas de los puntos $x_i, x_j \in X$ [59, 121]. En la Figura 1.6, se muestra un ejemplo de una envoltura convexa para nueve puntos en $\mathbb{R}^2$

Definición 18 $conv(X) = \left\{\sum_{i=1}^{k} \alpha_i x_i | x_i \in X, \alpha_i \geq 0\ \forall i = 1, 2, \ldots, k,\ \ \sum_{i=1}^{k} \alpha_i = 1, k \in \mathbb{R}\right\}$.

Teorema 9 *El $conv(X)$ de un conjunto X es el menor conjunto convexo que contiene a ese conjunto [191].*

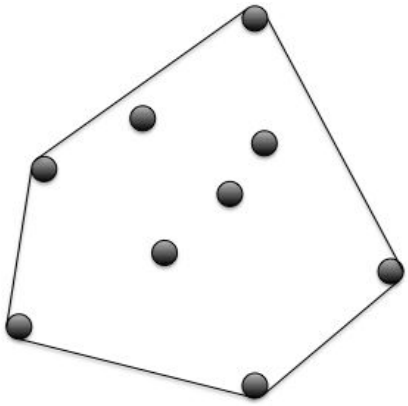

Figura 1.6: Ejemplo de envoltura convexa

Una clase muy importante de los conjuntos convexos es la de conos convexos [14]. De manera informal, se define a un **cono convexo** como un conjunto convexo cuyos elementos son todos los segmentos de línea que salen del origen. En la Figura 1.7 se muestra un ejemplo de cono convexo.

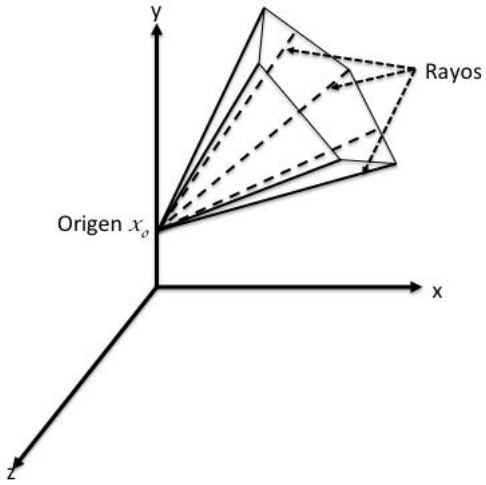

Figura 1.7: Ejemplo de cono convexo

A continuación, se formaliza la definición de cono convexo:

Definición 19 *Si un conjunto convexo $\mathcal{C}$ satisface: $\lambda x \in \mathcal{C}$ para todo $x \in \mathcal{C}$ y todo $\lambda \geq 0$, entonces se dice que $\mathcal{C}$ es un cono convexo*

Un **rayo** es una colección de puntos que satisface $x_0 + \lambda d : \lambda \geq 0$, donde d es un vector distinto de cero al cual se le denomina **dirección del rayo**. Un **conjunto poliédrico** o **poliedro** $\mathcal{P}$ es la intersección de un número finito de semiespacios; a un conjunto poliédrico acotado se le denomina **politopo** [14]. En la Figura 1.8 se muestra un ejemplo de politopo en $\mathbb{R}^3$.

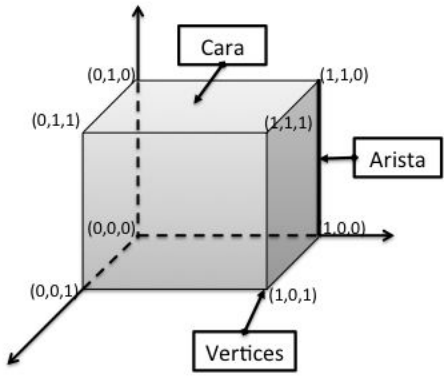

Figura 1.8: ejemplo de politopo en $\mathbb{R}^3$

A continuación, se define a un poliedro y a un polítopo como:

Definición 20 *Un $\mathcal{P} \in \mathbb{R}^n$ es el conjunto de puntos x que satisface $\mathcal{P} = \{x \in R^n : Ax \leq b\}$, donde: A es una matriz tal que $A \in R^{m \times n}$ y b es un vector tal que $b \in R^m$. Si A y b son racionales , entonces $\mathcal{P}$ es un **poliedro racional** [121].*

Definición 21 *Un politopo convexo es resultado por la intersección finita no vacía de semiespacios cerrados engendrados $h(X)$ [191].*

Teorema 10 *Todo politopo es un conjunto convexo cerrado [191].*

C) Funciones convexas y cóncavas Sobre la idea de conjunto convexo, se definen las funciones cóncavas y convexas, las cuales desempeñan un papel importante en la optimización [14].

Definición 22 *Dado un conjunto convexo y no vacío $\mathcal{F} \subseteq \mathbb{R}^n$ y $f : \mathcal{F} \to \mathcal{R}$. La función f es convexa en $\mathcal{F}$ si y sólo si para todo par de puntos $x_i, x_j \in \mathcal{F}$ y $\lambda \in [0, 1]$ se satisface que*

$$f(\lambda x_i + (1 - \lambda)x_2) \leq \lambda f(x_i) + (1 - \lambda)f(x_2) \tag{1.1.6}$$

[179, 59]

En la Figura 1.9, se ejemplifica una función convexa.

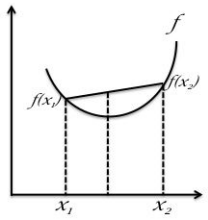

Figura 1.9: Ejemplo de función convexa

Definición 23 *Dado un conjunto convexo y no vacío* $\mathcal{F} \subseteq \mathbb{R}^n$ *y* $f : \mathcal{F} \to \mathcal{R}$*. La función* f *es estrictamente convexa en* $\mathcal{F}$ *si y sólo si para todo par de puntos* $x_i, x_j \in \mathcal{F}$ *y* $\lambda \in [0, 1]$ *se satisface que*

$$f(\lambda x_i + (1 - \lambda)x_2) < \lambda f(x_i) + (1 - \lambda)f(x_2) \tag{1.1.7}$$

[59]

Definición 24 *Dado un conjunto convexo y no vacío* $\mathcal{F} \subseteq \mathbb{R}^n$ *y* $f : \mathcal{F} \to \mathcal{R}$*. La función* f *es cóncava en* $\mathcal{F}$ *si y sólo si para todo par de puntos* $x_i, x_j \in \mathcal{F}$ *y* $\lambda \in [0, 1]$ *se satisface que*

$$f(\lambda x_i + (1 - \lambda)x_2) \geq \lambda f(x_i) + (1 - \lambda)f(x_2) \tag{1.1.8}$$

[59]

En la Figura 1.10, se ejemplifica una función cóncava.

Definición 25 *Dado un conjunto convexo y no vacío* $\mathcal{F} \subseteq \mathbb{R}^n$ *y* $f : \mathcal{F} \to \mathcal{R}$*. La función* f *es estrictamente cóncava en* $\mathcal{F}$ *si y sólo si para todo par de puntos* $x_i, x_j \in \mathcal{F}$ *y* $\lambda \in [0, 1]$ *se satisface que*

$$f(\lambda x_i + (1 - \lambda)x_2) > \lambda f(x_i) + (1 - \lambda)f(x_2) \tag{1.1.9}$$

[59]

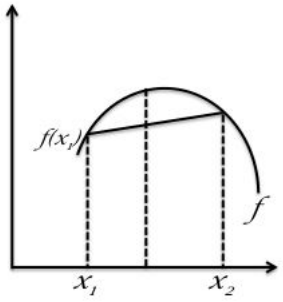

Figura 1.10: Ejemplo de función cóncava

El teorema 11 establece un criterio de convexidad de las funciones diferenciables:

Teorema 11 *Sea f una función diferenciable en $\mathcal{F}$. Entonces f es convexa si, y sólo si:*

$$f(x) \geq f(x_0) + \triangle f(x_0(x - x_0)) \tag{1.1.10}$$

para todo $x, x_0 \in \mathcal{F}$.

Definición 26 *Sea f una función convexa definida sobre un conjunto $\mathcal{F} \subseteq \mathbb{R}$ (convexo no vacío), entonces la función $-f$ se define como una función cóncava en $\mathcal{F}$*

Teorema 12 *Dada una función f doblemente diferenciable, si su segunda derivada $f''(x)$ es positiva, entonces f es convexa; si $f''(x)$ es negativa, entonces es cóncava. Si $f''(x)$ es cero entonces f no es cóncava ni convexa (véase Figura 1.11).*

Definición 27 *Una función f definida sobre un conjunto $\mathcal{F} \subseteq \mathbb{R}$ (convexo no vacío), es lineal, si y sólo si, para todo par de puntos $x_i, x_j \in \mathcal{F}$ y $\lambda \in [0, 1]$ se satisface que:*

$$\begin{aligned} f(\lambda x_i + (1 - \lambda)x_2) &\geq \lambda f(x_i) + (1 - \lambda)f(x_2) \\ f(\lambda x_i + (1 - \lambda)x_2) &\leq \lambda f(x_i) + (1 - \lambda)f(x_2) \end{aligned} \tag{1.1.11}$$

En la Figura 1.12, se ejemplifica una función lineal.

Algunas de las propiedades de las funciones cóncavas y convexas son:

- Dada una familia de funciones convexas, representada con ($f_i \; \forall i = 1 \ldots m$, definida sobre el conjunto $X \subseteq \mathbb{R}^n$ convexo y no vacío, la función $f = \sum_{i=1}^{n} f_i$ es convexa en X.

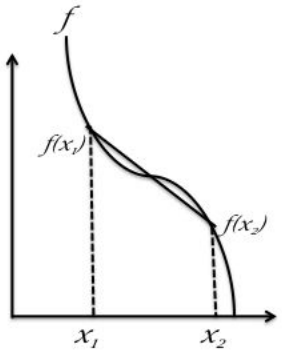

Figura 1.11: Ejemplo de función ni cóncava ni convexa

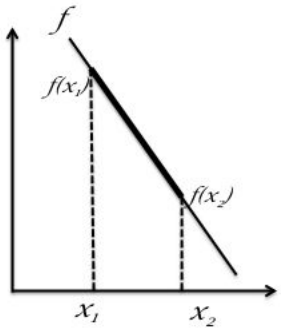

Figura 1.12: Ejemplo de función lineal

- Sea $f : X \subseteq \mathbb{R}^n \to \mathbb{R}$ y sea X un conjunto convexo y no vacío, entonces se tiene:
  - f es convexa, si y sólo si, θf, con $\theta > 0$, es convexa.
  - f es estrictamente convexa, si y sólo si, θf, con $\theta > 0$, es estrictamente convexa.
  - f es convexa, si y sólo si, θf, con $\theta < 0$, es cóncava.
  - f es estrictamente convexa, si y sólo si, θf, con $\theta < 0$, es estrictamente cóncava.
- Una función $f : X \to \mathbb{R}$ lineal es convexa y cóncava.

1.1.1.2. Óptimo local y global

El concepto de "punto extremo" es fundamental para definir si un punto es óptimo; en términos generales, se dice que $x \in \mathcal{F}$ es un punto extremo, si y sólo si, x no puede ser expresado como una combinación convexa estricta de cualquier par de puntos distintos en $\mathcal{F}$ [14]. En la definición 28 se formaliza dicho concepto.

Definición 28 *Dado $x \in \mathcal{F}$. Si no existen $x_1, x_2 \in \mathcal{F}$, tales que:*

$$\begin{gathered} x = \lambda x_1 + (1-\lambda)x_2 \\ \textit{donde:} \\ x_1 \neq x_2 \\ \lambda \in (0,1) \end{gathered} \tag{1.1.12}$$

entonces se dice que x es un punto extremo de $\mathcal{F}$.

Con base en las ideas de conjunto convexo y la de punto extremo, se concluye que dado un punto x en conjunto convexo X, si no existe ningún segmento de recta (no degenerado) en X que contenga a x en su interior relativo, entonces x es un punto extremo de X. Al conjunto de todos los puntos extremos de X se le denomina **perfil de** X y se le denota como: $ext(X)$.

Teorema 13 (Minkowski, Krein-Milman) *Todo cuerpo convexo $X \in \mathbb{R}$ es la envoltura convexa de sus puntos extremos [140].*

Un punto extremo es un punto frontera; sin embargo, no todos los puntos frontera son puntos extremos.

Localizar los puntos extremos es fundamental para la optimización, particularmente para la programación lineal, ya que en ellos se encuentran los valores máximos o mínimos de un función.

Definición 29 *Sea x un punto en el dominio de una función f . Si $f'(x) = 0$, o bien, $f'(x)$ no está definida; entonces, se le denomina a x como un valor crítico y $(x, f(x))$ es un punto crítico.*

Si en un intervalo $[a, b]$ entorno al punto x, la función f toma el mayor (o menor) valor, entonces se dice que x es máximo (o mínimo). A continuación, se definen los conceptos de máximo y mínimo.

Definición 30 *Sea f una función con dominio $\mathcal{F}$. Si para un punto x^* se satisface: $f(x^*) \geq f(x)$ $\forall x \in \mathcal{F}$, entonces se dice que x^* es un **máximo global** de f.*

Si se satisface la desigualdad $f(x^*) > f(x)$ para todos $x \neq x^* \in \mathcal{F}$, entonces se dice que: x^* es un punto **máximo global estricto**.

Definición 31 *Sea f una función con dominio $\mathcal{F}$. Si para un punto x^* se satisface: $f(x^*) \leq f(x)$ $\forall x \in \mathcal{F}$, entonces se dice que x^* es un **mínimo global** de f.*

Si se satisface la desigualdad $f(x^*) < f(x)$ para todos $x \neq x^* \in \mathcal{F}$, entonces se dice que: x^* es un punto **mínimo global estricto**. Con base en lo anterior y al concepto de conjunto, se puede concluir que: un elemento x^* en un conjunto $\mathcal{F}$ es un **elemento máximo**, si no existe un $x_1 \in \mathcal{F}$ tal que $x^* < x_1$. En contraste, el elemento $x^* \in \mathcal{F}$ es un **elemento mínimo** de $\mathcal{F}$ si no existe un elemento $x_1 \in \mathcal{F}$ tal que $x^* > x_1$.

Generalmente, se dice que un punto x es un **máximo local**, si $f(x) \leq f(x_i)$ $\forall x_i \in X'$, donde X' es un intervalo semiabierto de $\mathcal{F}$. Por otro lado, se dice que x es un **mínimo local**, si $f(x) \leq f(x_i)$ $\forall x_i \in X'$. En la Figura 1.13, se ejemplifican las ideas de mínimo global y local.

Si x^* es mínimo global también será una mínimo local; ya que al ser $f(x^*)$ el valor más pequeño de f en $\mathcal{F}$ se implica que $f(x^*)$ sea el valor más pequeño f en X'. Por otro lado un máximo global también es una máximo local.

A continuación, se dan las reglas de la primera y la segunda derivada; las cuales, permiten identificar un mínimo (o máximo) local.

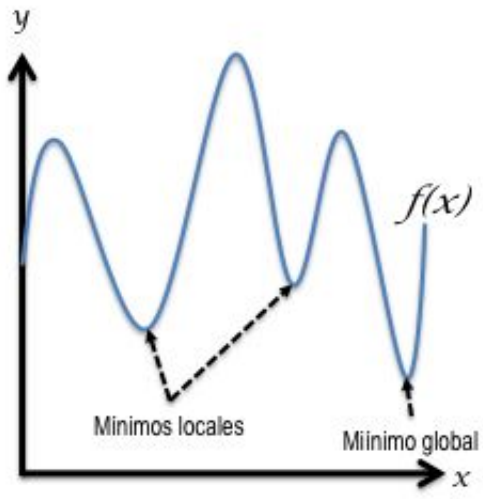

Figura 1.13: Ejemplo de mínimo local y global

Prueba 1.1 (Prueba de la primer derivada) *Sea f una función continua diferenciable en intervalo (a, b), excepto posiblemente en el punto crítico $x \in (a, b)$. Si al moverse a lo largo de f cerca de x se satisface que:*

- *f' cambie de negativo a positivo en x, entonces f tiene un mínimo local en x.*
- *f' cambie de positivo a negativo en x, entonces f tiene un máximo local en x.*
- *f' no cambie de signo en x, entonces f no tiene un extremo local en x.*

[234]

En el ejemplo 1.2 se muestra el uso de la prueba de la primera derivada.

Ejemplo 1.2

Dada la función $f(x) = 5x^4 - 6x^2 + 1$. Utilizar la prueba de la primera derivada con el fin de encontrar y caracterizar los puntos extremos locales. Como primer paso, se determinan los puntos críticos de la función.

$$f'(x) = \frac{d}{dx}5x^4 - 6x^2 + 1$$

$$f'(x) = 20x^3 - 12x$$

igualando $f'(x)$ *a cero*

$$20x^3 - 12x = 0$$

$$5x^3 - 3x = 0$$

$$x(5x^2 - 3) = 0$$

lo cual implica que:

$$x = 0$$

o bien

$$x = \pm\sqrt{\frac{3}{5}}$$

En las tablas 1.1, se caracterizan los tres puntos críticos de la función.

Tabla 1.1: Comportamiento de $f(x) = 5x^4 - 6x^2 + 1$ en los puntos críticos.

Intervalo	signo de f'	Comportamiento de f
$-\infty < -\sqrt{\frac{3}{5}}$	$-$	Decrecimiento
$-\sqrt{\frac{3}{5}} < 0$	$+$	Crecimiento
$0 < \sqrt{\frac{3}{5}}$	$-$	Decrecimiento
$\sqrt{\frac{3}{5}} < \infty$	$+$	Crecimiento

Tabla 1.2: Características de los puntos críticos.

x	$f(x)$	$f'(x)$	Tipo de extremo local
$-\sqrt{\frac{3}{5}}$	-0.8	0	mínimo
0	1	0	máximo
$\sqrt{\frac{3}{5}}$	-0.8	0	mínimo

En la Figura 1.14, se gráfica la función.

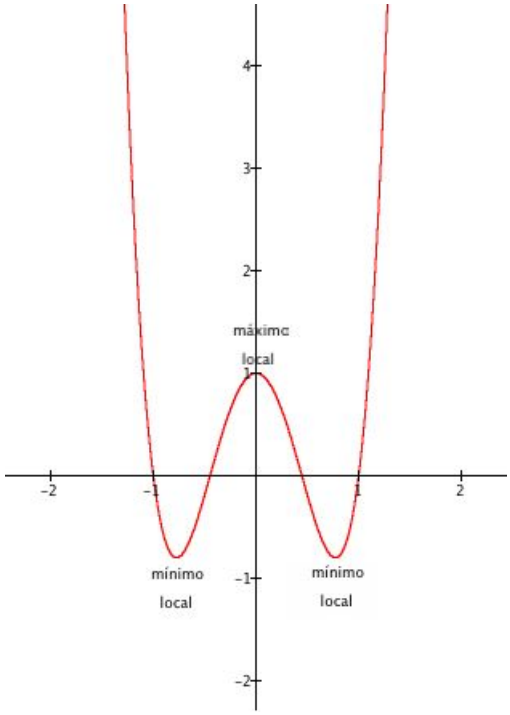

Figura 1.14: $f(x) = 5x^4 - 6x^2 + 1$

Prueba 1.2 (Prueba de la segunda derivada) *Sea $f(x)$ una función dos veces diferenciable en un intervalo abierto (a, b), el cual contiene a un punto (x)*

- *Si $f'(x) = 0$ y $f''(x) < 0$, entonces f tiene un máximo local en x.*
- *Si $f'(x) = 0$ y $f''(x) > 0$, entonces f tiene un mínimo local en x.*
- *Si $f'(x) = 0$ y $f''(x) = 0$, entonces la prueba falla; puesto que f puede tener un máximo local, mínimo local, o ninguno de ellos en x.*

[234]

1.1.1.3. Espacio de búsqueda y espacio factible

Definición 32 *Dado $x \in \mathcal{F}$, un vector d es una dirección factible de x si existe un $\overline{\alpha} > 0$ tal que $x + \alpha d \in \mathcal{F}$ para todos los $0 \leq \alpha \leq \overline{\alpha}$.*

En modelos los optimización se definen de manera implícita el espacio de búsqueda $\mathcal{S}$ al definir las variables de decisión. $\mathcal{S}$ es un conjunto finito o infinito contable de puntos, integrado por todas las soluciones candidatas; es decir, este conjunto puede verse como un rectángulo $n-$dimensional generado por la intersección del dominio de cada una de las variables [123]. En contraste, el espacio factible $\mathcal{F}$ es $\mathcal{F} \subseteq \mathcal{S}$ definido por un conjunto adicional de m restricciones ($m \geq 0$). Lo anterior se esquematiza en la Figura 1.15.

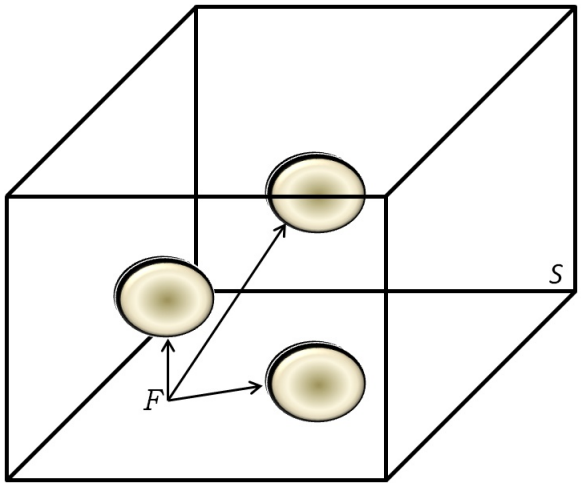

Figura 1.15: Espacio de búsqueda $\mathcal{S}$ y región factible $\mathcal{F}$ Fuente [152].

Definición 33 *Dado un espacio de búsqueda $\mathcal{S}$ junto con región factible $\mathcal{F}$, un problema de optimización es buscar un elemento $x \in \mathcal{F}$ tal que:*

$$f(x) \text{ es mejor que } f(y) \text{ para toda } y \in \mathcal{F} \tag{1.1.13}$$

[152]

Cualquier $x \in \mathcal{F}$ satisface el conjunto de m restricciones. Se denomina **restricción activa** aquella inecuación o ecuación cuyo lado derecho es igual al lado izquierdo.

De manera informal, se dice que el vecindario de un punto $x \in \mathcal{F}$ es el conjunto abierto $N(x) \subseteq \mathcal{F}$ formado por aquellos puntos "cercanos" al punto x (incluyendo al propio punto x). En otras palabras, $N(x)$ es un mapeo que asigna a cada elemento $x \in \mathcal{S}$ un conjunto de elementos $y \in \mathcal{S}$. Lo anterior

se esquematiza en la Figura 1.16 La definición formal de vecindario es:

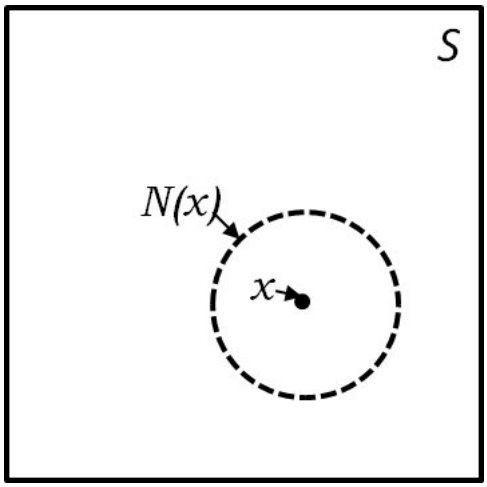

Figura 1.16: Vecindario de una potencial solución x Fuente [152].

Definición 34 *Dado un problema de optimización con instancias* (F, c) *, un vecindario es un mapeo* $N(x)$ *sobre el espacio de búsqueda* $\mathcal{S}$*, tal que:*

$$N(x) : \mathcal{S} \rightarrow 2^{\mathcal{S}} \tag{1.1.14}$$

[210]

Ahora, se define el vecindario en términos de una función distancia sobre el espacio de búsqueda como:

Definición 35 *Sea* d *una función distancia sobre el espacio de búsqueda* $\mathcal{S}$*, tal que:*

$$d : \mathcal{S} \times \mathcal{S} \rightarrow \mathbb{R} \tag{1.1.15}$$

por lo tanto el vecindario $N(x)$ *se define como:*

$$\begin{gathered} N(x_i) = \{x_i : x_i \in \mathcal{S} \wedge d(x_i, x_j) \leq \epsilon\} \\ \textit{para algún } \epsilon \geq 0 \end{gathered} \tag{1.1.16}$$

Todas las soluciones contenidas en el vecindario de solución x pueden ser encontradas desde x a través de sólo un movimiento.

Teorema 14 *Dada alguna instancia de un problema de optimización* $(\mathcal{F}, f)$*, donde* $\mathcal{F} \subseteq \mathbb{R}^n$ *convexo y* f *es una función convexa* $\mathcal{F}$*, entonces el vecindario* $N(x)$ *se define como:*

$$\begin{gathered} N(x) = \{x_i : x_i \in \mathcal{F} \wedge d(x_i, x_j) \leq \epsilon\} \\ \textit{para algún } \epsilon > 0 \\ \textit{donde: } d(x_i, x_j) = \sqrt{\sum_{i=1}^{n}(x_i - x_j)^2} \end{gathered} \tag{1.1.17}$$

1.1.1.4. Paisaje

Un **paisaje** es una tripleta (S, N, f) donde: S es en conjunto de soluciones admisibles, N es un operador de vecindario, y $f : X \to \mathbb{R}$ es una función de aptitud [35].

Una función de aptitud sirve para evaluar un conjunto de atributos: valor de la función objetivo, fatibilidad, entre otros; para una particular solución admisible; por ende una función de aptitud sirve para cuantificar la calidad de cada solución.

La **aptitud del paisaje** [5] es una representación de la estructura del espacio de búsqueda; y se define por: el valor de la función de aptitud , un conjunto de soluciones admisibles y un operador de vecindario. La estructura y propiedades de la aptitud de paisaje juegan un rol importante para determinar el grado de dificultad de un problema y el método más adecuado para explorar el espacio de búsqueda.

Un concepto importante emanado de la aptitud de paisaje es **vecindario neutral** $(N_n(s))$ de una solución $s \in S$, el cual es un vecindario de la solución con el mismo valor de la función de aptitud; en otras palabras,$N_n(s) = \{s' \in N(s) | f(s') = f(s)\}$ [35].

1.1.1.5. NP-Completos

Probablemente ya se tenga alguna idea intuitiva sobre lo que es un problema, un algoritmo y el tiempo de corrida de un algoritmo; sin embargo, resulta necesario formalizar estas ideas antes de analizar la dificultad de un problema. En la optimización son de interés los problemas de decisión[6]y los problemas de optimización.

Existe una correspondencia entre ambas clases de problemas; que dado un problema de optimi-

[5] Concepto propuesto por Wright.

[6] Un problema de decisión es aquel cuya solución es simplemente "sí" o "no" [74, 129].

zación Π se expresa como un problema de decisión Π' con la imposición de un valor límite sobre la función objetivo (o valores límites sobre el conjunto de funciones objetivo) [43].

Generalmente, los problemas de optimización son más difíciles de resolver que sus equivalentes problemas de decisión [129]; es decir, un problema de optimización requiere para resolverse como mínimo la misma cantidad de recursos utilizados para resolver su correspondiente problema de decisión. Un problema de decisión Π consiste simplemente en el conjunto de D_Π formado por las instancias y un subconjunto de las $Y_\Pi \subseteq D_\Pi$ de instancias cuya respuesta es afirmativa [74]. Considérese los siguientes ejemplos:

Ejemplo 1.3

Problema de subgrafos isomorfos. Instancias: Dados los grafos $G_1 = (V_1, E_1)$ *y* $G_2 = (V_2, E_2)$ *Pregunta: ¿ El grafo* G_1 *contiene un subgrafo* G_1' *que sea isomórfico a* G_2*?, para contestar esta interrogante se necesita determinar la existencia o no del subgrafo* $G_1' = (V_1' \subseteq V_1, E_1' \subseteq E_1)$ *tal que* $|V_1'| = |V_2|$ *,*$|E_1'| = |E_2|$ *,* $f : V_2 \to V_1'$ *que satisfaga* $\{u, v\} \in E_2$ *si y sólo si* $\{f(u), f(v)\} \in E_1'$.

Ejemplo 1.4

Problema del agente viajero. Instancias: Un conjunto finito de ciudades $C = \{c_1, c_2, c_3, \ldots, c_m\}$, *una función de distancia* $d(c_i, c_j) \in Z^+$ *para cada par de ciudades* $c_i, c_j \in C$ *y un valor* $B \in Z^+$. *Pregunta: Existe un tour entre todas las ciudades en* C, *tal que la longitud total de éste no sea mayor a* B; *es decir, existe un ordenamiento de la ciudades en* $C < c_{\pi(1)}, c_{\pi(2)}, \ldots, c_{\pi(m)} >$ *tal que* $[\sum_{i=1}^{m-1} d(c_{\pi(i)}, c_{\pi(i+1)})] + d(c_{\pi(m)}, c_{\pi(1)}) \leq B$.

El ejemplo 1.4 es un problema de decisión derivado de un problema de optimización, ya que el problema de optimización del agente viajero (TSP por la siglas en inglés de *travelling salesman problem*), en el cual se requiere encontrar el tour con costo mínimo, puede ser asociado con un problema de decisión; donde se incluya una condición de forzamiento tal que el conjunto de recorridos posibles quede restringido a aquellos cuyo valor sea menor o igual a B.

a) Máquina de Turing

Generalmente, se recurre a los conceptos de lenguaje y máquina de Turing, para describir y analizar a los problemas decisión, puesto que cualquiera de estos problemas se puede representar como "lenguaje", siendo equivalente el proceso de resolver el problema con el proceso de reconocimiento

del correspondiente lenguaje [74].

Definición 36 *Sea $\sum$ un alfabeto* [7] *con al menos dos símbolos y sea $\sum*$ el conjunto finito formado por todas las cadenas finitas generadas a partir de $\sum \bigcup \varnothing$. El lenguaje L sobre $\sum$ es un subconjunto $\sum*$ [74, 39]. Considerese el ejemplo 1.5*

Ejemplo 1.5

Dados $\sum = \{0, 1\}$ entonces $\sum*$ contendrá a las cadenas $\varnothing, 0, 1, 00, 01, 10, 11, 000, 010, 001, \ldots$, si se define a L como un subconjunto de $\sum*$ tal que L contenga a todas las cadenas de no más de dos elementos entonces $L = \{\varnothing, 0, 1, 00, 01, 10, 11\}$.

Definición 37 *Un lenguaje L satisface las siguientes propiedades:*

- *Si $c \in \sum$, $L(c) = \{c\}$. Esto es una cadena de una sola letra.*
- $L(\varnothing) = \varnothing$
- *Dados E_1 y E_2 dos expresiones regulares* [8] *sobre un alfabeto, se cumplen las siguientes propiedades:*
  - $L(E_1|E_2) = L(E_1) \cup L(E_2)$.
  - $L(E_1 \cdot E_2) = L(E_1) \circ L(E_2)$
  - $L(E_1\star) = L(E_1)\star$
  - $L((E_1)) = L(E_1)$

  [167]

La correspondencia entre los problemas de decisión y el lenguaje está dada por el esquema de codificación especificado en el problema; el esquema de codificación e de un problema D_Π provee la manera apropiada para describir a cada una de las instancias de D_Π en cadenas de símbolos sobre un Σ; por lo tanto, la codificación genera una partición en tres clases del conjunto $\Sigma\star$, que son: A)

[7] Un alfabeto es un conjunto finito de símbolos, diferente al conjunto vacío.

[8] Una expresión regular, también denominado patrón, es una expresión que describe un conjunto de cadenas sin enumerar sus elementos. Una expresión regular es un conjunto numerable sobre un alfabeto finito; en contraste la cantidad de lenguajes distintos sobre un alfabeto finito es no numerable.

las cadenas no codificadas para ninguna instancia de D_Π, B) aquellas codificaciones de las instancias de D_Π cuya respuesta es "no" y C) aquellas codificaciones de las instancias de D_Π cuya respuesta es "sí". Esta tercera clase de cadenas es el lenguaje asociado a D_Π y e.

$$L[D_\Pi, e] = \left\{ \begin{array}{c} \Sigma \text{ sea el alfabeto usado por} \\ x \in \Sigma\star \quad e \text{ y } x \text{ sea una codificación dentro de } e \\ \text{para una instancia } I \text{ tal que } I \in Y_\Pi \end{array} \right\} \tag{1.1.18}$$

Si un resultado es válido para el lenguaje $L[D_\Pi, e]$, entonces también es válido para el problema D_Π en la codificación del esquema e. Si e y e' son dos esquemas de codificación razonables para D_Π, entonces las propiedades se mantienen tanto para $L[D_\Pi, e]$ y $L[D_\Pi, e']$ o bien para ninguno. En el concepto de **razonable esquema de codificación** se involucra la idea de consistencia. Una máquina de Turing M [9] es un dispositivo teórico computacional, compuesta por una unidad de control de estados finitos (ejemplo algoritmo) y una cinta [10] magnética en ambas direcciones infinitas, la comunicación entre la unidad de control y la cinta sólo se da a través de una cabeza de lectura-escritura, la cual, es capaz de moverse en ambas direcciones de la cinta a una celda adyacente, posteriormente el cabezal puede leer y escribir los símbolos de un alfabeto. Una máquina de Turing realiza una tarea a través de combinar algunas de las siguientes operaciones: a) avanzar el cabezal lector/escritor hacia la derecha o b) avanzar el cabezal lector/escritor hacia la izquierda. Lo anterior se esquematiza en la Figura 1.17. Una M es un dispositivo automático de los símbolos sobre las celdas de una cinta magnética infinita, generando a su vez una salida sobre la propia cinta. A continuación, se formaliza el concepto de M y algunas ideas relacionadas:

Definición 38 *Una máquina de Turing M es una tupla $(\Sigma, \Gamma, Q, \delta)$ donde Σ, Γ, Q son conjuntos finitos no vacíos con $\Sigma \subseteq \Gamma$ y $b \in \Gamma - \Sigma$. El conjunto de estados Q contiene a: $q_o, q_{aceptar}, q_{rechazar}$. La función de transformación δ debe satisfacer:*

$$\delta : \{Q - \{q_{aceptar}, q_{rechazar}\}\} \times \Gamma \to Q \times \Gamma \times \{-1, 1\}$$

se asume que los conjuntos Q y Γ son conjuntos disjuntos [39].

[9] Concepto introducido por Allan Turing en 1937

[10] La cinta está dividida en celdas, cada una de ellas es capaz de almacenar un símbolo de una alfabeto prefijado $\sum \bigcup \varnothing$.

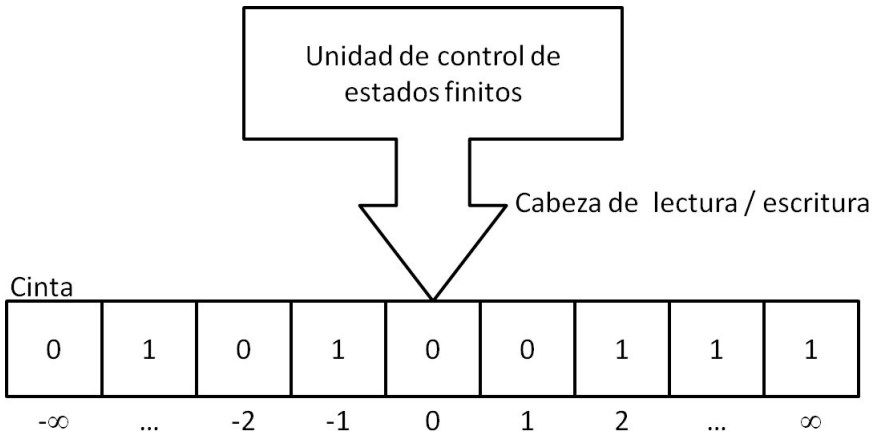

Figura 1.17: Visualización de una máquina de Turing.

La operación que debe realizar M está determinada por su función de transformación; dado un estado y valor actual se produce un nuevo estado y valor a través de su movimiento en una dirección. Es decir, dada la función de transición $\delta(q, s) = (q', s', h)$, ésta indica que M está actualmente en el estado q escaneando el símbolo s; en el nuevo estado q', el cabezal de la cinta se habrá movido hacia la derecha o hacia la izquierda dependiendo si h es -1 o 1, colocándose en una nueva celda donde se imprimirá el símbolo s'.

Definición 39 *Una configuración C de M es una cadena x, q, y donde: $x, y \in \Gamma*, y \neq \varnothing \wedge q \in Q$ [39].*

Ahora bien, sea el lenguaje aceptado por M se denota como $L(M)$ y se asocia a un alfabeto Σ de tal forma que:

$$L(M) = \{w \in \sigma * | M \text{ acepte } w\} \tag{1.1.19}$$

entonces para una máquina M con una entrada $w \in \Sigma*$; M deberá realizar una secuenciación de configuraciones ($C_0, C_1, \ldots$, tales que $C_0 = q_0 w$ y $C_i \overrightarrow{M} C_{i+1}$ para cada i en la C_{i+1} operación), para decidir sobre aceptar o rechazar w como un elemento del lenguaje asociado a M. Si el total de operaciones necesarias para decidir sobre w, es igual al número de configuraciones menos uno, se dice que el cómputo es finito, en cualquier otro caso el cómputo es infinito. Denótese con $t_M(w)$ al número de operaciones requeridas por M para procesar una entrada w.

Lema 3 *Un lenguaje $L(M)$ es aceptable para una máquina de Turing si y sólo si $L(M)$ es el lenguaje de salida de M; o bien, si un lenguaje $L(M)$ es decidible, entonces es aceptable.*

Sea $T_M(n)$ el número de operaciones necesarias por M para procesar una entrada de tamaño n tal que:

$$T_M(n) = \text{máx}\{t_M(w)|w \in \Sigma^n \text{ y } n \in \mathbb{N}\} \tag{1.1.20}$$

donde: Σ^n sea el total de cadenas de longitud n. Se dice que M corre en un tiempo polinomial si existe una constante k tal que para toda n se satisfaga que $T_M(n) \leq n^k + k$ [39]. Si para cada $\delta(q, s)$ sólo existe una posibilidad de ejecución, se trata de una máquina de Turing determinista, mientras que en el caso de que dado un $\delta(q, s)$ existan dos o más posible acciones se trata de una máquina de Turing no determinista; a continuación se formalizan estas definiciones:

Definición 40 *Dada una máquina de Turing si la función de transformación se define como $\delta(q, s) = (q', s', h)$ para todo estado q y para cada símbolo en la cinta s, entonces M es una máquina de Turing determinista (MD).*

Definición 41 *Dada una máquina de Turing si la función de transformación se define como $\delta(q, s) = \{(q'_1, s'_1, h_1), (q'_2, s'_2, h_2), \ldots, q'_k, s'_k, h_k\}$ donde:$k \in \mathbb{N}, k \geqq 2$,para todo estado q y para cada símbolo en la cinta s entonces M es una máquina de Turing no determinista (MND).*

Posteriormente, se acuñó la idea de una máquina universal; dicha idea ha permitido plantear de manera abstracta la posibilidad de diseñar, o encontrar en la naturaleza, un sistema autorregulado capaz de imitar el funcionamiento de cualquier otro sistema [98]. Pese a la imposibilidad práctica de diseñar una maquina universal esta idea se ha utilizado como un paradigma en el desarrollo tecnológico actual. La máquina universal de Turing [11] (MUT) posee las características de universalidad y programabilidad. Por ende, una MUT emula a cualquier MT y soluciona cualquier problema que pueda ser resuelto a través de un algoritmo [131]. La definición formal de una MUT es la siguiente.

Definición 42 *Dada una M_1 que recibe en la cinta una descripción de otra M_2 y el contenido de la cinta de M_2. Si M_1 es capaz de producir el mismo resultado que el obtenido por la cinta de M_2,*

[11] Idea concebida por Marvin Minsky

entonces M_1 se llama máquina universal, pues es capaz de simular el comportamiento de cualquier M. [241]

b) Tesis de Turing y Church

Definición 43 *La **Tesis de Turing** establece que las funciones que pueden ser calculadas mediante un método definido coincide con la clase de las funciones calculables mediante una Máquina de Turing.*

Corolario 1 *Un problema tiene solución si existe una máquina de Turing capaz de calcularlo [92].*

La **Tesis de Church** establece que las funciones y problemas computables son precisamente los que pueden ser resueltos con una Máquina de Turing. En otras palabras, la Tesis de Church dice: que cualquier proceso que se reconoce naturalmente como un algoritmo puede ser llevado a cabo por una MT [97]. A continuación, se formaliza la Tesis de Church

Definición 44 *Cuando existe un método efectivo (algoritmo) para obtener valores de una función matemática, la función puede ser calculada por una MT [92]. En otras palabras, Toda función computable es recursiva y todo conjunto decidible es recursivo [208].*

Probar las Tesis de Church y de Turing conlleva una gran dificultad, debido a la ambigüedad de algunos conceptos involucrados en ellas. Sin embargo, muchos avances y resultados sólidos en la ciencia, se encuentran sustentados en estas tesis; pues las ideas propuestas por Turing y Church fincaron el **paradigma de computación clásica**. En este paradigma, el concepto de compatibilidad (una serie de procesos lógicos) se ligó a un conjunto de procesos mecánicos de una MT, presuponiendo que MT era regida por las leyes de física clásica, al asociar las ideas de mecánica cuántica con las posibilidades de construir ordenadores más potentes.

c) Clase P y NP

Considerando lo anterior, los problemas de decisión se clasifican en:

- **Indecidibles**: son aquellos problemas que no pueden solucionarse en forma algorítmica; generalmente a estos problemas se les puede describir, pero no se pueden representar o resolver. Algunos ejemplos de problemas indecidibles son: A) Dada una M arrancada, determinar si M

se detiene con la cinta vacía; B) dada una M, determinar si M se detiene frente alguna (o todas) las entradas posibles; C) dada una M, determinar si el lenguaje que acepta M es finito; D) dadas M_1 y M_2, determinar si ambas máquinas aceptan el mismo lenguaje; E) dadas M_1 y M_2, determinar si se detienen frente la misma entrada [167].

- **Decidibles**: son aquellos problemas para los cuales existe al menos un algoritmo que puede decidir para cada posible frase de entrada si dicha frase pertenece o no al lenguaje del problema de decisión. A continuación, se da la definición de problema de decisión en términos de lenguaje:

  Definición 45 *Un* ***problema de decisión*** Π *es un par* (L_X, L_Y)*; donde:* L_X *es un lenguaje decidible en tiempo polinomial y* $L_Y \subseteq L_X$ *. Los elementos de* L_X *son llamados instancias de* Π*; los elementos de* L_Y *son las instancias afirmativas de* Π *-denotadas como sí-instancias- ; mientras que* $L_X \setminus L_Y$ *son las instancias negativas de* Π *-denotadas como no-instancias- . Un* ***algoritmo para un problema de decisión*** (L_X, L_Y) *es un algoritmo capaz de calcular una función* $f : L_X \to \{0, 1\}$ *tal que:* $f(x) = 1$ *si* $x \in L_Y$ *y* $f(x) = 0$ *si* $x \in L_Y \setminus LX$ *[121].*

  A su vez este tipo de problemas se subdivide en clase P y clase NP

  - *Clase de problemas* ***P***: son aquellos problemas de decisión que pueden solucionarse por un algoritmo con un número de pasos limitados por un polinomio en función del tamaño de entrada, en términos de lenguaje la clase P se define como:

    $$P = \{L | L = L(M) \text{ para alguna máquina M de Turing que corre en tiempo polinomial}\} \tag{1.1.21}$$

  - *Clase de problemas NP*: es la colección de aquellos problemas de decisión, para los cuales, sólo se conocen algoritmos de solución no-determinísticos en tiempo polinomial; lo anterior, indica que una M no determinística resuelve en tiempo polinomial esta clase de problemas.

    Otra definición equivalente utiliza el lenguaje L_R de una relación binaria R de chequeo entre dos lenguajes[12], el lenguaje L_R se define como: $L_R = \{w\sharp y | R(w, y)\}$. Entonces: la

[12]Dado $R \subseteq \Sigma* \times \Sigma*_1$, sea L_R el lenguaje definido $\Sigma \bigcup \Sigma_1 \bigcup \{\sharp\}$ donde:$\sharp$ indica que el símbolo no esta Σ

clase NP se forma por aquellos problemas cuyo lenguaje L sobre Σ satisface la siguiente condición: Existe un $k \in \mathbb{N}$ y una relación de chequeo R de tiempo polinomial, que cumple para toda $w \in \Sigma*$ la relación $w \in L \Leftrightarrow y(|y| \leq |w|^k$ y $R(w.y))$ donde:$|y|$ y $|w|$ denota la longitud de la cadena respectiva [74].

Existe una relación entre las clases P y NP; ya que, dado un problema $\Pi \in P$ dicho problema Π pertenecerá también a NP. En otras palabras, si Π es un problema de decisión que se soluciona a través de un algoritmo determinístico en tiempo polinomial, entonces Π puede ser resuelto en un tiempo polinomial por un algoritmo no determinístico. Lo anterior ha llevado a suponer que $P \subseteq NP$. La evidencia teórica hasta el momento apunta hacia $P \subset NP$; sin embargo, no se ha demostrado que $P \neq NP$ [74, 129]. En la Figura 1.18 se esquematizan las posibles relaciones entre P y NP.

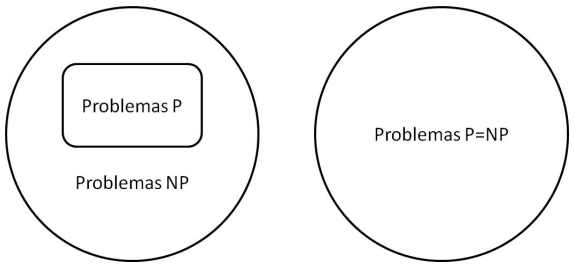

Figura 1.18: Posibles relaciones entre los grupos P y NP.

d) Transformación polinomial

Una idea fundamental para establecer la relación entre los grupos P y NP es la de **transformación polinomial** , también llamada **reducción polinomial**. La transformación polinomial de un lenguaje $L_1 \subseteq \Sigma_1*$ en un lenguaje $L_2 \subseteq \Sigma_2*$ es una función [13] $f : \Sigma_1* \rightarrow \Sigma_2*$ que satisface las siguientes condiciones [74, 129]:

- Existe un programa en una DM capaz de calcular la f en un tiempo polinomial
- Para todo $x \in \Sigma_1*$, $x \in L_1 \Leftrightarrow f(x) \in L_2$. Es decir, existe una regla de asociación entre L_1 y

[13]Dados dos conjuntos A y B una función es una regla que asigna un elemento único $f(x) \in B$ a cada elemento $x \in A$

L_2 de tal forma que a cada elemento $x \in L_1$ se le asigna un único elemento $f(x) \in L_2$.

La transformación polinomial entre los dos lenguajes L_1 a L_2, denotado por $L_1 \propto L_2$, involucra el siguiente lema:

Lema 4 *Si $L_1 \propto L_2$ y si $L_2 \in P$, se implica que $L_1 \in P$. En caso que $L_2 \notin P$ entonces $L_1 \notin P$ [74].*

Dados los problemas de decisión Π_1 y Π_2 asociados con los esquemas de codificación e_1 y e_2 respectivamente, existe $\Pi_1 \propto \Pi_2$ si existe una función tal $f : L[\Pi_1, e_1] \rightarrow L[\Pi_2, e_2]$. Generalmente, se omite el esquema de codificación, pues se asume que se utiliza un esquema razonable (computable) y por ende la $\Pi_1 \propto \Pi_2$ se asume como una función $f : D_{\Pi_1} \rightarrow D_{\Pi_2}$, la cual satisface las siguientes condiciones:

- La función f puede ser calculada por un algoritmo en un tiempo polinomial.
- Para todo $I \in D_{\Pi_1}$, $I \in Y_{\Pi_1} \Leftrightarrow f(I) \in Y_{\Pi_2}$

En otras palabras, la reducción polinomial consiste en reducir un problema Π_1 en otro Π_2 a través de una función f, esta función es calculada por medio de un algoritmo A en tiempo polinomial. f es denominado función de reducción, mientras que A se le llama algoritmo de reducción. A continuación, se formaliza el concepto de reducción polinomial.

Definición 46 *Dados los problemas de decisión $\Pi_1 = (L_{X_1}, L_{Y_1})$ y $\Pi_2 = (L_{X_2}, L_{Y_2})$. Se dice que Π_1 se **tranforma polinomialmente** en Π_2 si existe una función $f : L_{X_1} \rightarrow L_{X_2}$ computable en tiempo polinomial; tal que $f(x_1) \in L_{Y_2}$, para todo $x_1 \in L_{Y_1}$; y $f(x_1) \in L_{X_2} \setminus L_{Y_2}$, para todo $x_1 \in L_{X_1} \setminus L_{Y_1}$ [121]. En la Figura 1.19 se esquematiza la idea de reducción polinomial.*

A continuación, se muestra un ejemplo de reducción polinomial tomado de [244].

Ejemplo 1.6

Reducción polinomial entre Cobetura de Vértices y Clique

Definición de los problemas:

- *Cobetura de Vértices.*

  Instancia: Dada una gráfica $G = (V, E)$ y un entero positivo k.

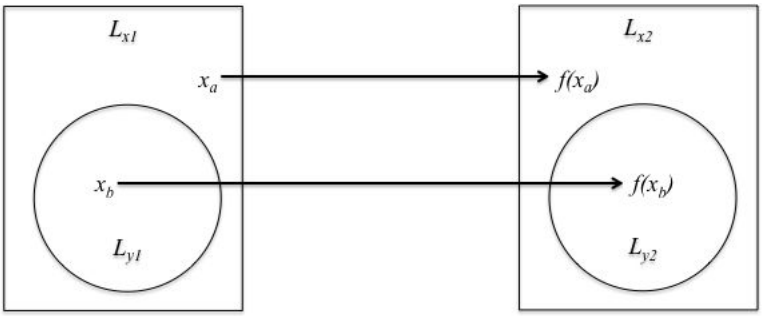

Figura 1.19: Reducción polinomial Fuente [43]

Pregunta: ¿Existe un subconjunto de los vértices ($S \subseteq V$); tal que: $|S| \leq k$ y si $(u, v) \in E$ entonces $u \in S$, $v \in S$ o ambos?

- *Clique*

  Instancia: Dada una gráfica $G = (V, E)$ no dirigida y un entero positivo $k \leq |V|$.

  Pregunta: ¿Existe un caln [14] V' de tamaño k o más, el cual es un subconjunto de los vértices ($S \subseteq V$) tal que: $|S| \geq k$ y cada par de vértices en S se encuentran unidos por una arista en E?

Cobertura de Vértices $\propto$ Clique:

Reducción por equivalencia simple

Premisa inicial:

- *Para cualquier gráfica $G = (V, E)$, existe una gráfica complementaria $G' = (V, E')$, donde: $E' = (u, v)|(u, v) \notin E$.*

Probar que el problema de Clique es equivalente a Cobertura de Vértices

Se presume que G tiene un clan S, con $|S| = k$

[14] Un clan, también llamado clique, es un subconjunto de vértices del grafo que están todos ellos conectados entre si.

Sea $S' = V - S$, por lo tanto $|S'| = |V| - k$

Considere que para cualquier arista $(u, v) \in E'$ las siguientes oraciones son verdaderas:

$(u, v) \notin E$

Ya sea u o v no están en S; por lo tanto, S conforma un clan.

Ya sea u o v están en S'; por lo tanto (u, v) está cubierto por S'.

Con base en lo anterior, S' es una cobertura de G.

Probar que el problema de Cobertura de Vértices es equivalente a Clique

Se presume que G' tiene una cobertura S', con $|S'| = |V| - k$

Sea $S = V - S'$, por lo tanto $|S| = k$

Considere que para cualquier arco $(u, v) \in E'$ las siguientes oraciones son verdaderas:

u, v o ambos pueden estar en S'.

Si u y v no estan en S'; entonces $(u, v) \in E$.

Con base en lo anterior, se deduce que S es un clique de G

Considerando lo anterior se deduce que: La gráfica G tiene un clan de tamaño k; si y sólo si G' tiene una cobertura de tamaño $|V| - k$

e) Clase $co - NP$

Al conjunto de problemas de decisión complementarios a los de la clase NP se les denomina $co - NP$. Por problema complementario se entiende aquel que cuyas respuestas positiva o negativa están invertidas. En el Ejemplo 1.7, se muestran dos problema $co - NP$.

Ejemplo 1.7

a *$co - NP$ de gráfica Hamiltoniana.*

Instancia: Dada una gráfica $G = (V, E)$. Pregunta: ¿La gráfica G es no Hamiltoniana?

b *$co - NP$ de suma de subconjuntos Instancia: Dado un conjunto finito de enteros . Pregunta: ¿Existe un subconjunto de enteros _distinto de conjunto vacío_, cuya suma no sea cero?*

A continuación, se formaliza la definición de $co - NP$.

Definición 47 *Para un problema de decisión* $\Pi = (L_x, L_y)$*, su problema de decisión complementario es* $(x, x \setminus y)$*. La clase* $co - NP$ *consiste de todos aquellos problemas cuyos complementos están en* NP *[121].*

El complemento de un $\Pi \in P$ también está en P. En la Figura 1.20 se muestran las posibles relaciones entre P, NP y $co - NP$, cabe señalar la conjetura más común de la relación existente entre NP y $co - NP$ implica que $NP \neq co - NP$.

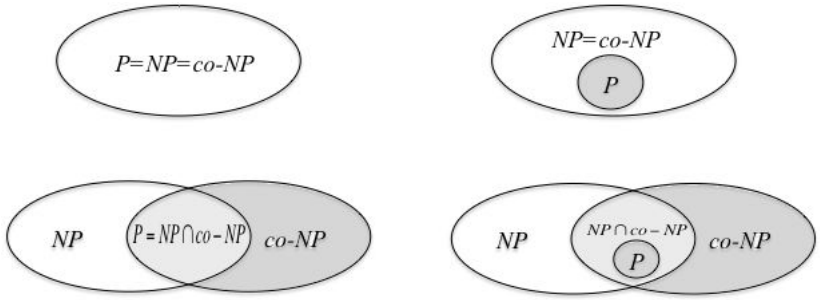

Figura 1.20: Conjeturas entre las relaciones P, NP y $co - NP$ [43]

f) NP-Completos

Los problemas NP-Completos se definen como problemas muy difíciles en NP [40]. Se dice que un problema Π es NP-Completo si para cada problema $\Pi' \in NP$ existe una reducción en tiempo polinomial tal que $\Pi' \propto \Pi$. A continuación, se da la definición de NP-Completo.

Definición 48 *Un problemas* X *es NP-Completo si* $X \in NP$ *y todo problema* NP *se reduce a* X *[129].*

El conjunto de los problemas NP-Completos es muy importante, pues hasta ahora cualquiera de estos problemas no puede ser solucionado de forma exacta por ningún algoritmo determinístico conocido en un tiempo polinomial; es decir, resolver un problema NP-completo en el peor caso es una tarea intratable [43]; ya que no es posible que un algoritmo exacto, encuentre la solución óptima con esfuerzos computacionales aceptables. En la Figura 1.21, se representa el conjunto de posibles relaciones entre P, NP y NP-Completos.

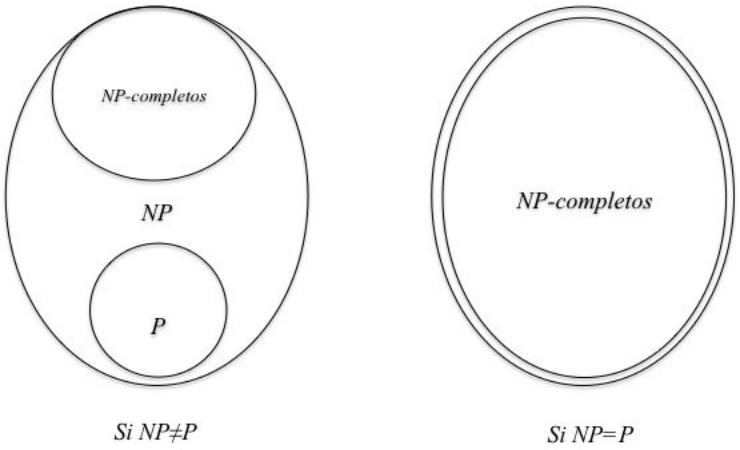

Figura 1.21: Posibles relaciones entre P, NP y NP-Completos Fuente [129]

El teorema de Cook fue la primera prueba de existencia de problemas NP-Completos. Este teorema establece que el problema de satisfacibilidad (SAT) es NP-Completo. Antes de continuar, se definen los conceptos implicados en el teorema de Cook.

Definición 49 *Problema de Satisfacibilidad (SAT)*
Instancia: Un conjunto finito X de variables booleanas y una familia Z de claúsulas[15]*. sobre X.*
Pregunta: ¿El conjunto Z es satisfacible [16] *?*

Teorema 15 *(**Teorema de Cook**) El SAT es NP-Completo [39]*

El Teorema de Cook implica que $NP = P$, si y sólo si, el problema SAT es un P [129]. En otras palabras, el teorema de Cook dice que si SAT puede resolverse a través de algoritmo polinomial entonces todo problema NP puede ser resuelto en un número polinomial de pasos. Cook también mostró que el problema 3-SAT [17] es NP-completo

[15] Una **cláusula** sobre un conjunto X es un conjunto de literales. Una literal es una variable proposicional (literal con polaridad positiva) o la negación de una variable proposicional (literal con polaridad negativa)

[16] Una familia de claúsulas sobre X es **satisfacible** si y sólo si existe para una asignación de verdad -una asignación de verdad para un conjunto X es una función $T : X \to veradadero, falso$- que satisfaga simultaneamente todas las cláusula

[17] Problema 3-SAT

Instancia: Un conjunto X de variables y una colección de Z claúsulas sobre X, cada claúsula contiene exacta-

Teorema 16 *El problema 3-SAT es NP-Completo [121]*

Karp [1972] [40, 121], en su trabajo "Reducibility Among Combinatorial Problems", profundizó en el trabajo de Cook; en dicho trabajo Karp mostró las implicaciones del teorema de Cook en problemas de optimización combinatoria y además en listo 21 problemas combinatorios del tipo NP-Completos . En lo siguientes teoremas y corolarios, se muestran algunos de los resulatdos obtenidos por Karp.

Corolario 2 *Los problemas de Cobertura de vértices y Clique son NP-Completos [40].*

Corolario 3 *El problema de la* ***cobertura exacta***[18] *es NP-Completo [40].*

Teorema 17 *El problema del conjunto estable o independiente* [19] *es NP-Completo [121]*

Teorema 18 *El problema del ciclo Hamiltoniano* [20] *es NP-Completo [121]*

Corolario 4 *El problema de un* ***ciclo Hamiltoniano dirigido*** [21] *es NP-Completo [40]*

mente tres literales

Pregunta: ¿El conjunto Z es satisfactible?

[18]Problema de cobertura exacta[235]

Instancia: Un conjunto finito X de elementos, una colección finita S de subconjuntos de X

Pregunta: ¿ Existe una colección S' tal que $S' \subseteq S$ y que cada elemento de X aparezca en S' exactamente una vez ?

[19]Problema del conjunto estable o independiente [74]

Instancia: Un grafico $G = (V, E)$ y un entero $k \leq |V|$

Pregunta: ¿ G contiene un conjunto independiente de vértices ($S \subseteq V$) de tamaño k, tal que $|S| \geq k$ y que no hay dos vértices en S, los cuales se unan por una arista en E ?

[20]Problema del ciclo Hamiltoniano [43]

Instancia: Un gráfico $G = (V, E)$.

Pregunta: ¿ La gráfica G tiene un ciclo Hamiltoniano; el cual es sucesión de aristas adyacentes que visita todos los vértices de G una sola vez?

[21]Ciclo Hamiltoniano dirigido [74]

Corolario 5 *El problema de un* ***ciclo Hamiltoniano no dirigido*** *es NP-Completo [40]*

Corolario 6 *El problema del* ***agente viajero*** *(TSP) es NP-Completo [40]*

Teorema 19 *El problema de matching 3-dimensional (3DM)* [22] *es NP-Completo [121, 74]*

Teorema 20 *El problema de partición* [23] *es NP-Completo [121, 74]*

Corolario 7 *El problema de* ***suma de subconjuntos*** [24] *es NP-Completo [121]*

Los teoremas y corolarios anteriores muestran algunos problemas NP-Completos, los más conocidos, si se desea más información de estos problemas se puede revisar [74, 43, 121].

Para probar que un problema Π es NP-completo se debe utilizar la propiedad transitiva **reducir a**, la cual, dice: "Si Π_1 es un problema NP-completo; Π_2 es un problema NP y puede demostrarse que $\Pi_1 \propto \Pi_2$, entonces Π_2 es un problema NP-completo" [129]. Implementar la propiedad "reducir a" dado un problema Π_2 involucra realizar los siguientes cuatro pasos:

- Probar que Π_2 es un problema NP.

Instancia: Una gráfica dirigida $G = (V, E)$

Pregunta: ¿Existe en G un ciclo Hammiltoniano dirigido?

[22] Problema de matching 3-dimensional [74]

Instancia: Un conjunto $M \subseteq W \times X \times Y$, donde: W, X e Y son conjuntos disjuntos que tienen el mismo número q de elementos

Pregunta: ¿Existe en M un acoplamiento o matching M', tal que $M' \subseteq M$, $|M'| = q$ y que ningún par de elementos en M' tenga algún elemento en común?

[23] Problema de partición [74]

Instancia: Un conjunto finito A y un formato $s(a) \in \mathbb{Z}^+$ para cada $a \in A$

Pregunta: ¿Existe un $A' \subseteq A$, tal que $\sum_{a \in A'} s(a) = \sum_{a \in A - A'} s(a)$?

[24] Problema de suma de subconjuntos [74]

Instancia: Un conjunto finito A y un formato $s(a) \in \mathbb{Z}^+$ para cada $a \in A$, y un entero B

Pregunta: ¿Existe $A' \subseteq A$, tal que la suma de los formatos en A' es B?

- Seleccionar un problema Π_1 el cual se sabe es NP-completo.
- Construir una transformación f entre Π_1 y Π_2, tal que $\Pi_1 \propto \Pi_2$.
- Probar que f es una transformación polinomial.

Los problemas $co - NP$-Completos son los problemas complemetarios a los problemas NP-Completos. En la Figura 1.22 se muestra la conjetura más común de la relación entre los problemas NP-Completos y los $co - NP$-Completos.

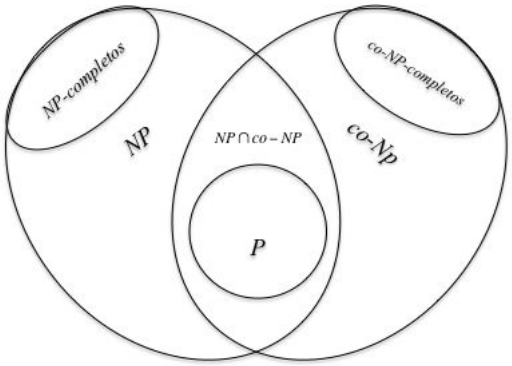

Figura 1.22: Conjeturas entre las relacion entre NP-Completos y $co - NP$-completos [121]

Aquellos problemas en NP para los cuales no se tiene la certeza de que pertenezcan a P ni a NP-completo se les denomina problemas $NP - intermedios$ [247]. En otras palabras, si $P \neq NP$ existe un conjnto no vacío en NP, el cual se forma por $NP \setminus (P \cup Np - Completos)$, a dicho conjunto se le denomina $NP - intermedios$ [52]. La idea de $NP - intermedio$ fue propuesta por Ladner.

A continuación, se muestran algunos de los problemas que se presume son $NP - intermedios$

- Problema de isomorfismo de gráficas

  Instancia: Un par de gráficas $G_1 = (V, E_1)$ y $G_2 = (V, E_2)$

  Pregunta: ¿Existe un mapeo f entre los vértices ($f : V \rightarrow V$), tal que $\{u, v\} \in E_1 \Leftrightarrow \{f(u), f(v)\} \in E_2$?

- Descomposición en factores primos

  Instancia: Dados los números naturales m y n, tales que $m < n$

Pregunta: ¿n tiene un factor primo que sea mayor o igual que m?

- Logaritmo Discreto

  Instancia: Dada los numeros naturales g, h, k, n, tales que $k < n$

  Pregunta: ¿Existe un número e , tal que $g^e = h$ en modulo n?

El concepto de problemas $NP - intermedios$ ha sido el fundamento para el desarrollo de algoritmos cuánticos; para mayor información sobre este tema véase [247, 52, 171]

g) NP-Duros

La idea de problemas $NP - Duros$ es importante en la teoría de complejidad computacional. El conjunto de problemas $NP - Duros$ se intregra por aquellos problemas que son al menos tan difíciles como un problema de NP; por ende, se dice que un problema Π es $NP - Duro$ si todos los algoritmos exactos conocidos para resolver dicho problema tienen una complejidad temporal exponencial para el peor caso [65]. A continuación, se formaliza el concepto de $NP - Duro$.

Definición 50 *Un problema de optimización o un problema de decisión* Π *es llamado* $NP - Duro$ *si todo problema en* NP *pueden ser transformados polinomialmente a* Π *[121]. En otras palabras, dado un problema* Π *con lenguaje* L *y cualquier problema* $\Pi' \in NP$ *con lenguaje* L'*; si existe una transformación polinomial de* L' *a* L *(*$L' \propto L$*), entonces se dice que* Π *es* $NP - Duro$.

Cabe mencionar que no todos los problemas NP son $NP - Duros$ ni todos los problemas $NP - Duros$ son NP . En la Figura 1.23, se muestran la posible relacion entre $NP-Duro$, $NP-Completos$ y NP.

Se dice que un problema Π es $NP - fácil$ si Π puede transformarse polinomialmente a cualquier problema NP. Si un problema Π es simultáneamente $NP - fácil$ y $NP - Duro$; entonces, se dice que Π es $NP - equivalente$ [121].

Los problemas de optimización asociados a problemas de decisión NP conforman el grupo NPO[25]; dichos problemas de optimización son al menos tan difíciles como el problema de la clase NP [74, 129].

[25]El conjunto de problemas NPO se integra por los problemas de optimización que pueden ser resueltos por un algoritmo no determinístico en un tiempo polinomial.

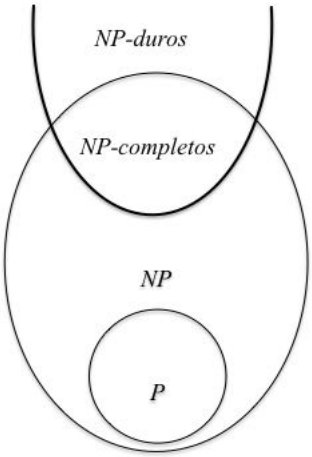

Figura 1.23: Conjeturas entre las relacion entre NP, $NP-Duros$ y $NP-Completos$ [163]

1.1.1.6. Localidad y Descomposición

Se puede entender la **localidad** de un problema como una relación entre la distancia $d(s_i, s_j)$ de dos soluciones $s_i, s_j \in S$ y la diferencia de los valores obtenidos de la función objetivo de s_i y s_j $(f(s_i) - f(s_j))$ [210].

Se dice que una instacia tiene alta localidad si las soluciones en el vecindario de las soluciones tienen un valor similar en la función objetivo; en contraste, se dice que una instacia tiene baja localidad si el vecindaro de las soluciones tiene un valor diferente en la función objetivo.

La localidad de un problema impacta de forma significativa los resultados que se pueden obtener de métodos de búsqueda guiados. Se requiere una alta localidad para el buen funcionamineto de un método de búsqueda .

Se dice que un problema Π es descomponible, si éste puede descomponerse en pequeños problemas independientes uno del otro, los cuales son fácilmente resolubles de modo individual y que en conjunto permiten encontrar la solución de Π. Los problemas multietapa, los problemas de programación lineal de gran tamaño y los problemas estocásticos de gran tamaño son ejemplos de problemas cuya resolución se puede abordar mediante técnicas de descomposición.

- Problemas descomponibles.

Resolver la derivada:

$$\frac{\delta(x^2+2x)}{\delta(x)} = \frac{\delta(x^2)}{\delta(x)} + \frac{\delta(2x)}{\delta(x)}$$

- Problemas no descomponibles.

  El problema de apilar y desapilar bloques no es descomponible porque los subproblemas no son independientes entre sí.

A continuación, se describen algunas de las técnicas de descomposición.

- Descomposición de Benders: propone separar en subproblemas las decisiones tomadas en diferentes etapas; por ende, se necesita que las decisiones de una etapa sólo dependan de las consecuencias de las decisiones tomadas en la etapa anterior.

- Relajación lagrangiana: consiste en eliminar las restricciones complicadas de problema, de forma tal que se puedan resolver independientemente las distintas partes del problema

1.2. Diseño de algoritmos

En secciones previas, se ha utilizado la idea de algoritmo; sin embargo, antes de continuar resulta necesario definir formalmente qué es "algoritmo" para un problema de optimización.

Definición 51 *Sea* Π *un problema de optimización definido como:* $\Pi = (X, (S_x)_{x\in X}, c, meta)$*, donde:* X *es un lenguaje decidible sobre* $\{0,1\}$ *en tiempo polinomial;* S_x *es un subconjunto de* $\{0,1\{^\star$ *para cada* $x \in X$*, existe un polinomio* p *tal que* $magnitud(y) \leq p(magnitud(x))$ *para todo* $y \in S_x$ *y todo* $x \in X$ *y el lenguaje* $\{(x,y)|x \in X, y \in S_x\}$ *y* $\{x \in X|S_x = \emptyset\}$*es decidible en tiempo polinomial;* $c : \{(x,y)|x \in X, y \in S_x\} \longrightarrow \mathbb{Q}$ *es una función computable en tiempo polinomial; por útimo meta es maximizar o mínimizar.*

Un ***algoritmo*** *para un problema de optimización* $\Pi = (X, (S_x)_{x\in X}, c, meta)$ *es un algoritmo capaz de calcular para cada entrada* $x \in X$ *con* $S_x \neq \emptyset$ *una solución factible* $y \in S$*. Se dice que un algoritmo es* ***exacto*** *si para todo* $x \in X$ *con* $S_x \neq \emptyset$ *se encuentra* $c = \{(x,y)\} = OPT(x)$*.*

Todo algoritmo es un procedimiento computable, si cumple las siguientes caracterśticas:

- **Finito** Todo algoritmo debe tener un número finito de pasos.
- **Preciso** Todo algoritmo debe indicar a detalle el orden en que se realiza cada uno de los pasos; es decir, un algoritmo debe ser descrito con claridad de tal forma que permita entenderlo y leerlo facilmente.
- **Definido** Todo algoritmo debe evitar la ambigüedad en los pasos.
- **Contar con entrada y salida** La entrada se conforma por los datos necesarios por el algoritmo para su ejecución. La salida se conforma por los resultados obtenidos por el algoritmo.
- **Repetible** Si un algoritmo se repite varias veces con los mismos datos de entrada, los resultados deben ser los mismos. Esta propiedad permite verificar a todo algoritmo.
- **Efectividad** Todo algoritmo debe llegar al objetivo para el que fue diseñado.

Ahora bien, una vez que se ha definido un algoritmo y sus características, se abordará el tema de diseño de algoritmos. De manera general, se define al diseño de algoritmos como el área del conocimiento en la que se conjuntan las teorías y técnicas implicadas en la construcción de algoritmos para la solución de problemas. Además, el análisis de algoritmos permite establecer propiedades sobre la eficiencia de los algoritmos; lo cual, posibilita la comparación entre alternativas.

En las subsecciones siguientes se abordaran algunos de los conceptos implicados en el diseño de algoritmos; así como el conjunto de técnicas usadas con este propósito

1.2.1. Complejidad algorítmica

La **complejidad algorítmica** es una métrica teórica para estimar el costo en tiempo y/o espacio de un algoritmo; la **complejidad temporal** es el tiempo que emplea un algoritmo en ejecutarse dada alguna entrada.

Teorema 21 *(**Teorema de Böhm-Jacopini**) Todo procedimiento computable puede ser implementado en un lenguaje de programación que combine el uso de tres estructuras básicas de control [165]; las cuales son: secuenciales, de decicsión e iterativas.*

Se puede decir, de manera general que cada una de las estrategias básicas se compone a su vez de un conjunto de operaciones. A continuación, se describen las operaciones básicas más comúnmente utilizadas en los algoritmos.

- Asignación: Establecer un valor a alguna variable.
- Operaciones aritméticas: son operaciones básicas aritméticas (suma, resta, multiplicación y división).
- Operaciones lógicas: Es la comparación entre dos números.

A continuación, se muestra el costo de algunas de las instrucciones más usadas en los algoritmos.

- El costo de constantes, comentarios y declaración de variables es de 0.
- El costo de expresiones y asignaciones es la suma de las operaciones implicadas, cada una de las expresiones tiene un costo de 1 (si se llama a una función se sumará el costo de la llamada)
- En sentencias condicionales (If, $Else$), el costo será la suma de las expresiones para uno u otro caso.
- En sentencias iterativas simples (For , $While$), el costo será el producto de los ciclos necesarios.

La **complejidad de espacio** es una métrica, la cual se centra en tratar de determinar la cantidad de memoria necesaria para ejecutar un algoritmo hasta llegar a completar una tarea. El avance tecnológico proporciona hoy en día un incremento sustancial en la cantidad de memoria disponible, por lo que, generalmente el análisis de los algoritmos se centra en determinar el costo temporal de los procedimientos. El análisis de la eficiencia temporal de los algoritmos consta de dos fases:

- El análisis a posteriori ofrece una medida real, consiste en medir el tiempo de ejecución del algoritmo para unos valores de entrada dados y en un ordenador concreto.
- El análisis a priori proporciona una medida teórica, que consiste en obtener una función de acotamiento (por arriba y/o por abajo) del tiempo de ejecución, para unos valores de entrada dados. Esta medida ofrece estimaciones del comportamiento de los algoritmos de forma independiente del ordenador.

El orden de los algoritmos es una función ($f(n)$) que acota asintóticamente, ya sea superior (O grande), inferior (Ω) o superior e inferiormente (Θ) a la función estimada de complejidad temporal de un algoritmo ($g(n)$), para una determinada entrada a partir de una n_0.

Generalmente, se utiliza la notación O grande para referirse al orden de un algoritmo, la función $f(n)$ es asintóticamente superior a la función de complejidad temporal $g(n)$.

1.2.2. Recursión e Iteración

Generalmente, se dice que un objeto es recursivo si se implica en su construcción o definición al de objeto mismo; en términos matemáticos una **función es recursiva** [26] si posee la propiedad de llamarse a sí misma de manera directa o indirecta (a través de otra función) [53]. En contraste, en la iteración se construye o se define un objeto por la agregación correlativa de las partes desde la primera hasta la última; en términos matemáticos, una **función iterada** [27] es una función compuesta consigo misma, en forma repetida, en un proceso llamado iteración.

Un método iterativo trata de resolver un problema (como una ecuación o un sistema de ecuaciones) mediante aproximaciones sucesivas a la solución, empezando desde una estimación inicial. En el Algoritmo 2 se muestra un método iterativo .

En contraste, un método recursivo está definido a partir de sí mismo, además estos métodos

[26] Las funciones recursivas se definen a partir de las funciones primitivas recursivas; ya que éstas permiten desarrollar funciones parciales, agregando el operador de búsqueda no acotadas.

Definición 52 *(Funciones recursivas primarias I) Se dice que la función $f(x_1, x_2, \ldots, x_n)$, está compuesta por $g(x_1, x_2, \ldots, x_m)$ y $h_i(x_1, x_2, \ldots, x_n)$ con $1 \leq i \leq m$, si para todos los números naturales $x_1, \ldots, x_n$ se satisface $f(x_1, x_2, \ldots, x_n) = g[h_1(x_1, x_n) \ldots h_m(x_1, x_n)]$ [136].*

Definición 53 *(Funciones recursivas primarias II) Se dice que la función $f(x_1, x_2, \ldots, x_n)$ es recursiva primitiva de $g(x_1, \ldots, x_{n-1})$ y $h(x_1, \ldots, x_{n+1})$ si para todos los números naturales k, $x_2, \ldots, x_n$ se satisfce que $f(0, x_2, \ldots, x_n) = g(x_2, \ldots, x_n)$ y $f(k+1, x_2, \ldots, x_n) = h[k, f(k, x_2, \ldots, x_n), x_2, \ldots, x_n]^n$ [136]*

[27] Una función iterada en un conjunto X se define formalmente como:

Definición 54 *Dado un conjunto X y sea $f : X \to X$ una función. Se define el iterado $n-$ésimo (f^n) de f mediante la función identidad (f^0) de X y $f^{n+1} = f \circ f_n$*

Algoritmo 2: Método iterativo para cálcular la potencia de un número real elevado a un entero

Input: Un número $a \in \mathbb{R}$ que será la base y un número $b \in \mathbb{Z}$ que será el exponente

Output: Un número c que será el valor de elevar la base a a la potencia b

```
1 c=1
2 for i = 1; i ≤ b; i++. do
3 |   c = c * a
4 end
```

presentan, básicamente las siguientes características: a) autorreferencia; b) se requiere una condición de paro -la cual evita que el algoritmo genere llamadas infinitas sobre sí mismo -; y c) existe un caso general del algoritmo en el que se evidencia la autoreferencia. En el algoritmo 3 se musetra un método recursivo .

Algoritmo 3: Método recursivo para cálcular la potencia de un número real elevado a un entero

Input: Un número $a \in \mathbb{R}$ que será la base y un número $b \in \mathbb{Z}$ que será el exponente

Output: Un número c que será el valor de elevar la base a a la potencia b

```
1 Llamar a la función potencia potencia(a, b).
2 if b = 0 then
3 |   c = 1
4 end
5 c = a * potencia(a, b − 1)
```

Los métodos iterativos y recursivos son fácilmente confundibles ya que ambos usan estructuras de control (véase Teorema 21); la diferencia entre ambos estriba en que los métodos iterativos emplean una estructura de repetición; en contraste, los métodos recursivos involucran una estructura de selección.

Principio 1.1 *Para todo método de solución recursivo se puede encontrar un método solución iterativo equivalente, mientras que lo contrario no siempre es cierto.*

Hay que considerar varios aspectos al decidir si se implementa un algoritmo en forma iterativa o de forma recursiva; algunos de los cuales son: a) carga computacional -la recursión de un algoritmo conlleva una repetida invocación de algoritmo; por lo cual generalmente, se incurre en un gasto de tiempo y de memoria que más elevada que la versión iterativa del algoritmo- ; b) redundancia; c) complejidad del algoritmo; d) concisión; e) legibilidad; entre otros.

A continuación, se describen las técnicas más utilizadas en el diseño de algoritmos.

1.2.3. Técnicas para el diseño de algoritmos

1.2.3.1. Estrategia divide y vencerás

La estrategia divide y vencerás constituye una herramienta poderosa para el diseño de algoritmos. Esta estrategia consiste en dividir el problema Π en un conjunto de problemas independientes ($\pi = \{\Pi_1, \ldots, \Pi_n\}$) más pequeños y posteriormente, construrir la solución de Π a partir de resolver el conjunto de problemas π. Es decir, la estrategia "divide y vencerás" consiste en 'fraccionar" un problema Π de tamaño n en problemas más pequeños de tal forma que la solución de dichos problemas permita construir fácilmente una solución del problema original.

En el Algoritmo 4, se muestra la estructura, habitual, de la estrategia divide y vencerás.

Un problema pueden ser abordado a través de la técnica divide y vencerás, si este problema posee las siguientes características:

- El problema es descomponible en un conjunto de subproblemas independientes y disjuntos.
- La solución final se puede expresar como la combinación de las soluciones de los subproblemas.

A continuación, se muestra un ejemplo de aplicación de la estategia divide y vencerás para resolver el problema de ordenamiento de datos.

Ejemplo 1.8

Problema de ordenamiento.

Instancia: Un conjunto $A = \{a_1, a_2, \ldots, a_k\}$ *de* k *números.*

Pregunta: Determinar una permutación (reordenamiento) $a'_1, a'_2, \ldots, a'_k$ *del conjunto original, tal que* $a'_1 \leq a'_2 \leq \ldots \leq a'_k$.

Algoritmo 4: Método general "divide y vencerás".

Input: Un problema Π

Output: Solución del problema Π

```
1  if Π es suficientemente pequeño then
2      Utilizar un algoritmo específico para Π.
3  else
4      Descomponer Π en π = {Π_1, ..., Π_n}.
5      for i = 1; i ≤ n; i++ do
6          Resolver el problema Π_i a través de aplicar recurrentemente el algoritmo divide y vencerás.
7          Asignar a y_i la solución obtenida de Π_i.
8      end
9      Recombinar (y_1, ..., y_n) para obtener la solución del problema Π.
10     Asignar a y la solución obtenida de Π.
11 end
```

El problema de ordenamiento cumple con las características necesarias para aplicar la estrategia divide y vencerás; ya que el problema de ordenar una lista se puede dividir en problemas de ordenación parcial de sus sublistas. Existe un tamaño crítico ($k = 2$ o $k = 3$) para el cual una lista parcialmente ordenada es, además, una lista ordenada.

En este ejemplo, la instancia del problema de ordemaniento a resolver es: encontrar un ordemamiento creciente del conjunto $A = 9, 6, 5, 4, 9, 2$. Para solucionar esta instancia se utilizara la estategia divide y vencerás, a través de implemetar el algoritmo Quick sort (véase algoritmo 5).

En la Figura 1.24 se muestra la aplicación del algoritmo Quick sort para odenar la lista $A = 9, 6, 5, 4, 9, 2$; a través del cual, se obtiene la lista ordenada $A' = 2, 4, 5, 6, 9, 9$.

1.2.3.2. Programación dinámica

La programación dinámica fue introducida por R. Bellman en 1957 en su libro "Dynamic Programic" [237]. Debe verse a la programación dinámica como un principio general aplicable a problemas de optimización , el cual utiliza la propiedad de descomponibilidad (véase sección 1.1.1.6) y el principio de optimalidad (véase principio 1.2) para obtener una relación de recurrencia. En otras palabras,

Algoritmo 5: Algoritmo Quick sort

Input: Un conjunto de números $a_l, a_{l+1}, \ldots, a_f$

Output: Una secuencia ordenada $a_l, a_{l+1}, \ldots, a_f$

```
1  if l ≥ f then
2      Return
3  else
4      X = a_l
5      i = l + 1
6      j = f
7      while i < j do
8          while (a_j ≥ X) ∧ (j ≥ l + 1) do
9              j = j − 1
10         end
11         while (a_i ≤ X) ∧ (i ≤ l + 1) do
12             i = i + 1
13         end
14         if i < j then
15             a_i ↔ a_j
16         end
17     end
18 end
```

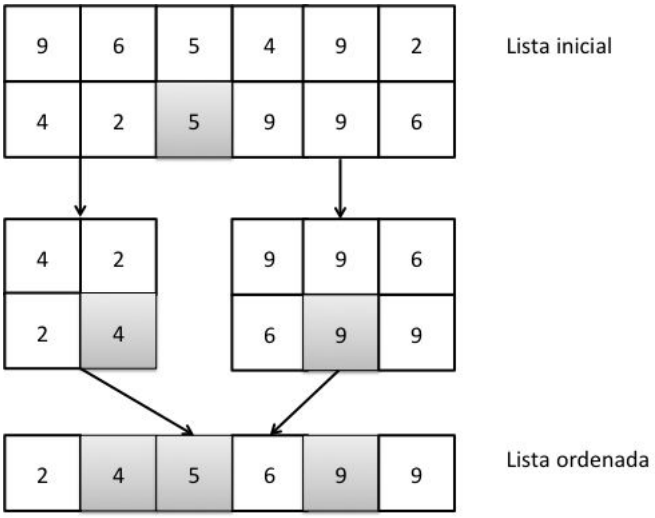

Figura 1.24: Aplicación del algoritmo Quick sort

la programación dinámica consiste en descomponer un problema en subproblemas, cada uno de los cuales se resuelve aplicando de manera recurente el mismo método [129]

Principio 1.2 *Una política óptima tiene la propiedad de que, cualesquiera que sean los estados y decisiones iniciales (es decir el control), las decisiones restantes deben constituir una política óptima con respecto al estado resultante del la primera decisión [237]. En otras palabras, las decisiones futuras para las etapas* [28] *restantes formarán una política óptima independiente de las políticas adopadas en etapas anteriores [232].*

Por medio de la programación dinámica es posible resolver problemas complejos a través de descomponerlo en una serie de problemas más simples. Para que un problema pueda ser resuelto con la técnica de programación dinámica, debe cumplir con ciertas características [248, 232, 15]:

- El problema se puede dividir en etapas ($t = 1, 2, \ldots, T$) y en cada una de ellas se toma una decisión.
- Cada etapa tiene asociada un estado [29] ($S(t) = 1, 2, \ldots, T$) relacionado con el estado previo.

[28] Cada etapa debe tener asociado una o más decisiones (problema de optimización), cuya dependencia de las decisiones anteriores está dada exclusivamente por lasvariables de estado

[29] Por estado entiéndase la información que se necesita en cualquier etapa para tomar una decisión óptima [248]

- La decisión tomada en cualquier etapa describe el modo en que el estado en la etapa actual se transforma en el estado en la etapa siguiente.
- Dado el estado actual, la decisión óptima para cada una de las etapas restantes no tiene que depender de los estados ya alacanzados o de las decisiones tomadas previamenete.
- Si los estados del problema se clasifican dentro de uno de T etapas, debe existir una **recursión** que relacione el beneficio o costo obtenido de las etapas anteriores con el costo o beneficio generado en las etapas subsecuentes.

En el Algoritmo 6, se muestran los pasos para desarrollar un algoritmo basado en programación dinámica para resolver un problema.

Algoritmo 6: Método para desarrollar un algoritmo basado en programación dinámica

Input: Un problema Π a resolver

Output: Una solución al problema basada en programación dinámica

```
1  if El problema es descomponible then
2      Dividir el problema en subproblemas.
3      if Es posible aplicar el principio de optimalidad. then
4          Determinar la relación de recurrencia.
5          Determinar el conjunto de subproblemas distintos a resolver.
6          Identificar los subproblemas con solución trivial.
7          Resolver los subproblemas con un enfoque ascendente utilizando para ello la relación de
           recurrencia determinada.
8          Determinar la solución óptima del problema Π basado en la información previamente
           calculada.
9      else
10         No se puede aplicar la programación dinámica para reslover este problema.
11     end
12 else
13     No se puede aplicar la programación dinámica para reslover este problema.
14 end
```

En el ejemplo 1.9 se muestra la aplicación de la programación dinámica para resolver una instancia del problema de la mochila binario.

Ejemplo 1.9

De manera general, se puede entender al problema de la mochila binario como: "dada una mochila de capacidad C y un conjunto de n objetos, cada uno de ellos tiene un peso p_i y un valor asociado b_i, encontrar la combinación de objetos que puedan ser trasportados en la mochila de tal forma que se maximice el valor de dicha combinación ". El modelo general del problema de la mochila binario es el siguiente:

$$\begin{gathered} \text{máx} \sum_{1 \leq i \leq n} b_i x_i \\ \textit{sujeto a:} \\ \sum_{1 \leq i \leq n} p_i x_i \leq C \\ x_i \in \{0,1\} \wedge x_i \in \mathbb{Z} \end{gathered} \tag{1.2.1}$$

donde: b_i es el beneficio del objeto i; p_i es el peso del objeto i; x_i es la decisión de seleccionar del objeto i, la cual es 1 si se selecciona y 0 en otro caso; C es la capacidad de la mochila.

En este ejemplo la instancia del problema de la mochila a resolver es: Dada una mochila con capacidad 6 kg y un conjunto de cuatro objetos, los pesos y beneficios de estos objetos; mostrados en la Tabla 1.4, encontrar la combinación de objetos que maximice los beneficios. En el Algoritmo 7, se muestra un método para resolver el problema de la mochila binario basado en una estrategia de programación dinámica.

Tabla 1.3: Peso y beneficio de cuatro objetos.

Objeto	Peso kg	Beneficio
A	1	2
B	3	7
C	1	8
D	2	12

El resultado implica que $x_1 = 0, x_2 = 1, x_3 = 1$ y $x_4 = 1$, con un beneficio de 27 unidades y un peso de 6 kg.

Algoritmo 7: Algoritmo basado en Programación dinámica para resolver el problema de la mochila

Input: La capacidad de la mochila C, un conjunto de n, cada uno ligado a un beneficio b_i y un peso p_i

Output: La combinación óptima de objetos que maximizan el beneficio

```
1  g_{n,C} ← ∅
2  for i = 0; i ≤ n; i++ do
3  |   g_{i,1} = 0;
4  end
5  for j = 0; j ≤ C; j++ do
6  |   g_{1,j} = 0;
7  end
8  for i = 1; i ≤ n; i++ do
9  |   for j = 1; j ≤ C; j++ do
10 |   |   if j < p_i then
11 |   |   |   g_{i,j} = g_{i-1,j} ;
12 |   |   else
13 |   |   |   if g_{i-1,j} ≥ g_{i-1,j-p_i} + b_i then
14 |   |   |   |   g_{i,j} = g_{i-1,j}
15 |   |   |   else
16 |   |   |   |   g_{i,j} = g_{i-1,j-p_i} + b_i
17 |   |   |   end
18 |   |   end
19 |   end
20 |   Rastrear en g la solución del problema
21 end
```

Tabla 1.4: Resultados obtenidos por el algoritmo 7

Objeto	Peso						
	0	1	2	3	4	5	6
	0	0	0	0	0	0	0
A	0	2	2	2	2	2	2
B	0	2	2	7	9	9	9
C	0	8	8	8	15	17	17
D	0	8	12	20	22	22	27

1.2.3.3. Algoritmos voraces

Los algoritmos voraces -también llamados algoritmos glotones, algoritmos codiciosos o algoritmos *greedy* por su nombre en inglés- son comunmente utilizados para encontrar la solución de problemas de optimización. Estos algoritmos son fáciles de diseñar y eficientes al encontrar un solución rápida en problemas.

Los algoritmos voraces son miopes; ya que, toman decisiones con la información que tienen disponible de forma inmediata, sin tener en cuenta sus efectos futuros; es decir, estos algoritmos pueden tomar decisiones sin tener en cuenta consecuencias futuras.

Típicamente, los componentes característicos de los algoritmos voraces son:

- *Conjunto de candidatos*: lo conforman todos aquellos elementos que pueden formar parte de la solución.
- *Conjunto de seleccionados*: lo conforman todos aquellos elementos considerados y seleccionados para la solución.
- *Conjunto de rechazados*: lo conforman todos aquellos elementos examinados y eliminados para la solución.
- *Función solución*: determina si un conjunto de elementos es o no una solución del problema (ignorando si es óptima o no).

- *Función de selección* : permite escoger al candidato que sea el más prometedor.
- *Función factible*: comprueba si es posible completar el conjunto añadiendo algún otro candidato.
- *Función objetivo*: determina el valor (u costo) de una solución y es la que se pretende maximizar o minimizar.

En el Algoritmo 8 se muestra la estructura general de los algoritmos voraces.

Algoritmo 8: Algoritmo voraz genérico

Input: Instancia del problema Π

Output: Solución de Π obtenida por un algoritmo voraz

```
1  C_C ← conjunto de candidatos.
2  C_S ← ∅ % lista de solución.
3  C_R ← ∅ % lista de rechazados
4  while C_c ≠ ∅ y C_S no este completa do
5      Seleccionar un elemento x de C_C
6      if C_S ∪ x es viable then
7          Añadir x a C_S
8          Eliminar x de C_C
9      else
10         Añadir x a C_R
11         Eliminar x de C_C
12     end
13 end
14 if C_S es solución then
15     C_S es solución del problema Π
16 else
17     Regresar "No existe una solución del problema Π"
18 end
```

En el ejemplo 1.10 se muestra la aplicación de un algoritmo voraz (algoritmo de Prim) para

resolver una instancia del problema de árbol de expansión mínima.

Ejemplo 1.10

El problema del árbol de peso mínimo es: dada una gráfica $G = (V, E)$ ponderada encontrar un árbol de expansión [30] *no dirigido para el cual el peso total de las aritas en el árbol sea el menor posible. Considérese la siguiente definición:*

Definición 55 *El peso de un árbol generador es la suma de los pesos de sus ramas. Un árbol generador mínimo es un árbol con peso mínimo.*

Para resolver este problema se han desarrollado varias técnicas como el algoritmo Prim; el algoritmo de Kruskal entre otros.

La instacia a resolver en este ejemplo es: Una pequeña ciudad, en el norte del país, tiene un sistema de calles que comunican las 10 áreas de recreación y emblematicas de la ciudad, dichas áreas son: A) mirador, B) fuerte, C) parque , D) ayuntamiento, E) iglesia, F) parque botánico, G) presa, H) mercado, I) cuevas con pinturas rupestres, J) museo. El sistema modelado, a través de una gráfica ponderada, se muestra en la Figura 1.25, donde el peso representa la distancia en kilometros que hay entre los sitios.

Para resolver esta intancia, se utiliza el algoritmo de Prim (véase algoritmo 9)

En las Figuras 1.26, 1.27, 1.28 y 1.29, se muestra la aplicación el algoritmo de Prim en la instancia de interés.

El árbol encontrado involucra un peso de 29 kilometros.

1.2.3.4. Método vuelta atrás

El método de vuelta -también conocido como *Backtracking* ,por su nombre en inglés-, puede ser aplicado en la solución de un gran número de problemas.

Esta estrategia, básicamente, consiste en una búsqueda sistemática a través de todas las configuraciones posibles dentro de un espacio de búsqueda. Por ende, esta estrategia es de naturaleza recursiva y construye la solución con base al conjunto de soluciones parciales generadas; similar al

[30] Dada una gráfica $G = (V, E)$, un árbol de expansión esta compuesto por todos los vértices y algunas de las aristas de G

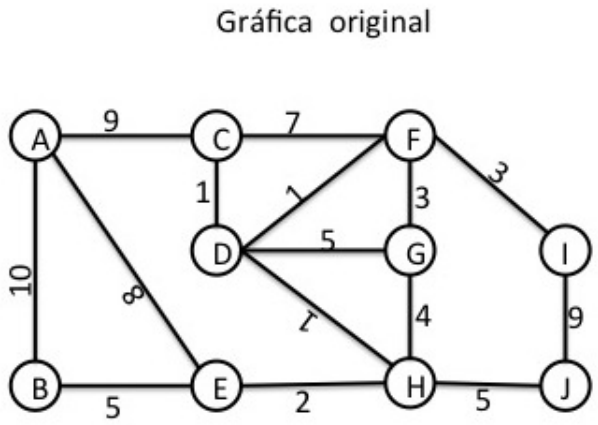

Figura 1.25: Modelo de la ciudad.

Algoritmo 9: Algoritmo de Prim [96]

Input: Una gráfica conexa $G = (V, E)$

Output: Un árbol de peso mínimo

$n = |V|$ Elegir al azar un vértice $v_1 \in V$

$N = V - v_1$

$P = v_1$ $T = \emptyset$

while $|T| < n - 1$; **do**

    Seleccionar la arista con menor peso de G que conecte a un vertice $v_x \in P$ con un vertice v_y en N.

    if *(v_x, v_y no forma ciclos con los miembros en T)* **then**

        Añadir v_y a P ($P = P \cup v_y$).

        Añadir v_x, v_y al árbol ($T = T \cup (v_x, v_y)$).

    end

    Borrar v_y de $N(N = N - v_y)$

end

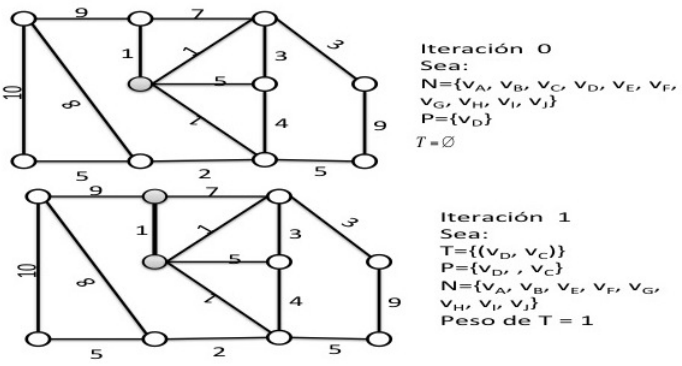

Figura 1.26: Aplicación el algoritmo de Prim.

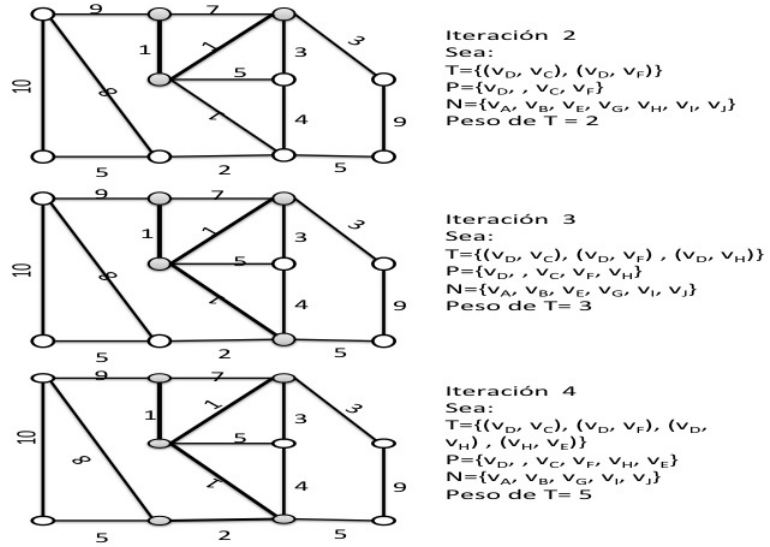

Figura 1.27: Aplicación el algoritmo de Prim (continuación).

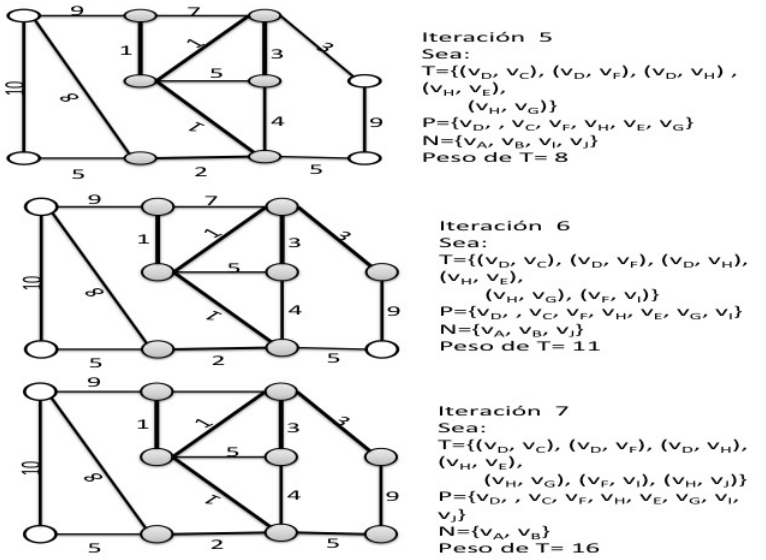

Figura 1.28: Aplicación el algoritmo de Prim (continuación).

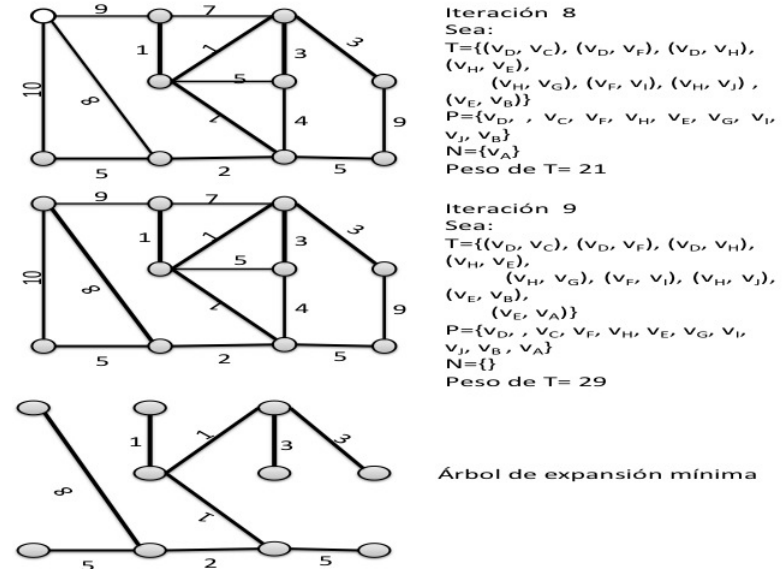

Figura 1.29: Aplicación el algoritmo de Prim (continuación).

recorrido en profundidad en árbol expansión; como resultado de este recorrido se puede llegar a una solución del problema (una hoja del árbol), o bien, se llega a una solución parcial del problema que no se puede acompletar (un nodo fracaso); en este caso se deben eliminar los elementos añadidos a esta solución iniciando con el más reciente y terminando con el elemento insertado en primer lugar.

Las características típicas de los problemas que se pueden resolver por el método de vuelta atrás son:

- La solución final debe ser expresable mediante una secuencia de decisiones.
- El problema puede requerir una: a) una solución factible, b) el conjunto de todas las soluciones fatibles y c) la solución óptima.

Los componentes característicos del método vuelta atrás son [57]:

- **Conjunto de restricciones explícitas**: Conjunto de reglas que definen es el dominio x_i de cada decisión.
- **Conjunto de restricciones implícitas**: Conjunto de reglas que definen las relaciones entre los elementos del problema.
- **Soluciones posibles**: Conjunto de soluciones que satisfacen todas las soluciones explícitas del problema.
- **Soluciones factibles**: Subconjunto de soluciones posibles que satisfacen todas las soluciones explícitas e implícitas del problema.
- **Soluciones óptimas**: Subconjunto de soluciones factibles con el que se encuentra el mejor resultado posible de la función objetivo.

El método de vuelta atrás no sigue reglas fijas en la búsqueda de las soluciones. Es más bien, un proceso de prueba y error que construye gradualmente una solución. En el Algoritmo 10 se muestra la estructura general de método de vuelta atrás.

A continuación, se ejemplifica el uso de la estategia del método vuelta atrás para reslover una instacia del problema del número mínimo de monedas.

Algoritmo 10: Algoritmo general de la estategia vuelta atrás

Input: Un problema Π

Output: Solución del problema Π

```
1  S_S = ∅
2  % conjunto de soluciones S_C = ∅
3  % conjunto de candidatos. S_P ⟵ conjunto de soluciones posibles
4  Seleccionar una nueva opción x ∈ S_P.
5  Exito = 0
6  Incluir x de S_C. while Exito ≠ 1 ∨ |S_P| ≠ 0 do
7      if x es aceptable then
8          Incluir x en S if S es una solución incompleta then
9              Generar todos los candidatos de la solución que son extensión de x.
10             Incluir todos los candidatos en S_C
11             S_P = S_P − x
12             Seleccionar una nueva opción x ∈ S_C.
13             Reutilizar el método vuelta atrás para acompletar la solución.
14         else
15             Procesar la solución S Exito = 1
16         end
17     else
18         Eliminar todos los candidatos en S_C que incluyan x
19         Eliminar de S_P
20     end
21 end
```

Ejemplo 1.11

El problema del número mínimo de monedas se define como: "dadas n de monedas, de valores $v_1, v_2, \ldots, v_n$ respectivamente y una cantidad C, se debe determinar el número mínimo de monedas necesarios para sumar C " . El algoritmo 11 es un procediemiento basado en la estategia vuelta atrás para resolver el problema del número mínimo de monedas.

Algoritmo 11: Algoritmo de número mínimo de monedas

Input: Un conjunto de n y una cantidad C

Output: Si existe, el mínimo número de monedas cuya suma es C

```
1  suma = 0
2  long = 0
3  long_optima = n + 1
4  k = 1
5  sec = ∅
6  opt = ∅
7  if suma = C then
8      if long < long_optima then
9          opt ← sec
10         long_optima = long
11     end
12 else
13     while (k ≤ n) ∧ (1 + long < long_optima) ∧ (suma + v_k ≤ C) do
14         sec[1 + long] ← v_k
15         Usar con el resto de monedas, tomando el resultado de la casilla superior.
16         Usar con una moneda del tipo actual y el resto con el resultado que se hubiera obtenido al
           utilizar la cantidad actual a la que se le ha restado el valor de la moneda actual.
17         k = k + 1
18     end
19 end
```

La instancia del problema del número mínimo de monedas a resolver es: dadas 4 monedas de

1,2,2,5 pesos respectivamente; encontrar el mínimo número de monedas para igualar a 7 pesos. En la Figura 1.30, se muestra la aplicación del algoritmo 11 para resolver la instancia anterior.

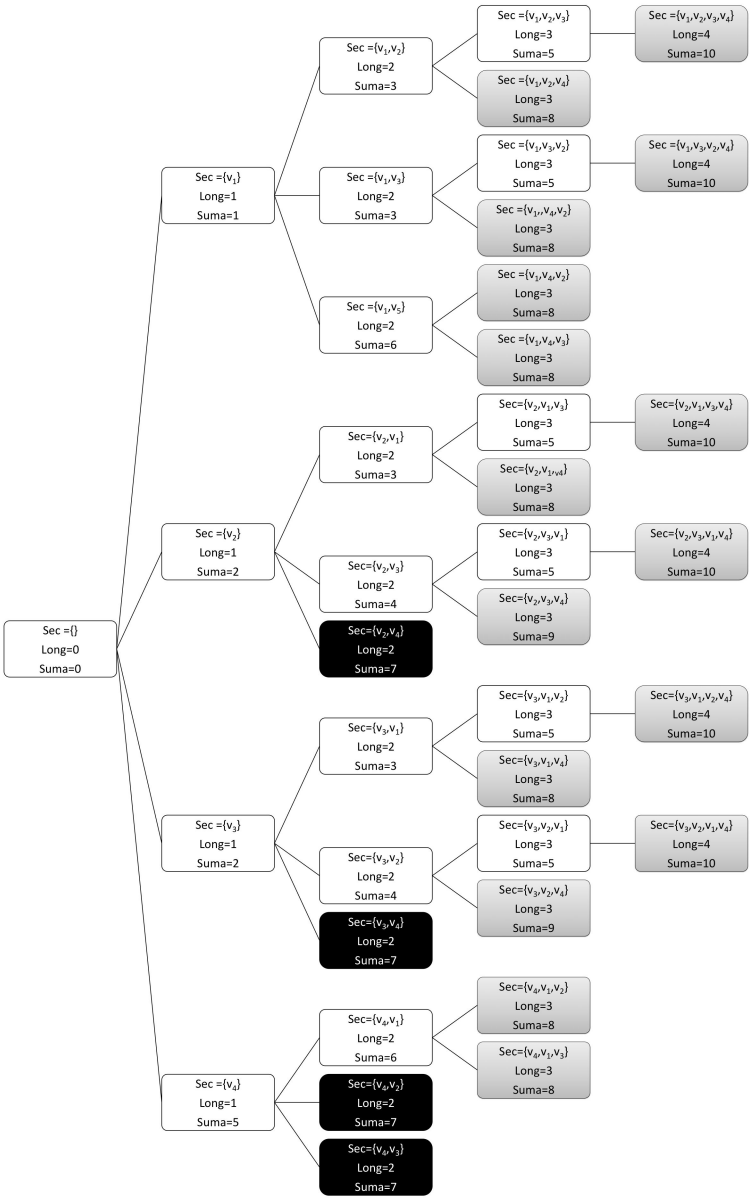

Figura 1.30: Solución al problema del número mínimo de monedas.

En la Figura 1.30, se puede observar que existen 4 soculciones factibles, las cuales son:

$(v_2, v_4); (v_3, v_4); (v_4, v_2)$ *y* (v_4, v_3) *y que el resto de hojas en el árbol de búsqueda representan soluciones infactibles, pues sobrepasan la cantidad de 7 pesos que son requeridos.*

1.2.3.5. Ramificación y poda

La estrategia de ramificación y poda, es una variante del método vuelta atrás; sin embargo, dada su importancia ˍ ya que es quizá una de las más eficientes para resolver problemas combinatorios grandes [129] -se analizará de manera particular-. Cabe mencionar que esta estategia es eficiente en casos promedios; ya que en los peores casos el espacio de soluciones tiende a ser muy grande.

Generalmente, se suele interpretar al método de ramificación y acotamiento como un árbol de soluciones, donde cada rama es una posible solución posterior a la actual; en este árbol un nodo vivo es aquel nodo que puede ser ramificado. Al conjunto de nodos vivos en un árbol que no han sido analizados hasta el tiempo t se les conoce como lista de nodos vivos (LNV); es decir, todo nodo dentro de la lista hasta el tiempo t no ha sido examinado.

Para cualquier nodo vivo i en árbol se debe conocer:

- una *cota superior* de beneficio -o costo- $(C_S(i))$ que se pueda alcanzar a partir de i;
- una *cota inferior* de beneficio -o costo- $(C_I(i))$ que se pueda alcanzar a partir de i; y
- un *ualor estimado* de beneficio -o costo estimado- $(V(i))$ que se puede encontrar a partir de i.

El método de ramificación y poda, básicamente, consta de tres etapas, las cuales son:

- **Selección**: en esta etapa se selecciona un nodo vivo x del conjunto de soluciones posibles.
- **Ramificación**: en esta etapa se construyen todos los decendientes posibles de la solución x y se colocan como nodos hijos del nodo vivo x. Tanto la etapa de selección como la de ramificacón están en función de tipo de recorrido que se realice en el árbol de búsqueda, algunos de los recorridos más comúnes son: a) seleccionar el nodo vivo con mayor beneficio y en caso de empate escoger el primer nodo introducido a la lista de nodos vivos (LC-FIFO); b) seleccionar el nodo vivo con mayor beneficio y en caso de empate escoger el último nodo introducido a la lista de nodos vivos (LC-LIFO); entre otros.

- **Acotamiento o poda** en esta etapa se eliminan algunos de los nodos creados en la etapa anterior, lo cual permite disminuir el espacio de búsqueda. En términos generales, si la rama A es peor la rama B, entonces A debe ser descartada. A continuación, se analizan las estrategias de poda:

  - En el caso de maximización se debe podar el nodo i si: El nodo i en sí representa una solución no factible.

    Si $C_S(i) \leq C_I(j)$ para algún nodo j generado con anterioridad.

    $C_S(i) \leq V(s)$ para algún nodo s generado con anterioridad, el cual es una solución del problema

  - En el caso de minimización se debe podar el nodo i si: El nodo i en sí representa una solución no factible.

    Si $C_S(i) \geq C_I(j)$ para algún nodo j generado con anterioridad.

    $C_S(i) \geq V(s)$ para algún nodo s generado con anterioridad, el cual es una solución del problema.

El método de ramificación y poda finaliza cuando encuentra la solución, o bien cuando se agota el conjunto de nodos vivos. En el Algoritmo 12, se muestra la estructura general del algoritmo de ramificación y acotamiento.

En el ejemplo 1.12 se muestra la aplicación de la estategia de ramificación y acotamiento para resolver una instancia del problema de asignación de personal .

Ejemplo 1.12

El problema de asignación de personal se define como:"dado un conjunto de n personas ($P = \{P_1, P_2, \ldots, P_n\}$) *ordenados linealmente, donde* $P_1 < P_2 < \ldots < P_n$*; y un conjunto de n trabajos* ($J = \{J_1, J_2 \ldots, J_n\}$) *parcialmente ordenados. Sea* $C_{i,j}$ *el costo de asignar* P_i *a* J_j*. Una asignacion factible asigna cada persona un trabajo, y a ningún par de personas les asigna el mismo trabajo. El problema consiste en encontrar una asignación factible tal que minimice el costo asociado ".*

Una solución para este problema puede verse como una tupla $x_{sol} = \{x_1, x_2, \ldots, x_n\}$*, donden: la*

Algoritmo 12: Método general de ramificación y poda

Input: Un problema Π

Output: Si existe, la solución del problema Π

```
1  Fijar los valores iniciales de cada variable de decisión dentro de su dominio, en función del objetivo
   de la optimización.
2  while no se haya encontrado una solución a Π o bien no queden nodos vivos do
3      if Se satisfacen el conjunto de restricciones para los valores actuales de cada variable then
4          if Aún existen valores sin explorar para alguna variable then
5              if Aún existen valores sin explorar en el dominio de la variable inicial then
6                  Asignar un valor sin explorar a la variable actual
7              else
8                  Elegir otra variable de decisión, a la cual aun sea posibile explorar su dominio.
9                  Colocar a la nueva variable como variable actual.
10                 Asignar un valor sin explorar a la variable actual.
11             end
12         else
13             No hay solución del problema Π
14         end
15     else
16         No hay solución del problema Π
17     end
18 end
```

posición del elemento x_i señala a la persona P_i; y el valor de x_i indica la tarea asignada a P_i. Por lo tanto, en una solución parcial $x_i = 0$ si a la persona P_i aun no se le asigna una tarea; en contraste, $x_i \in 1, n$ si a la persona P_i ya se le asignó una tarea.

La instacia del problema de asignación de personal abordada en este ejemplo es: Una empresa ha preseleccionado 3 candidatos para ocupar 3 puestos de trabajo en dicha empresa. Los puestos de trabajo consisten en manejar 3 máquinas diferentes (un trabajador para cada máquina). La empresa puso a prueba a los 3 trabajadores en las 3 máquinas, realizando el mismo trabajo todos ellos en cada una de las máquinas, los tiempos obtenidos por cada trabajador se muestran en la Tabla 1.5. Para resolver esta instancia se implementó el algoritmo 13.

Tabla 1.5: Tiempos obtenidos en el muestreo

		Máquinas		
		J_1	J_2	J_3
Trabajadores	P_1	10	9	7
	P_2	8	7	6
	P_3	5	10	9

En la Figura 1.31, se muestran los resultados, obtenidos de aplicar el algoritmo 13 a la instancia de interes. La solución encontrada $x_{sol} = \{3, 2, 1\}$ implica que a P_1 se le asigne J_3; P_2 se le asigne J_2; y a P_3 se le asigne J_1, esta asignación conlleva un tiempo de 19 unidades.

Algoritmo 13: Método de ramificación y poda para el problema de asignación de personal

Input: n, Costos de asignación

Output: Si existe, vector x_{sol}

```
1  raíz ← vector de ceros de orden 1 × n
2  raíz.nivel = 0; raíz.bact = 0
3  Añadir raíz a LNV
4  C ← cota superior de la raiz
5  x_sol ← ∅; V(x_sol) ← ∞
6  while LNV ≠ ∅ do
7      Seleccionar un nodo_x de la LNV
8      LNV = LNV − nodo_x.
9      if CI(nodo_x) ≤ C then
10         Ramificar a nodo_x (véase algoritmo 14) for cada nodo_y hijo del nodo_x do
11             if nodo_y.nivel = n then
12                 nodo_y.es.solución = 1
13             else
14                 nodo_y.es.solución = 0
15             end
16             if nodo_y.es.solución = 1 then
17                 if V(nodo_y) < V(x_sol) then
18                     V(x_sol) ← V(nodo_y)
19                     x_sol ← sol_y
20                     C ← V(x_sol)
21                 end
22             else
23                 Insertar los hijos del nodo_x a la LNV
24                 C = mín{C, CS(nodo_y)}
25             end
26         end
27     else
28         Podar a nodo_x
29     end
30 end
```

Algoritmo 14: Método de ramificación usado en el problema de asignación de personal

Input: $nodo_x$,n

Output: hijos del nodos

```
for i = 1; 1 ≤ n; i++ do
    if Aun no se utiliza i then
        Crear un nodo_y
        nodo_y.nivel = nodo_x.nivel + 1
        sol_y ← sol_x
        sol_y[nivel] ← i
        V(nodo_y) ← V(nodo_x) + C_{nodo_y.nivel}
        nodo_y.Tareas.usadas ← Añadir (nodo_y.Tareas.usadas, i)
        Calcular la CI(nodo_y)
        Calcular la CS(nodo_y)
        Insertar nodo_y a los hijos del nodo_x
    end
end
```

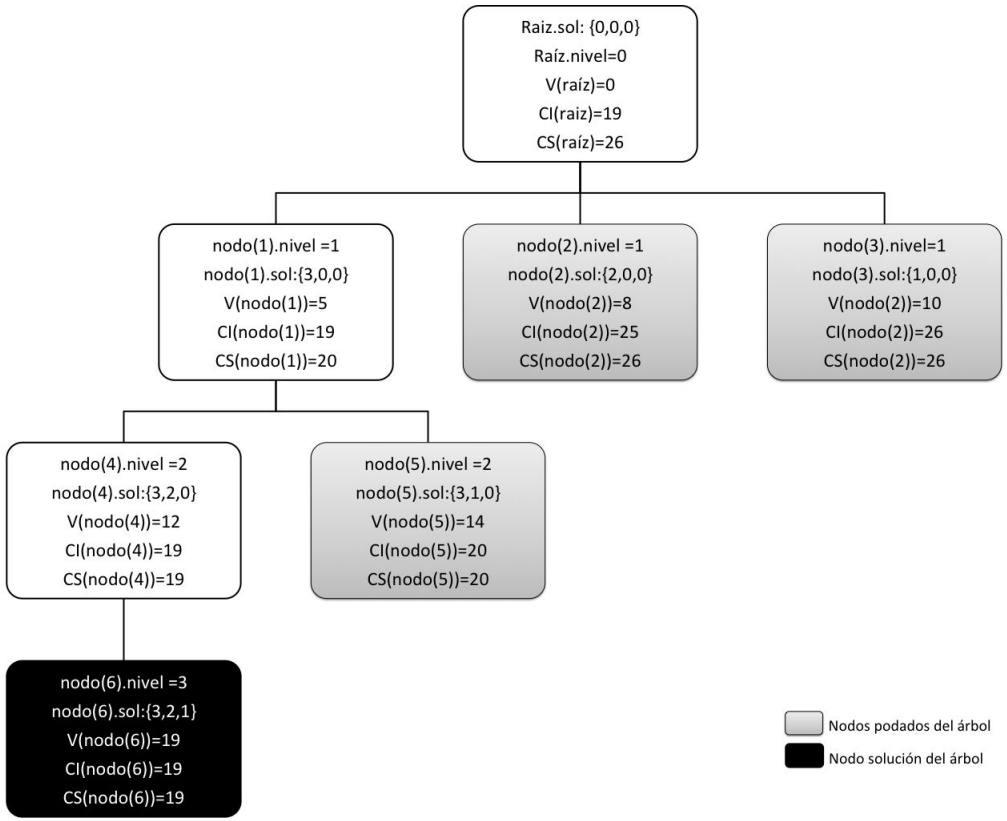

Figura 1.31: Solución al problema de asignación de personal.

La información presentada en este capítulo será utilizada en los posteriores . En el siguiente capítulo se revisarán algunos de los métodos de solución más utilizados para resolver problemas de decisión y de optimización.

Capítulo 2

Métodos de optimización

En la actualidad, los tomadores de decisiones se enfrentan, cada vez con más frecuencia, a problemas complejos de optimización y de decisión; por ende, en las últimas décadas, se han desarrollado y perfeccionado métodos para la solución de dichos problemas. Cabe mencionar, hasta hoy en día, no existe un método universal para la solución de problemas; más bien, cada uno de los métodos existentes sirve para resolver eficientemente a un subconjunto de éstos.

En la introducción se mecionó la forma general en la que se clasifican los métodos de optimización. Esta clasificación se esquematiza en la Figura 2.1.

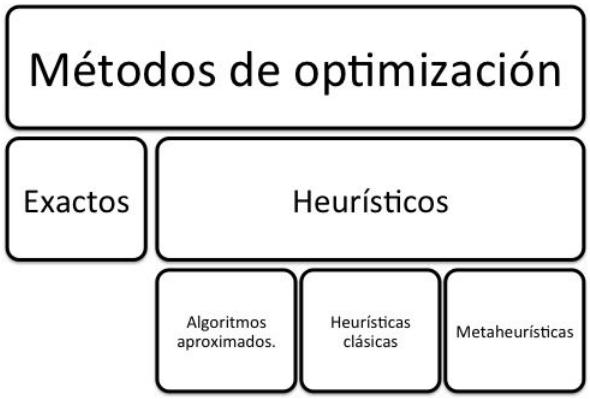

Figura 2.1: Clasificación de los métodos de optimización

En el presente capítulo se analizan las características y propiedades de algunos de los métodos más utilizados en la programación matemática.

2.1. Métodos exactos de optimización

Los métodos exactos de optimización son aquellos métodos que garantizan encontrar la solución óptima de la instancia a resolver. Generalmente, estos métodos son útiles para un subgrupo específico de problemas; ya que estos métodos aprovechan y utilizan las propiedades y características de los problemas en la búsqueda de la solución.

Cabe señalar, que se prefiere utilizar un método exacto a uno heurístico para resolver un problema, siempre y cuando, éste exista, y su orden éste acotado polinomialmente. Por ejemplo: para encontrar el árbol de peso mínimo en una gráfica $G = (V, E)$, se pueden implementar los algoritmos de Prim y Kruskal -con complejidad $O(|V|^2)$ y $O(|V|^2 log_2|V|)$ respectivamente- los cuales, tienen un orden polinomial; en contrate, para resolver el problema del agente viajero no existe un método exacto para solucionarlo; por ende, se recurre al uso de heurísticos para resolverlo.

A continuación, se analizan algunos de los métodos exactos de optimización diseñados para resolver los problemas de programación lineal; programación lineal entera y programación no lineal; ya que, las instancias de interés en este trabajo pueden ser solucionadas de manera exacta por alguna de estas técnicas.

2.1.1. El problema de programación lineal

2.1.1.1. Conceptos básicos

La **programación lineal** (PL) es una técnica de la programación matemática u optimización, la cual se utiliza para resolver un modelo matemático compuesto por una función objetivo lineal y un conjunto de restricciones también lineales; adicionalmente, en este modelo todas las variables decisión asumen un valor no negativo en el dominio de los reales. En la ecuación 2.1.1, se muestra el modelo de un problema de programación lineal.

$$\begin{gathered} \text{mín o máx} \sum_{i=1}^{n} c_i * x_i \\ \text{sujeto a:} \\ a_{i,j} * x_i \leq, \geq \text{ o} = b_j \text{ para todo } j = 1, \ldots, m \\ x_i \geq 0 \\ x \in \mathbb{R}^n, c \in \mathbb{R}^n, A \in \mathbb{R}^{n \times m}, b \in \mathbb{R}^n \end{gathered} \tag{2.1.1}$$

El modelo de programación lineal satisface las suposiciones de: a) proporcionalidad [1], b) aditividad [2], c) divisibilidad [3] y d) certidumbre [4]. Algunos de estos supuestos se exponen a mayor detalle en el Anexo A.

Una instancia de PL se puede expresar en diferentes formas equivalentes a través de manipulaciones apropiadas; en la tabla 2.1) se muestras dichas formas equivalentes.

Tabla 2.1: Formas equivalentes del problema de PL

Nombre de la forma	Caso de la minimización	Caso de la maximización
Canónica	$\text{mín } c^T x$ sujeto a: $Ax \geq b$ $x \geq 0$	$\text{máx } c^T x$ sujeto a: $Ax \leq b$ $x \geq 0$
Estándar	$\text{mín } c^T x$ sujeto a: $Ax = b$ $x \geq 0$	$\text{máx } c^T x$ sujeto a: $Ax = b$ $x \geq 0$

[1] La supocisión de proporcionalidad implica que la contribución de cada variable en la función objetivo y en el lado derecho de cada una de restricción es proporcional al valor de la variable.

[2] La suposición de aditividad implica que la contribución de cada variable en la función objetivo y en el lado derecho de cada una de restricción es independiente de la contribución de las otras variables.

[3] La supocisión de divisibilidad requiere que todas las variables de decisión puedan asumir valores fraccionarios.

[4] La suposición de certidumbre requiere conocer con certeza todos los parámetros del modelo.

Generalmente, las instancias de PL se expresan en forma mixta, la cual implica que algunas resticciones son ecuaciones y otras inecuaciones; además, alguna de las variables de decisión puede no ser positiva. Cualquier problema PL en forma mixta, a través de manipulaciones matemáticas adecuadas, se puede expresar en equivalente forma canónica o estandar (véase el ejemplo 2.1)

Ejemplo 2.1

Para el siguiente problema de PL en forma mixta

$$\text{máx}\, 2x_1 - 3x_2$$

sujeto a:

$$x_1 + x_2 \leq 3$$

$$x_1 \leq \tfrac{3}{2}$$

$$x_1 \geq 0$$

Su correspondiente forma estándar es:

$$\text{máx}\, 2x_1 - 3(x_2^+ - x_2^-)$$

sujeto a:

$$x_1 + x_2^+ - x_2^- + x_3 = 3$$

$$x_1 - x_4 = \tfrac{3}{2}$$

$$x_1 \geq 0$$

$$x_2^+ \geq 0$$

$$x_2^- \geq 0$$

y su pertinente forma canónica es:

$$\text{máx}\, 2x_1 - 3(x_2^+ - x_2^-)$$

sujeto a:

$$x_1 + x_2^+ - x_2^- \leq 3$$

$$-x_1 \leq -\tfrac{3}{2}$$

$$x_1 \geq 0$$

$$x_2^+ \geq 0$$

$$x_2^- \geq 0$$

Las ideas de poliedro, región factible y óptimo (expuestas en [14, 248, 232], véase también el capítulo 1) han sido la base para el desarrollo de métodos de solución de programción lineal. Dichos conceptos se conjuntan en el teorema fundamental de la programación lineal (véase Teorema 22).

Teorema 22 *(**Teorema fundamental de la programación lineal**) Dado el programa lineal*

$$\text{mín}\, c^T x$$

sujeto a:

$$Ax = b$$

$$x \geq 0$$

donde: $A_{m \times n}$ es una matriz de rango m;

- *Si existe una solución factible* [5]*; entonces existe una solución básica* [6]*.*
- *Si existe una solución factible óptima; entoces existe una solución factible básica óptima*

En la actualidad, existen métodos eficientes y eficaces para solucionar el problema de PL; algunos ejemplos son: algoritmo simplex, algoritmo elipsoidal de Khachiyan, algoritmo de Karmarkar entre otros. A continuación, se analizan los métodos más utilizados en la PL.

2.1.1.2. Métodos de solución

a) Método Simplex

El método simplex explota la estructura combinatoria del problema de programación lineal; ya que la solución óptima de una instancia de PL se puede encontrar al analizar el conjunto (finito) de puntos extremos del poliedro formado por las restricciones [17].

Con base en lo anterior, la idea básica del método simplex consiste en partir de un punto extremo (solución básica factible) x_{extemo}; si existe una solución adyacente $x_{adyacente}$ que mejore el valor de la función objetivo; entonces $x_{extemo} \leftarrow x_{adyacente}$; se continúa con este proceso hasta que los puntos

[5] Es una solución para la que se satisfacen todas las resticciones

[6] Una solución básica para $Ax = b$ se obtiene haciendo $n - m$ variables iguales a cero, posteriormente se resuelve el sistema de ecuaciones de m ecuaciones con m variables. Así se asume que al hacer las $n - m$ variables iguales a cero se llega a valores únicos para las m variables restantes; es decir, las columnas de las m variables son linealmente independientes

adyacentes a x_{extemo} no mejoren el valor de la función objetivo. La esencia del método simplex resinde en reconocer la optimalidad de un punto extremo solución dado con base en consideraciones locales, sin tener que enumerar todas las soluciones básicas factibles [14]. El algoritmo 15 muestra el modelo general del algoritmo simplex.

El algoritmo simplex tiene una complejidad del orden exponencial ($O(2^m)$), ya que el algoritmo puede recorrer todos los vértices del poliedro para encontrar la solución óptima (véase Teorema 23). Sin embargo, para la mayoría de los problemas prácticos el algoritmo requiere de $\frac{3m}{2}$ -en raras ocasiones de $3m$- iteraciones para solucionarlos [14].

Teorema 23 *El algoritmo simplex a lo más realizará* $\binom{m}{n}$ *iteraciones antes de terminar [121]*

Dedido a la gran variedad en las intancias de PL, se han desarrollado variantes del algoritmo simplex, las cuales permiten de manera natural a un conjunto particular de instancias de PL. Algunos variantes, se muestran en el anexo D

b)Algoritmos de punto interior

Estos algoritmos explotan la estructura continua del problema de PL [17]. Generalmente, estos algoritmos utilizan las siguientes ideas: a) tomar del interior de la región factible un punto, el cual ayude a alcanzar la solución óptima; b) moverse en la dirección que se mejore lo más rápido posible el valor de la función objetivo; y c)transformar la región factible de modo que se coloque la solución de prueba actual cerca del centro; lo cual, permitirá una mejora grande cuando se aplique la idea b) [9].

El algoritmo de Dikin es el primer método de punto interior. Este método se basó en la idea de elipse (véase definición 56) [125, 170]. Este algoritmo tiene un orden exponencial.

Definición 56 *Una elipse es un conjunto* $E(A,x) = (z-x)^T A^{-1}(z-x) \leq 1$*, para una matriz* A *simetrica positiva definida [121]*

Algoritmo elipsoidal Khachiyan

El algoritmo elipsoidal de Khachiyan demostró que el problema de PL puede solucionarse en tiempo polinomial; de donde, se mostró que el problema de PL está en la clase P; lo anterior se

Algoritmo 15: Pseudocódigo del algoritmo simplex [179]

Input: Instancia de PL a resolver

Output: Solución de la instancia

criterio de paro$\leftarrow$ $\begin{cases} \bar{c} = c - z \geq 0 & \text{en minimización} \\ \bar{c} = c - z \leq 0 & \text{en maximización} \end{cases}$

solución óptima $\leftarrow$ No;

solución no acotada $\leftarrow$ No;

Determinar una solución básica factible inicial.

repeat

  if *Se satisface el criterio de paro* **then**

    solución óptima $\leftarrow$ Si;

  else

    Elegir algún j, tal que: $\begin{cases} \bar{c_j} < 0 & \text{en minimización} \\ \bar{c_j} > 0 & \text{en maximización} \end{cases}$;

    $\Theta \leftarrow \oslash$;

    for $i = 1 : n$ **do**

      if $x_{i,j} \geq 0$ **then**

        $\Theta_i \leftarrow \frac{x_{i,0}}{x_{i,j}}$

      else

        $\Theta_i \leftarrow \infty$

      end

    end

    $\Theta_k = \text{mín}(\Theta_i)$ **if** $\Theta_k = \infty$ **then**

      solución no acotada $\leftarrow$ Si;

    else

      k sea la fila asociada al Θ_k;

      Pivotear sobre $x_{i,k}$ para determinar las nueva solución;

    end

  end

until *solución óptima = No y solución no acotada = No*;

formaliza en el Teorema 24

Teorema 24 *El problema de PL* $\in P$.

La idea básica de funcionamiento del algoritmo Khachiyan es la siguiente: se inicia con una elipse suficientemente grande, para contener el conjunto de soluciones problema (S) - se supone $S \neq \emptyset$-. En cada iteración k se checa si el centro de la elipse x_k es una solución factible. Si $x_k \notin S$; entonces se procede a construir una sucesión idonéa de elipses, que decrecen monótonamente de volumen; para ello, se toma un hiperplano (que contiene a x_k). Posteriormente, se genera una elipse (más pequeña que la original), que contenga a todas las soluciones factibles; se continúa este procedimeinto hasta que $x_k \in S$ (condición de término del algoritmo) [14, 121].

El algoritmo Khachiyan tiene un orden $O[(m+n)^6 L]$, donde: $L = \lceil 1 + \log m + \log n + \sum_j (1 + \log(1+|c_j|)) + \sum_i \sum_j (1+\log(1+|a_{ij}|) + \sum_i (1+\log(1+|b_i|))\rceil$. Sin embargo, el algoritmo de Khachiyan no fue usado de manera frecuente para la resolución de problemas prácticos; ya que, este algoritmo, requería mucha memoria y su tiempo de ejecución es cercano a la cota superior.

Algoritmo Karmarkar

Este algoritmo, desarrollado por Narendra Karmarkar [117], es un procediemieto de punto interior, de complejidad polinómica ($O(n^4 L)$), y fue un gran cambio en la programación matemática.

El algoritmo de Karmarkar requiere formular la instancia de PL de acuerdo al modelo mostrada en la ecucaión 2.1.2.

$$\begin{gathered} \text{mín}\, c^T x \\ \text{sujeto a:} \\ Ax = 0 \\ \mathbf{1}x = 1 \\ x \geq 0 \end{gathered} \tag{2.1.2}$$

donde: A es una matriz $m \times n$ con rango m, $n \geq 2$; $\mathbf{1}$ es un vector hilera formado exclusivamente de unos. Además se deben de cumplir las siguientes hipótesis [14]:

- El punto $x_0 = \left(\frac{1}{n}, \frac{1}{n}, \ldots, \frac{1}{n}\right)$ es factible en el problema

- El valor óptimo del objetivo del problema 2.1.2 es cero.

En términos generales, el funcionamiento del algoritmo de Karmarkar es el siguiente: en la $k-$ésima iteración se inicia con una solución x_k factible; después se utiliza una trasformacion proyectiva sobre el espacio factible del problema de PL, tal que $x_k \rightarrow [1, 1, 1, \ldots, 1]$ sea el punto central del espacio trasformado; posteriormente, se avanza en dirección del gradiente proyectado para determinar un nuevo punto solución x'_k (dicho punto debe estar contenido en el espacio trasformado); finalmente se evaluá el punto x'_k en su correspondiente posición del espacio original y se actualiza $x_k \leftarrow x'_k$. Se repite este procedimento hasta satisfacer la condición de paro.

2.1.2. El problema de programación lineal entera

2.1.2.1. Conceptos básicos

El problema de programación lineal entera (PLE) se diferencia del problema PL; ya que no satisface la propiedad de divisibilidad; en otras palabras, en este problema se requiere que un conjuto ($Y : Y \neq \emptyset$ y $Y \subseteq X$) de las variables de decisión asuma valores enteros. En la ecuación 2.1.3, se muestra el modelo general de programación entera.

$$\begin{gathered} max \text{ o } \min cx \\ \text{sujeto a:} \\ Ax \leq, \geq, \text{o} = b \\ x \geq 0 \\ x_j \in \mathbf{N} \text{ si } x_j \in Y \\ x_j \in \mathbf{R} \text{ si } x_j \notin Y \end{gathered} \tag{2.1.3}$$

De acuerdo a las características, las instancias de PLE se clasifican en: a) casos de programación entera pura (si todas las variables de decisión asumen valores enteros); b) casos de programación entera mixta (si sólo algunas variables asumen valores enteros); y c) casos de programación entera binaria (si cada variable variable toma valor cero o uno).

2.1.2.2. Métodos de solución

En la actualidad, se utilizan varios métodos para resolver el problema PLE, algunos de ellos retoman las estrategias de diseño de algoritmos analizadas en la sección 1.2.3 del capítulo anterior. A continuación, se describen algunos de los métodos más utilizados en la solución del problema de PLE.

- **Enumeración exhaustiva**. Este método, también es denominado método de enumeración explícita, consiste en enumerar todas las soluciones posibles del problema (ya que existe un límite finito para el número de soluciones posibles) y la selección de la mejor. Este método se vuelve impráctico a medida que aumenta el número de variables y su rango, ya que se incrementa rapidamente el número de combinaciones a examinar para encontrar la solución óptima.

- **Ramificación y acotamiento**.Este método, también llamado *branch and bound* (su nombre en inglés), se basa en la estrategia de ramificación y poda (véase sección 1.2.3.5) con algunas variaciones (dichas variaciones dependen de las características de la instancia a resolver). El funcionamiento general de este método es: a) Comenzar con una relajación del problema (no considerar restricciones de integralidad), denominado problema PL relajado; b) Resolver el problema de PL relajado c) Seleccionar una variable x_j que requiera ser entera y tenga un valor fraccionario entre los enteros i e $i+1$ d) Generar dos subproblemas problemas mutuamente excluyente. En el primer problema es el subproblema anterior más la restricción $x_j \leq i$ y el segundo es el subproblema anterior con la reticción $x_j \geq i$; f) Actualizar la cota g) Podar las ramas del árbol que represente: i) un problema infactible ii) una solución peor a la cota, y iii) una solución que satisfaga las condiciones de integralidad del problema original; se repite este procedimiento hasta satisfacer el criterio de paro. El método de ramificación y corte realiza una enumeración parcial de las posibles soluciones de un problema; ya que se eliminan conjuntos de soluciones en la etapa de poda.

- **Enumeración implícita** . Este método se utiliza en instancias de PLE del tipo binario; ya que en este método se utiliza el hecho de que cada variable debe ser igual a 0 o 1 para simplificar

las componentes de ramificar y de acotar a fin de determinar eficazmente cuándo un nodo no es factible.

- **Plano de corte**. En este método, al igual que en el de Ramificación y Acotamiento, se utiliza el problema PL relajado para generar una solución inicial. La diferencia radica en modificar el espacio de la soluciones al añadir una nueva restricción, denominada corte, que no corta ninguna otra solución factible; se repite este procedimiento hasta satisfacer el criterio de paro.

- **Programación dinámica**. Ya que, algunos problemas PLE pueden ser resueltos mediante un algoritmo recursivo, por lo cual se utiliza la programación dinámica (véase sección 1.2.3.2) . Un problema pueda ser abordado por esta técnica siempre y cuando la solución al problema se alcance a través de una secuencia de decisiones, una en cada etapa y la secuencia de decisiones cumpla el principio de optimalidad. Algunos de los problemas que se resuelven eficientemente con esta técnica son: problema de la mochila y problema de la ruta más corta.

2.1.3. El problema de programación no-lineal

2.1.3.1. Conceptos básicos

Generalmente, se dice que programación matemática es no lineal si la funcion objetivo, o bien, un conjunto de las restricciones no son lineales. El problema de programación no lineal (PNL) se puede formular a través del modelo mostrado en la ecuación 2.1.4

$$\begin{aligned}
&\text{máx o mín } z = f(x) \\
&\text{sujeto a:} \\
&g_1(x) \geq, \leq \text{ o } = b_1 \\
&g_2(x) \geq, \leq \text{ o } = b_2 \\
&g_3(x) \geq, \leq \text{ o } = b_3 \\
&\vdots \\
&g_m(x) \geq, \leq \text{ o } = b_m \\
&x_i^L \leq x_i \leq x_i^U \text{ for all } i = 1, 2, 3, \ldots, n \; x_i \in \mathbb{R}
\end{aligned} \tag{2.1.4}$$

Donde $f : \mathbb{R}^n \to \mathbb{R}$. $x_i^L \leq x_i \leq x_i^U$ rango factibles de las variables de decisión, los cuales definen $\mathcal{S}$. Las restricciones del problema son $g_1(x), \ldots, g_m(x)$ las cuales definen $\mathcal{F}$.

A continuación, se presentan conceptos y condiciones fundamentales implicados en las solución del problema de PNL (véase sección 1.1.1.2).

Teorema 25 ***Teorema de Weierstrass*** *Si dada una instancia del problema (PNL), el conjunto $\mathcal{F}$ es compacto y $\mathcal{F} \neq \emptyset$ y la función objetivo f es continua en $\mathcal{F}$; entonces, la instancia posee máximo y mínimo globales.*

Teorema 26 ***Teorema Local - Global*** *Si dada una instancia del problema (PNL), el conjunto $\mathcal{F}$ es convexo y la función objetivo f es continua y cóncava (resp. convexa) en $\mathcal{F}$, entonces cualquier máximo (resp. mínimo) local es global.*

Las instancias del problema de PNL se clasifican, en función de sus cartacterísticas, en caso irrestricto del problema PNL (si el conjunto de restricciones $g = \emptyset$) y caso restricto del problema PNL (si el conjunto de restricciones $g \neq \emptyset$). A continuación, se analizarán cada uno; así como los métodos exactos diseñados para resolverlos.

2.1.3.2. Caso irrestricto del problema PNL

La estructura general del caso irrestricto del problema PNL se muestra en la ecuación 2.1.5

$$\begin{gathered} \text{mín}_{(x \in \mathcal{F})}\, f(x) \\ \text{sujeto a:} \\ x_i^L \leq x_i \leq x_i^U \text{ for all } i = 1, 2, 3, \ldots, n \; x_i \in \mathbb{R} \end{gathered} \tag{2.1.5}$$

donde:

$$f : \mathbb{R}^n \to \mathbb{R}$$

$$\mathcal{F} \subseteq \mathbb{R}^n$$

$$x_i^L \leq x_i \leq x_i^U \text{ rango factibles de las variables de decisión}$$

El conjunto de condiciones necesarias y suficientes para que un punto $x^\star \in \mathcal{F}$ sea un óptimo local de f involucra los siguientes teoremas y proposiciones :

Teorema 27 ***Condición necesaria*** *Dado un punto $x^\star \in \mathcal{F}$ en el cual la función f tiene un mínimo local (o máximo). Si f es diferenciable el $x^\star$, entonces $\nabla f(x^\star) = 0$ [11]*

Teorema 28 ***Condición suficiente*** *Dado un punto $x^\star \in \mathcal{F}$ en el cual la función f doblemente diferenciable. Si $\nabla f(x^\star) = 0$ y $z^T \nabla^2 f(x^\star) > 0$ para todo vector z diferente del vector de ceros, entonces f tiene un mínimo local en $x^\star$. Si $\nabla f(x^\star) = 0$ y $z^T \nabla^2 f(x^\star) < 0$ para todo vector z diferente del vector de ceros, entonces f tiene un máximo local en $x^\star$ [11]*

Proposición 3 ***Condición de optimalidad***

1. *Si $x^\star$ es un mínimo local de f sobre $\mathcal{F}$, entonces $\nabla f(x^\star)'(x - x^\star) \geq 0$ para todo $x \in \mathcal{F}$.*

2. *Si f es convexa sobre $\mathcal{F}$, entonces la condición 1 es suficiente para que $x^\star$ minimize f en $\mathcal{F}$.*

2.1.3.3. Métodos de solución del caso irresticto

En la actualidad, existen varios métodos para solucionar del caso irrestricto del problema de PNL; los cuales, de acuerdo a sus características se pueden clasificar en: métodos de búsqueda en un intervalo sin derivadas (e.g: método de búsqueda exhaustiva, método de búsqueda Fibonacci, método de búsqueda de la sección áurea, entre otros); métodos de búsqueda en un intervalo con derivadas (e.g: método de bisecciones sucesivas, método de interpolación cúbica, método de Newton-Raphson, entre otros); métodos de búsqueda multidimensionales sin derivadas (e.g: método de cordenadas cíclicas, método de Hooke y Jeeves, método de Rosenbrock, entre otros); métodos de búsqueda multidimensionales con derivadas (e.g: método del gradiente, método de Newton, método Quasi - Newton) y métodos de direcciones conjugadas (e.g: método de gradiente conjugado, método de Fletcher-Reeves, entre otros). A continuación, se describen algunos de los procedimientos más utilizados para resolver el caso irrestricto del problema de PNL.

- **Búsqueda exhaustiva**. Este método se utiliza en instancias con una sóla variable y su funcionamiento general implica utilizar de manera sucesiva la ecuación

$$x_{k+1} = x_k + h$$

donde: h es un cambio suficientemente pequeño, para no pasar sobre el mínimo. El valor de h es positivo si se tiene evidencia que el mínimo está a la derecha, en contraste es negativo si el mínimo está a la izquierda.

- **Búsqueda de la sección aurea**. Este método asume que la función $f(x)$ es unimodal [7]. La sección áurea se obtiene dividiendo un segmento en dos partes de forma que la proporción de la parte menor frente a la parte mayor coincida con la proporción entre la mayor y la longitud total del segmento. El funcionamiento general de este método es el siguiente: dado un intervalo $[a, b]$ se generan dos nuevos puntos $x_1 = a + (1 - \alpha)(b - a)$ y $x_2 = a + \alpha(b - a)$, para algún $\alpha \in \left(\frac{1}{2}, 1\right]$ convenientemente elegidos. De acuerdo a los valores obtenidos por $f(x_1)$ y $f(x_2)$ el intervalo de incertidumbre en el siguiente paso será (a, x_2) o bien (x_1, b); ya que la idea del método consiste en elegir un α de tal forma, y según el caso, el valor de x_{1_k} coincida con $x_{2_{k-1}}$; o bien, x_{2_k} coincida con $x_{1_{k-1}}$.

- **Método de gradiente**. Es un procedimento iterativo que pertenece a la familia de métodos de gradiente, a este procedimento también se le denomina método de Cauchy de máximo descenso y se utiliza para minimizar una función de varias variables. El gradiente de una función f se denota por $\nabla f(x)$ y se define como $f(x) = \left[\frac{\partial f(x)}{x_1}, \frac{\partial f(x)}{x_2}, \ldots, \frac{\partial f(x)}{x_n}\right]$. El funcionamiento general de este método es el siguiente: se inicia con un punto x_1, $k = 1$ y un $\epsilon > 0$; mientras que $\|\nabla f(x_k)\| \not< \epsilon$, asignar $d_k = -\nabla f(x_k)$ y determinar a λ_k como una solución del problema:

$$\min_{\lambda \geq 0} f(x_k + \lambda d_k)$$

  posteriormente, calcular $x_{k+1} = x_k + \lambda d_k$ y $k = k + 1$, por ende $x_k \leftarrow x_{k+1}$; si $\|\nabla f(x_k)\| < \epsilon$; entonces, se para el algoritmo, ya que el punto actual x_k es una aproximación aceptable al verdadero mínimo. La principal desventaja de este método descenso de gradiente es el hecho de que la convergencia puede ser muy lenta para cierto tipo de funciones o valores iniciales.

- **Método de Newton**. Es un procedimento iterativo que pertenece a la familia de métodos de gradiente, se basa en el uso de una línea tangente como aproximación de $f(x)$ cerca de

[7] Una función $f(x)$ es unimodal en el intervalo $[a, b]$ si existe un punto $\tilde{x}$, tal que $f(x)$ es creciente en $[a, \tilde{x}]$ y decreciente en $[\tilde{x}, b]$

los puntos donde el valor de la función es cero. La idea del método es comenzar en un punto x_0, posteriormente se determina un nuevo punto x_{k+1} a través de la ecuación $x_{k+1} = x_k - [\nabla^2 f(x_k)]^{-1} \nabla f(x_k)$ [8]; el procedimento termina, cuando $x_k \approx x_{k-1}$. El método de Newton converge muy rápidamente, pero tiene los siguientes inconvenientes: se requiere calcular la segunda derivada de la función $f(x)$; y la convergencia depende en gran medida de la forma que adopte la función en las proximidades del punto de iteración. El principal problema del Método de Newton es que la matriz $\nabla^2 f(x_k)$ debe ser definida positiva. Generalmente, se desconoce toda información acerca de la solución óptima del problema; por lo cual, no se tiene un buen criterio para seleccionar x_0.

- **Métodos Quasi-Newton**. Es un conjunto de procedimentos diseñados para imitar el método de Newton; pero sólamente utilizando información de primer orden; por lo cual, se utiliza una matriz A_k que se aproximará a la matriz Hessiana. Los dos procedimeintos Quasi-Newton más utilizados son: el método de Davidon-Fletcher-Powell (DFP) y el método de Broyden-Fletcher-Goldfarb-Shanno (BFGS). Otra diferencia de este conjunto de métodos con el método de Newton, es que el segundo sólo requiere de un punto inicial, mientras que los procedimientos Quasi-Newton, generalmente, requieren dos.

- **Métodos de direcciones conjugadas**. El propósito de esta familia de métodos es mejorar la tasa de convergencia del método de descenso más rápido, sin incurrir en la sobrecarga computacional del método de Newton [17]. La idea básica en la que descansan estos métodos consiste en construir una base de vectores ortogonales y utilizarla para realizar la búsqueda de la solución en forma más eficiente. Una de las ventajas de estos métodos radica en que basta con asegurar la ortogonalidad de un nuevo miembro con respecto al último que se ha construido, para que automáticamente, esta condición se cumpla con respecto a todos los anteriores. El más importante de los procedimientos de gradiente conjugado es el método de gradiente conjugado.

[8] El elemento $\nabla^2 f(x_k)$ es la matriz Hessiana H evaluada en el punto x_k

2.1.3.4. El caso restricto del problema PNL

La estructura general del caso resticto del problema PNL se muestra en la ecuación2.1.6

$$\begin{aligned} &\text{mín}_{x\in\mathcal{F}\subseteq\mathcal{S}} \;\; f(x) \\ &g_j(x) \leq 0 \;\forall j = 1, 2, \ldots, p \\ &h_j(x) = 0 \;\forall j = p+1, \ldots, m \\ &x = (x_1, \ldots, x_n) \in \mathbb{R}^n \end{aligned} \tag{2.1.6}$$

donde: n es el número de variables de decisión; la función objetivo ($f : \mathbb{R}^n \to \mathbb{R}$) se define sobre el espacio de búsqueda($\mathcal{S}$) y la región factible ($\mathcal{F}$) formada por la intersección de los semiepacios definidos por las reestricciones -sean desigualdades $g : \mathbb{R}^n \to \mathbb{R}^p$; o bien, igualdades $h : \mathbb{R}^n \to \mathbb{R}^m$- tal que $\mathcal{F} = \{x \in \mathbb{R}^n : g_j(x) \leq 0, \forall j = 1, 2, \ldots, p; h_j(x) = 0, \forall j = p+1, \ldots, m; m \leq n\}$ [123, 150, 24, 151]

A continuación, se analizan las condiciones necesarias y suficientes involucradas en la solución del caso restricto del problema PNL.

Proposición 4 ***Condición necesaria - Multiplicadores de Lagrange****. Sea $x^\star$ un mímimo de la función f sujeta a $h(x) = 0$. Si los gradientes de las restricciones $\nabla h_1(x^\star), \ldots, \nabla h_1(x^\star)$ son linealmente independientes, entonces existe un único vector $\lambda^\star = (\lambda_1^\star, \ldots, \lambda_m^\star)$, llamado vector multiplicador de Lagrange, tal que:*

$$\nabla f(x^\star) + \sum_{i=1}^{m} \nabla h_i(x^\star) = 0$$

Adicionalmente, si f y h son funciones continuas y doblemente diferenciable, entonces:

$$y' \left(\nabla f(x^\star) + \sum_{i=1}^{m} \nabla^2 h_i(x^\star) \right) y \geq 0 \; y \in V(x^\star)$$

donde: $V(x^\star)$ es el subespacio de variaciones factibles de primer orden:

$V(x^\star) = \{y | \nabla h_i(x^\star)' y = 0 \; \forall i = 1, \ldots, m\}$ *[17]*

Proposición 5 ***Condición de suficiencia de segundo orden****. Sean f y h funciones continuas y doblemente diferenciables. Si dados los vectores $x^\star \in \mathbb{R}^n$ y $\lambda^\star$ se satisface:*

$$\nabla_x L(x^\star, \lambda^\star) = 0, \; \nabla_y L(x^\star, \lambda^\star) = 0$$

$$y'\nabla^2_{xx}L(x^\star,\lambda^\star)y > 0 \;\; \textit{para todo } y \neq 0 \;\; \textit{con} \;\; \nabla h(x^\star)'y = 0$$

entonces, $x^\star$ es un mínimo local estricto de f sujeto a $h(x) = 0$. Además, existen los escalares $\gamma > 0$ y $\epsilon > 0$ tales que:

$$f(x) \geq f(x^\star) + \frac{\gamma}{2} \parallel x - x^\star \parallel^2 \;\; \forall x \;\; con \;\; h(x) = 0 \;\; y \;\; \|x - x^\star\|$$

[17]

Proposición 6 ***Condición necesaria de Karush-Kuhn-Tucker (KKT)****. Sea $x^\star$un mímimo local del problema*

$$\text{mín } f(x)$$

$$\textit{sujeto a } h_1(x) = 0, \ldots, h_p(x) = 0$$

$$g_1(x) \leq 0, \ldots, g_r(x) \leq 0$$

donde: f, h_i y g_j son funciones continuas y diferenciables de $\mathbb{R}^n$ a $\mathbb{R}$. Si $x^\star$ es regular [9]*, entonces existen unos únicos multiplicadores de Lagrange $\lambda^\star = (\lambda^\star_1, \ldots, \lambda^\star_p)$ y $\mu^\star = (\mu^\star_1, \ldots, \mu^\star_r$, tales que:*

$$\nabla_x L(x^\star, \lambda^\star, \mu^\star) = 0$$

$$\mu^\star_j \geq 0 \; j = 1, \ldots, r$$

$$\mu^\star_j = 0 \; j \notin A(x^\star)$$

donde: $A(x^\star)$ es el conjunto de restricciones activas[10] *en $x^\star$.*

Si, además, f, h y g son funciones continuas doblemente diferenciables, entonces

$$y'\nabla^2_{xx}L(x^\star,\lambda^\star,\mu^\star)y \geq 0$$

$$\textit{para todo } y \in \mathbb{R}^n \;\; \textit{tal que:}$$

$$\nabla h_i(x^\star)'y = 0, \; \forall i = 1, \ldots, p$$

$$\nabla g_j(x^\star)'y = 0, \; \forall j \in A(x^\star)$$

[17]

[9] Un punto x es regular, cuando el Jacobiano de las restricciones, evaluado en es punto, es de rango máximo m, en otras palabras, las m filas de la matriz son linealmente independientes

[10] Una restricción es activa sólo si se cumple como una igualdad entre el miembro del lado izquierdo y el derecho $A(x) = j|g_j = 0$

2.1.3.5. Métodos de solución del caso restricto

Los métodos de solución del casos restricto del problema PNL, se pueden clasificar en los siguientes grupos:

- **Métodos duales**. Estos procediemietos se utilizan, cuando el problema dual asociado al problema primal permite, entre otras cosas, resolver el primero en forma más sencilla, aprovechando las propiedades que el segundo tiene (la concavidad de la función objetivo, la menor dimensión, la simplicidad de las restricciones, entre otros). La idea básica de estos problemas es la siguiente: dado un problema primal en la forma siguiente:

$$\text{mín } f(x)$$
$$\text{sujeto a:}$$
$$x \in \mathcal{F} \subseteq \mathbb{R}^n$$
$$h_i = 0 \; i = 1, \ldots, p$$
$$g_j \leq 0 \; j = 1, \ldots, r$$

donde: f, h y g son funciones continuas. Si la función de Lagrange ($L : \mathbb{R}^{n+p+r} \rightarrow \mathbb{R}$) del problema primal es: $L(x, \lambda, \mu) = f(x) + \sum_{i=1}^{p} \lambda_i h_i + \sum_{j=1}^{r} \mu_j g_j$, entonces la función lagrangiana dual es $\theta(\lambda, \mu) = \text{ínf}_x \left(f(x) + \sum_{i=1}^{p} \lambda_i h_i + \sum_{j=1}^{r} \mu_j g_j \right)$; ya que, la función lagrangiana dual proporciona una cota inferior del valor óptimo ($x^\star$), del la instancia que se desea resolver, entonces se debe satisfacer que $\theta(\lambda, \mu) \leq x^\star$ con cualquier valor de λ y μ. Con base en lo anterior, se puede construir el **problema dual lagrangiano** asociado con el problema original, el cual es:

$$\text{máx } \theta(\lambda, \mu)$$
$$\text{sujeto a:}$$
$$\lambda \in \mathbb{R}^p \mu \geq 0$$

la solución óptima de este problema ($d^\star$) representa una cota inferior del valor de $x^\star$. Si $d^\star = x^\star$, entonces existe una dualidad fuerte entre los problemas; en contraste, si $d^\star \leq x^\star$, entonces existe una dualidad débil entre los problemas.

Proposición 7 *Si, el problema primal tiene una solución óptima, entonces, el problema dual también tendrá una solución óptima. Ambas soluciones óptimas tendrán el mismo valor.*

Para que $x^\star$ sea la solución óptima del problema primal y $(\lambda^\star, \mu^\star)$ sea la solución del problema dual, es necesario y suficiente que: $x^\star$ sea una solución factible del primal; $\mu^\star \geq 0$; $\mu_j^\star = 0$ para todo $j \notin A(x^\star)$ y $x^\star \in arg\,\text{mín}_{x\in\mathcal{F}} L(x, \lambda, \mu)$ [17]

- **Programación geométrica**. Este conjunto de técnicas requiere que el problema se encuentre modelado en forma geométrica; lo cual, implica que en el problema primal la función objetivo y las restricciones sean posinomios [11]; y además, que las variables x_i sean estrictamente positivas . El problema dual asociado se caracteriza por ser un problema de programación convexa; por ende, se recurre a los algoritmos de programación convexa para resolverlos.

- **Métodos de penalización**. Transforman el problema restringido en uno nuevo en el que las restricciones se incorporan a la función objetivo por medio de una selección adecuada de parámetros de penalización. Estos algoritmos transforman el problema restringido en una sucesión de problemas sin restricciones; en los cuales, se utiliza a función de penalización para dirigir la búsqueda del óptimo. Estos procedimientos se clasifican en:

  - **Métodos de punto exterior**. Se caracterizan por utilizar una secuencia de soluciones de los problemas sin restricciones que sólo contiene puntos no factibles. Estos procedimientos utilizan a la función de penalización $P(x, r) = f(x) + k\psi(h(x), g(x))$; donde: k es el parámetro de penalización; y la función de penalización es $\psi(h(x), g(x)) = \begin{cases} 0 & \text{si } x \in \mathcal{F} \\ > 0 & \text{si } x \notin \mathcal{F} \end{cases}$, sobre el punto x
  - **Métodos de punto interior**. También conocidos como métodos barrera. Se caracterízan por utilizar una secuencia de soluciones de los problemas sin restricciones, que sólo contienen puntos factibles. Estos procediemientos utilizan a la función de penalización $P(x, r) = f(x) + k\phi(h(x), g(x))$; donde: k es el parámetro de penalización; y ϕ es la función de penalización; la cual forma una barrera infinita a lo largo del contorno de la región factible.

- **Métodos de linealización parcial**. En estos procedimientos se resuelve el problema original, a través de aproximaciones por subproblemas linealizados planteados apartir de desarrollos

[11] Un posinomio es una función de la forma $f(x) = \sum_{j=1}^{k} c_k \prod_i^n x_i^{a_{ij}}$

lineales de Taylor de la función objetivo y de las restricciones. En estos procedimientos se utilizan y modifican las ideas implicadas en los métodos de gradiente (vistos para el caso irrestricto). Estos métodos no convergen desde puntos de partida arbitrarios [45].

- **Métodos de multiplicadores de lagrange**. Se basan en el uso de una función objetivo expandida, por ende se suma a la función objetivo original y el producto de cada restricción multiplicada por el correspondiente multiplicador de Lagrange [45]
- **Método de los multiplicadores o del lagrangiano aumentado**. Estos procediemientos tratan de evitar el mal condicionamiento de la matriz Hessina implicada en los métodos de penalización a través de imponer un valor máximo al factor de penalización [45].
- **Programación cuadrática**. Un problema de programación cuadrática es aquel donde cada término en la función objetivo es de grado 2, 1 o 0 y todas las resticciones son lineales [248]. Los métodos de solución desarrollados para este problema se basan en las condiciones de KKT. Por ende, resolver el problema primal equivale a resolver un sistema de ecuaciones de KKT. El método de Wolfe es una alternativa eficaz para resolver problemas de programación cuadrática.

La tarea, esencial, de las técnicas de optimización es encontrar el punto meta (el mejor resultado con base al criterio de decisión), dentro del espacio de soluciones [139]. Lo cual, sólo se puede asegurar a través de algoritmos exactos. Existen problemas complejos, en los cuales no es posible implementar estos algoritmos [12] -ya sea porque su implementación implica un alto consumo de recursos, o bien, no se conoce un algoritmo exacto para el problema particualr-. Entonces, se recurre a utilizar un método heurístico. En la sección siguiente se analizarán algunos de estos procedimientos.

2.2. Métodos heurísticos para la optimización

Cuando un problema no puede ser resuelto por un procedimiento exacto, se puede modificar (simplificar) en el modelo; o bien, emplear alguno de los procedimiento heurísticos, a fin de resolver

[12] Un problema de optimización puede no ser resuelto por un algoritmo exacto por alguna de las siguientes razones: a) el espacio de búsqueda es complejo; b) la simplificación del modelo conlleva a una solución poco úti, por ende no es recomedable modificar el modelo; c) la función de evaluación que describe la calidad de las soluciones varía con el tiempo y/o tiene ruido y d) las soluciones posibles están altamente restringidas [147]

el problema de interés. Generalmente, se dice que las técnicas heurísticas son procedimientos que permiten obtener buenas soluciones de un problema con recursos razonables[43].

Martí [142] expone las razones por las que se pueden utilizar métodos heurísticos, las cuales son:

- No se conoce ningún método exacto para la resolución del problema.
- La implementación del mejor método exacto para la resolución del problema es muy costosa.
- Se requiere la alta flexibilidad que ofrecen los métodos heurísticos para la manipulación y resolución de problemas difíciles.
- Como parte de algún procedimiento de optimización:
  - Se usa para generar soluciones iniciales.
  - Se usa como un paso intermedio del procedimiento

Como se mencionó con aterioridad, los métodos heurísticos se dividen en: heurísticas, algoritmos de aproximación y metaheurísticas. En las siguientes secciones, se analizarán cada uno de estos grupos.

2.2.1. Heurísticas

De manera general, se puede decir que una heurística es un razonamiento plausible que sirve para resolver un problema; a continuación, se formaliza este concepto.

Definición 57 *Las técnicas heurísticas son métodos o procedimientos para resolver un problema, que no son producto de un riguroso análisis formal. En la investigación de operaciones, es un procedimiento para el que se tiene un alto grado de confianza en que encontrarán soluciones de alta calidad con un costo computacional razonable, aunque no garantiza optimalidad o factibilidad [145]*

Existen muchos métodos heurísticos de naturaleza sumamente diferente (varios de ellos se han diseñados para tratar un problema en específico con poca posibilidad de generalización o aplicaciones a otros problemas) por lo que resulta complicado dar una clasificación completa y excluyente.

Una clasificación de los procedimientos heurísticos más conocidos es la siguiente (cabe mencionar que las categorías no son excluyentes). **Métodos de descomposición**: el problema original se

descompone en subproblemas más sencillos, manteniendo en memoria la correspondencia entre los subproblemas y el problema original. **Métodos de reducción**: consiste en identificar características y propiedades que cumplen mayoritariamente el conjunto de buenas soluciones e introducirlas como restricciones al problema original, con la finalidad de comprimir el espacio de soluciones factibles. **Métodos inductivos**: los cuales, consisten en generalizar una versión más simple del caso completo, donde se analizan las aplicaciones y restricciones del modelo más sencillo en el problema original. **Métodos Constructivos**: consisten en construir paso a paso una solución del problema, generalmente, son procedimientos de carácter determinista y suelen estar basados en la mejor elección en cada iteración. **Métodos de búsqueda local**: también conocidos como procedimientos de mejora local, comienzan con una solución inicial del problema, la cual se mejora progresivamente en cada iteración.

Un ejemplo de estos procedimientos es ascenso en la montaña; el cual funciona básicamente de la siguiente manera: a) se toma una solución inicial; b) se busca un vecino con mejor calidad; c) si existe dicho vecino entonces se sustituye la solución actual por el vecino encontrado y d) se repiten los pasos b) y c) satisfacer el criterio de paro . Para mayor información sobre métos heurísticos véase [192, 62]

2.2.2. Algoritmos de aproximación

Un algoritmo aproximado tiene como finalidad encontrar soluciones de calidad y cuyos tiempos de ejecución están acotados por valores conocidos. Este término fue introducido por Ronald L. Graham, 1966 [94].

Un algoritmo A es un algoritmo de aproximación para un $\pi \in NPO$; si A es capaz de generar una solución factible S de π en tiempo polinomial bajo un esquema de aproximación[13], que garantice que la solución S se encuentra contenida a $p(n)$-veces de aproximación [14] de la solución óptima [43].

[13]Un esquema de aproximación para un problema de optimización es un algoritmo que toma como entradas no sólo las instancias del problema, sino también, se busca fijar un elemento ϵ ($\epsilon > 0$) tal que $1 + \epsilon$ acote el factor de aproximación del algoritmo A [43].

[14]Un problema de optimización tiene un radio de aproximación $p(n)$ si para alguna entrada de tamaño n, el costo C para generar una solución por el algoritmo A queda acotado por un factor $p(n)$ en razón del costo $C\star$ asociado a producir el óptimo, por lo cual $p(n)$ satisface $p(n) \geq \text{máx}\left(\frac{C}{C\star}, \frac{C\star}{C}\right)$ [43].

Cabe hacer notar que: ϵ no es parte de la entrada de los algoritmos.

Los algoritmos de aproximación se pueden dividir en las siguientes clases [210]:

- **Esquema de aproximación totalmente polinomial** (frecuentemente, denominado FPTAS o FPTAS -por las siglas en inglés de *fully polynomial time approximation technique* -). Conjunto de algoritmos de aproximación $A_{\epsilon\epsilon>0}$, tal que cada A_ϵ corre en tiempo polinomial con una entrada de tamaño n y su resultado está en un radio de aproximación de $1+\epsilon$ del óptimo, y además, su ejecución depende polinomialmente del parámetro ϵ.

- **Esquema de aproximación a tiempo polinomial** (frecuentemente, denominado PAS o PTAS -por las siglas en inglés de *polynomial time approximation scheme*-) . Conjunto de algoritmos de aproximación $A_{\epsilon\epsilon>0}$, tal que cada A_ϵ corre en tiempo polinomial con una entrada de tamaño n y su resultado está en un radio de aproximación de $1+\epsilon$ del óptimo y además su ejecución depende exponencialmente del parámetro ϵ.

- **Aproximación por factor constante**. (frecuentemente, denominado APX -por las siglas en inglés de *a polynomial-time r-approximate algorithm*-). Conjunto de algoritmos de aproximación que garantizan que su radio de aproximación es una constante y que su ejecución es en tiempo polinomial con una entrada de tamaño n.

2.2.3. Metaheurísticas

2.2.3.1. Conceptos generales

Las técnicas metaheurísticas son estrategias inteligentes para alcanzar buenas soluciones a los problemas, utilizando una cantidad razonable de recursos. Dichos procedimientos se caracterizan por poseer una fase de intensificación y otra de diversificación. Estos métodos pueden aplicarse a una gran variedad de problema, ya que son métodos de propósito general; algunos ejemplos son: recocido simulado (SA), búsqueda tabú (TS), búsqueda en vecindades variables (SS), algoritmos bio-inspirados (Algoritmos Genéticos, Sistema inmúne), algoritmos sociales (optimización por colonia de hormigas, optimización por nubes de partículas, algoritmo de sociedades y civilizaciones, algoritmos culturales), entre otros.

Generalmente, se dice que una metaheurística es un método de solución que organiza una interacción entre un procedimiento de búsqueda local y una estrategia de alto nivel (que permita escapar de los óptimos locales, así como robustecer el método de búsqueda en el espacio de soluciones) [90]. Es decir, las técnicas metaheurísticas son estrategias inteligentes para delinear mecanismos y mejoras a procedimientos heurísticos generales para resolver problemas con un alto rendimiento [145] A continuación, se define formalmente el término metaheurística en terminos formales

Definición 58 *Una metaheurística A_M es una tupla de los siguientes componentes [218]:*

$$A_M = (T_M, \Xi_M, \mu_M, \lambda_M, \phi_M, \sigma_M, U_M, \tau_M)$$

donde:

T_M es el conjunto de elementos que manipulan la metaheurística, en el cual se encuentra contenido el espacio de búsqueda.

$\Xi_M = \{(E_1, D_1), (E_2, D_2), (E_3, D_3), \ldots, (E_v, D_v)\}$ es el conjunto de v pares. Donde E_i es la variable de estado de A_M y D_i es el dominio de dicha variable, en la iteración i.

μ_M es el número de soluciones con las que trabaja A_M en un paso.

λ_M es el número de nuevas soluciones generadas en cada iteración de A_M.

ϕ_M es el operador que genera nuevas soluciones a partir de las existentes, por ende,

$$\phi_M : T_M^{\mu_M} \times \prod_{i=1}^{v} D_i \times T_M^{\lambda_M} \rightarrow [0, 1]$$

, esta función debe satisfacer lo siguiente:

$$\sum_{y \in T_{M^{\lambda_M}}} \phi_M(x, t, y) = 1$$
$$\forall\, x \in T_M^{\mu_M}$$
$$\forall\, t \in \prod_{i=1}^{v} D_i$$

σ_M es una función que permite seleccionar las soluciones que serán manipuladas en la siguiente iteración A_M, por ende:

$$\sigma_M : T_M^{\mu_M} \times T_M^{\lambda_M} \times \prod_{i=1}^{v} D_i \times T_M^{\mu_M} \rightarrow [0, 1]$$

esta función debe satisfacer lo siguiente:

$$\sum_{z \in T_{M^{\mu_M}}} \sigma_M(x, y, t, z) = 1$$
$$\forall\, x \in T_M^{\mu_M}$$
$$\forall\, y \in T_M^{\lambda_M}$$
$$\forall\, t \in \prod_{i=1}^{v} D_i$$
$$\forall\, z \in T_M^{\mu_M}$$
$$\sigma(x, y, t, z) \geq 0$$

U_M es un procedimiento de actualización de las variables de estado de la metaheurística; por ende:

$$U_M : T_M^{\mu_M} \times T_M^{\lambda_M} \times \prod_{i=1}^{v} D_i \times \prod_{i=1}^{v} D_i \rightarrow [0, 1]$$

, esta función debe satisfacer lo siguiente:

$$\sum_{u \in \prod_{i=1}^{v} D_i} U_M(x, y, t, u) = 1$$
$$\forall\, x \in T_M^{\mu_M}$$
$$\forall\, y \in T_M^{\lambda_M}$$
$$\forall\, t \in \prod_{i=1}^{v} D_i$$

τ_M es el la función de decisión sobre el término de la metaheurística por tanto $\tau_M : T_M^{\mu_M} \times \prod_{i=1}^{v} D_i \rightarrow [verdadero]$

Dréo establece que la mayoría de las técnicas metaheurísticas comparten las siguintes características [61]:

- Son en alguna medida estocásticas, esta aproximación permite contener la explosión combinatoria.
- Generalmente tienen un origen en problemas discretos esto tiene ventajas para los problemas continuos.
- Se inspiran en analogías ya sean físicas (recocido simulado, difusión simulada) biológicas (algoritmos evolutivos, sistema inmune) etiológicas (colonia de hormigas, enjambre de partículas), memoria humana (redes neuronales) y procesos artísticos (búsqueda de la armonía).

2.2.3.2. Taxonomía de las metaheurístcas

Tomando como base una (o algunas) de las características y particularidades de las metaheurísticas, éstas pueden ser divididas en grupos exluyentes a manera de ejemplo: a) métodos de trayectoria (manejan sólo una solución) vs. métodos poblaciones (manejan un conjunto de soluciones); b) aleatorias vs. determinísticas; c) con memoria vs. sin memoria; d) bio-inspiradas vs. no bio-inspiradas; entre otras. Estos grupos son utilizados a fin de clasificar a las metaheurísticas; sin embargo, en la literatura no existe una clasificacion única de las metaehursticas. A continuación se analizaran algunas de las clasificaciones más utilizadas.

Dréo et al clasifican a las metaheurísticas en: a) métodos basados en trayectoria y b) métodos basados en poblacion. Se dice que una metaheurística se basa en trayectoria, si emplea una sola solución durante su ejecución, algunos ejemplos son: recocido simuladado, búsqueda tabú, entre otros. En cotraste, se dice que una metaheurística se basa en población, si ésta emplea un conjunto de soluciones durante su ejecución.

Por su parte, Melián y otros autores clasifican a las metaheurística en función del tipo de heurística ivolucrada en la metaheurística: a) **técnicas de relajación** son aquellos procediemientos en los cuales se realizan relajaciones (modificaciones) al problema original a fin de facilitar su solución, b) **técnicas constructivas** son aquellos procedimientos que van incorporando elementos a una estructura inicialmente vacía a fin se obtener una solución, c) **técnicas de búsqueda** son aquellos procediemientos en los cuales se presupone la existencia de una solución y por lo tanto se utilizan las ideas de recorrer y explorar el espacio de soluciones, a fin de encontrarla d) **técnicas evolutivas** son aquellos procediemientos que consiste en generar, seleccionar, combinar y reemplazar un conjunto de soluciones; e) **técnicas de descomposición** son aquellos procedimientos que para resolver un problema implican la descomposición de éste en un conjunto determinando subproblemas y a partir de éstos se construye la solución del problema original y f) **técnicas de aprendizaje** son aquellos procedimientos capaces de adquirir, utilizar y explotar la información generada durante el propio proceso de solución.

Brito et al [23] proponen dividir a las metaehuríticas el los siguientes grupos: a) procedimientos de **búsqueda local**: son aquellos metaheurísticas que seleccionan una nueva solución dentro del

vecindario de la solución actual; b) procedimientos de **búsquedas globales** son aquellos procedimentos que implican los métodos de búsqueda local combinada con una estrategia que les permita escapar de óptimos locales; las siguientes estrategias (volver a comenzar la búsqueda desde otra solución inicial; modificar la estructura de entornos, y permitir movimientos que empeoran la solución actual); c)procedimientos de **búsqueda poblacional** son aquellas metaheurísticas que utlizan un conjunto de soluciones que recorren el espacio de búsqueda en cada iteración, a fin de determinar la solución d) **otros porcedimientos metaheurísticos**

Debido a la variedad existente en las clasificaciones de las metaheurísticas, en este trabajo se ocupó una clasificación basada en la división de Dréo et al, la cual se muestra en la Figura 2.2. Con base en esta clasificación se analizan algunas de las metaheurísticas más utilizadas.

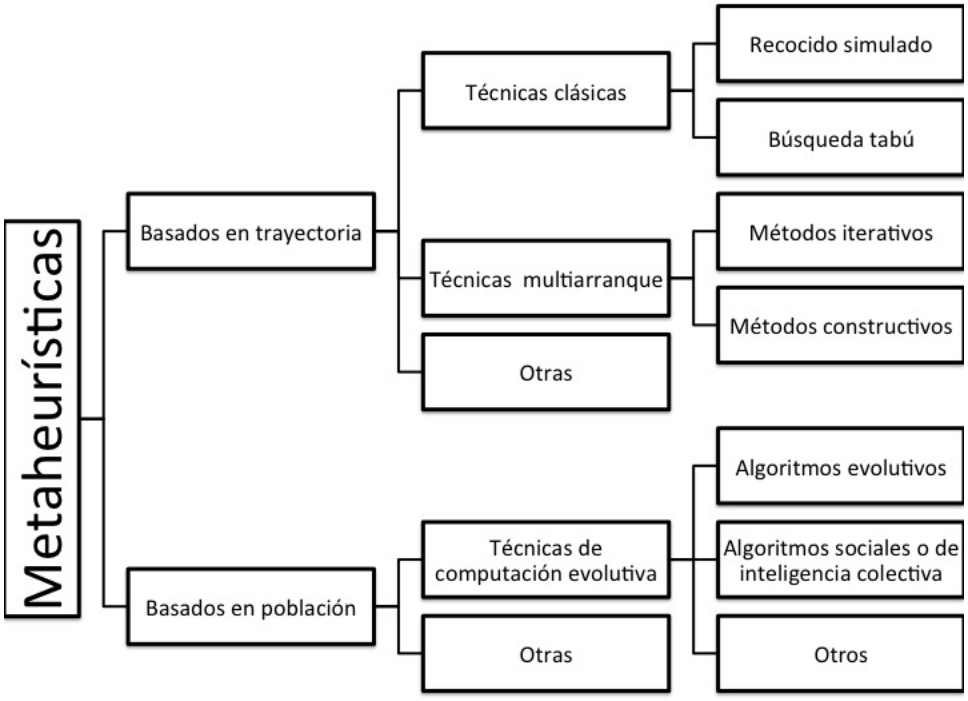

Figura 2.2: Clasificación de las metaheurísticas.

2.2.3.3. Técnicas basadas en trayectoria

A) Técnicas clásicas

A.1) Recocido Simulado

La técnica de recocido simulado (SA por las siglas en inglés de *simulated annealing*), es una de las metaheurística denominadas clásicas, se caracteriza por su simplicidad, robustez, adaptabilidad, eficiencia y eficacia. SA fue propuesto por Kirkpatrick, Gelatt y Vecchi, 1983 [119]. La cual, emula el proceso de recocido de metales[15]; el diseño y desarrollo del SA se basa en el algoritmo de Metrópolis[16], este procedimiento permite construir una cadena de Markov de las probabilidades de transición entre los estados de un material. En la Tabla 2.2 se muestra la analogía construida entre los procesos de recocido de metales y optimización.

Tabla 2.2: Analogía entre el temple simulado de metales y la optimización

Proceso	Optimización	Metalurgía
Estado inicial del sistema	Solución inicial	Configuración excitada de las moléculas
Mejor estado	Solución óptima	Configuración fundamental de las moléculas
Estimación por	Función objetivo	Estado de mínima energía
Estimación con	Costo de la solución	Energía de la configuración
Proceso unitario	Cada iteración	Cambio del estado energético

En el diseño del algoritmo SA se utilizaron principios, leyes y conceptos de la mecánica estadísti-

[15]Proceso térmico en el que se somete a algún metal o aleación en estado sólido a una temperaturas definidas por suficiente tiempo lo cual generará una excitación en las partículas del material, posteriormente se controlará la velocidad de enfriamiento del material para alcanzar una configuración de las partículas altamente cristalina. El fin de este tratamiento térmico es mejorar las propiedades físicas y mecánicas de los metales.

[16]El algoritmo de Metrópolis se puede describir como: dado un material en un estado inicial u y con energía e_u, se ocasionará una pequeña perturbación generando un nuevo estado v con energía e_v . Si, la energía e_v es menor que e_u la probabilidad de transición ($p_{aceptar}$) del estado u al v es uno en caso contrario $p_{aceptar} = e^{-\frac{e_u - ev}{kT}}$, donde k es la constante de Boltzmann, y T es la temperatura actual del sistema.

ca[17] y de la termodinámica estadística[18]. El SA es un procedimiento de búsqueda local, cuya idea fundamental es la existencia de una probabilidad $P_{aceptar}$ que permita aceptar siempre a toda nueva solución que mejore a la actual y eventualmente una nueva solución con un peor valor de función objetivo que la actual ($P_{aceptar}$), a medida que avanza el algoritmo $P_{aceptar}$ debe tender a cero, se utiliza la probabilidad $P_{aceptar}$ con el fin de escapar de óptimos locales [210], $P_{aceptar}$ se calcula mediante la ecuación 2.2.1

$$P_{aceptar} = \begin{cases} 1 & si \quad f(x_{nueva}) \text{ es mejor que } f(x_{actual}) \\ e^{-\frac{\delta E}{T_i}} & si \quad f(x_{nueva}) \text{ es peor que } f(x_{actual}) \end{cases} \tag{2.2.1}$$

donde: δE es la diferencia de aptitud entre la nueva solución x_{nueva} y la solución actual x_{actual}, véase la ecuación 2.2.2

$$\delta E = f(x_{nueva}) - f(x_{actual}) \tag{2.2.2}$$

T_i es la temperatura del sistema en la i-ésima iteración, la temperatura debe ser actualizada de forma tal que se satisfaga que $T_i \approx 0$ cuando $i \rightarrow$ máximo número de iteraciones ; algunas de las posibles formas de cálculo son:

- Principio geométrico para el decremento:

$$T_i = \alpha * T_{i-1} \tag{2.2.3}$$

  donde: $\alpha \in [0, 1]$ es una constante.

- Ley de decremento adaptativo:

$$T_i = (1 - T_{i-1} * (\frac{\Delta T_{i-1}}{\sigma^2(T_{i-1})})) * T_{i-1} \tag{2.2.4}$$

  donde: $\sigma^2(T_{i-1}) = f^2_{T_{i-1}} - (f_{T_{i-1}})^2$; f denota la función objetivo; ΔT_{i-1} es el factor de adaptibilidad (puede ser una constante).

En la Figura 2.3, se muestra la utilidad de aceptar soluciones peores, con el objeto de escapar de óptimos locales. Debe hacerse notar que la probabilidad para aceptar estados peores cambia a

[17]Disciplina científica, en la cual, se busca predecir las propiedades macroscópicas de un sistema a partir de las propiedades moleculares.

[18]Parte de la mecánica estadística que estudia los sistemas en equilibrio.

medida que avanza el proceso de optimización; ya que, en el recocido de metales a medida que decrece la temperatura es más difícil moverse de un estado a otro (véase Figura 2.4); de manera similar en el SA, la probabilidad de aceptar soluciones de menor calidad desciende en función del número de iteraciones ejecutadas.

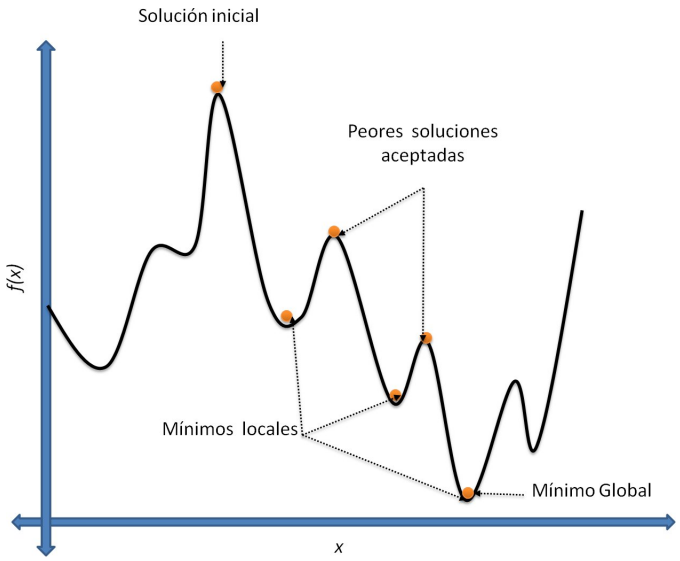

Figura 2.3: Particularidades del SA para escapar de óptimos locales.

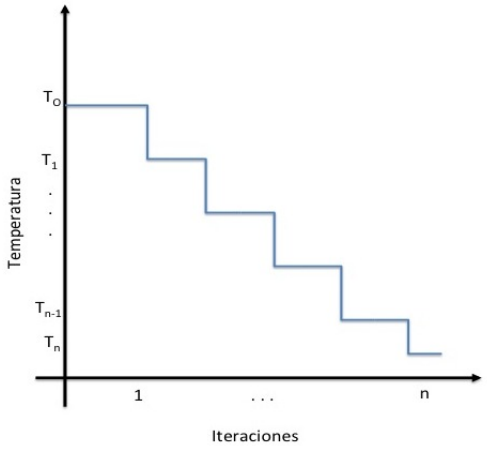

Figura 2.4: Cambios en la temperatura del SA.

Obsérvese que si no se aceptara un estado peor al actual, se quedaría atrapado en un mínimo local, símil a mover una pelota en una tubería, cuando se aplica poca fuerza a la pelota esta quedará atrapada en alguno de los valles de la tubería sin que necesariamente este sea el valle más profundo, a medida que se aplica mayor fuerza a la pelota esta tendrá mayor posibilidad de alcanzar el mínimo global. En el Algoritmo 16, se muestra el funcionamiento del SA simple.

Algoritmo 16: Pseudocódigo recocido simulado.

```
1  Crear una solución inicial. S_1 ← solución inicial.
2  f_0 ← valor de la función objetivo obtenido por S_1.
3  T_0 ←temperatura inicial.
4  α ← tasa de enfriamiento número de iteraciones ← criterio de paro.
5  k ← 1
6  for i = 1 : 1 : número de iteraciones do
7      T_i = T_{i-1} * α.
8      Generar de forma aleatoria una nueva solución al problema en la vecindad de S_1.
9      S_2 ← nueva solución.
10     f_1 ← valor de la función objetivo dado por S_2.
11     if f_1 es mejor que f_0 then
12         S_1 ← S_2.
13         f_0 ← f_1.
14     else
15         δE = f_0 − f_1
16         if rand ≤ e^{δE/(k*T_i)} then
17             S_1 ← S_2.
18             f_0 ← f_1.
19         end
20     end
21 end
```

A.2) Búsqueda Tabú.

La técnica de búsqueda Tabú (TS por las siglas en inglés de *tabu search*) emula el mecanismo fundamental de la ingenuidad (intuición) humana, sin considerar elementos aleatorios y asumiendo que la única razón para escoger una peor solución es evitar realizar una tarea que se hizo previamente [33].

TS fue propuesta por Glover en 1989 como una alternativa determinística a SA. El procedimemiento de TS combina mecanismos de búsqueda local con el uso una memoria (lista tabú) con el objeto de guiar la búsqueda de forma inteligente; ya que la lista tabú permite escapar de óptimos locales.

En términos generales, se puede describir como: a) se selecciona una solución inicial; b) se crea una nueva solución por medio de búsqueda local, esta nueva solución no debe estar contenida en la lista tabú; c) se actualiza la lista tabú (añadiendo el nuevo elemento creado y retirando de ella al elemento más viejo que la lista contenga); d) se actualiza la solución actual; y e) se repinten los pasos b), c) y d) hasta satisfacer el criterio de paro. El tamaño de la lista tabú influye de manera significativa en el desempeño del algoritmo al resolver un problema y en la calidad de las soluciones obtenidas. En el Algoritmo 17 se detalla el funcionamiento de TS simple.

Coello menciona que TS se basa en los siguiente elementos [33]: a) el uso de estructuras flexibles de memoria (basadas en atributos), que le permitie un manejo más adecuado de los criterios de evaluación y la información histórica obtenida en la búsqueda; b) un mecanismo asociado de control basado en la interacción entre los elementos que limitan el proceso de búsqueda; y c) el uso de memorias de diferente duración; lo cual, permite implentar estrategias que intensifiquen y diversifiquen la búsqueda.

B) Técnicas multiarranque

Las técnicas multiarranque son un conjunto de metaheurísticas, las cuales combinan procedimemientos contructivos (los cuales se utilizan para generar soluciones de partida para la búsqueda local de acuerdo a algún tipo de criterio) con métodos de búsqueda local (los cuales, parten de las soluciones anteriores y a partir de ellas recorren una parte del espacio de búsqueda a fin de mejorar la solución actual). En el Algoritmo 18, se muestra el funcionamiento básico de un método multiarranque general Las técnicas multiarranque se pueden clasificar, de acuardo a sus características,

Algoritmo 17: Pseudocódigo búsqueda tabú

```
1  Crear una solución inicial (S_0);
2  Crear una lista tabú (T), tal que T = ∅;
3  while no se satisfaga el criterio de paro do
4  |  Tomar la solución S_1 con mejor aptitud que esta en el vecindario de S_0 y a su vez no éste
   |  contenida en T.
5  |  Añadir S_1 a la lista tabú.
6  |  if |T| > l then
7  |  |  Eliminar el más viejo elemento contenido en T.
8  |  end
9  |  if S_1 es mejor que S_0 en términos de la función objetivo then
10 |  |  S_1 ← S_0
11 |  end
12 end
```

Algoritmo 18: Pseudocódigo búsqueda multiarranque general

```
1  Crear una solución inicial S_0 ;
2  Mejor solución encontrada_i ← s_i
3  while no se satisfaga el criterio de paro do
4  |  s'_i ← sea la solución encontrada por búsqueda local a partir de s_i
5  |  if s'_i es mejor que Mejor solución encontrada_i then
6  |  |  Mejor solución encontrada_i ← s'_i
7  |  end
8  |  Genera un nuevo conjunto de soluciones S_k
9  end
```

en los siguientes grupos: a) métodos iterativos de modificación de la solución encontrada (e.g: búsqueda local iterada, búsqueda local guiada y búsqueda en vecindades variables) y b) métodos constructivos de la solución inicial (e.g: GRASP). A continuación, se describen, brevemente, algunas de las metaheurísticas consideradas como técnicas multiarranque.

B.1) Métodos iterativos

B.1.1) Búsqueda local iterada

Búsqueda local iterada (ILS por las siglas en inglés de *iterated local search*) consiste básicamente en generar una secuencia de soluciones a través de búsqueda local[19] colocado en un esquema de repetición; a fin de mejorar los resultados obtenidos. El funcionamiento básico de este procedimiento se muestra en el Algoritmo 19.

Algoritmo 19: Pseudocódigo búsqueda local iterada

```
1 Generar una solución inical S_0
2 S* ← S_0
3 while no se satisfaga el criterio de paro do
4 |   Sea S' una solución obtenida a partir de perturbar, transformar o cambiar S*.
5 |   Sea S'* la solución encontrada por búsqueda local a partir de S'.
6 |   if Satisface un criterio de aceptación then
7 |   |   S* ← S'*
8 |   end
9 end
```

B.1.2) Búsqueda local guiada

Búsqueda local guiada (GLS por las siglas en inglés de *guided local search*) consiste básicamente en realizar cambios dinámicos en la función objetivo. Este procediemiento explota el conocimiento sobre las características o propiedades de las soluciones, a fin de poder discriminar entre una solución y otra.

En GLS la diversificación es resultado de la modificación sistemática de la función objetivo; mientras que, la intensificación es consecuencia de la búsqueda local. El grado de diversificación es regulado a través del parámetro λ. El funcionamiento básico de este procedimiento se muestra en el Algoritmo 20.

[19]La idea básica de los métodos basados en búsqueda local es crear iterativamente un vecindario de soluciones

Algoritmo 20: Pseudocódigo búsqueda local guiada

Generar una solución inical S_0

$k = 0$

for $i = 1 : 1 : M$; *donde* M *es el número de propiedades* **do**

    $p_i = 0$, donde: p_i es el parámetro de penalización de la propiedad i

end

while *no se satisfaga el criterio de paro* **do**

    $I_i = 1$ si la propiedad i esta presente en la solución, en caso contrario $I_i = 0$

    $h = f + \lambda \sum_{i=1}^{M} p_i I_i$

    Sea S_{k+1} el resultado de aplicar búsqueda local comenzando en el punto S_k y utlizando la función de aptitud h

    for $i = 1 : 1 : M$ **do**

        $u_i(S_{k+1}) = I_i(S_{k+1}) \times \frac{c_i}{(1+p_i)}$

    end

    for *toda* i *que maximize* $u_i(S_{k+1})$ **do**

        $p_i = p_i + 1$

    end

    $k = k + 1$

end

Regresar la mejor solución $S^{\star}$ encontrada de acuerdo a la función f

B.1.3) Búsqueda en vecindades variables

Búsqueda en vecindades variables (VNS por las siglas en inglés de *variable neighborhood search*) consiste básicamente en combinar la búsqueda local con una estructura dinámica del vecindario, propuesta por Hasen y Mladenovié [100] . Rothlauf [210] menciona que VNS se basa en los siguientes elementos: a) La estructura del espacio de búsqueda depende de la métrica utilizada y además es diferente para distintos operadores de búsqueda y representaciones; b) un mínimo global (o máximo global) es el óptimo global para todas las posibles métricas y además, es independiente de los operadores de búsqueda; y c) los óptimos locales no están distribuidos aleatoriamente en el espacio de búsqueda; pero lo óptimos locales contienen alguna solución óptima para un subproblema.

VNS consta de tres fases, las cuales son: a) Shaking: se selecciona aleatoriamente un elemento S' en $k-$ésima vecindad de la solución actual; b) búsqueda local: determinar una solución S'' a través de aplicar búsqueda local partiendo de S'; y c) movimiento: se decide si se acepta o no la nueva solución.

El funcionamiento básico de este procedimiento se muestra en el Algoritmo 21.

Algoritmo 21: Pseudocódigo búsqueda en vecindades variables

```
1  Seleccionar un conjunto de estructuras de vecindad N_k, k ∈ 1, ..., k_max
2  Sea S una solución inicial. while no se satisfaga el criterio de paro do
3      k = 1 while k < k_max do
4          Elegir aleatoriamente una solución S' en el vecindario N_k(S).
5          Sea S'' el óptimo local encontrado tras aplicar búsqueda local en el vecindario N_k(S)
           iniciando en el punto S'.
6          if f(S'') es mejor que f(S) then
7              S ← S'' k = 1
8          else
9              k = k + 1
10         end
11     end
12 end
```

B.2) Métodos constructivos

B.2.1) GRASP

La búsqueda glotona aleatoria adaptativa, generalemte, llamado GRASP (por las siglas de *greedy randomized adaptive search procedure*) es un procedimiento de búsqueda global multiarranque . GRASP fue propuesto por Feo y Resende [66] ; la idea básica de éste es que en cada iteración se construya un conjunto de soluciones y porteriormente se utilize estas soluciones como punto inical de búsquedas locales; se repite lo anterior hasta que se haya satisfecho el criterio de paro y se devuelve como salida del procedimiento la mejor solución encontrada. La forma básica de este procedimiento se muestra en el Algoritmo 22. **C)**

Algoritmo 22: Pseudocódigo GRASP

```
1 Mejor_Solución ← ∅ while no se satisfaga el criterio de paro do
2 |   Construir de manera glotona y aleatorizada una solución S (véase el algoritmo 23 );
3 |   Determinar la solución (S') de aplicar búsqueda local a partir de S
4 |   if S' es mejor que Mejor_Solución then
5 |   |   Mejor_Solución ← S'
6 |   end
7 end
8 return [Mejor_Solución]
```

Algoritmo 23: Construir una solución glotona aleatorizada

```
1 S = ∅ while no se haya construido la solución do
2 |   Crear Lista Restringida de Candidatos (LRC), a través de una función de selección
3 |   Seleccionar aleatoriamente un elemento s ∈ LRC
4 |   S = S ∪ s
5 |   Actualizar el conjunto de candidatos.
6 end
```

Otras

C.1) FANS

Los métodos de búsqueda por entornos adaptativos y difusos (FANS por las siglas en inglés de *fuzzy adaptive neighborhood search*) son un conjunto de procedimientos que combinan y utilizan: a) un operador para construir soluciones; b) una función difusa para valorar y calificar las soluciones; c) un mecanismo para administrar y adaptar el comportamiento o características del operador; y d) de vecindario, para generar y seleccionar una nueva solución[186, 187].

La idea básica de este algoritmo es la siguiente: a) dada una solución inical (x_{actual}) buscar sobre su vecindario una solción aceptable $x_{aceptable}$, usando una valoración disfusa. b) existen dos posibles resultados los cuales son: b.1) se encuentra un $x_{aceptable}$, entonces $x_{aceptable} \leftarrow x_{actual}$ y b.2) no se encuentre $x_{aceptable}$, entonces, se modifica el vecindario; y c) cuando las condiciones de búsqueda lo requieren, se aplica un mecanismo de reinicialización clásico para generar una nueva solución. Se repiten los pasos b) y c) hasta satisfacer el criterio de paro.

2.2.3.4. Técnicas basadas en poblaciones

. Las técnicas basadas en poblaciones se caracterizan por utilizar un conjunto de soluciones que interactúan entre sí y recorren conjuntamente el espacio de soluciones. Además, de los movimientos aplicables a la población existen otros operadores a considerar para generar nuevas soluciones a partir de las ya existentes. El resultado proporcionado por este tipo de algoritmos depende fuertemente de la forma en que manipula la población

A) Técnicas de computación evolutiva

Las técnicas de computación evolutiva se inspiran en la evolución y/o comportamiento de seres vivos [124]; en otras palabras, estos métodos retoman los conceptos de evolución biológica y social, selección natural, genética.

Coello establece que los principales paradigmas que rigen la computación evolutiva son los siguientes [219]:

- Codificar las estructuras que se replicarán, denominada individuos.
- Determinar las operaciones que afectarán a los individuos.
- Utilizar una función de aptitud, la cual servirá para determinar qué tan buena es una solución con respecto a las demás.

- Deteminar un mecanismo de selección

En la Figura 2.5, se muestran la clasificación general de las técnicas de computación evolutiva.

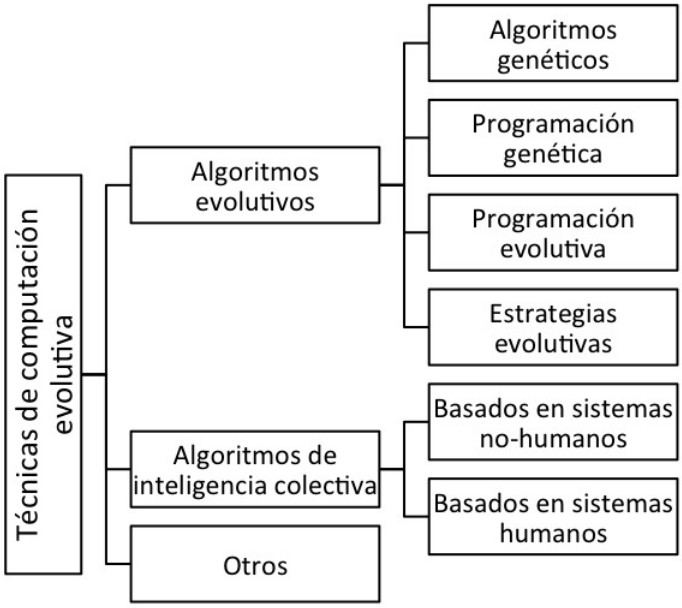

Figura 2.5: Clasificación de las técnicas de computación evolutiva

A.1) Algoritmos evolutivos

De manera general, se dice que los algoritmos evolutivos (AE) son algoritmos metaheurísticos poblacionales de búsqueda aleatoria [124] que se inspiran en la evolución, retomando los conceptos de selección natural y genética [236]. En otras palabras, los algoritmos evolutivos basan su funcionamiento en la teoría de la evolución de especies, la supervivencia del más apto y la trasmisión de características de padres a hijos [147].

En el Algoritmo 24, se muestra el funcionamiento general de un algoritmo evolutivo.

Michalewichz menciona que los principales elementos de los algoritmos evolutivos son los siguientes: [152]

- **Representación**. Implica un mapeo entre espacio de posibles soluciones al espacio de codificación de soluciones con una estructura determinada.
- **Función de aptitud**. También se le denomina función fitness y sirve para determinar qué tan buena es una solución con respecto a las demás; es decir, es un instrumento que permite valorar y juzgar.

Algoritmo 24: Pseudocódigo de un algoritmo evolutivo

```
1 t ← 0.
2 Generar una población de soluciones (P(t))
3 Evaluar a los individuos en P(t).
4 while No se satisfaga el criterio de paro do
5 |   t ← t + 1.
6 |   Seleccionar un conjunto de individuos a partir de P(t − 1).
7 |   Transformar a los individuos seleccionados para generar a P(t).
8 |   Evaluar a los individuos en P(t).
9 end
```

Existen cuatro tipos de funciones fitness, los cuales son: a) fitness puros[20]; b) fitness normalizado [21], c)fitness estandarizado [22]; y d) fitness ajustado[23]

- **Operadores de variación**. Son los mecanismos por los cuales se generará una nueva generación de individuos. Los operadores genéticos introducen diversidad en una población de soluciones [67]. Los operadores más utilizadas son:
  - Cruza: Forma un nuevo individuo a través de combinar los genotipos[24] de un conjunto de individuos seleccionados, a partir de un criterio; a dichos individuos se les denomina padres . Los mecanismos de cruza más comúnmente empleados son: a) cruza en un punto b) cruza en multiples

[20] El fitness puros da la aptitud de un individuo en téminos del objetivo(s) del problema original

[21] El fitness normalizado da la aptitud de una solución con respecto al resto de soluciones presentes en la población

[22] El fitness estandarizado es una modificaciín del fitness puro dado por la ecuación

$$\text{fitness estandarizado} \begin{cases} \text{valor de fitness puro} & \text{si es un problema de minimización} \\ \text{máximo valor fitness puro} - \text{fitness puro} & \text{si es un problema de maximización} \end{cases}$$

tal que la aptitud de individuo crecerá a medida que el valor de su función de aptitud se aproxime a cero.

[23] El fitness ajustado es una modificación del fitness estandatizado, dado por la ecuación

$$\text{fitness ajustado} = \frac{1}{1 + \text{fitness estandarizado}}$$

por lo tanto, el valor de la función de aptitud oscila en $[0, 1]$; la aptitud de individuo crecerá a medida que el valor de su función de aptitud se aproxime a uno.

[24] Un genotipo es una representación de las soluciones, la cual permite trasmitir y manipular dicha información a través de generaciones

puntos; c) cruza uniforme y d) cruza aritmética.

- Mutación: Implica pequeños cambios aleatorios en el genotipo del padre, a fin de generar un nuevo individuo alterando la información .
- Reordenamiento: Cambia el orden de la información contenida en el padre.
- Copia. Consiste simplemente en copiar un individuo seleccionado en la nueva generación.

En la práctica, la implementación de los operadores de variación se realiza sobre la representación utilizada y el paisaje (*landscape*) generado por la función de evaluación [61]. Cabe recalcar que la interacción entre la función fitness y los operadores de variación en gran parte detemina la eficiencia en el proceso de búsqueda.

- **Mecanismos de selección, reproducción y reemplazo**. Los mecanismos de selección actúan sobre los individuos de la solución actual y tienen como propósito escoger a través de un criterio individuos, los cuales serán utilizados como base para crear la nueva población. Estos mecanismos permiten intensificar la búsqueda. Los procesos más utilizados en la selección son: la ruleta, el torneo y el elitismo; los dos primeros procedimentos seleccionan a los individuos a través de una función de probabilidad; la cual, asigna a cada individuo una probabilidad considerando los resultados obtenidos por éste. En contrate, el elitismo se basa en tomar las mejores soluciones, sin ningún tipo de competencia.

  Ya que, con frecuencia, en los algoritmos evolutivos el tamaño de la población es fijo se debe seleccionar qué individuos se deben reemplazar (eliminar para que un nuevo individuo tome su lugar) en la población actual para generar la nueva solución. Esta actividad, involucra fijar un porcentaje de reemplazo y determinar un método de reemplazo que serán utilizados por el algoritmo. Los métodos de reemplazo más útilizados son: a) aleatorio[25]; b) reemplazo de padres[26] c) reeplazo de similares[27] y d) reemplazo de los peores individuos[28]

- **Población inicial**. La población inicial debe ser lo suficientemente grande (considerando las propiedades del espacio de búsqueda); y diversa, tal que los individuos contenidos en ella, sean una muestra significativa de los individuos contenidos en el espacio de búsqueda.

Con base en lo anterior, se formaliza el concepto de algoritmo evolutivo en la siguiente definición.

[25] se elimina cualquier individuo en la población y su lugar es ocupado por un nuevo individuo

[26] las nuevas soluciones ocupan el lugar que tenían sus padres

[27] Cada individuo de la descendencia reemplazará a un individuo en la población con un ajuste similar al suyo

[28] se selecciona aleatoriamente entre los peores individuos de la población aquellos que serán remplazados por la descendencia

Definición 59 *Un algoritmo evolutivo (AE) es el conjunto*

$$AE = \{I, \phi, \Omega, \psi, s, i, \mu, \lambda\}$$

donde:

I *es el espacio de individuos;*

$\mu \in \mathbb{N}$ *es el número de individuos en la población.*

$\lambda \in \mathbb{N}$ *es el número de individuos seleccionados.*

$\phi : I \to \mathbb{R}$*, es la función de aptitud, la cual asigna valores reales a los individuos.*

Ω *es un conjunto de operadores probabilísticos*

ψ *es la función de transición entre generaciones, y describe el proceso crear una población* $P(t+1)$ *al aplicar los operadores variacion y mecanismos de selección sobre una población* $P(t)$*.*

s *es el operador de selección, es decir este operador permite determinar cómo tomar* λ *individuos a partir de* μ *individuos.*

i *es el criterio de paro.*

Los algoritmos evolutivos son capaces de converger a la solución óptima, si se ejecutan un número muy grande de veces. A continuación, se formaliza el concepto de convergencia evolutiva.

Definición 60 ***(Convergencia evolutiva).*** *Sea* $(X_t : t \geq 0)$ *una secuencia de poblaciones generadas por algún EA y sea* $F_t = \text{mín}\{X_{t,1}, X_{t,2}, \ldots, X_{t,n}\}$ *el mejor valor de la función objetivo en la población de tamaño* $n < \infty$ *generado en un tiempo* $t \geq 0$*. Se dice que un AE converge completamente (con una probabilidad media de 1) en un mínimo global* $f^\star = \text{mín}\{f(x)\}$ *para la función* $f : \mathcal{F} \to \mathbb{R}$ *si una secuencia de números aleatorios no negativos* $(D_t; t \geq 0)$ *con* $D_t = f^\star - F_t$ *converge completamente a cero [124].*

En [195, 133, 123, 36, 63, 135, 68], se muestran algunas aplicaciones de estos algoritmos.

A.1.1) Los algoritmos genéticos.

Los algoritmos genéticos (propuestos por John Hollan 1975 con el objetivo de comprender los mecanismos subyacentes de auto-adaptación de sistemas [61]) son un subconjunto de los algoritmos evolutivos.

La idea básica de los algoritmos genéticos es la siguiente: a) establecer una codificación[29] apropiada de las soluciones en el espacio de búsqueda y una forma de evaluar la calidad de cada individuo en función objetivo (u objetivos) de la instancia a resolver; b) generar una población inicial; c) evaluar la aptitud de cada individuo en la población inicial; d) seleccionar, a través de un mecanismo de selección, un conjunto

[29]Las codificaciones de las soluciones generalmente son: binarias, enteras o reales.

de individuos de la población que servirán como base para generar una nueva población e) utilizar las operaciones de cruce y mutación para generar una nueva población f) evaluar la aptitud de cada individuo en la población. Se repiten los pasos d) al f) hasta satisfacer el criterio de paro. En el Algoritmo 25, se muestra el funcionamiento de un AE simple.

Algoritmo 25: Pseudocódigo de algoritmo genético canónico

```
1  Generar una poblacion inicial de tamaño N.
2  Evaluar la funcion de aptitud en cada individuo de la población.
3  if N es impar then
4  |  N_1 ← N + 1
5  else
6  |  N_1 ← N
7  end
8  while No se cumpla el criterio de paro do
9  |  for i = 1 : 1 : N do
10 |  |  Cruzar dos individuos seleccionados de la anterior generación; con lo cual, se obtienen
   |  |  dos descendientes.
11 |  |  if se satisface cierta probabilidad then
12 |  |  |  Mutar los dos descendientes.
13 |  |  end
14 |  |  Evaluar la función de aptitud en los dos descendientes.
15 |  |  Insertar los dos descendientes en la nueva generación.
16 |  end
17 end
```

Holland, también desarrolló la **teoría del esquema**, la cual muestra la relaccion existente entre la eficiencia de un algorimo genético al explorar el espacio de búsqueda y la mejora en las soluciones.

Un **esquema** es un conjunto de individuos con genes no definidos en algunos lugares [124]; en otras palabras, un esquema es un elemento formado del patrón de coincidencias de los individuos en una población

[67]. Geométricamente, un esquema puede representar un híperplano en el espacio de búsqueda, por lo cual, generalmente, se representan con una H.

El teorema del esquema afirma que en número de esquemas con alta aptitud aumenta en una población de generación en generación; por ende, la proporción de individuos que representan en esquema H en el tiempo t $(m(H,t))$ es menor que $(m(H,t+1))$. Por ende, el teorema del esquema establece una cota inferior sobre la calidad de las soluciones.

Teorema 29 ***(Teorema del esquema)***

$$m(H,t+1) \geq m(H,m) * \frac{f(H)}{< f >} * \left[1 - p_c \frac{d(H)}{l-1}\right] * [1 - p_m * o(H)]$$

donde: $f(H)$ representa la aptitud del esquema H; $< f >$ la media poblacional de la aptitud; $d(H)$ es la longitud del esquema; $o(H)$ orden del esquema; p_c probabilidad de cruza y p_m probabilidad de mutación.

A partir del teorema del esquema, se desarrolló la **hipótesis de construcción de bloques**[30] (BBH por las siglas en inglés de *building block hypothesis*); en términos generales esta hipótesis establece que los esquemas de orden inferior se combinan y compiten entre sí hasta generar esquemas de orden superior repitiendo este proceso hasta generar la solución óptima del problema.

Con base en lo anterior, se puede decir que un algoritmo genético enfatiza el rol de la contrucción de bloques y la cruza [67].

En [153, 95, 216], se muestran algunas aplicaciones de estos algoritmos.

A.1.2)Programación genética.

La programación genética, propuesta por Smith en 1980, opera sobre árboles que representan programas o circuitos [67]. Por ende, esta técnica trata que una población de árboles (programas) evolucionen a través de transmitir su herencia de generación en generación; en otras palabras, el objetivo de la programación genética es crear programas que evolucionen y mejoren su capacidad para solucionar problemas.

Los nodos terminales, representan variables independientes del problema; mientras que, los nodos internos del árbol representan funciones, siendo las funciones más comúnmente utilizadas las siguientes:

- Funciones boleanas: AND, OR, NOT, XOR.
- Funciones aritméticas: PLUS, MINUS, MULT, DIV.
- Sentencias condicionales: IF, THEN, ELSE, CASE, SWITCH
- Sentencias para iteraciones: WHILE, FOR, REPEAT..UNTIL

[30] Un **bloque** es un subestructura de un esquema que contribuye de manera positiva a la aptitud de un individuo.

En el Algoritmo 26, se muestra el funcionamiento general de la programación genética.

Algoritmo 26: Pseudocódigo de programación genética

```
1 Genera una población inicial.
2 while No se cumpla el criterio de termino do
3 |   Seleccionar individuos (para reproducción y eliminación), considerando su calidad.
4 |   Combinar y/o variar los individuos seleccionados para generar nuevos individuos.
5 |   Agregar y eliminar individuos.
6 end
```

Con base en lo anterior, se puede observar que la mayor diferencia entre programación genética y algoritmos genéticos es la representación de los individuos.

A.1.3) Programación evolutiva

La programación evolutiva (EP por las siglas en inglés de *evolutionary programming*) fue propuesta por Fogel en 1966. La EP involucra la estructura de autónoma de estados finitos con mutaciones y selecciones iteradas.

La idea básica de estos procedimientos es la siguiente: dado un conjunto de soluciones se perturba a cada individuo, con base en una distribución Gaussiana (normal) con media cero, (la perturbación es un medio de mutación); posteriormente se decide replazar o no a los individuos de la población; se repite este procedimiento hasta satisfacer el criterio de paro.

Cabe mencionar que en la EP no se utiliza la cruza como un operador de variación. La EP enfatiza los nexos de comportamiento entre padres e hijos, en vez de buscar emular operadores genéticos específicos lo cual difiere de los AG.

A.1.4) Estrategias evolutivas

Las estrategias evolutivas fueron propuestas por Schwefel y Rechenberg en 1973. Estos son métodos estocásticos con paso adaptativo, que permiten resolver problemas de optimización númerica, los cuales se caracterizan por utilizar una representación a través de vectores reales, que mediante los procesos de mutación (perturbación Gaussiana) y de recombinación evolucionan hasta alcanzar el óptimo.

Cada individuo es una solución potencial del problema. La representación de cada individuo consta de dos tipos de variables: las variables objeto (representan los posibles valores de las variables de decisión, estas

variables son principalmente afectadas por la mutación) y las variables estratégicas (son los valores de los parámetros que rigen el proceso evolutivo y son principalmente afectadas por la selección).

Existen dos tipos de estrategias evolutivas, las cuales son: a) simples (si utilizan un sólo individuo, por lo cual un nuevo individuo se genera como $x_i' = x_i + \sigma N \backsim (0,1))$ y b) avanzadas (si utilizan un conjunto de soluciones; por lo cual, un nuevo individuo se genera como $x_i' = x_i + \sigma N \backsim (\mu, \lambda))$.

A.2) Algoritmos sociales.

Los algoritmos sociales, también denominados algoritmos de inteligencia colectiva o algoritmos de inteligencia de partículas, se basan en las propiedades emergentes[31], resultado de las interacciones de un grupo organizado de seres vivos (una sociedad[32]); estas propiedades permiten que la sociedad en su conjunto sea capaz de realizar tareas o actividades que un sólo individuo es incapaz de hacer. En otras palabras estas metaheurísticas se basan en el comportamiento emergente[33] resultado de las interacciones socio-ambientales de un grupo de seres vivos. En la naturaleza varios organismos -como son: aves, hormigas, peces, bacterias, células, el ser humano entre otros- presentan un comportamiento social.

Las ideas fundamental en los algoritmos sociales son la auto-organización y comunicación. En términos generales, la **auto-organización** es un proceso, a través del cual, las interacciones locales entre los elementos de un sistema causan la emergencia del patrón global de un sistema [61]. Por otra parte, la **comunicación** se define como el proceso mediante el cual se puede transmitir información de un individuo a otro; por ende, este proceso es el principal instrumento para la modificación del comportamiento de los individuos en una sociedad.

Los algoritmos sociales explotan las siguientes características de una sociedad o enjambre:

- Compuesto de agentes simples
- Generalmente, descentralizado, ya que:
  - no hay un plan global de comportamiento
  - no existe un individuo que controle de manera directa el comportamiento de los demás.
- Una sociedad es flexible, ya que:
  - responder a cambios (internos y externos)

[31] sincronización, formación de patrones y estructuras, transiciones estado, conectividad, adaptación, cooperación

[32] Una sociedad es una agrupación natural o pactada de individuos, con características a fines, y que interactúan entre sí para formar una comunidad

[33] El comportamiento emergente es un comportamiento complejo, que aparece de forma imprevista, como consecuencia de una serie de acciones simples.

- no existe un modelo explícito de entorno.
- El comportamiento de un individuo se ve afectado por su entorno (comunicación con otros agentes), y un individuo afecta a su entorno en forma local con su experiencia.

- Una sociedad es robusta, ya que su éxito no depende de un sólo individuo

A.2.1) No basados en sistemas humanos.

Estos algoritmos se basan en la idea de que un agente, el cual forma parte de un conjunto de agentes similares, toma una decisión -con base a un componente social y un componente individual- a fin de alcanzar la mejor posición posible. Con base en la decisión tomada cada agente se mueve en el espacio de soluciones. El comportamiento de cada uno de los agentes afecta el comportamiento de sus vecinos.

A.2.1.1) Optimización por nube de partículas

Los métodos de optimización por nube de partículas (PSO por las siglas en inglés de *particle swarm optimization*) se inspira en el comportamiento del vuelo de bandadas de aves y el movimiento de los bancos de peces. Este procedimiento fue desarrollado por Jammes Kennedy y Russell Eberhart en 1995 e imita en comportamiento de las parvadas en busca de comida en un área; los pájaros no saben dónde está la comida, pero conocen su distancia a otras aves de su parvada; por lo cual, la estrategia más eficaz para hallar la comida es que cada ave busque en su entorno y posteriormente seguir aquella que se encuentre más cerca de la comida.

El PSO es un sistema multiagente; es decir, un sistema donde las partículas son los agentes simples que se mueven por el espacio de búsqueda y que guardan (y posiblemente comunican) la información sobre la mejor solución que han encontrado; cada partícula utiliza un fitness sobres su posición y velocidad, para dirigir su movimiento. En cada momento, el movimiento de cualquier partícula en el espacio se ve guiado por el movimiento de las partículas con el mejor valor de fitness. En términos generales, una nube (también denominada cúmulo o enjambre) es una población; y una partícula es un individuo o solución.

Formalmente, la nueva posición x_i^{t+1} de una partícula se determina a través de la siguiente ecuación

$$x_i^{t+1} = x_i^t + v_i^{t+1} \tag{2.2.5}$$

donde: v_i^{t+1} es la velocidad de la partícula; la cual se determina como:

$$v_i^{t+1} = v_i^t + \varphi_1(p_i - x_i^t) + \varphi_2(p_g - x_i^t) \tag{2.2.6}$$

donde: φ_n es un número aleatorio $U \backsim (0, \varphi_{max})$. φ_1 representa la experiencia individual; mientras que φ_2 representa la comunicación social.

p_i mejor posición encontrada por la i-ésima partícula.

p_g mejor posición encontrada en el vecindario de la i-ésima partícula.

En el Algoritmo 27 se muestra el funcionamiento del PSO general.

Algoritmo 27: Pseudocódigo PSO genérico

```
1  k = 0; Generar un engambre inicial de N partículas
2  for i = 1 : 1 : N do
3  |  Inicializar los parámetros x_i^0, v_i^0
4  |  p_i ⟵ x_i^0
5  end
6  Evaluar la aptitud de cada partícula en el enjambre.
7  for i = 1 : 1 : N do
8  |  p_g ← posición de la partícula con mejor aptitud en el vecindario.
9  end
10 while No se cumple el criterio de terminación do
11 |  k = k + 1
12 |  for i = 1 : 1 : N do
13 |  |  v_i^k = v_i^{k-1} + φ1(p_i − x_i^{k-1}) + φ2(p_g − x_i^{k-1})
14 |  |  x_i^k = x_i^{k-1} + v_i^k
15 |  |  if x_i^k es mejor que p_i then
16 |  |  |  p_i ⟵ x_i^0
17 |  |  end
18 |  end
19 end
20 Evaluar la aptitud de cada partícula en el enjambre.
21 Determinar la partícula con mejor aptitud.
```

El PSO ha sido aplicado exitosamente en diferentes campos de investigación, algunos ejemplos son: optimización de funciones numéricas, entrenamiento de redes neuronales, aprendizaje de sistemas difusos,

clasificación de imágenes, minería de datos, agente viajero, ingeniería química, entre otros.

A.2.1.2) Optimización por colonia de hormigas

La optimización por colonia de hormigas (ACO por las siglas en inglés de *particle swarm optimization*) fue introducido por Dorigo, Maniezzo y Colorni, 1996 [60], imita el comportamiento colectivo de las hormigas; puesto que una colonia de hormigas: capaz de realizar tareas difíciles (desde el punto de vista de una hormiga individual) y resolver problemas (como encontrar la ruta más corta entre el hormiguero y una fuente de alimento), a través de la mutua colaboración y comunicación de las hormigas (por medio del depósito y seguimiento de un rastro químico).

En este algoritmo la colonia de hormigas es un conjunto de agentes que se comunican indirectamente vía una modificación dinámica de su medio ambiente; por ende, la solución encontrada de algún problemas esta basada en la experiencia colectiva [61].

Las coincidencias entre la colonia de hormigas artificial y la colonia real son las siguientes: a) ambas colonias se componen de una población de individuos que trabajan conjuntamente para alcanzar una cierta meta; b) el objetivo de las hormigas reales es encontrar una fuente de comida, en contraste las hormigas artificiales buscan encontrar la solución de un problema de optimización dado y se requiere la cooperación entre los individuos para encontrar buenas soluciones; y c) una colonia es una población de agentes simples, independientes y asíncronos que cooperan para encontrar una buena solución del problema (o fuente de comida). En contraste, las diferencias entre las colonias artificiales y reales son las siguientes: a) las hormigas artificiales viven en un mundo discreto y su movimiento es secuencial a través de un conjunto de estados finitos, b) los cambios en la concentración (depósitos y evaporación) de feromónas en ambas sociedades es diferente; en el caso del sistema artificial la actualización es en algunas de las hormigas, c) algunas implementaciones requieren que los agentes artificiales tengan habilidades, que no poseen las hormigas reales, a fin de resolver el problema de interés [161].

La idea básica del ACO es la siguiente: a) lanzar una colonia de N hormigas; b) para cada una de las hormigas en la colonia construir una solución del problema a interés (las soluciones se generan de manera probabilística guiándose por la concentración de feromona artificial) y c) actualizar la concentración de feromona artificial. Los pasos anteriores se repiten hasta satisfacer el criterio de paro. En el Algoritmo 28, se muestra el funcionamiento básico del ACO.

En [22, 60, 61, 161], se muestran algunas aplicaciones de este algoritmo

A.2.1.3) Colonia Artificial de Abejas

Las abejas resuelven problemas difíciles todos los días; como por ejemplo: ellas visitan flores en múltiples

Algoritmo 28: Pseudocódigo ACO general

```
1 Inicializar rastros de feromonas.
2 while no se satisface el criterio de paro do
3     Construir una colonia de hormigas ( Con base en la concentración de feromonas; se contruye una
      solución; la cual inicialmente vacía s^p y posterioremente se añade un componente elegido entre los
      vecinos).
4     Aplicar una técnica de búsqueda local (Se realiza una búsqueda local a las soluciones construidas
      y las soluciones óptimas locales; a fin de poder decidir qué feromónas se deben actualizar).
5     Actualizar los rastros de feromónas (Se evapora la concentración de feromona en el espacio de
      búsqueda y posterioremente se aumenta su concetración de en aquellas regiones prometedoras).
6 end
```

ubicaciones, debido a que utilizan mucha energía para volar, encuentran la ruta que implica el menor tiempo posible. Las colonias de abejas, al igual que las de hormigas, son capaces de realizar hazañas extraordinarias que un individuo por sí sólo es incapaz de hacer.

El algoritmo de colonia artificial de abejas (ABC por las siglas en inglés de *artificial bee colony*), fue propuesto por Dervis Karaboga en 2005. Este procedimiento imita el comportamiento de las abejas en su búsqueda de alimento; ya que, en el ABC, las abejas artificiales se mueven en un espacio de búsqueda en función de su experiencia colectiva e individual; por ende, este procediemiento combina métodos de búsqueda local y global.

Karaboga define los sisguentes elementos como los componentes del ABC [116]:

- **Fuente de alimento**: En la natutraleza el valor que una colmena le asigna a una fuente de alimento depende de varios factores (proximidad a la colmena, calidad, facilidad de extracción, entre otros), en el caso de las sociedad artificial, y por simplicidad, los agentes asignan un valor numérico a sus fuentes de comida.
- **Abejas Empleadas**: Estas abejas se asocian a una fuente de comida, la cual están explorando y de la cual conocen su informacion (tal como: distancia a la colmena, ubicación y beneficio). Las abejas empleadas comparten, con cierta probabilidad, la información de sus fuentes de alimento con las demás abejas
- **Abejas No Empleadas** Estas abejas en constante búsqueda de una fuente de alimento. Existen dos tipos, los cuales son:

- **Abejas exploradoras**, también llamadas *scauts*, son aquellas encargadas de buscar nuevas fuentes de alimentos.
- **Abejas observadoras**, también llamadas abejas en espera, son aquellas abejas que esperan en la colmena y eligen una fuente de alimento con base en la información compartida por las empleadas o por otras exploradoras en la colmena.

El ABC requiere para su ejecución pocos parámetros los cuales son: a) **Número de soluciones** (NS) Este valor establece el número de fuentes de comida, y el número de abejas de cada tipo que serán utilizadas; b) **Máximo número de ciclos** (MNC) es el criterio de paro utilizado por el algoritmo; y c) **Límite** define el número máximo de ciclos que una fuente de alimento sin mejorar puede ser conservada, por una abeja, antes de ser reemplazada por una encontrada por una abeja exploradora.

En el Algoritmo 29, se muestra el funcionamiento general del ABC

Algoritmo 29: Pseudocódigo ABC general

```
1  Generar una población inicial con NS abejas de cada tipo.
2  Crear para cada abeja empleada una fuente de alimento.
3  Evaluar la fuente de alimento de cada una de las empleadas .
4  k = 0
5  while k ≤ MNC do
6      Producir nuevas soluciones para las abejas empleadas y evaluarlas.
7      Conservar la mejor solución entre la actual y la candidata.
8      Seleccionar en función de la aptitud, las regiones que serán visitadas por las abejas observadoras.
9      Generar nuevas soluciones para las abejas observadoras y evaluarlas.
10     Conservar la mejor solución entre la actual y la candidata.
11     Decidir, con base en el límite, si se deben abandonar fuentes de alimento.
12     Remplazar las fuentes de alimento que fueron abandonadas.
13     Determinar la mejor fuente de alimento encontrada hasta el momento.
14     k = k + 1;
15 end
```

A.2.1. 4) Algoritmos basados en bacterias

Esisten dos algoritmos basados en sistemas de bacterias los cuales son: a) optimización basada en el comportamiento quimiotáctico de bacterias y b) optimización de forrajeo bacteriano.

La optimización basada en el comportamiento quimiotáctico[34] de bacterias (QB-OA por las siglas en ingés de *optimization algorithms based on a model of bacterial chemotaxis*), fue propuesto por Bremermann en 1974 y simula el desplazamiento de bacterias.

La optimización de forrajeo bacteriano (BFO por la siglas en inglés de *bacterial foraging optimization*), fue propuesto por Kevin Passino [184, 185], se basa en el proceso de búsqueda de alimento de la bacteria *Escherichia coli* (E-Coli). Por ende, en el BFO se considera el desplazamiento (concentración y dispersión), reproducción, muerte de las bacterias.

En el BFO las bacterias son agentes, los cuales utilizan su percepción del entorno y su interacción con otros agentes, como base para moverse (cambiar de lugar o dirección) en busca de fuentes de nutrientes.

Los parámetros utilizados por este algoritmo son los siguientes: a) número de bacterias (soluciones); b) máximo número de generaciones (criterio de paro); c) número de ciclos quimiotácticos (veces que cada bacteria se moverá); d) factor de escalamiento (determina el movimiento de bacterias) y e) porcentaje del tamaño de paso. En el Algoritmo 30, se muestra el funcionamieno general del BFO

En las últimas décadas, se han generado varios algoritmos sociales inspirados en sistemas no humanos -además, de los previamente presentados- algunos de los cuales son: a) algoritmo de las luciérnagas, desarrollado por Yang en 2009 [250], b) algoritmo multiobjetivo inspirado en el comportamiento de las luciérnagas [212], c) optimizacion por grupos de leones [242], d) algoritmo de murciélagos[120], entre otros.

A.2.2) Basados en sistemas humanos.

Estos procedimientos imitan comportamientos en sociedades humanas y se basan en la siguiente idea:"los individuos en una sociedad se adaptan, reaccionan y evolucionan más rápido a cambios en medio socio-cultural en comparación a los cambios producidos por la herencia genética" [84, 83].

A continuación, se describen algunos de los algoritmos sociales basados en sistemas humanos desarrollados en las últimas décadas,

A.2.2.1) Algoritmos culturales

Introducidos por Reynolds, 1997 como un medio para la simulación de la evolución cultural [199]; en estos algoritmos cada individuo se describe en términos de un conjunto de rasgos o comportamientos, o

[34]Fenómeno en el cual las bacterias y otras organismos uni o multicelulares dirigen sus movimientos de acuerdo a la concentración de ciertas sustancias químicas en su medio ambiente.

Algoritmo 30: Pseudocódigo BFO general

```
1  Inicializar una población con N bacterias (soluciones).
2  Evaluar la población de bacterias .
3  k = 0
4  while k ≤ máximo número de generaciones do
5      for i = 1 : 1 : N do
6          Mov = 0
7          while i = 1 : 1 : N do
8              La i−ésima bacteria realizar un movimiento en función de la información obtenida de su
               medio y de sus congéneres. Mov = Mov + 1
9          end
10     end
11     Se mueren las bacterias en regiones agrestes.
12     Se reproducen las bacterias en regiones confortables.
13     Se elimina y remplaza la bacteria menos competitiva de la población actual.
14     k = k + 1;
15 end
```

bien como un mapeo generalizado de sus experiencias, los rasgo de cada individuo pueden ser modificados e intercambiados [199], en un comportamiento social dinámico, los rasgos y características de los individuos se pasan a la siguiente generación a través de mecanismos por motivos sociales.

Los algoritmos culturales consisten básicamente en dos componentes: el espacio de población y el espacio de creencias [144]. En la Figura 2.6, se muestra una esquematización de los componentes de los algoritmos culturales:

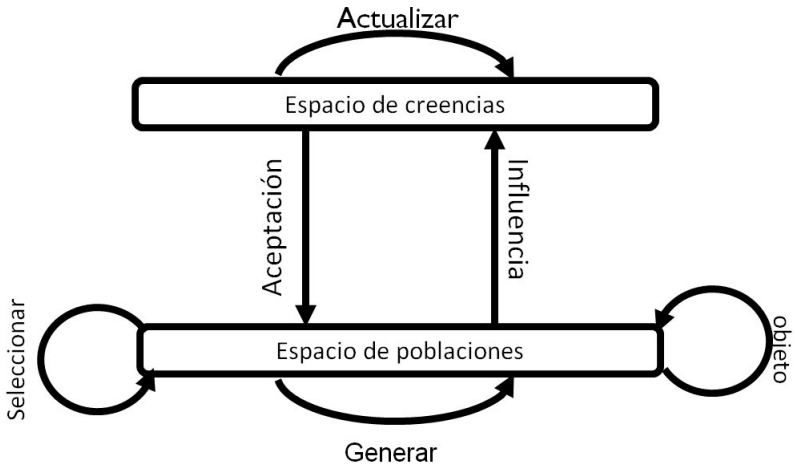

Figura 2.6: Marco conceptual de los algoritmos culturales Fuente [112]

En el Algoritmo 31, se muestra el funcionamiento general de los algoritmos culturales.

En [1, 31, 38, 37, 201] se muestran algunas aplicaciones de estos algoritmos.

A.2.2.2) Algoritmo de Sociedad y civilización

Introducido por Ray y Liew, 2003 [196], este algoritmo imita el desarrollo de civilizaciones[35]. La idea básica se derivó de las siguientes observaciones: a) una sociedad emerge y avanza por las relaciones cooperativas entre sociedades; b) los individuos en una sociedad interactúan con otros a fin de mejorar. Los individuos extraen información de su medio ambiente a través de la interacción intrasocial, mientras que cada sociedad compite con otras sociedades para atraer a los individuos más aptos [196].

También, se han generado varios algoritmos sociales inspirados en sistemas humanos -además, de los

[35]Una civilización es un sistema complejo que comprende un conjunto de sociedades y sus relaciones intra e intersociales; es decir, una civilización es una entidad cultural que aglutina un sentido semiinconsciente de unidad las ideas, el conocimiento, las artes, las costumbres y las creencias que caracterizan a un grupo humano.

Algoritmo 31: Pseudocódigo general de los algoritmos culturales

```
1  Generar una población inicial.
2  Inicializar el espacio de creencias.
3  Evaluar la población inicial.
4  repeat
5      Interacción entre los individuos y cambios en el espacio poblacional.
6      Evaluar a cada individuo mediante una función de aptitud.
7      Seleccionar a los padres para la siguiente generación.
8      Actualizar el espacio de creencias.
9      Aplicar la variación en los operadores.
10 until Satisfacer condición de paro;
```

previamente presentados- algunos de los cuales son: a) algoritmo de competencia imperialista, propuesto por Atashpaz-Gargari en 2007 [10], b) Algoritmo de anti-cultura, propuesto por Tang y Li en 2008 [233], entre otros. Se puede consultar [169] donde se da un marco conceptual sobre los algoritmos sociales, además de describir tres algorimtos sociales.

A.3 Otros algoritmos basados en técnicas de computación evolutiva

A.3.1) Sistema inmune artifical

El sistema inmune biológico tiene la capacidad de proteger a un organismo de la presencia de agentes infecciosos, así como de reparar las células dañadas o eliminarlas cuando sea necesario. Las características más importantes del sistema inmune es su capacidad de memoria.

El sistema inmune biológico involucra el principio de selección clonal, el cual dice lo siguiente:"los linfocitos que tienen anticuerpos con la afinidad adecuada para el antígeno son estimulados y sometidos a un proceso de reproducción por clonación; es decir, se crean múltiples copias de ellos. Estos nuevos clones sufren un proceso de mutación a gran escala (llamada hipermutación) con lo que se incrementa considerablemente la variedad de su repertorio. Esto permite que algunos de ellos incrementen aún más su afinidad hacia el antígeno. Los clones que ahora resulten ser más afines al antígeno son denominados células efectoras y se adhieren al antígeno procediendo a su neutralización y eliminación".

Leandro Nunes de Castro y Jonathan Timmis definen al sistema inmune artificial como: "Los sistemas inmunes artificiales son sistemas adaptativos, inspirados por la teoría inmunológica, funciones, principios y

modelos inmunológicos observados, los cuales son aplicados a la solución de problemas."

Sistema Inmune Artificial (AIS por las siglas en inglés de *artificial immune systems*) se refiere al diseño de sistemas computacionales basados en un mecanismo observado en la naturaleza, con la finalidad de solucionar problemas complejos. El sistema inmune (SI) puede ser visto como un poderoso sistema de procesamiento de información, cuyas características más importantes son: a) Memoria, el SI es capaz de recordar antígenos que se han presentado en el pasado. b) Aprendizaje. la respuesta del SI, ante repetidas apariciones de un antígeno es cada vez más rápida y eficiente, por eso se dice que posee aprendizaje. c) Robusto y tolerante a fallas. Esto significa que el sistema es capaz de responder adecuadamente aún en el caso de que falten algunos de sus elementos. Además, tiene la capacidad de recuperarse de errores; y d) Diverso. El SI es capaz de generar una gran diversidad y reconocer casi cualquier tipo de antígeno que se le presente.

A.3.2) Búsqueda armónica

Introducida por Zong Woo Geem, 2001, quien combinó conceptos de ingeniería y música [77]. La armonía puede entenderse como el arte de combinar y organizar acordes, para conseguir la emoción deseada (alegría, melancolía, tristeza, etc.) en los espectadores; es la habilidad de combinar sonidos desde un punto de vista artístico. En términos formales: se refiere al estudio estético de la simultaneidad (estudio vertical) en secuencias musicales. Un acorde consiste en tres o más notas diferentes que suenan al mismo tiempo. La búsqueda armónica imita el proceso de perfeccionamiento del estado de armonía en la producción (innovación) musical, evaluando el estándar estático para cada innovación. El grado en que se alcanza el estándar estático, permite evaluar a la combinación de sonidos, de cada uno de los instrumentos en cada ejecución. Lo que es similar a evaluar en la función objetivo los valores de las variables de decisión [80]. En la tabla 2.3 se resume la analogía existente entre la innovación musical y la optimización.

Tabla 2.3: Analogía entre queda armónica en el jazz y la optimización

Proceso	Optimización	Innovación musical
Mejor estado	óptimo global	Armonía fantástica
Estimación por	Función objetivo	Estándar estético
Estimación con	Asignar valores a las variables	Ejecutar cada instrumento
Proceso unitario	Cada iteración	Cada Práctica.

Fuente: [80]

En la innovación musical se utiliza una combinación de las siguientes operaciones básicas:

- **Recordar**: Utilizar algún acorde que se encuentre disponible en memoria.
- **Adaptar**: Realizar una modificación a un acorde disponible en memoria antes de ejecutar.
- **Crear**: Implementar un nuevo acorde.

En esta metaheurística se incorporan estructuras y estrategias de otras metaheurísticas, como ejemplo, la memoria armónica es similar a la lista tabú de búsqueda tabú: la capacidad de variar y adaptar desde principio a fin de la ejecución; el vector de búsqueda por medio de las consideración del ritmo de armonía es análogo a las estrategias de manejo de información utilizado por los algoritmos genéticos. En la tabla 2.4 se resumen los parámetros utilizados por HS.

En el algoritmo 32 se muestra el funcionamiento del HS. El método HS básicamente se compone de

Algoritmo 32: Algoritmo general HS

```
1 Inicializar proceso de optimización.
2 Genera una memoria de armonía inicial.
3 while No se cumple el criterio de terminación do
4 |   Generar una nueva armonía.
5 |   Actualizar memoria
6 end
```

cinco pasos [128], los cuales se describen a continuación:

- **Inicializar proceso de optimización**. En él se ingresan los parámetros, y datos del problema de optimización necesarios para la ejecución del algoritmo.
- **Inicializar la memoria de armonía**. En esta fase se asignan aleatoriamente valores a las variables de decisión. Y se verifica el cumplimiento de restricciones; de no ser así, se genera otra solución. Posteriormente, se evalúa la función objetivo mediante el conjunto de valores factibles (x^i) y finalmente se construye la matriz de memoria armónica.
- **Improvisar una nueva armonía** . Se forma un nuevo vector de soluciones $x^i = \{x_1^i, x_2^i, \ldots, x_N^i\}$, considerando la memoria de armonía y los parámetros.

Tabla 2.4: Parámetros básicos de HS

Parámetro	Símbolo	Características
Tamaño de la memoria de armonía	HMS	Es un valor discreto mayor o igual a uno.
Máximo número de improvisaciones	$MaxImp$	Es un valor discreto mayor o igual a uno
Consideración del ritmo en la memoria de armonía	$HMCR$	Es un valor real, que oscila entre 0 a 1. Este parámetro es la probabilidad de elegir un valor histórico contenido en la memoria (HM). Entre más cercano sea el valor de HMCR a uno, menor será la posibilidad de improvisar un nuevo valor para la variable
Parámetro de ajuste del ritmo.	PAR	Es un valor real, que oscila entre 0 a 1. Este parámetro es la probabilidad de generar un valor sin considerar la memoria
Ancho de la banda	b	Es un valor real mayor o igual a cero.

- **Actualizar memoria armónica**. Si el nuevo vector de armonía (x_i') al ser evaluado con la función objetivo es mejor que el peor vector armónico en memoria (x_{peor}); entonces, x_i' remplazará a x_{peor}.
- **Cumplir criterio de paro**. Repetir los dos pasos anteriores hasta satisfacer el criterio de paro. Generalmente se utiliza el número de iteraciones (improvisaciones).

HS a sido utilizado en una amplia variedad de problemas, ejemplo: problemas clásicos de optimización global[243, 127, 78], diseño de redes hidráulicas [76] composición musical [79] Ruteo de vehículos [81], diseño de domos geodésicos [214], Alineamiento múltiple de secuencias [158, 159]

A.3.3) Algoritmos Meméticos

Los algoritmos meméticos (MA), propuestos por Dawkins en 1976, son técnicas que combinan sinérgicamente conceptos tomados de otras metaheurísticas, tales como la búsqueda basada en poblaciones y la mejora local. Los MA son metaheurísticas basadas en población, por ende genera un conjunto de soluciones candidatas para el problema considerado. En el Algoritmo 33, se muestra el funcionamiento general de un MA.

Algoritmo 33: Algoritmo general MA

```
1  Generar, aleatoriamente, una población P(t).
2  while no se satisfaga el criterio de parada do
3      Calcular la aptitud para los individuos de la población P(t)
4      Elegir un subconjunto de P(t) de acuerdo a su aptitud, el cual se denota como M(t).
5      Guardarlo en M(t).
6      Combinar y variar los individuos de M(t) y generar un conjunto M'(t).
7      Guardarlo en M'(t).
8      Mejorar los individuos de M'(t) con búsqueda local.
9      Calcular la aptitud para los individuos en M'(t).
10     Generar una población P(t) considerando a los individuos en M'(t).
11 end
```

B) Otros algoritmos poblacionales.

B.1) Heurística de concentración

La heurística de concentración (HC) fue propuesta por Rosing en 1997 [209]. En la HC óptimo local es una fuente de información acerca de la estructura de una parte de la solución óptima; por ende, se espera que un conjunto de aquéllos den información sobre todas las partes de ésta. En el algoritmo 34 se muestra el funcionamiento de HC.

2.2.3.5. Extensiones de los métodos heurísticos

En los últimos años, se han generado una gran variedad de aportaciones en el campo de las metaheurísticos, entre los que destacan: la adaptación de una metaheurística a un problema en particular; el desarrollo de métodos híbridos; la generación de nuevos procedimientos, híperheurísticas, entre otros.

Algoritmo 34: Algoritmo general HC

1 Generar, aleatoriamente, N solucione iniciales y aplicar búsqueda local en cada uno de ellas.

2 Registrar las m mejores diferentes soluciones obtenidas.

3 Definir al Conjunto de concentración (CS) como el conjunto de elementos (metapatrón).

4 Aplicar un método (exacto o heurístico) al problema original pero restrngiendo la búsqueda con el CS

La **variante de un método metaheurístico** puede ser entendida de manera general como un cambio o modificación (provisional o definitivo) en alguna (s) fase (s) de un procedimiento original; que permita un mejor manejo y resolución de un problema particular. Esta modificación no debe afectar la estructura básica del procedimiento original; es decir, no se alterará la estructura esencial de la estrategia utilizada. Un ejemplo de lo anterior, se tiene en la aplicación de recocido simulado (originalmente diseñado para problemas discretos) para resolver problemas continuos o bien los cambios hechos a PSO para el manejo de problemas discretos.

Por otro lado, los procedimientos **metaheurísticos híbridos** son derivados de la combinación de los conceptos y estrategias de varias técnicas metaheurísticos y heurísticas (dos o más procedimientos); las metaheurísticas padre deben estar documentadas y comprobadas para una amplia variedad de problemas. La idea de combinar los procedimientos heurísticos y metaheurísticos se basa en el concepto de **vigor híbrido**: es el nivel de mejora de aptitudes de varias técnicas cuando se implementan fusionadas, en contraste a la implementación individual, para resolver un problema de optimización. Es decir, el vigor híbrido supone que al conjuntar dos o más métodos de solución (denominados padres), se potencializan las fortalezas de cada uno, y a su vez, se disminuyen sus debilidades.

Algunas metaheurísticos, han surgido de la combinación de otros procedimientos, por ejemplo: la búsqueda glotona aleatoria adaptativa (GRASP por las siglas en inglés de *greedy randomized adaptive search procedures*); pero que se han diferenciado de los procedimientos híbridos por la incorporación de tácticas diferentes a las proporcionas por los métodos padres para la resolución de problemas. Es decir, una nueva metaheurística se convierte en un algoritmo estructurado con al menos una regla o estrategia no tradicional, en la estructuración del algoritmo debe considerarse el alcanzar la mayoría de las propiedades deseables para las metaheurísticas.

En la literatura consultada no hay una definición sobre qué es un **nuevo algoritmo metaheurístico** o qué características debe poseer un método para ser considerado como un nuevo procedimiento. Sin embargo,

cuando se hace referencia a que una heurística es nueva, se contrastan sus características estructurales en las fases de intensificación y diversificación con los procedimientos actuales enfatizando sus características únicas (véase [251, 80]. Por ende, un **nuevo procedimiento metaheurístico** puede ser entendido como aquel procedimiento que cuenta con al menos una regla o estrategia no tradicional o utilizada en los métodos tradicionales.

El termino **hiper heurísticas** fue acuñado para referirse a la idea "heurística que elije heurísticas", o en otras palabras "heurística que genera heurísticas". Las hiper -heurísticas son procedimientos heurísticos y metaheurísticos caracterizados comúnmente por buscar la automatización en la designación y ajuste de parámetros para resolver problemas NP-completos o NP-duros.

La evolución de la heurísticas y metaheurísticos en los últimos años las ha llevado a disminuir la diferencia existente (o grado de error), entre la solución encontrada (solución heurística) y la solución óptima. Sin embargo, para ello' se ha recurrido a un aumento en el consumo de recursos (tiempo y memoria de cómputo).

Capítulo 3

Sistemas sociales, creatividad y música

El presente capítulo tiene los siguientes objetivos: a) presentar ideas y conceptos vinculados con los sistemas creativos; b) mostrar ideas y conceptos vinculados con la música, y la composición musical y las relaciones c) exponer las relaciones entre optimización y música y d) desarrollar una analogía entre los procesos de optimización y composición musical.

Este capítulo se encuentra dividido en tres secciones, las cuales son: a) sociedad y redes sociales, se describe el concepto de sociedad humana y se caracterizan las redes sociales; b) creatividad, se analizan los procesos creativos y sus implicaciones en los sitemas sociales; y c) música se define el concepto de música y se caracteriza el proceso de composición musical.

3.1. Sociedad

El término sociedad proviene del latín *sociĕtas* lo que significa: compañerismo, asociación, unión. Generalmente, se utiliza el término sociedad para referirse a cualquier asociación o grupo de seres vivos, con ciertas semejanzas o coincidencias en su constitución o en sus actividades, que interaccionan entre sí y con su medio ambiente.

El concepto anterior es muy amplio; por ende, es necesario acotarlo. En este trabajo, se utiliza el término

sociedad para referirse a una sociedad humana. Una **sociedad humana** puede ser definida como: la unión[1] de seres humanos interrelacionados a través de vínculos, que interactúan entre sí y con su medio, para alcanzar un bien común; los vínculos permiten: el intercambio incesante de pensamientos, ideas, conocimientos, afectos y sentimientos; y además, una constante comunicación sobre beneficios, servicios, necesidades entre otros [240].

3.1.1. Cultura

La sociedad humana posee y reproduce muchas de las características de la sociedad animal -como son cooperación, especialización, continuidad, entre otras-. Esto se debe a que el modo social de vida es un estadio en la evolución biológica previo al surgimiento del ser humano; sin embargo, existe una diferencia entre las sociedades animal y la humana, la cual es la "cultura" [6].

La palabra **cultura** proviene del latín *cultura* que significa cultivo. En la actualidad, esta palabra se utiliza en una amplia variedad de situaciones (con diferentes significados) que van desde utilizarla para referirse a un punto de estatus o distinción; hasta aplicarla en sentido artístico para referirse a alguna disciplina artística o a los productos emanados de ella. En sociología y áreas afines, se emplea el término cultura para referirse al conjunto de tradiciones, usos y costumbres de un grupo social, etc. [84].

Definición 61 *Cultura es un complejo que comprende conocimiento, arte, creencia, moral, usos y costumbres adquiridas por el hombre en cuanto es miembro de una sociedad; es decir, la cultura surge de la interacción social y consiste en contenidos de conocimiento y pautas de conducta que han sido socialmente aprendidos [6]*

Las unidades más pequeñas de la cultura son los rasgos. Un **rasgo cultural** puede ser definido como una característica mayoritaria (creencia, costumbre, forma de hacer algo) en una población que emerge de la interacción de los individuos en la sociedad. A un conjunto de rasgos culturales claramente diferenciables se les denomina **complejo cultural**

La cultura de un grupo social determinado se puede analizar y caracterizar, a través del punto de vista antropológico; lo cual, implica definir y estudiar los rasgos sociales de dicho grupo a los siguientes niveles: a) descriptivo (enumeración de características); b) histórico (tradiciones y patrimonio social); c)

[1] Un sólo hombre no puede formar una sociedad; ya que se requiere de un conjunto de individuos que se agrupe. Debe hacerse notar que la unidad resultante de la unión no elimina las diferencias entre sus integrantes, sino agrupa la diferencia entre sus miembros, de modo tal que éstos preservan su individualidad

normativo (normas y valores de comportamiento); d) psicológico (dispositivos y aprendizaje para la solución de problemas); e) estructuras (organización); y f) genética (productos, artefactos ideas y símbolos).

3.1.2. Red social

Las sociedades humanas han sido analizadas por varias disciplinas científicas, como son: sociología, medicina, inteligencia artificial, entre otras; lo que ha llevado a generar un gran número de ideas, conceptos, modelos y técnicas para su estudio.

Una de estas ideas es la de **red social**, desarrollada desde la segunda mitad del siglo XX, como una amplicación de la teoría moderna de la comunicación al tejido de interacciones que se configura alrededor de las personas. Esta idea se basa en un enfoque multidisciplinario, el cual analiza las estruturas sociales a través de la teoría de graficas [2].

En términos generales, una red social es un modelo que mapea todos los lazos relevantes entre un conjunto de individuos (personas u organizaciones). Antes de definir, formalmente una red social se analizan las siguientes definiciones sobre red social, disponibles la literatura:

- Barnes [12] define a una red social como un conjunto de puntos unidos a través de líneas; los puntos representan a las personas y a los grupos; y las líneas indican que las personas que están relacionadas entre sí.
- Michell define a una red social como un conjunto específico de vínculos entre un conjunto definido de personas, con la propiedad de que las características de esos vínculos como un todo pueden usarse para interpretar la conducta social de las personas implicadas [58].
- Una red social es una estructura social compuesta de individuos (personas, organizaciones u otras entidades), los cuales están conectados por uno o varios tipos de relaciones (amistad, parestesco, intereses comunes, intercambios económicos, relaciones sexuales, creencias, conocimientos, prestigio, entre otros) [143].
- Zhang define a una red social como un conjunto de actores (también llamados nodos o miembros de la red) conectados por una o más relaciones [70]

[2]La teoría de grafos, también llamada teoría de gráficas es una rama de la matemáticas discretas que estudia las propiedades de los grafos (también llamadas gráficas). Un grafo es un par $G = (V, E)$, donde: V es un conjunto finito, no vacío, cuyos elementos se denominan vértices (nodos o puntos) y E es un conjunto de pares de vértices $V \times V$ que define una relación R, de modo que si los vértices están en la relación, existe al menos una arista que los une [86]

- Rizo menciona que una red social es un sistema abierto horizontalmente, el cual aglutina a conjuntos de personas con un problema común. El principal atributo de esta estructura social es la construcción de interacciones para la resolución de problemas y satisfacción de necesidades [207]

A partir de las definiciones anteriores y el concepto de sistema complejo se construyó la definición 62 sobre red social; ésta será la que se empleará en el presente trabajo.

Definición 62 *Una red social es un sistema complejo y se encuentra compuesta por un conjunto de actores (personas, organizaciones u otras entidades sociales), denominados nodos o miembros de la red, conectados por una o varias de relaciones (amistad, parentesco, intereses comunes, entre otras), llamadas vínculos o aristas. Las interacciones entre los agentes determinan las propiedades, características y funcionamiento de la red social.*

Los elementos básicos de una red social son: a) **nodos** (cada uno de los actores en la red social -cada nodo- es complejo; ya que poseé las capacidades de comunicación, interacción social y procesamiento inteligente de información [239]) b) **vínculos** (son cada uno de los arcos en la red e indican la existencia de algún tipo de relación entre los nodos en la gráfica; pueden ser no dirigidos, o bien, dirigidos) c) **comunicación** (es el principal vínculo entre los miembros de la red; por ende, es fundamental, ya que permite el intercambio de información y conocimiento; es decir, el proceso de comunicación implica que un actor social trasmitirá un mensaje a otros miembros de la red, previo proceso de razonamiento y además será capaz de recibir informacion de otros nodos [239]) d) **sistema de vínculos** (es el conjunto de relaciones entre los nodos); e) **intercambio entre agentes** (son las interacciones entre los actores que generan un flujo de recursos, información y conocimiento) y f) **apioyo social** (las interacciones entre los actores generan un cambio en los vínculos de los actores). Otras de las características de las redes sociales se presentan en el anexo E.

Chritakis y Fowler [30] mencionan que una red social posee las siguientes normas generales:

- Los agentes de red social organizan y reorganizan continuamente la estructura de dicha red.
- Los agentes de una red se ven influenciados por los vínculos e interacciones con otros agentes.
- Las redes sociales poseen propiedades y funciones que sus miembros no poseen, no controlan ni perciben.

3.2. Creatividad

3.2.1. Conceptos básicos

La creatividad ha sido objeto de estudio por varias ciencias como son: biología, la educación, la filosofía, estudios soci-culturales, inteligencia artificial, neurociencias etc. pero, aún representa uno de los temas más misteriosos del comportamiento humano [134].

Definición 63 *Creatividad es el proceso inteligente para unir, juntar, asociar, conectar, integrar o combinar diferentes ideas ya existentes (previamente no relacionadas) de manera no habitual, inesperada, sorpresiva e innovadora, a fin de producir nuevas ideas que serán más complejas y mejor adaptadas para algunos propósitos[49]. La creatividad es resultado de un momento de inspiración (instante de claridad inexplicable e incomprensible); o bien, es causada por un proceso recursivo de estudio, análisis y adaptación sobre alguna idea o pensamiento [110].*

Definición 64 *Un proceso creativo es una sucesión, en el tiempo, de acciones o acontecimientos que dan lugar a un cambio del sistema original que satisfaga un conjunto de objetivos en un determinado momento [99].*

Dada la alta complejidad de la creatividad humana, es necesario investigarla desde más de un punto de vista a fin de entenderla apropiadamente. En este trabajo se analizan dos diferentes niveles de creatividad, los cuales son: creatividad personal y creatividad socio-cultural [134]. A continuación, se describen cada uno de ellos.

3.2.2. Creatividad personal y socio-cultura

La creatividad personal, también llamada creatividad individual, es un proceso psicológico cíclico de razonamiento e imaginación con el fin de generar conocimiento. En otras palabras, la creatividad individual es una cualidad humana personal e intransferible para generar y comunicar ideas y conocimiento (originales y no convencionales) para resolver problemas.

De acuerdo con Shalley, [222] las condiciones necesarias para la creatividad son las siguientes: a) habilidad (es el conocimiento y experiencia del individuo sobre un objeto o idea particular), b) motivación intrínseca (necesidad o deseo); y c) actividades cognitivas (una actividad cognitiva es la acción mental mediante la cual un individuo es capaz de recibir, organizar, integrar, relacionar y modificar información de la realidad a fin de construir conocimiento).

Un pensamiento creativo a nivel personal posee algunas de las siguientes características [134]:

1. El pensamiento debe ser novedoso y poseer un valor para el individuo y su cultura.
2. El pensamiento debe ser no convencional.
3. Involucra la reformulación o replanteamiento del problema que se desea resolver.
4. Es resultado de un arduo proceso de razonamiento e imaginación.

La creatividad socio-cultural, también denominada creatividad colectiva, es un proceso cíclico de interacción y comunicación entre los individuos de un grupo para la construcción de conocimiento. En otras palabras, la creatividad social es un proceso de evaluación, filtrado, aceptación y aprendizaje social del conocimiento para la solución de problemas; dicho proceso es de carácter alocéntrico, ético y constructivo.

Csikszentmihalyi modeló el sistema de creatividad socio-cutural (véase Figura 3.1); en este modelo se considera que las personas tomen información y conocimiento de su cultura y la transformen; posteriormente, el individuo comunica su transformación a algunos de los demás miembros del sistema social; quienes la analizan, evalúan y en su caso aceptan. Si un cambio en el conocimiento es aceptado por la sociedad[3], entonces éste se incluye en el dominio [4] y trasmitido a las nuevas generaciones a través de la cultura.

La creatividad individual puede generar cambios a nivel social, a través, de la conunicación entre las personas. De manera similar, la creatividad social puede generar cambios en el comportamiento individual.

3.2.3. Sociedades artificiales y creatividad

En sociedades artificiales el término cultura artificial ha llevado a una confrontación científica, a un reto en las ciencias afines a la inteligencia artificial; pero también ha sido una fuente de ideas, conceptos útiles que permiten la emulación y estudio de varios sistemas humanos complejos.

Los niveles de complejidad cultural son: a) En la mente de cada agente existen una multitud y una gran variedad de pensamientos; b) entre las mentes de los agentes existe conocimiento cultural distribuído; y c) la cognición de los agentes [84]. Por lo anterior, la creatividad artificial es una representación mínima de los objetos y procesos culturales donde sólo se consideran las funciones esenciales; es decir, es una mezcla primitiva de las propiedades intelectuales, sociales y de medio ambiente de un grupo de humanos.

El término creatividad artificial fue acuñado en la inteligencia artificial y es utilizado para referirse a modelos computacionales que han sido desarrollados para simular e imitar los procesos mentales involucrados

[3]Se dice que un cambio es aceptado por la sociedad, si la mayoría de sus miembros admiten dicho cambio

[4]En términos generales, el dominio está formado por el sistema de símbolos de la cultura, el lenguaje y la notación específica de esa área

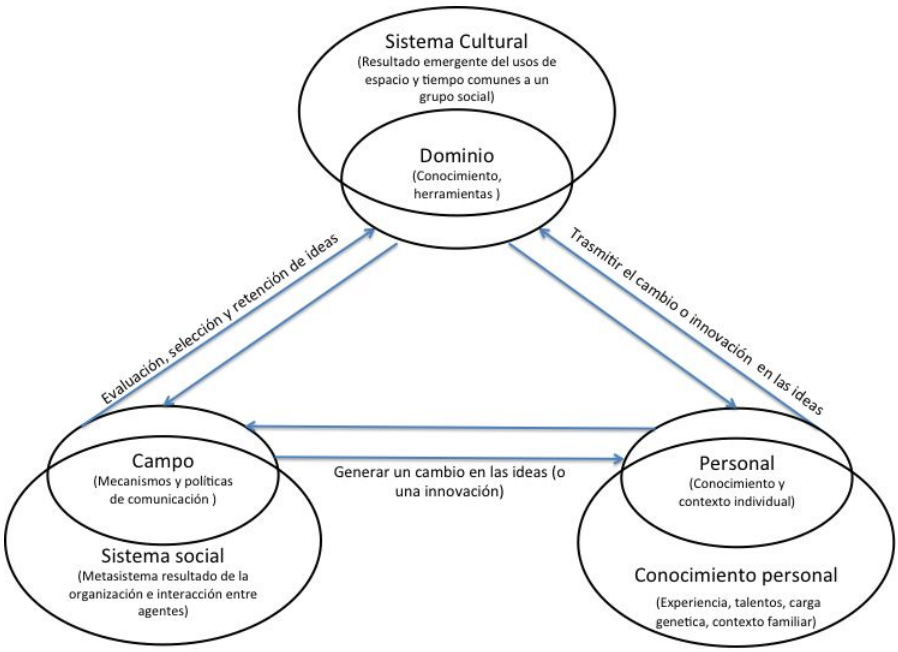

Figura 3.1: Modelo de sistema creativo Funte:[102]

por el ser humano para generar nuevas obras de arte [220]. Estos modelos pueden basarse en creatividad personal o bien en creatividad socio-cultural.

Los algoritmos de creatividad artificial, emulan en cierta medida la capacidad creativa de un artista, se componen de dos procesos: uno "generador" y otro "evaluador". El primero de los procesos, es sencillo de implementar en una máquina acogiéndose a la idea de que la creatividad es resultado de un proceso recursivo de perfeccionamiento; por ende, para la ejecución de esta tarea se conforma un conjunto de reglas, pasos e instrucciones basándose en la experiencia de los artistas o en la teoría disponible. Sin embargo, que una máquina efectué el segundo proceso representa un arduo trabajo; por lo cual, algunos algoritmos requieren de la intervención humana mientras otros implementan conceptos relacionados con heurísticos. Lo anterior se debe a que los algoritmos de creatividad artificial tienen serias implicaciones filosóficas que van desde ¿cómo reemplazar a los sentimientos humanos?¿cómo se reemplaza a la imaginación, a la inspiración y a la intuición humana?, etc [41].

En la actualidad, se han generado y utilizado varios algoritmos de creatividad artificial para la generación de elementos artísticos en la música, en la literatura y las artes visuales. Un método es considerado como un algoritmo de creatividad artificial si parte de las siguientes premisas esenciales: a) el modelo contiene una sociedad de agentes situados en un entorno cultural; b) ningún agente puede afectar directamente el

comportamiento de otro; c) existen reglas que dirigen el comportamiento global del sistema; d) los agentes interactúan con otros agentes para intercambiar ideas, artefactos y opiniones; e) los agentes interactúan con el medio ambiente para acceder a los símbolos culturales; f) los agentes pueden evaluar la calidad de los artefactos de los agentes con los que tienen un vínculo y decidir si la información o artefactos le son útiles.

3.3. La música

La música está presente en la vida cotidiana, ya sea, en la recreación, en la comunicación, en el arte, en la cultura o simplemente dispersa en el medio circundante [182]. Ésta es una de las expresiones artísticas, puesto que es una actividad intelectual cognitiva y profunda que se basa en el paradigma estético.

La música tiene una gran influencia en el ser humano; lo cual, la ha colocado como un elemento de identidad personal y social [182]; ya que, el sonido impacta en la conducta individual y colectiva, además, es una de las formas más antiguas de expresión [197]. El significado musical es tan amplio como la cultura misma, la religión o la propia sociedad en que se desarrolla [177].

El significado de la palabra música ha cambiado a lo largo de la historia. A continuación, se muestran algunas acepciones utilizadas para el término música:

1. El origen etimológico de la palabra música proviene del griego “musikè” que significa “el arte de las musas”.
2. Según el compositor Claude Debussy (1862-1918), la música es un total de fuerzas dispersas expresadas en un proceso sonoro que incluye: el instrumento, el instrumentista, el creador y su obra, un medio propagador y un sistema receptor [168].
3. Música es el arte de combinar sonidos en el tiempo, el cual es inmaterial y efímero, pues pasado el momento de interpretación sólo queda en la memoria [182].
4. Según Claude Lévi-Strauss(1908-2009), la música es un lenguaje con algun significado al menos para la inmensa mayoría de la humanidad, aunque sólo un pequeño número de personas son capaces de crearlo; además, es el único lenguaje con los atributos contradictorios inteligible e intraducible [130].
5. Música es sonido en movimiento, lo que sugiere contrastes sonoros generados por un compositor con la finalidad de derivar en emoción estética.
6. Música es la combinación consistente del tiempo (duración del sonido), altura (sonido grave o agudo) y la intensidad del sonido [7].

7. Música es el arte que combina sonidos con el tiempo para conseguir un efecto estético además sirve para comunicar sentimientos y sensaciones entre las personas [21].
8. Según Anthony Storr (1920-2001), la música puede ser considerada como una forma de comunicación entre las personas, aunque generalmente no se tiene claro el mensaje que se comunica [229].

En este trabajo se define música como:

Definición 65 *La música es el arte, la ciencia y el lenguaje que organiza motivos (unidades musicales), considerando los principios de melodía, de ritmo y de armonía a fin de generar y trasmitir un mensaje entre el compositor y el espectador.*

En la definición anterior se considera como unidad musical básica al "motivo"; ya que éste desempeña una función dentro de la música semejante a la realizada por la palabra en el lenguaje. A continuación, se dan algunas definicines relacionadas con motivo.

Definición 66 *Motivo es un conjunto de notas que forman una unidad melódica, rítmica y armónica de una obra musical; es decir, los elementos que configuran un motivo son interválicos y rítmicos, los cuales al combinarse producen una forma o contorno reconocible que usualmente implican armonía inherente [221]. El uso conciente del motivo debe producir: unidad, relación, coherencia, lógica, inteligibilidad y fluidez en una obra musical [221].*

Definición 67 *Una nota es una representación simbólica de las características de un sonido o un silencio que forma una pieza (o fragmento) musical.*

Definición 68 *El sonido es la sensación percibida por el oído al recibir vibraciones en un medio, causadas por cuerpos vibratorios. Dadas las características fisiológicas del oído y de las zonas auditivas en la corteza cerebral, existen cuatro cualidades fundamentales que determinan el carácter de un sonido, las cuales son: altura (tono), timbre, duración, intensidad (volumen).*

Definición 69 *El silencio es ausencia de sonido y su única cualidad es la duración.*

En la Tabla 3.1 se esquematizan las principales características y cualidades del sonido y del silencio, así como su expresión musical.

Tabla 3.1: Expresión musical de las cualidades del sonido y el silencio

Cualidad	Causa	Efecto	Figuras musicales	Expresión musical.
Altura o tono	Frecuencia o número de vibraciones por segundo	Sonidos grave o agudo	Tonos y semitonos de escalas musicales. Intervalos. Melodía y armonía	Pentagrama, notas, claves, escalas.
Intensidad o volumen	Amplitud o tamaño de la vibración de onda respecto al punto de reposo	Sonido fuerte o débil.	Acentos, matices expresivos y pasos de matices	Reguladores y matices (piano, forte, mezzoforte).
Duración	Tiempo de permanecia o ausencia de la onda sonora	Sonido o silencio largos o cortos	Ritmos o silencio, acelerandos y reiterados, tempo	Figuras. Compases Tempo (adagio, allegro, vivo).
Timbre	Forma de la onda	Color del sonido	Contrastes tímbricos vocales e instrumentales	Instrumentos armónicos. Familias de instrumentos. Tipos de orquestas.

A raíz de que la música es un sistema de comunicación desfasado; se requieren elementos que permitan una eficiente trasmisión del mensaje desde el compositor hasta el público; la escritura musical provee a los compositores de las herramientas de expresión para plasmar y trasmitir su mensaje. En otras palabras, la simbología musical en unión con reglas de composición ejecuta en la música una función similar a la realizada por el alfabeto en conjunción a las reglas gramaticales para la literatura.

Al proceso de organización lógico y coherente del sonido y el silencio, a fin de producir música se le denomina composición musical, el cual será abordado en la siguiente subsección.

3.3.1. El proceso de composición musical

La palabra componer proviene del latin *componĕre*, que significa "juntar varias cosas para formar otra cosa que se expresa"; a partir de lo anterior, se podría suponer que el proceso de composición musical se limita a la concatenación de sonidos y silencios; lo cual, es una idea errónea, pues éste es un proceso creativo para la combinación de imaginación, elementos musicales y conocimientos a fin de obtener una obra musical que cumpla con cierto estándar estético. Es decir, el procesos de composición es más complejo que la simple asociación de elementos, ya que este proceso involucra los sistemas cretivos socio-cultural y personal para la gereneración de un producto artístico.

A nivel personal involucra que cada compositor, de manera iterativa y estructurada, selecciona, usa, combina, adecúa, genera melodías en función de su conocimiento (reglas, pasos, la abstracción, la implementación, el análisis y la corrección de ideas, principios instrucciones) experiencia e intuición. A nivel socio-cultural conlleva que cada compositor a partir de sus interacciones sociales obtiene y analiza la información sobre el grado de aceptación, de su obra y de las propuestas de otros músicos; con base en lo cual, obtiene y modifica su conocimiento, lo que conlleva a la emergencia de un patrón sobre las características de una obra que satisfaga los gustos del público.

3.3.2. Algoritmos utilizados para la generación de música

La aplicación de algoritmos en la música es tan vieja como el proceso de composición [110] y a partir de la década de los 50 se ha implementado la composición asistida por computadora. Los algoritmos en la música usan una variedad de caminos que incluyen desde la síntesis del sonido, la toma de muestras y composición. Los algoritmos de composición aplican una serie de reglas e instrucciones para obtener resultados [41].

A continuación, se mencionan algunos de los algoritmos desarrollados para la composición musical:

- Algoritmo genéticos para la composición musical asistida por computadora [105]. En este caso se

utiliza un sistema basado en un algoritmo genético para seleccionar operadores que serán utilizados en el proceso de composición; para ello el autor implementa un procesos quede transformación de un patrón musical inicial en otro patrón diferente.

- GenJam [18]. Este caso se trata de un algoritmo genético interactivo que imita la improvisación del jazz.
- Composición con algoritmos genéticos [109]. A través de este algoritmo el autor implementa cambios estocásticos en "frases iniciales´´ con el principal objetivo de conseguir çomposiciones que suenen bien´´. En esta propuesta el autor genera un conjunto de pequeñas frases, cada una de ellas es evaluada para decidir si es o no aceptada; en caso de ser rechazada se reordenan los elementos de la frase de distintas maneras hasta conseguir el un buen orden.
- Algoritmo de composición como un proceso creativo [110].
- Alice [41, 42]. Este algoritmo crea música a través de búscar la continuidad y la formal de la lógica de la música utilizada como base.
- SARA [41, 42], por las siglas en ingés de *simple snalytic recombinant algorithm*. En este algoritmo se implementan de manera racional los cambios temporales en una cadena de Márkov utilizada como modelo inicial.

Parte II

Diseño del método de composición musical

Capítulo 4

Diseño y desarrollo del método de composición musical

En este capítulo se construye el modelo general del algoritmo metaheurístico propuesto en esta tesis; a esta técnica se le ha denominado "método de composición musical o MMC". En el planteamiento y desarrollo del MMC se utilizaron los conceptos y las ideas expuestos en los capítulos precedentes .

La estructura del presente es la siguiente: en la primera sección se examinan conceptos básicos para el diseño de una metaheurística; en la segunda, se analizan las relaciones entre composición musical y optimización y se plantea una analogía entre ambos procesos; en la última sección se propone y desarrolla el MMC.

4.1. Conceptos básicos para el diseño y desarrollo

El diseño [1] y desarrollo de buenos métodos heurísticos y metaheurísticos es un proceso no trivial que requiere de una buena dosis de imaginación, intuición, conocimiento (véase Figura 4.1) e información [236].

Además, durante el diseño y desarrollo de las técnicas metahurísticas se deben considerar las propiedades deseables[2] en estos procedimientos, las cuales son: [142]:

[1] De manera general, diseño se puede definir como una actividad creativa que conlleva la planificación y materialización de ideas y conceptos (originales) para la búsqueda de una solución en cualquier campo

[2] Una propiedad deseable son todas aquellas que favorecen el interés práctico y teórico

Figura 4.1: Conocimientos implicados en el diseño de metaheurísticas

- **Simple**. La metaheurística debe estar basada en un principio sencillo y claro; fácil de comprender.
- **Precisa**. Los pasos y fases de la metaheurística deben estar formulados en términos concretos.
- **Coherente**. Los elementos de la metaheurística deben deducirse naturalmente de sus principios.
- **Eficaz**.Debe existir una alta probabilidad de alcanzar soluciones óptimas de casos realistas con la metaheurística.
- **Eficiente**. Se debe realizar un buen aprovechamiento de recursos computacionales: tiempo de ejecución y espacio de memoria.
- **Adaptable**. Debe ser capaz de adaptarse a diferentes contextos de aplicación o modificaciones importantes del modelo.
- **Robusta**. El comportamiento de la metaheurística debe ser poco sensible a pequeñas alteraciones del modelo o contexto de aplicación.
- **Interactiva**. Debe permitir que el usuario pueda aplicar sus conocimientos para mejorar el rendimiento del procedimiento.
- **Múltiple**. Debe suministrar diferentes soluciones alternativas de alta calidad entre las que el usuario pueda elegir.
- **General**[3]. Debe ser utilizable con buen rendimiento en una amplia variedad de problemas.
- **Autónoma**[4] . Debe permitir un funcionamiento autónomo, libre de parámetros o que se puedan establecer automáticamente.

Wallas [137] menciona que la generación de un nuevo método de solución de problemas involucra un proceso recursivo de las siguientes fases: a) preparación (consiste en la recolección y análisis de información sobre el problema a resolver y los métodos de solución); b) incubación (involucra el proceso mental de conexión de conceptos; es decir, en esta fase se genera el constructo sobre el problema a resolver); c) inspiración (es donde sucede el proceso creativo, a fin de unir conceptos disjuntos a fin de hallar una táctica para resolver el problema); y d) verificación (es la aplicación de pruebas prácticas que permitan verificar la validez de la táctica desarrollada).

Este proceso continúa hasta alcanzar un equilibrio entre el grado de cercanía de la solución obtenida con respecto a los recursos utilizados para su generación; es decir, se realiza una serie de aproximacimaciones, a

[3]El uso y explotación de la información de un problema para generar un procedimiento *ad hoc* para resolverlo, generalmente, se contrapone con la propiedad de generalidad

[4]En la actualidad, no existe ninguna metaheurística que sea autónoma

fin de alcanzar un nivel alto de balance entre los recursos necesarios y la calidad de la solución encontrada, lo cual se esquematiza en la Figura 4.2.

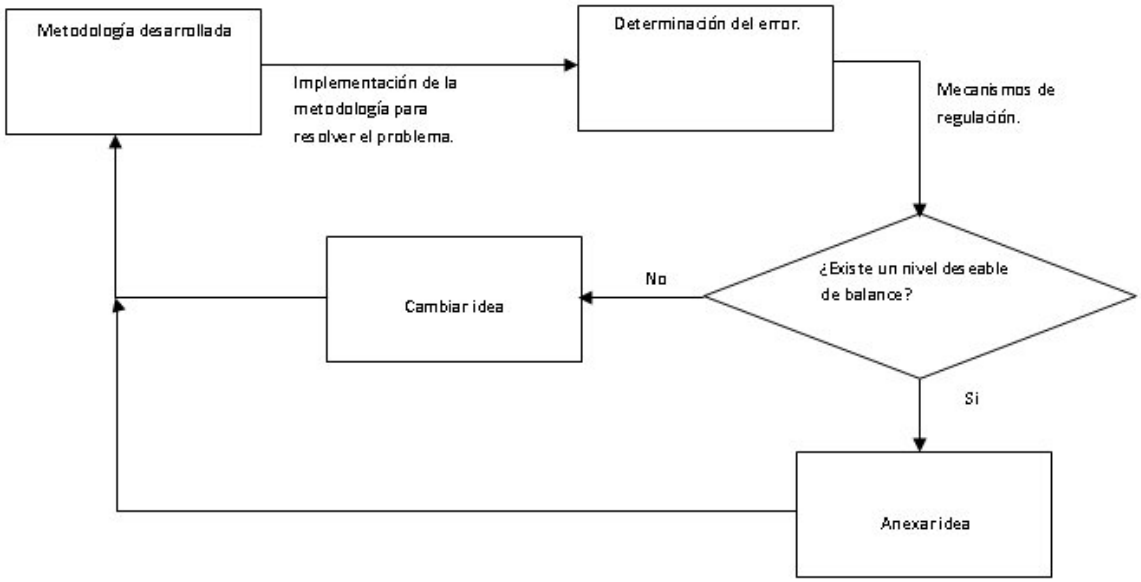

Figura 4.2: Mecanismo de Piaget para la búsqueda de balance [27]

El nivel de balance existente entre los recursos requeridos por una metaheurística y la cercanía de la solución encontrada con la óptima; influye en el grado de aceptación que tiene este mecanismo de solución para un problema determinado.

4.2. Relaciones entre composición musical y optimización

4.2.1. Sistema creativos, arte y composición musical

Todo pensamiento, en particular el arte, refleja la realidad de un sociedad; ya que muestra la pluralidad de los puntos de vista, la fugacidad caleidoscópica y contradictoria del mundo [72].

De acuerdo con Marx, el arte, la cultura, la religión, ciencia, filosofía, entre otros, conforman la **superestructura social**; ésta emerge de la **infraestructura social** (relaciones y modo de producción) y constituyen la conciencia social [72, 213]. De manera general, se puede decir que la superestructura se

conforma con las ideas, conceptos, opiniones, patrones de conducta que siguen los miembros de un grupo social [93]. La superestructura influye en el artista; ya que, éste constantemente define y redefine el **campo artístico** basado en la información disponible. El campo artístico se conforma de los medios de producción (recursos tecnológicos para generar una obra de arte); la relaciones sociales de producción específicas de un área artística (relaciones entre los artistas, público, intermediarios, marchands[5], instituciones, comercializadoras entre otros) [72]. Con base en lo anterior y al modelo del sistema de creatividad socio-cultural, se desarrolló el modelo de creatividad del proceso artístico, el cual se muestra en la Figura 4.3

La composición musical es un proceso creativo (a nivel personal y el socio-cultural) a través del cual, se genera una obra de arte. A nivel individual, la creatividad de un compositor es causada por momentos de genialidad (destellos de genialidad) o a través de un proceso recursivo de razonamiento sobre un pensamiento, (trabajo duro) [110]. En contraste, la creatividad socio-cultural se produce por la interacción de los miembros sociales, lo que genera cambios en la estructura social.

En el caso de la música, la supraestructura se conforma con los patrones y las convenciones generados por los gustos y preferencias de un determinado grupo social. Por ende, los estilos o géneros musicales surgen de los elementos comunes en un conjunto de piezas musicales individuales; por ejemplo: las pieza de pop contemporáneo se producen a partir de la combinación de un conjunto pequeños de armonías y melodías [16].

4.2.2. Sistemas multiagente, comportamiento social y algoritmos sociales

De manera general, un **agente** es una entidad capaz de reaccionar a los cambios en su entorno y desarrollar procesos inteligentes. Un **sistema multiagente** es un sistema constituido por un número de agentes que interactúan entre sí.

En las décadas pasadas, el comportamiento social del ser humano ha sido objeto de estudio por varias ciencias, como son: la sociología, la antropología, la biología, las ciencias de las computación entre otras. Generalmente, estos estudios eran unidisiplinarios. Recientemente, el estudio multidisiplinario de este tema ha tomado fuerza, con lo que se ha favorecido el flujo y la generación de conocimiento entre las distintas disciplinas que analizan y estudian a las sociedades humanas.

Los algoritmos sociales son resultados de la interacción entre ciencias; éstos emulan los fenómenos sociales [169] a través de las interacciones en un sistema multiagente; donde el conjunto de agentes (son el grupo

[5]El término "marchand" se utiliza para referirse a los expertos en el mundo del arte que ofrecen al público las obras de los artistas que han seleccionado con anterioridad

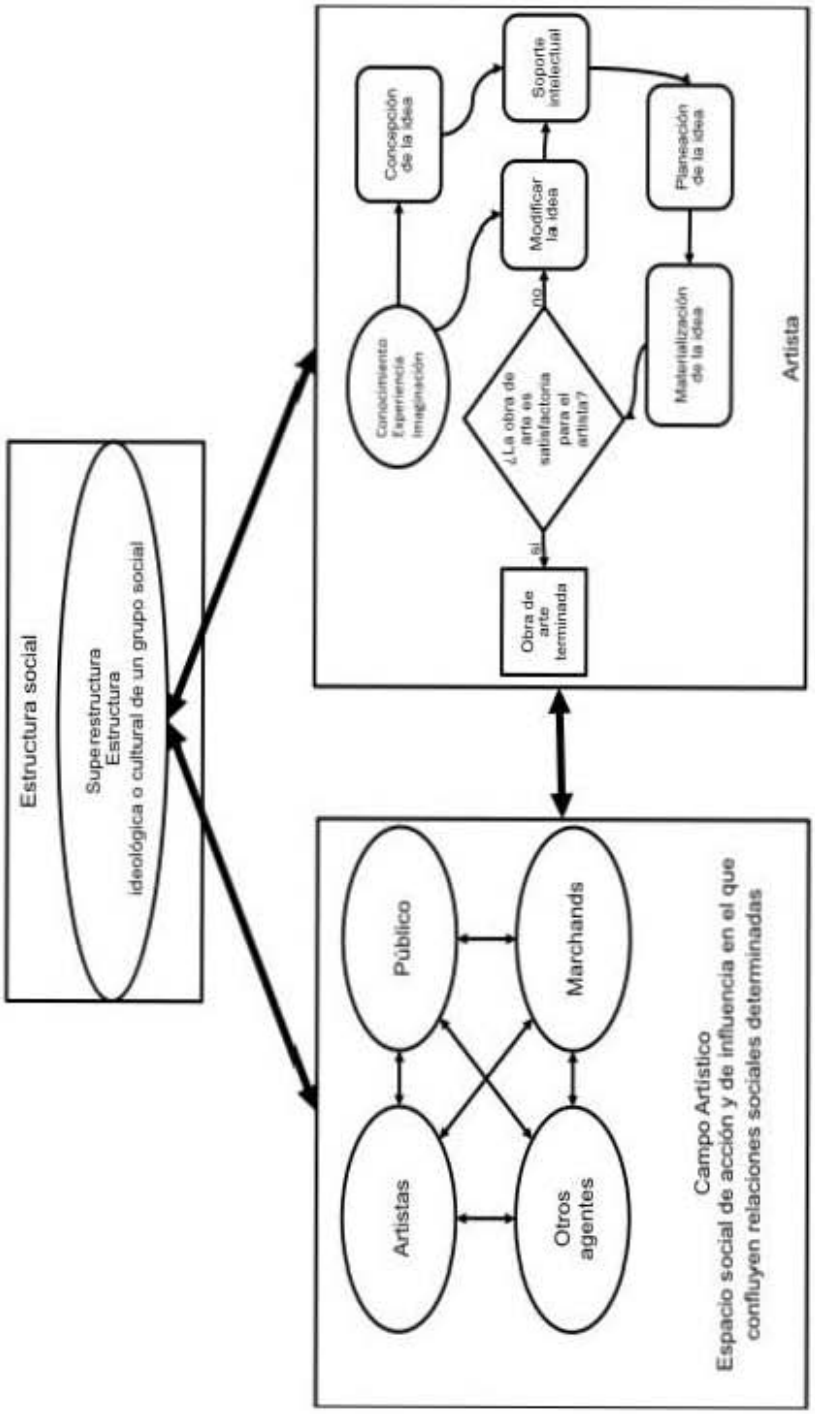

Figura 4.3: Sistema creativo para la producción artística

social) y las reglas de comportamiento (relaciones observadas entre los individuos de la sociedad). En la sección 2.2.3.4 de este trabajo, se presentaron algunos de estos algoritmos utilizados frecuentemete en la solución de problemas de optimización.

4.2.3. Analogía entre optimización y composición musical

Una analogía es una forma de razonamiento en el cual una idea o concepto se infiere a partir de las similitudes con otra idea o concepto en ciertos aspectos, lo cual se realiza sobre la base del conocimiento [49, 42] .

En la construcción una analogía entre los proceso de optimización y composición musical, se tomarón como base las ideas siguientes:

1. Se asume que la actividad de composición musical conlleva un número finito de pasos [41]
2. De manera natural el proceso de composición se puede percibir como un algoritmo; pues la necesidad de componer es similar a la necesidad de resolver un problema [41]
3. Un sistema de creatividad socio-cultural implica que los individuos sean capaces de aprender de sus experiencias además de poder utilizar esta información en futuras decisiones.
4. Los cambios posibles a una obra musical se basan en la alusión a trabajos previos, recombinación de ideas e influencias de otros compositores a través del aprendizaje adquirido y los destellos de genialidad.

El proceso de composición musical es un proceso creativo (a nivel personal y socio-cultural), basado en el cononocimiento e imaginación del compositor, para la producción de una pieza musical, el cual surge por la necesidad y/o deseo de un compositor; en este proceso, se seleccionan y organizan motivos musicales[6], a fin de crear una obra musical que genera la máxima satisfacción del artista. De manera similar el proceso de optimización[7] se inicia con la necesidad de resolver un problema para la toma de decisiones.

Adicionalmente, el compositor aplica repetidamente arreglos a su obra musical (pieza musical); hasta conseguir el mayor grado de satisfacción deseado, análogamente al optimizar se recurre a una serie de

[6]En música, un motivo es la mínima unidad significativa de sonido; es decir, un motivo es una sucesión de notas representativas de una pieza musical. Un motivo en música es similar a una palabra en el lenguaje; por ende, un compositor debe combinar adecuadamente los motivos a fin de transmitir eficientemente al público el mensaje deseado.

[7]En este proceso se involucra el conocimiento y creatividad humana a fin de precisar y modelar adecuadamente el problema posteriormente a través de los métodos de solución se asignan y combinan valores a las variables de decisión, a fin de generar el mejor valor posible en la función objetivo (o funciones objetivo), es un proceso que garantiza tomar la mejor decisión posible.

iteraciones a fin de encontrar la mejor solución (óptima). En la Tabla 4.1 se muestra la analogía entre optimización y composición musical.

Tabla 4.1: Comparación entre optimización y composición musical, Fuente [156, 157].

Proceso	Optimización	Composición musical
¿ Para qué se realiza?	Resolver un problema de programación matemática.	Componer una pieza musical.
¿Qué se desea obtener?	Óptimo global	Una obra musical que genere el mayor grado de satisfacción al artista .
¿Cómo se estima la calidad de los resultados obtenidos?	Función objetivo	Grado de satisfacción del artista .
¿Cuáles son los elementos de decisión?	Variables de decisión	Motivos musicales
¿Qué se hace con los elementos de decisión?	Asignarles valor	Asignarles un conjunto de notas.
¿Pueden ser vistos como procesos recursivos finitos?	Si	Si
¿ Cuál es el proceso unitario de mejora?	Iteración	Arreglo

Con base en lo anterior, se adaptó el modelo de sistema creativo propuesto por Liu en [134]; el cual, fue guía en el desarrolló de la metaheurística propuesta. En el modelo adaptado, los agentes son los compositores, ya que ellos pueden crear y modificar las obras. Lo anterior se muestra en la Figura 4.4

En la siguiente sección, se describe el método creado a partir de la analogía y modelo multiagente generados.

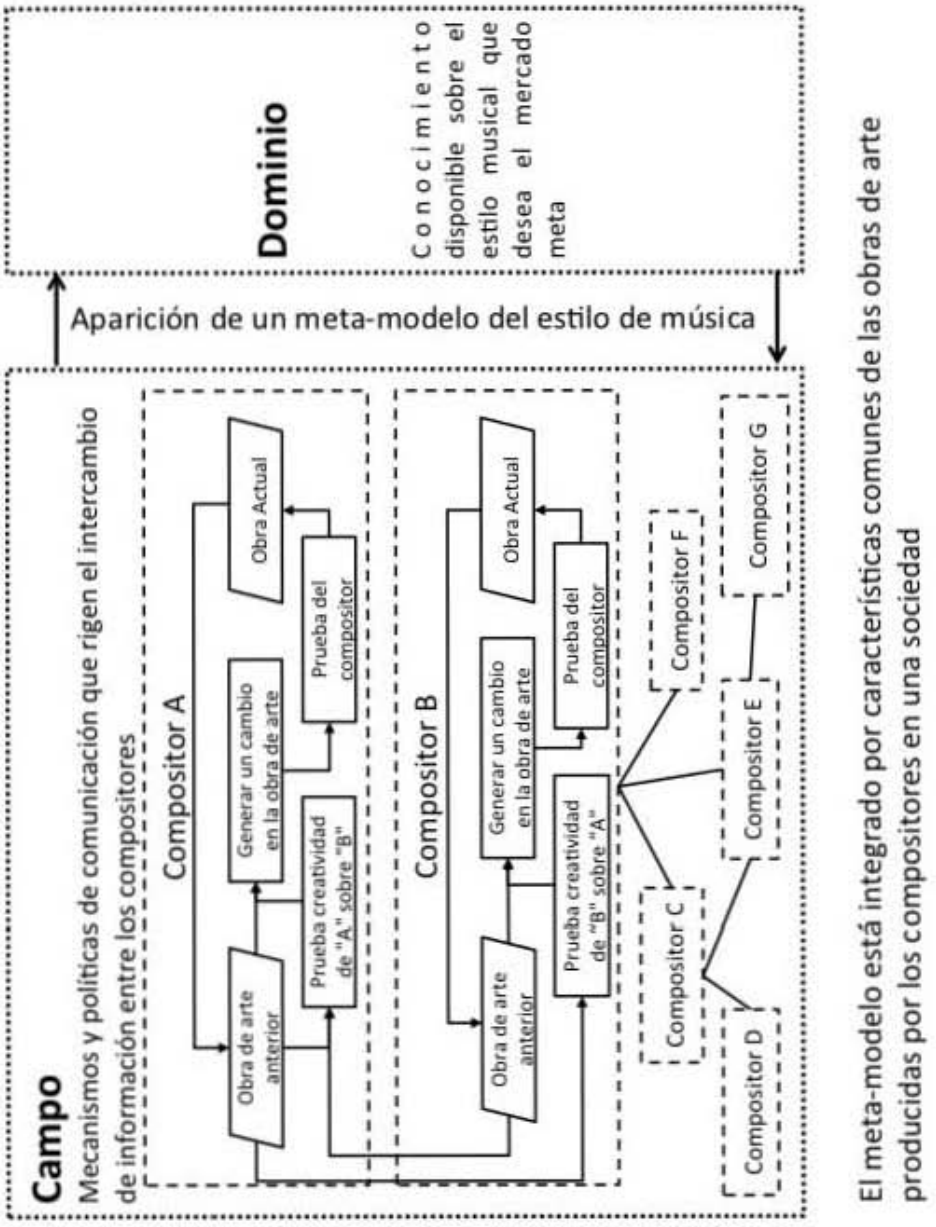

Figura 4.4: Modelo del sistema creativo en la composición musical Fuente [156, 157].

4.3. Método de composición musical

La estructura básica del algoritmo propuesto es la siguiente: inicialmente, se genera una sociedad artificial de Nc compositores y se definen las reglas de interacción entre los agentes que la conforman. Entonces, para cada uno de los compositores en la sociedad, aleatoriamente, se crea un conjunto de Ns temas que se registran en la partitura asociada a ese compositor($P_{\star,\star,i}$) -la partitura servirá como la memoria del compositor- Posteriormente y hasta satisfacer el criterio de paro, se realiza lo siguiente: a) se actualizan los vínculos entre los compositores de la sociedad; b) los compositores interactúan entre sí; cada uno de ellos analiza la información recibida de los demás compositores y selecciona las datos que tomara de su entorno, a este conjunto de datos se le denomina ideas adquiridas socialmente ($ISC_{\star,\star,i}$); c) el $i-$ésimo compositor construye su conocimiento ($KM_{\star,\star,i}$), con base en su $P_{\star,\star,i}$ y las ideas adquiridas socialmente; en términos generales, $KM_{\star,\star,i} = P_{\star,\star,i} \cup ISC_{\star,\star,i}$; d) cada compositor genera una nueva melodía ($x_{\star,new}$), a partir de su conocimiento y los posibles destellos de genialidad; después, este compositor determina el grado de satisfacción alcanzado con $x_{\star,new}$ y e) finalmente, el $i-$ésimo compositor, con base en el grado de satisfacción, debe decidir si $x_{\star,new}$ remplazará a algún elemento de su partitura.

Las etapas del MMC se pueden agrupar en las siguientes fases: a) **Inicializar el proceso de optimización** incluye desde ingresar la información al algoritmo hasta la generación de la partitura inical para cada agente ; b) **interacción entre los agentes** incluye desde la actualización de los vínculos en la red social hasta el intercambio de información entre los compositores; c) **generación y evaluación de una nueva melodía para cada uno de los agentes** involucra desde la construcción de $KM_{\star,\star,i}$ hasta determinar el nivel de satisfaccíon alcanzado por $x_{\star,new}$; d) **actualizar la obra de arte para cada uno de los agentes** involucra decidir si $x_{\star,new}$ reemplazará a alguna melodía en la partitura ; e) **construir el conjunto de soluciones** se toma la mejor melodía de cada uno de los agentes. Cabe mencionar que las fases b), c) d) y e) se repiten hasta alcanzar el criterio de paro. En el Algoritmo 35, se muestra con mayor detalle el funcionamiento general del algoritmo propuesto.

En las siguientes subsecciones, se describe cada una de las fases del MMC.

4.3.1. Descripción del método de composición musical

4.3.1.1. Inicializar el algoritmo

En esta fase se alimenta al algoritmo con la instancia de optimización a resolver. Además, se asigna un valor numérico a los parámetros necesarios para la ejecución del MMC. Estos parámetros así como sus

Algoritmo 35: Algoritmo MMC básico

```
1  Crear una sociedad artificial con reglas de interacción entre los agentes.
2  for cada uno de los agentes de la sociedad do
3  |  Generar aleatoriamente una obra musical (Para esta actividad se considera la información
   |  sobre la instancia a resolver)
4  end
5  while No se satisface el criterio de paro do
6  |  Actualizar los vínculos de la red social.
7  |  Intercambiar información entre agentes.
8  |  for cada uno de los agentes de la sociedad do
9  |  |  Actualizar la matriz de conocimiento.
10 |  |  Generar y evaluar una nueva melodía (x_{⋆,new})
11 |  |  Actualizar la partitura .
12 |  end
13 |  Construir el conjunto de soluciones encontradas con la mejor melodía de cada compositor.
14 end
```

Tabla 4.2: Características de los parámetros del algoritmo MMC

Nombre del Parámetro	Símbolo	Descripción
Máximo número de arreglos	máx $\_arrangement$	máx $\_arrangement \in \mathbb{N}$
Factor de genialidad sobre innovación	ifg	$ifg \in [0 \quad , \quad 1]$
Factor de genialidad sobre cambio	cfg	$cfg \in [0 \quad , \quad 1]$
Factor de cambio entre las relaciones de los agentes	$fcla$	$fcla \in [0 \quad , \quad 1]$
Número de compositores	Nc	$Nc \in \mathbb{N} \backslash [0 \quad , \quad 2)$ ya que una sociedad es un grupo de personas que interaccionan entre sí, se requiere que existan al menos dos individuos.
Número de acordes en las obras de arte	Ns	$Ns \in \mathbb{N} \backslash [0 \quad , \quad 3)$ En la música un acorde es una combinación de tres o más tonos que suenan simultáneamente; la armonía es la ejecución concurrente de acordes; por tanto, se decidió establecer los posibles valores de este parámetro dentro de $3, 4, 5, \ldots, \infty$.

características se muestran en la Tabla 4.2

Los factores de genialidad (ifg y cfg) son probabilidades, que emulan la espontaneidad del ingenio humano en los procesos creativos. El parámetro ifg es la probabilidad que el $i-$ésimo compositor genere una nueva melodía completa, sin recurrir a la información en $KM_{\star,\star,i}$. En contraste, el parámetro cfg es la probabilidad de que el $i-$ésimo compositor genere algunos de los elementos en la nueva melodía sin utilizar su conocimiento actual. Por otro lado, el parámetro $fcla$ es la probabilidad de que el $i-$ésimo compositor modifique algunas de sus relaciones con los otros agentes en la sociedad.

Al determinar un valor para los parámetros máx _$arrangement$ y Nc, se debe considerar que el número de evaluaciones (Ne) que realizará el algoritmo se relaciona directamente con estos elementos como lo muestra la ecuación 4.3.1.

$$Ne = Nc * \text{máx}\_arrangement \tag{4.3.1}$$

Inicialmente, se genera un conjunto de Nc compositores y se definen las normas de comportamiento, también denominadas reglas de interacción, para los agentes en la sociedad.

- Se determinan las caractrísticas de los arcos (vínculos) entre los miembros de la sociedad, estos vínculos son los canales de comunicación y por ende, influyen en el flujo de la información entre los agente. Si el conjunto de arcos en la red son dirigidos, entonces se indica que las relaciones entre los agentes no son simétricas entre sí. En contraste, si los arcos son no dirigidos las relaciones entre los miembros de la sociedad serán simétricas.
- Se establecen las reglas de análisis y selección de información que serán ocupadas por los agentes en la sociedad. La norma de análisis le permiten al $i-$ésimo compositor determinar si la información que proviene del $k-$ésimo compositor le es útil o no. En contraste, el código de selección le permite elegir al $i-$ésimo agente información que adquirirá del $k-$ésimo compositor. En la sección 4.3.1.2 se exponen con mayor detalle estas reglas.
- Se fija la condición para que el $i-$ésimo compositor actualice su partitura. En la sección **??**, se trata con mayor detalle esta condición.

Posteriormente, se produce para cada uno de los compositores una partitura inicial; la cual, generalmente, se crea de manera aleratoria $P_{\star,\star,i}$ por medio del Algoritmo 36 .

4.3.1.2. Interacción entre agentes

En esta fase, se emula el comportamiento dinámico de las redes sociales humanas (en las cuales, cada persona es capaz de estructurar y reestructurar sus vínculos, con otros seres humanos, en función de sus deseos o necesidades) y adicionalmente, se imitan las capacidades cognitivas del ser humano para aprender de su entorno. Por ende, esta fase se divide en las siguientes sub-fases: a) actualización de vínculos entre los compositores e b) intercambio de información.

a) **Actualizar vínculos entre los compositores.**

El objetivo de esta subfase es mimetizar los cambios sociales en el tiempo; para ello, el algortimo utiliza el parámetro $fcla$ y la estrategia mostrada en el Algoritmo 37.

Algoritmo 36: Generación de las partituras iniciales

Input: Número de variables de decisión (n), rango posible para cada una de las variables de desición $[x_\star^L, x_\star^U]$,Nc, Ns

Output: Las partituras asociadas a cada agente en la sociedad

```
for i = 1 : Nc do
    for j = 1 : Ns do
        for l = 1 : n do
            x_l ← Valor aleatorio tomado de [x*^L, x*^U]
            P_⋆,⋆,i ← x_l
        end
    end
end
```

En la primera parte del algoritmo 37, se construye la red social inicial, es de hacerse notar que en esta construcción se evita la generación de bucles [8] en la red social. En la segunda parte del mismo se producen cambios aleatorios en los arcos de la red en el tiempo t con respecto a la red del el tiempo $t-1$ tal y como se muestra en la Figura 4.5.

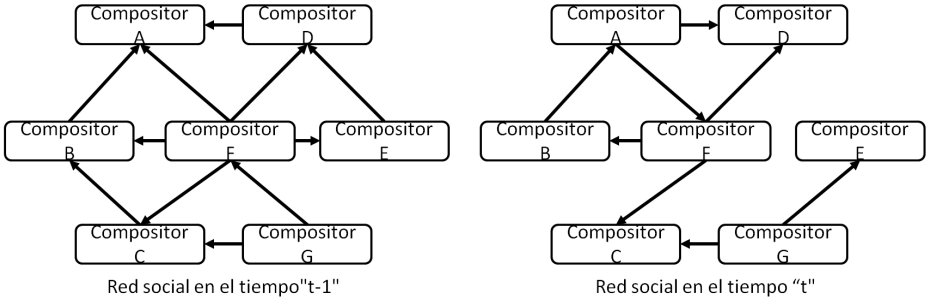

Figura 4.5: Cambio en la red social fuente [156, 157].

Posterior a la actualización de los vínculos se ejecuta la siguiente subfase.

[8] Un bucle o lazo en una gráfica es un arco cuyo nodo destino es el mismo que el nodo origen

Algoritmo 37: Actualización de arcos en la red social

Input: v, Nc, $fcla$, sociedad artificial previa

Output: Actualización de los vínculos en la red social

```
% En esta rutina el valor de v indica arreglo. % if v = 1 then
    for i = 1 : Nc do
        for k = 1 : Nc do
            if i ≠ k then
                if rand < 0.5 then
                    Creación de un vínculo entre el compositor i y el compositor k .
                end
            end
        end
    end
else
    for i = 1 : Nc do
        if rand < fcla then
            Elegir aleatoriamente un compositor k, tal que i ≠ k.
            Cambiar la relación entre ambos compositores.
        end
    end
end
Verificar que cada uno de los agentes tiene por lo menos un vínculo .
```

b)**Intercambio de información**.

En esta subfase cada uno de los compositores adquiere información de su entorno, con base en la política de interacción (regla de análisis y selección de información).

En este trabajo la norma de análisis es la siguiente: " si existe un arco con origen en el nodo k y destino en el nodo i, el $i-$ésimo compositor compara la melodía con peor grado de satisfacción en su partitura $x_{i-worst}$ contra la melodía con peor grado de satisfacción en la partitura del $k-$ésimo compositor. Si $x_{k-worst}$ es mejor que $x_{i-worst}$ entonces el agente i decide que la información del agente k le es útil ". La norma de selección es la siguiente: "si el compositor i considera conveniente la información del compositor k, entonces, $i-$ ésimo compositor toma aleatoriamente una melodía de la $P_{\star,\star,k}$ asociada al $k-$ésimo compositor.

La política anterior se implementa por la metaheurística MMC a través de la rutina mostrada en el Algoritmo 38.

Algoritmo 38: Intercambio de información entre agentes

Input: $P_{\star,\star,i}$, Nc y función de satisfacción del $i-$ésimo compositor

Output: Conjunto de ideas adquiridas de cada uno de los compositores($SC_{\star,\star,i}$)

```
for i = 1 : Nc do
    ISC_{⋆,⋆,i} = ∅ x_{i−worst} ← melodía con el peor grado de satisfacción en P_{⋆,⋆,i}.
    for k = 1 : Nc ∧ k ≠ i do
        x_{k−worst} ← melodía con el peor grado de satisfacción en P_{⋆,⋆,k}.
        if Existe un arco en la red tal que su origen es el nodo k y su destino el nodo k then
            if f(x_{i−worst}) es menos satisfactoria que f(x_{k−worst}) then
                El compositor i toma aleatoriamente una melodía deP_{⋆,⋆,k} Se añade la melodía
                seleccionada a ISC_{⋆,⋆,i}.
            end
        end
    end
end
```

4.3.1.3. Generación y evaluación de una nueva melodía

En esta fase cada compositor creará una nueva melodía basado en su conocimiento previo y considerando los posibles instantes de ingenio. Esta fase se divide en las subfases: a) construcción del conocimiento previo y b) la creación y evaluación de una melodía.

a) **Construcción del conocimiento previo**.

En esta subfase cada uno de los compositores conjuntan las ideas adquiridas del medio con su partitura (memoria); a fin de generar un conocimiento $KM_{\star,\star,i} = P_{\star,\star,i} \bigcup ISC_{\star,\star,i}$, además, en esta subfase el $i-$ésimo compositor asigna la aptitud ($fitness(KM_{j',\star,i})$) a cada elemento en su $KM_{\star,\star,i}$. Para lo cual el procedimiento MMC utiliza la rutina mostrada en el Algoritmo 40.

Algoritmo 39: Construcción y evaluación del conocimiento previo

Input: $P_{\star,\star,i}$, $ISC_{\star,\star,i}$, Nc y $f(x)$

Output: Conocimiento del $i-$ésimo compositor y ponderación de cada uno de los elementos en el conocimiento ($fitness(KM_{j',\star,i})$

```
for i = 1 : Nc do
    KM_{⋆,⋆,i} = P_{⋆,⋆,i} ⋃ ISC_{⋆,⋆,i}.
    r ← número de melodías en el conocimiento del i−ésimo compositor.
    for j' = 1 : r do
        f(KM_{j',⋆,i}) ← Grado de satisfacción alcanzado por la j'−ésima melodía del i−ésimo
        compositor.
    end
    a_i = Σ_{j'=i}^{r} f(KM_{j',⋆,i}).
    for j' = 1 : r do
        fitness(KM_{j',⋆,i}) = (a_i − f(KM_{j',⋆,i})) / (a_i * (Nc − 1)).
    end
end
```

b) **Creación de una nueva melodía**

En esta subfase, cada compositor creará una nueva melodía, con base en su $KM_{\star,\star,i}$ y a sus ideas

innovadoras. Para ello, el procedimiento MMC utiliza la rutina mostrada en el Algoritmo 40.

Algoritmo 40: Creación de una nueva melodía

Input: $KM_{\star,\star,i}$, ifg, n, Nc

Output: Una nueva melodía ($x_{\star,new}$)

```
for i = 1 : Nc do
    if rand < (1 − ifg) then
        for l = 1 : n do
            if rand < (1 − cfg) then
                Se genera el elemento x_{l,new} de la nueva melodía a partir de KM_{⋆,l,i}.
            else
                Se toma aleatoriamente un elemento de [x_l^U, x_l^L] y se asigna a x_{l,new}
            end
        end
    else
        Aleatoriamente se genera toda la melodía x_{⋆,new}
    end
    Se determina el grado de satisfacción obtenido por la melodía x_{⋆,new} (f(x_{⋆,new})).
end
```

4.3.1.4. Actualizar la obra de arte

En esta fase, el $i-$ésimo compositor decide reemplazar o no una melodía en su partitura por la nueva melodía. En este trabajo la política de reemplazo es la siguiente: "Si el grado de satisfacción alcanzado por $x_{\star,new}$ es mayor que el alcanzado por $x_{i-worst}$ (melodía con peor grado de satisfacción en $P_{\star,\star,i}$), entonces el $i-$ésimo compositor remplaza a $x_{i-worst}$ por $x_{\star,new}$ en su partitura". En el Algoritmo 41, se muestra la rutina empleada por el procediemieto MMC.

Algoritmo 41: Actualizar obra del $i-$ésimo compositor

Input: $P_{\star,\star,i}$, $f(x)$ y Nc

Output: matriz $P_{\star,\star,i}$ actualizada

```
for i = 1 : Nc do
    f(x_{i-worst}) ← grado de satisfacción alcanzado con la peor melodía en P_{⋆,⋆,i}
    f(x_{⋆,new}) ← grado de satisfacción alcanzado por x_{⋆,new}.
    if f(x_{i-worst}) es peor que f(x_{⋆,new}) then
        Reemplazar x_{i-worst} con x_{⋆,new} en P_{⋆,⋆,i} .
    end
end
```

4.3.1.5. Construcción de un conjunto de soluciones

En esta fase se contruye el conjunto de soluciones $(Si, \star)$ a partir de la melodía con mayor grado de satifacción de cada uno de los compositores. En el Algoritmo 42 se muestra la rutina utilizada por el procedimiento MMC para dicho propósito.

Algoritmo 42: Construcción del conjunto de soluciones

Input: $P_{\star,\star,i}$, $f(x)$ y Nc

Output: Conjunto de soluciones $(S\star, \star)$

```
for i = 1 : Nc do
    Si, ⋆ ← melodía con mayor grado de satisfaccíon en P_{⋆,⋆,i}
end
```

4.3.2. Características del método de composición musical

En la Tabla 4.3 se muestra la analogía entre el proceso de composición musical y el procedimiento MMC.

Tabla 4.3: Similitudes entre proceso de composición musical y el procedimiento MMC

Proceso de composición musical	Procedimiento MMC
Es un sistema creativo	Es un sistema multiagente.
Requiere un número finito de pasos	El criterio de paro es un máximo número de arreglos a realizar.
Se utilizan las experiencias pasadas para la toma de decisiones	Se usan principios de reactividad y auto adaptación para sintonizar parámetros de la toma de decisiones a partir de los resultados obtenidos.
La creatividad puede deberse a destellos de genialidad y a un análisis recursivo de resultados.	Se utiliza un factor aleatorio decreciente para imitar los destellos de genialidad; el cual, es similar al parámetro de mutación en algoritmos genéticos. Por otra parte se utiliza un proceso recursivo para mejora de soluciones; que es similar al análisis recursivo para la mejora de resultados.
Se genera aprendizaje en el compositor	Se genera una matriz de partitura en la que se guardan los resultados que son utilizados para generar nuevas soluciones.
Se utiliza conjunto de reglas, pasos e instrucciones para mejor la obra actual.	Se utiliza la alusión, recombinación e influencia para modificar el conjunto de soluciones actuales.
Una pieza musical por principio de armonía requiere al menos tres fuentes de sonido que se ejecuten simultáneamente	Es una estrategia poblacional.

Tabla 4.4: Comparación del MMC contra otras metaheurísticas. Elaborado con base en [61]

Metaheurística	Intensificación	Diversificación
Recocido simulado	Búsqueda local básica	Movimientos que empeoran solución
Búsqueda Tabú	Búsqueda local básica	Lista tabú (memoria corto plazo)
Algoritmos Genéticos	Selección	Operadores de modificación (cruza, mutación)
ACO	Reglas de actualización de la feromona	Regla de construcción probabilística
Vecindades Variables	Búsqueda local básica	Criterio de aceptación y perturbación
Método de composicióon musical	Reglas de reemplazo de soluciones en la partitura. Metapatrón en la partitura y emergencia de un metapatrón en las melodías de la sociedad (emergencia de elementos en el contexto cultural)	Factores de genialidad, cambios en la red social, combinación de melodías y adquisición de ideas de su medio ambiente.

4.3.2.1. Comparación del MMC con otras metaheurísticas

En el procedimiento MMC la fase de intensificación es resultado de los política de actualización de la partitura lo que conlleva a la emergencia de un metapatrón de los elementos que componen a P y de elementos en el contexto cultural de la red social. La fase de diversificación es resultado del uso de factores de genialidad, cambios en la red social y de la combinación de melodía y adqusición de ideas de su entorno para su conocimiento. En la tabla 4.4 se comparan los elementos de intensificación y diversificación de varias metaheurísticas.

4.3.2.2. Representación del MMC por medio una red computacional

Una red computacional CN se define como tupla $CN = (N, k, a, f)$, donde: N es el conjunto de nodos; k es el conjunto de vínculos; a es un algoritmo y f es la función a computar por medio del algoritmo a [82]. El modelo de CN es de naturaleza general, lo cual permite utilizarlo para caracterizar, estudiar y comparar sistemas basados en inteligencia colectiva.

El MMC puede ser representado por una instancia de una CN, la cual es estocástica; los nodos son los compositores y las aristas son los vínculos entre los agentes; $f_{i,j}$ es la función utilizada para determinar el grado de satisfación obtenido por la $j-$ésima melodía del $i-$ésimo compositor.

Un nodo (compositor) posee una matriz de conocimiento KM (es la unión de las melodías en su partitura P y las adquiridas de otros compositores ISC), una función de satisfacción (f); una función de valoración de conocimiento $(f^{\text{valoración}}(KM, f))$; una política para la generación de una nueva melodía $(x_{new} = f^{\text{generación}}(KM_i, f, f^{\text{valoración}}))$ y una política de actualización del conocimiento $(P_{actualizada} = f^{\text{actualización}}(P, x_{new}, f))$. A las aristas en la red se asocia una política para el intercambio de interacción e intercambio de información entre los compositores $(f^{\text{interacción}})$ y una política para el apagado y encendido de los vínculos con otros compositores $(f^{\text{vínculos}})$. Lo anterior, se representa en la Figura 4.6

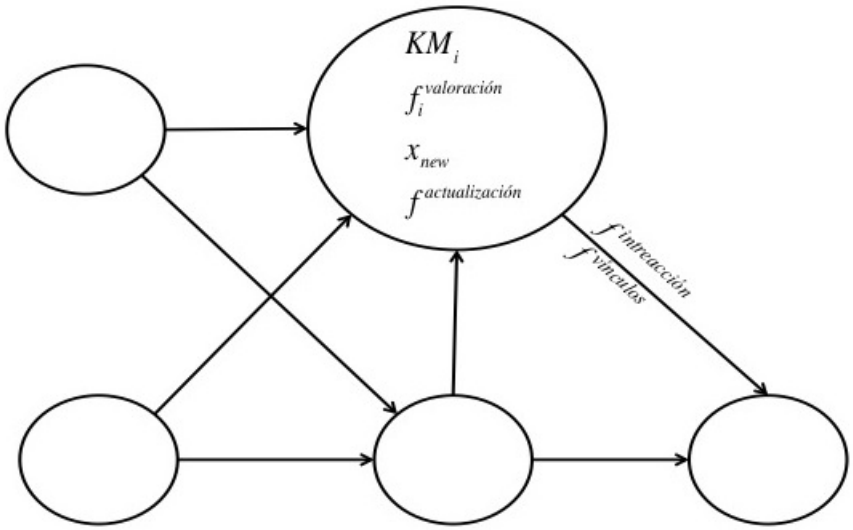

Figura 4.6: Esquema de la instanciación de la CN para el MMC

El MMC posee tres escalas de tiempo, las cuales son: a) “rápida” en la que el $i-$ésimo compositor generan y evalúan una nueva melodía, b) “media” en la cual el $i-$ésimo compositor compara, evalúa y adquiere información de su medio ambiente c) “lenta” en la cual el $i-$ésimo compositor decide establecer o eliminar sus vínculos. Por otra parte, el MMC poseé tres escalas dinámicas,las cuales son: a) **rápida** a nivel personal; b) **media** a nivel local; y c) **lenta** a nivel cultural.

Parte III

Aplicación del método de composición musical

Capítulo 5

Aplicación del MMC en la solución de instancias PLN irrestrictas

Este capítulo tiene el propósito de mostrar la adaptación y resultados obtenidos por el MMC al solucionar un conjunto de 22 instancias referenciales del problema PLN irrestricto; así como comparar los resultados obtenidos por el procedimiento propuesto con los encontrados por otras metaheurísticas sobre el mismo conjunto de problemas.

Por lo anterior, el presente capítulo se encuentra estructurado en cuatro secciones, las cuales son: marco de referencia; adaptación de MMC; experimentación y análisis de resultados.

5.1. Marco de referencia

En la sección 2.1.3.2 se describe y caracteriza al problema de PLN irrestricto y en la ecuación 2.1.5 se muestra el modelo general de este problema. El PLN irrestricto ha sido utilizado para modelar y analizar una gran variedad de casos reales, tales como: el diseño de equipos acústicos, procesamiento de imágenes, procesos químicos, análisis de datos, biología molecular, entre otros. Por ende, este problema ha sido objeto de una amplia investigación y como consecuencia se ha generado una amplia gama de métodos y técnicas para su solución.

Algunas instancias del PLN irrestricto han sido resueltas con métodos exactos como: Newton, gradiente, gradiente conjugado, y cuasi-Newton [113, 138, 230]; estos procedimientos se analizaron en la sección 2.1.3.

Por otro lado, algunas otras instancias se han resuelto por metaheurísticas, tales como: recocido simulado [101, 252], búsqueda tabú [29, 91, 89], algoritmos genéticos [5, 68, 216], algoritmos evolutivos [135] [1], programación evolutiva [68], algoritmos miméticos[175], búsqueda armónica [178, 243], optimización por nube de partículas [181, 132, 111], entre otros. En la sección 2.2.3 se revisaron estos procedimientos.

Cabe mencionar que, en la práctica, encontrar el óptimo global de una instancia del problema de la PNL es una tarea difícil; ya que, las características de esta instancia, influirán directamente en el desempeño de los métodos de solución. En la siguiente sección, se tratan las modificaciones hechas al MMC para resolver el PNL irrestricto.

5.2. Adaptación del MMC

5.2.1. Melodía y función de satisfacción

La estructura de una "melodía" (solución) utilizada en los casos del problema PNL irrestricto se muestra en la ecuación 5.2.1 .

$$\begin{gathered} x = [x_1, x_2, \ldots, x_n] \\ \text{tal que:} \\ x_l \in \mathbb{R}\ \forall l = 1 \ldots n \\ x_l \in [x_l^U, x_l^L]\ \forall l = 1 \ldots n \end{gathered} \tag{5.2.1}$$

donde: x es un nueva melodía; n es el número de variables decisión; x_l es la $l-$ésima variable de decisión; x_l^U y x_l^L son los límites inferior y superior, respectivamente, de la $l-$ésima variable de decisión.

El mecanismo para evaluar el grado de satisfacción alcanzado por la $j-$ésima melodía del $i-$ésimo compositor; implico que: "si la función objetivo es del tipo mín $f(x)$, entonces el nivel de satisfacción mejora a medida que $f(x)$ disminuye; en contraste, si la función objetivo es máx $f(x)$, entonces la satisfacción se incrementa a medida que $f(x)$ se incrementa".

5.2.2. Modificaciones a las fases del algoritmo

Para el manejo y solución de los casos PNL irrestrictos, se realizaron las siguientes modificaciones al MMC:

[1] El autor propone un algoritmo evolutivo para problemas cuadráticos binarios

- En la fase de **inicializar el algoritmo** sólo se modificó el mecanismo para la generación de partituras iniciales, como se muestra en el Algoritmo 43.

Algoritmo 43: Generación de las partituras iniciales

Input: n, Nc, Ns, x_l^U para todo $l = 1, 2, \ldots, n$ y x_l^L para todo $l = 1, 2, \ldots, n$

Output: $P_{\star,\star,\star}$

```
for i = 1 : Nc do
    for j = 1 : Ns do
        for l = 1 : n do
            rand ~ U[0, 1]
            P_{⋆,⋆,i} = x_l^L + (rand * (x_l^U − x_l^L))
        end
    end
end
```

- Dentro de la fase **generación y evaluación de una nueva melodía** en la creación de una nueva melodía se utilizó la rutina mostrada en el Algoritmo 44

Para le generación de un motivo se utilizaron dos rutinas; por ende, se produjeron dos variantes del MMC. Se denotó como MMC-o a la primera variante del MMC, en esta varinte se utilizo para generar una nueva melodía dos elementos en memoria; mientras, que se expresa como MMC-v a la segunda variante del MMC, en la cual se utilizo para generar una nueva melodía tres elementos en memoria.

5.3. Experimentación

5.3.1. Problemas de referencia

Se utilizaron 22 instancias de referencia, para probar y analizar el comportamiento del MMC en la solución de casos del problema PLN irrestricto, estas instancias han sido utilizados ampliamente en el desarrollo, prueba y análisis de otras metaheurísticas [216, 178, 243, 132, 111, 51]. Con base en sus características y propiedades, las instancias se dividen en los siguientes grupos: funciones unimodales, funciones multimodales

Algoritmo 44: Crear una nueva melodía

Input: $KM_{\star,\star,i}$, ifg, n, Nc, $f(x)$, $fitness(KM_{\star,\star,i})$, x_l^U para todo $l = 1, 2, \ldots, n$ y x_l^L para todo $l = 1, 2, \ldots, n$

Output: Una nueva melodía ($x_{\star,new}$)

```
for i = 1 : Nc do
    if rand < (1 − ifg) then
        for l = 1 : n do
            x_l^max ← máximo valor de x_l en KM_⋆,l,i.
            x_l^min ← mínimo valor de x_l en KM_⋆,l,i.
            KM_j,l,i ← probabilísticamente, seleccionar una melodía de KM_⋆,l,i, considerando
            fitness(KM_j,⋆,i)
            KM_j',l,i ← probabilísticamente, seleccionar una melodía de KM_⋆,l,i, considerando
            fitness(KM_j,⋆,i)
            if rand < (1 − cfg) then
                x_l,new ← Crear un nuevo motivo con base a la información seleccionada (véase
                Algoritmos 45 y 46)
            else
                if rand < 0.5 then
                    x_l,new = x_l^min + (rand ∗ (KM_j,l,i − x_l^min)).
                else
                    x_l,new = x_l^max − (rand ∗ (x_l^max − KM_j,l,i)).
                end
            end
        end
    else
        for l = 1 : n do
            x_l,new = x_l^U − (rand ∗ (x_l^U − x_l^L))
        end
    end
    Calcular el valor de la función objetivo obtenido por x_⋆,new (f(x_⋆,new)).
end
```

Algoritmo 45: Procedimiento para generar un nuevo motivo (variante 1 del MMC)

Input: $KM_{j',l,i}$ y $KM_{j,l,i}$

Output: $x_{l,new}$

1 $x_{l,new} = KM_{j,l,i} + (rand * (KM_{j',l,i} - KM_{j,l,i}))$

Algoritmo 46: Procedimiento para generar un nuevo motivo (variante 2 del MMC)

Input: $KM_{j',l,i}, KM_{j,l,i}$ y $KM_{\star,l,i}$

Output: $x_{l,new}$

1 $x_l^{rand} \leftarrow$ aleatoriamente seleccionar un elemento en la columna l de $KM_{\star,l,i}$

$x_{l,new} = x_l^{rand} + (rand * ((KM_{j',l,i} - KM_{j,l,i}) - (2 * x_l^{rand})))$

y funciones multimodales rotados. A continuación, se muestra la formulación de los casos considerados .

5.3.1.1. Funciones unimodales

Definición 70 *Una función es unimodal, si sólo tiene un óptimo (relativo o absoluto). En otras palabras, una función es unimodal sobre el intervalo $a \leq x \leq b$, si y solo sí es monótona a ambos lados del punto óptimo en el intervalo [217].*

A) Función de Schwefel 2.22:

$$\begin{gathered} \text{mín } f(x) = \sum_{l=1}^{n} x_l^2 * \prod |x_l| \\ -10 \leq x_l \leq 10 \text{ para todo } l = 1, 2, \ldots, n \\ \text{donde: } f(x^\star) = 0 \text{ con } [\ x_1 \ \ldots \ x_n\ ] = [\ 0 \ \ldots \ 0\ ] \end{gathered} \tag{5.3.1}$$

B) Función Rosenbrock:

$$\begin{gathered} \text{mín } f(x) = \sum_{l=1}^{n-1} (100(x_{l+1} - x_l^2)^2 + (x_l - 1)^2) \\ -30 \leq x_l \leq 30 \text{ para todo } l = 1, 2, \ldots, n \\ \text{donde: } f(x^\star) = 0 \text{ con } [\ x_1 \ \ldots \ x_n\ ] = [\ 1 \ \ldots \ 1\ ] \end{gathered} \tag{5.3.2}$$

C) Función Paso:

$$\begin{gathered} \text{mín } f(x) = \sum_{l=1}^{n} (\lfloor x_l + 0.5 \rfloor)^2 \\ -100 \leq x_l \leq 100 \text{ para todo } l = 1, 2, \ldots, n \\ \text{donde: } f(x^\star) = 0 \text{ con } [\ x_1 \ \ldots \ x_n\ ] = [\ 0 \ \ldots \ 0\ ] \end{gathered} \tag{5.3.3}$$

5.3.1.2. Funciones multimodales

Definición 71 *Una función es multimodal si sólo tiene más de un óptimo (relativo o absoluto). En otras palabras, una función es multimodal cuando tiene varias crestas (o valles) dentro del espacio vectorial donde se define [217].*

D) Función de Schwefel 2.26:

$$\begin{gathered} \text{mín } f(x) = 418.9829 - \sum_{l=1}^{n} x_l \sin(\sqrt{|x_l|}) \\ -500 \leq x_l \leq 500 \text{ para todo } l = 1, 2, \ldots, n \\ \text{donde: } f(x^\star) = 0 \text{ con } [\ x_1 \ \ldots \ x_n\ ] = [\ 420.9687 \ \ldots \ 420.9687\ ] \end{gathered} \tag{5.3.4}$$

E) Función de Rastrigin:

$$\begin{gathered} \text{mín } f(x) = \sum_{l=1}^{n} x_l^2 - 10\cos(2\pi * x_l) + 10 \\ -5.12 \leq x_l \leq 5.12 \text{ para todo } l = 1, 2, \ldots, n \\ \text{donde: } f(x^\star) = 0 \text{ con } [\ x_1 \ \ldots \ x_n\ ] = [\ 0 \ \ldots \ 0\ ] \end{gathered} \tag{5.3.5}$$

F) Función de Ackley:

$$\begin{gathered} \text{mín } f(x) = -20\exp^{-0.2(\sqrt{\frac{1}{n}\sum_{l=1}^{n} x_l^n})} - \exp^{\frac{1}{n}\sum_{l=1}^{n}\cos(2\pi * x_l)} + 20 + \exp^1 \\ -32 \leq x_l \leq 32 \text{ para todo } l = 1, 2, \ldots, n \\ \text{donde: } f(x^\star) = 0 \text{ con } [\ x_1 \ \ldots \ x_n\ ] = [\ 0 \ \ldots \ 0\ ] \end{gathered} \tag{5.3.6}$$

G) Función de Griewank:

$$\begin{gathered} \text{mín } f(x) = \frac{1}{4000}\sum_{l=1}^{n} x_l^2 - \prod_{l=1}^{n}\cos(\frac{x_l}{\sqrt{l}}) + 1 \\ -600 \leq x_l \leq 600 \text{ para todo } l = 1, 2, \ldots, n \\ \text{donde: } f(x^\star) = 0 \text{ con } [\ x_1 \ \ldots \ x_n\ ] = [\ 0 \ \ldots \ 0\ ] \end{gathered} \tag{5.3.7}$$

H) Función seis jorobas de camello:

$$\begin{gathered} \text{mín } f(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1 * x_2 - 4x_2^2 + 4x_2^4 \\ -5 \leq x_l \leq 5 \text{ para todo } l = 1, 2 \\ \text{donde: } f(x^\star) = -1.0316 \text{ con } [\ x_1 \ , \ x_2\ ] = [\ 0.08983 \ , \ 0.7129\ ] \end{gathered} \tag{5.3.8}$$

I) Función de Weierstrass:

$$\begin{gathered} \text{mín } f(x) = \sum_{l=1}^{n}(\sum_{k=0}^{20}(a^k\cos(2\pi b^k(x_l + 0.5)))) + n(\sum_{k=0}^{20}(a^k\cos(2\pi b^k * 0.5)))) \\ a = 0.5,\ b = 3 \\ -0.5 \leq x_l \leq 0.5 \text{ para todo } l = 1, 2, \ldots, n \\ \text{donde: } f(x^\star) = 0 \text{ con } [\ x_1 \ \ldots \ x_n\ ] = [\ 0 \ \ldots \ 0\ ] \end{gathered} \tag{5.3.9}$$

J) Función de Rastrigin no continua:

$$\begin{gathered}\text{mín } f(x) = \sum_{l=1}^{n}(y_l^2 - 10\cos(2\pi y_l) + 10) \\ y_l = \begin{cases} x_l & \text{Si} \quad |x_l| < \frac{1}{2} \\ \frac{round(2x_l)}{2} & \text{Si} \quad |x_l| \geq \frac{1}{2} \end{cases} \quad \forall l = 1, 2, \ldots, n \\ -5.12 \leq x_l \leq 5.12 \text{ para todo } l = 1, 2, \ldots, n \\ \text{donde: } f(x^\star) = 0 \;\; \text{con} \;\; [\;\; x_1 \;\; \ldots \;\; x_n \;\;] \;\; = \;\; [\;\; 0 \;\; \ldots \;\; 0 \;\;]\end{gathered} \tag{5.3.10}$$

K) Función de Schwefel:

$$\begin{gathered}\text{mín } f(x) = 418.9829(n - \sum_{l=1}^{n} x_l \sin(\sqrt{|x_l|})) \\ -500 \leq x_l \leq 500 \text{ para todo } l = 1, 2, \ldots, n \\ \text{donde: } f(x^\star) = 0 \;\; \text{con} \;\; [\;\; x_1 \;\; \ldots \;\; x_n \;\;] \;\; = \;\; [\;\; 420.9687 \;\; \ldots \;\; 420.9687 \;\;]\end{gathered} \tag{5.3.11}$$

5.3.1.3. Funciones multimodales rotadas:

L) Función hiper-elipsoidal rotada:

$$\begin{gathered}\text{mín } f(x) = \sum_{l=1}^{n}(\sum_{ll}^{l} x_{ll})^2 \\ -100 \leq x_l \leq 100 \text{ para todo } l = 1, 2, \ldots, n \\ \text{donde: } f(x^\star) = 0 \;\; \text{con} \;\; [\;\; x_1 \;\; \ldots \;\; x_n \;\;] \;\; = \;\; [\;\; 0 \;\; \ldots \;\; 0 \;\;]\end{gathered} \tag{5.3.12}$$

M) Función de esfera desplazada:

$$\begin{gathered}\text{mín } f(x) = \sum_{l=1}^{n} z_l^n + f_{bais} \\ -100 \leq x_l \leq 100 \text{ para todo } l = 1, 2, \ldots, n \\ z_l = x_l - 1 \text{ para todo } l = 1, 2, \ldots, n \\ f_{bais} = -450 \\ \text{donde: } f(x^\star) = -450 \;\; \text{con} \;\; [\;\; x_1 \;\; \ldots \;\; x_n \;\;] \;\; = \;\; [\;\; 1 \;\; \ldots \;\; 1 \;\;]\end{gathered} \tag{5.3.13}$$

N) Función de Schwefel desplazada:

$$\begin{gathered}\text{mín } f(x) = \sum_{l=1}^{n}(\sum_{ll}^{l} z_{ll})^2 + f_{bais} \\ -100 \leq x_l \leq 100 \text{ para todo } l = 1, 2, \ldots, n \\ z_l = x_l - 1 \text{ para todo } l = 1, 2, \ldots, n \\ f_{bais} = -450 \\ \text{donde: } f(x^\star) = -450 \;\; \text{con} \;\; [\;\; x_1 \;\; \ldots \;\; x_n \;\;] \;\; = \;\; [\;\; 1 \;\; \ldots \;\; 1 \;\;]\end{gathered} \tag{5.3.14}$$

O) Función de Rosenbrock desplazada:

$$\begin{gathered}\text{mín } f(x) = \sum_{l=1}^{n-1} 100(z_{l+1} - z_l^2)^2 + (z_{l-1})^2 + f_{bais} \\ -100 \leq x_l \leq 100 \text{ para todo } l = 1, 2, \ldots, n \\ z_l = x_l - 1 \text{ para todo } l = 1, 2, \ldots, n \\ f_{bais} = 390 \\ \text{donde: } f(x^\star) = 390 \text{ con } [\ x_1 \ \ldots \ x_n\ ] = [\ 1 \ \ldots \ 1\ ]\end{gathered} \tag{5.3.15}$$

P) Función Rastrigin desplazada:

$$\begin{gathered}\text{mín } f(x) = \sum_{l=1}^{n}((z_l^2 - 10\cos(2\pi * z_l) + 10) + f_{bais} \\ -5 \leq x_l \leq 5 \text{ para todo } l = 1, 2, \ldots, n \\ z_l = x_l - 1 \text{ para todo } l = 1, 2, \ldots, n \\ f_{bais} = -330 \\ \text{donde: } f(x^\star) = -330 \text{ con } [\ x_1 \ \ldots \ x_n\ ] = [\ 1 \ \ldots \ 1\ ]\end{gathered} \tag{5.3.16}$$

Q) Función de Rastrigin desplazada y rotada:

$$\begin{gathered}\text{mín } f(x) = \sum_{l=1}^{n}((z_l^2 - 10\cos(2\pi * z_l) + 10) + f_{bais} \\ -100 \leq x_l \leq 100 \text{ para todo } l = 1, 2, \ldots, n \\ z_l = (x_l - 1)M \text{ para todo } l = 1, 2, \ldots, n \\ M \text{ es una matriz de transformación lineal} \\ f_{bais} = -330 \\ \text{donde: } f(x^\star) = -330 \text{ con } [\ x_1 \ \ldots \ x_n\ ] = [\ 1 \ \ldots \ 1\ ]\end{gathered} \tag{5.3.17}$$

R) Función de Ackley rotada:

$$\begin{gathered}\text{mín } f(x) = -20\exp^{-0.2(\sqrt{\frac{1}{n}\sum_{l=1}^{n} y_l^2})} - \exp^{\frac{1}{n}\sum_{l=1}^{n}\cos(2\pi * y_l)} + 20 + \exp^1 \\ -32.768 \leq x_l \leq 32.768 \text{ para todo } l = 1, 2, \ldots, n \\ y_l = x_l \times M \text{ para todo } l = 1, 2, \ldots, n \\ \text{M es una matriz de transformación lineal} \\ \text{donde: } f(x^\star) = 0 \text{ con } [\ x_1 \ \ldots \ x_n\ ] = [\ 0 \ \ldots \ 0\ ]\end{gathered} \tag{5.3.18}$$

S) Función de Griewank rotada:

$$\begin{gathered}\text{mín } f(x) = \frac{1}{4000}\sum_{l=1}^{n} z_l^2 - \prod_{l=1}^{n}\cos(\frac{z_l}{\sqrt{l}}) + 1 + f_{bais} \\ -100 \leq x_l \leq 100 \text{ para todo } l = 1, 2, \ldots, n \\ z_l = (x_l - 1)M \text{ para todo } l = 1, 2, \ldots, n \\ M \text{ es una matriz de transformación lineal} \\ f_{bais} = -180 \\ \text{donde: } f(x^\star) = -180 \text{ con } [\quad x_1 \quad \ldots \quad x_n \quad] \quad = \quad [\quad 1 \quad \ldots \quad 1 \quad]\end{gathered} \tag{5.3.19}$$

T) Función de Weierstrass rotada:

$$\begin{gathered}\text{mín } f(x) = \sum_{l=1}^{n}(\sum_{k=0}^{20}(a^k \cos(2\pi b^k(y_l + 0.5)))) + n(\sum_{k=0}^{20}(a^k \cos(2\pi b^k * 0.5)))) \\ \text{where: } a = 0.5,\ b = 3,\ \ -5 \leq x_l \leq 5 \text{ para todo } l = 1, 2, \ldots, n \\ y_l = x_l \times M \text{ para todo } l = 1, 2, \ldots, n \\ \text{M es una matriz de transformación lineal} \\ \text{donde: } f(x^\star) = 0 \text{ con } [\quad x_1 \quad \ldots \quad x_n \quad] \quad = \quad [\quad 0 \quad \ldots \quad 0 \quad]\end{gathered} \tag{5.3.20}$$

U) Función de Rastrigin no continua rotada

$$\begin{gathered}\text{mín } f(x) = \sum_{l=1}^{n}(z_l^2 - 10\cos(2\pi z_l) + 10) \\ -5.12 \leq x_l \leq 5.12 \text{ para todo } l = 1, 2, \ldots, n \\ y_l = x_l \times M \text{ para todo } l = 1, 2, \ldots, n \\ \text{M es una matriz de transformación lineal} \\ z_l = \begin{cases} y_l & \text{Si} \quad |y_l| < \frac{1}{2} \\ \frac{redondear(2y_l)}{2} & \text{Si} \quad |y_l| \geq \frac{1}{2} \end{cases} \quad \text{para todo} l = 1, 2, \ldots, n \\ \text{donde: } f(x^\star) = 0 \text{ con } [\quad x_1 \quad \ldots \quad x_n \quad] \quad = \quad [\quad 0 \quad \ldots \quad 0 \quad]\end{gathered} \tag{5.3.21}$$

V) Función de Schwefel rotada

$$\begin{gathered}\text{mín } f(x) = 418.9829(n - \sum_{l=1}^{n} z_l \\ -500 \leq x_l \leq 500 \text{ para todo } l = 1, 2, \ldots, n \\ y_l = y_l' + 420.96 \text{ para todo } l = 1, 2, \ldots, n \\ y_l' = (x_l - 420.96) \times M \text{ para todo } l = 1, 2, \ldots, n \\ \text{M es una matriz de transformación lineal} \\ z_l = \begin{cases} y_l \sin(\sqrt{|y_l|}) & \text{Si} \quad |y_l| \leq 500 \\ 0.001(|y_l| - 500)^2 & \text{Si} \quad |y_l| > 500 \end{cases} \quad \text{for} l = 1, 2, \ldots, n \\ \text{donde: } f(x^\star) = 0 \text{ con } [\quad x_1 \quad \ldots \quad x_n \quad] \quad = \quad [\quad 420.9687 \quad \ldots \quad 420.9687 \quad]\end{gathered} \tag{5.3.22}$$

5.3.2. Diseño de experimentos

Con el objetivo de probar, analizar, evaluar la actuación del MMC (MMC-o y MMC-v) al solucionar instancias del PLN irrestricto se realizaron cuatro experimentos; en cada uno de ellos, se emplearon algunas de la instancias referenciales. Los resultados obtenidos con MMC se compararon contra las soluciones encontradas por otras metaheurístcas ejecutadas en condiciones similares (número de evaluaciones de la funcion objetivo y conjunto de instancias analizadas). A continuación, se describen los experimentos realizados, cabe mencionar que las características de estos es para comparar el comportamiento del MMC con otras metaheurísticas en las mismas circunstancias.

- **Experimento 1.** Este experimento consistió en probar el MMC-o sobre instancias 30-dimensionales de las funciones B), C), D), E), F), G), L), M), N), O), P), Q) y S). Una prueba implicó ejecutar el algoritmo en cada instancia, de tal forma que, $Ne = 50000$; para cada prueba se hicieron treinta repeticiones independientes registrando la mejor solución encontrada por MMC y su tiempo de ejecución.

  Los resultados obtenidos por el MMC se compararon con los datos obtenidos con: búsqueda armónica (HS) [80]; búsqueda armónica mejorada (IHS) [141]; búsqueda armónica global (GHS) [174] y búsqueda armónica auto-adaptativa (SHS) [178]. La información de estos algoritmos sobre el conjunto de instancias de prueba fue tomada de [178].

- **Experimento 2.** Este experimento consistió en probar el MMC-o y el MMC-v sobre instancias 10-dimesionales de las funciones E), F), G), I), J), K), Q), R), S), T), U) y V). Una prueba implico ejecutar el algoritmo en cada instancia, de tal forma que $Ne = 30000$; para cada prueba se hicieron treinta repeticiones independientes registrando la mejor solución encontrada por MMC y su tiempo de ejecución.

  Los resultados obtenidos por el MMC se compararon con los datos obtenidos con: PSO con inercia de peso (PSO-w) [223], PSO con factor de constricción (PSO-cf) [32], versión local del PSO-w (PSO-cf-local),versión local del PSO-cf (PSO-w-local) [118], optimización unificada del PSO (UPSO) [181], optimización totalmente informada del PSO (FIPS) [146], PSO basado en distancias de aptitud (FDR-PSO) ,[189] optimización cooperativa del PSO (CPSO-H) [238] y optimización con aprendizaje integral del PSO (CLPSO) [132]. La información de estos algoritmos sobre el conjunto de instancias de prueba fue tomada de [132, 178].

- **Experimento 3.** Este experimento consistió en probar el MMC-o sobre instancias 30-dimesionales de

las funciones A), C), L), D), E), F), G), M), N), O), P) y Q) y la instancia 2-dimensional de problema H). Una prueba implicó ejecutar el algoritmo en cada instancia, de tal forma que, $Ne = 50000$; para cada prueba se hicieron treinta repeticiones independientes registrando la mejor solución encontrada por MMC y su tiempo de ejecución.

Los resultados obtenidos por el MMC se compararon con los datos obtenidos con: *HS*, *IHS*, *GHS* y *SHS*. La información de estos algoritmos sobre el conjunto de instancias de prueba fue tomada de [178].

- **Experimento 4.** Este experimento consistió en probar el MMC-o y el MMC-v sobre instancias 100-dimesionales de las funciones A), C), L), D), E), F), G), M), N), O), P) y Q). Una prueba implicó ejecutar el algoritmo en cada instancia, de tal forma que, $Ne = 30000$; para cada prueba se hicieron treinta repeticiones independientes registrando la mejor solución encontrada por MMC y su tiempo de ejecución.

  Los resultados obtenidos por el MMC se compararon con los datos obtenidos con: *HS*, *IHS*, *GHS* y *SHS*. La información de estos algoritmos sobre el conjunto de instancias de prueba fue tomada de [178].

Con el objeto de facilitar la comparación entre los resultados obtenidos por las metaheurísticas, estos datos se normalizaron a través de la siguiente ecuación 5.3.23.

$$f(x^{normalized-\alpha}) = \frac{f(x^{method-\alpha}) - f(x^{\star})}{f(x^{worst\ \text{en}\ \beta}) - f(x^{\star})} \tag{5.3.23}$$

donde: $f(x^{\star})$ es el valor de la función objetivo en el punto óptimo global, $f(x^{method-\alpha})$ es el valor de la función objetivo promedio obtenido por la metaheurística α,$f(x^{worst\ \text{en}\ \beta})$ es el peor valor promedio encontrado por las metaheuristicas comparadas para la instancia β, y $f(x^{normalized-\alpha})$ es el valor normalizado de la función objetivo encontrado por la metaheurística α.

El valor de $f(x^{normalized-\alpha})$ oscila entre 0 y 1. Si el valor de $f(x^{normalized-\alpha})$ es cercano 0, entonces el valor de $f(x^{method-\alpha})$ está cerca $f(x^{\star})$. En contraste, si $f(x^{normalized-\alpha})$ tiende a 1, entonces $f(x^{method-\alpha})$ es lejano a $f(x^{\star})$.

Además, se aplicó la prueba de Wilcoxon[2] sobre los resultados obtenidos por las variantes del *MMC* y los datos obtenidos por las otras metaheurísticas analizadas. Los parámetros obtenidos por esta prueba son

[2]Ésta es una prueba de carácter no paramétrico, la cual sirve como método para comparar las medianas de dos poblaciones relacionadas y determinar si existen diferencias entre ellas.

h (es un valor binario, el cual muestra el resultado de la prueba de hipótesis) y p (es un valor real que oscila entre cero y uno, el cual indica el resultado de la prueba de simetría).

La hipótesis nula usada en esta prueba establece que los vectores de datos x e y son independientes: si la prueba arroja $h = 1$, entonces se rechaza la hipótesis nula con un nivel de significancia del 5 %, en contraste si $h = 0$ entonces se rechaza la hipótesis nula con un nivel de significación del 5 %.

5.3.3. Configuración de parámetros

La configuración de los parámetros del MMC se determinó a través del método de fuerza bruta, también llamado *brute force* [19] (veáse F). Para esta actividad se seleccionaron las instancias siguientes, función de Rastrigin no continua 10 dimensional, función de Schwefel 30-dimensional y función Rastringin rotada y desplazada 100-dimesional; ya que estos problemas son algunas de las instancias más difćieles .La configuración de los parámetros utilizó 100 pruebas, además cada experimento fue calibrado de manera independiente.

La configuración de parámetros determinada para el MMC-o y el MMC-v se muestran en las Tablas 6.2 y 5.2, respectivamente.

Tabla 5.1: Configuración de parámetros

	Experimentos			
Párametro	Primero	Segundo	Tercero	Cuarto
máx_$arrangement$	12500	7500	10000	1000
ifg	0.01	0.01	0.01	0.01
cfg	0.09	0.1	0.04	0.04
$fcla$	0.045	0.1	0.1	0.1
Nc	4	4	5	5
Ns	5	10	5	5

Con base en los resulados obtenidos durante el ajuste de los parámetros del *MMC*, se puede señalar que al parecer el algoritmo propuesto es más sensible a los cambios en $fcla$ que a cambios en los factores de genialidas ifg y cfg.

Tabla 5.2: Configuración de parámetros

	Experimentos			
Párametro	Primero	Segundo	Tercero	Cuarto
máx $\_arrangement$		7500	10000	1000
ifg		0.01	0.01	0.01
cfg		0.09	0.04	0.04
$fcla$		0.1	0.1	0.1
Nc		4	5	5
Ns		10	9	9

5.4. Resultados Numéricos

Los algoritmos MMC-o y MMC-v se implementaron en Matlab 7.10.0, y se ejecutaron en una computadora DELL Inspiron 1720. Cabe mencionar que la información mostrada en esta sección fue publicada en [156, 157]

5.4.1. Experimento 1

Los resultados obtenidos del primer experimento se muestran en las tablas 5.3 a la 5.5.

Tabla 5.3: Resultados del primer experimento (media ± desviación estandar) [156]

Problema	Óptimo global	HS	HIS	GHS	SHS	MMC
B	0	350.297 ± 266.691	624.323 ± 559.847	$\underline{49.669 \pm 59.161}$	150.93 ± 131.055	245.43 ± 297.927
C	0	4.233 ± 3.030	3.333 ± 2.196	0 ± 0.000	$\underline{0 \pm 0.000}$	1.0667 ± 1.143
D	0	30.262 ± 11.960	34.531 ± 10.400	0.042 ± 0.0503	$\underline{0.004 \pm 0.006}$	3.259 ± 1.245
E	0	1.391 ± 0.824	3.499 ± 1.829	$\underline{0.0086 \pm 0.0153}$	0.0177 ± 0.0675	0.465 ± 0.371
F	0	1.130 ± 0.407	1.893 ± 0.315	$\underline{0.0209 \pm 0.0217}$	0.484 ± 0.357	0.246 ± 0.0686
G	0	1.119 ± 0.041	1.121 ± 0.041	0.102 ± 0.176	$\underline{0.0505 \pm 0.035}$	0.720 ± 0.191
L	0	4297.816 ± 1362.148	4313.65 ± 1062.106	5146.176 ± 6348.79	$\underline{11.766 \pm 7.454}$	220.560 ± 154.72
M	-450	-443.553 ± 2.777	-438.815 ± 3.704	1353.211 ± 361.763	$\underline{-450 \pm 0.00}$	-449.142 ± 0.355
N	-450	3888.18 ± 1115.26	3316.60 ± 1519.41	18440.50 ± 4537.944	$\underline{-431.096 \pm 17.25}$	-224.97 ± 160.814
O	390	3790.70 ± 3271.57	5752.12 ± 3762.54	35046942 ± 22136432	2511.7 ± 3966.5	$\underline{1756.72 \pm 1383.8}$
P	-330	-329.129 ± 0.809	-328.057 ± 0.667	-263.272 ± 9.356	$\underline{-329.861 \pm 0.350}$	-329.319 ± 0.526
Q	-330	-274.687 ± 12.863	-270.695 ± 16.223	-192.096 ± 18.646	-232.14 ± 30.033	$\underline{-283.677 \pm 10.416}$
S	-180	547.869 ± 0.501	494.756 ± 6.717	546.62 ± 8686070.61	-47.189 ± 13.313	$\underline{-178.509 \pm 0.160}$

Tabla 5.4: Comparación entre las metaheurísticas

		B	C	D	E	F	G	L	M	N	O	P	Q	S
Función normailzada	1	2,3	1	2	2	2	1,2	3	3	3	3	3	3	1,3
	0.9			1										2
	0.8		2					1,2						
	0.7												4	
	0.6	1				1	5							
	0.5													
	0.4	5			1								1,2	
	0.3		5			4							5	
	0.2	4								1,2				4
	0.1			5	5	5	3							
	0		3,4	3,4	3,4	3	4	4,5	1,2,4,5	4,5	1,2,4,5	1,2,4,5		5

donde:

1 es HS 3 es GHS 5 es $MMC - o$

2 es HIS 4 es SHS

En las Tablas 5.3 y 5.4 se muestra que el algoritmo MMC genera buenos resultados en 7 de las 13 funciones referenciales. A mayor detalle al comparar los resultados obtenidos el MMC contra los obtenidos por HS y IHS; se observa que el MMC produce mejores resultados en todas las funciones de prueba. Al comparar el MMC contra GHS; se observa que el MMC genera para 7 de las 13 pruebas mejores resultados que GHS. Y por último, al comparar SHS contra MMC se observa que la propuesta genera en cuatro pruebas mejores resultados que las otras metaheurísticas.

Tabla 5.5: Resultados de la prueba de Wilcoxon primer experimento..

Parámetro	HS	IHS	GHS	SHS
p	0.1119	0.1008	0.3051	0.9183
h	0	0	0	0

En la tabla 5.5, se muestra que el algoritmo MMC-o tiene un comportamiento estadísticamente diferente sobre el conjunto de estas instancias que las metaheurísticas HS, IHS, GHS y SHS. En la 5.6 se muestra el tiempo de ejecución promedio en cada una de las instancias por el algoritmo MMC; cabe mencionar que no se incluyen los tiempos de las otras metaheurísticas por no estar disponibles en la literatura.

Tabla 5.6: Tiempo de ejecución primer experimento (media ± desviación estandar)

Instancia	Tiempo en segundos
B	42.37±5.075
C	39.439±2.366
D	50.326± 1.874
E	44.159± 4.196
F	49.344± 1.84
G	45.598± 4.643
L	46.153± 4.96
M	40.643± 3.744
N	61.071±3.032
O	43.986± 3.067
P	41.458± 2.383
Q	48.442± 4.982
S	44.898± 2.982

5.4.2. Experimento 2

Los resultados obtenidos por el segundo experimento se muestran en las tablas 5.7 a la 5.10.

Tabla 5.7: Resultados del segundo experimento (media ± desviación estandar)

Función	PSO-W	PSO-cf	PSO-W-local	PSO-cf-local
E	5.82E+00 ± 1.72E+00	1.25E-01 ± 2.27E+00	3.88E+00 ± 1.52E+00	9.05E+00 ± 1.87E+00
F	1.58E-14 ± 1.26E-07	9.18E-01 ± 1.00E+00	6.04E-15 ± 4.09E-08	5.78E-02 ± 5.08E-01
G	9.69E-02 ± 2.24E-01	1.19E-01 ± 2.67E-01	7.80E-02 ± 1.95E-01	2.80E-02 ± 2.52E-01
I	2.28E-03 ± 8.39E-02	6.69E-01 ± 8.18E-01	1.41E-06 ± 1.19E-03	7.85E-02 ± 2.26E-01
J	4.05E+00 ± 1.61E+00	1.20E+01 ± 2.23E+00	4.77E+00 ± 1.69E+00	5.95E+00 ± 8.78E+02
K	3.20E+02 ± 1.36E+01	9.87E+02 ± 1.66E+01	3.26E+02 ± 1.15E+01	8.78E+02 ± 1.71E+01
R	2.80E-01 ± 7.66E-01	1.19E+00 ± 1.06E+00	6.39E-15 ± 5.64E-08	2.56E-01 ± 7.30E-01
S	1.64E-01 ± 3.07E-01	1.38E-01 ± 3.27E-01	8.04E-02 ± 2.11E-01	7.90E-02 ± 2.36E-01
T	6.66E-01 ± 8.44E-01	2.17E+00 ± 1.14E+00	2.14E-01 ± 6.04E-01	1.20E+00 ± 1.10E+00
Q	9.90E+00 ± 1.94E+00	1.44E+01 ± 2.46E+00	9.25E+00 ± 1.66E+00	1.34E+01 ± 2.61E+00
U	1.02E+01 ± 1.89E+00	1.53E+01 ± 2.53E+00	1.09E+01 ± 2.02E+00	1.07E+01 ± 1.68E+00
V	5.69E+02 ± 1.47E+01	1.19E+03 ± 2.06E+01	4.72E+02 ± 1.75E+01	9.09E+02 ± 1.80E+01

Función	UPSO	FDR	FIPS	CPSO-H
E	1.17E+01 ± 2.47E+00	7.51E+00 ± 1.75E+00	2.12E+00 ± 1.15E+00	0.00E+00 ± 0.00E+00
F	1.33E+00 ± 1.22E+00	3.18E-14 ± 2.53E-07	3.75E-15 ± 1.46E-07	1.49E-14 ± 8.35E-08
G	1.04E-01 ± 2.66E-01	9.24E-02 ± 2.37E-01	1.31E-01 ± 3.05E-01	4.07E-02 ± 1.67E-01
I	1.14E+00 ± 1.07E+00	3.01E-03 ± 8.49E-02	2.02E-03 ± 8.00E-02	1.07E-15 ± 4.09E-08
J	5.85E+00 ± 1.77E+00	3.35E+00 ± 1.42E+00	4.35E+00 ± 1.67E+00	2.00E+00 ± 6.40E-01
K	1.08E+03 ± 1.64E+01	8.51E+02 ± 1.66E+01	7.10E+01 ± 1.22E+01	2.13E+02 ± 1.19E+01
R	1.00E+00 ± 9.63E-01	1.40E-01 ± 6.62E-01	2.25E-15 ± 3.92E-08	1.36E+00 ± 9.41E-01
S	7.76E-02 ± 2.53E-01	1.44E-01 ± 2.80E-01	1.70E-01 ± 3.55E-01	1.20E-01 ± 2.84E-01
T	2.61E+00 ± 9.74E-01	3.34E-01 ± 6.24E-01	5.93E-14 ± 4.31E-07	4.35E+00 ± 1.16E+00
Q	1.52E+01 ± 2.29E+00	9.25E+00 ± 1.58E+00	1.20E+01 ± 2.49E+00	2.67E+01 ± 3.26E+00
U	1.47E+01 ± 2.56E+00	1.07E+01 ± 1.96E+00	8.84E+00 ± 1.81E+00	1.90E+01 ± 3.01E+00
V	1.27E+03 ± 1.51E+01	1.07E+03 ± 1.49E+01	2.89E+02 ± 1.41E+01	9.67E+02 ± 1.92E+01

Función	CLPSO	MMC-o	MMC-v	
E	0.00E+00 ± 0.00E+00	1.10E-02 ± 1.20E-02	7.61E-09 ± 4.14E-08	
F	4.32E-14 ± 1.60E-07	1.13E-01 ± 7.91E-02	4.45E-04 ± 2.23E-03	
G	4.56E-03 ± 2.19E-01	9.47E-02 ± 4.93E-02	1.50E-06 ± 8.08E-06	
I	0.00E+00 ± 0.00E+00	1.34E-01 ± 4.28E-02	1.09E-08 ± 5.92E-08	
J	0.00E+00 ± 0.00E+00	1.28E-02 ± 1.79E-02	6.33E-09 ± 3.46E-08	
K	0.00E+00 ± 0.00E+00	8.98E-02 ± 1.16E-01	1.04E+02 ± 9.75E+01	
R	3.56E-05 ± 1.25E-02	2.00E+01 ± 1.41E-04	5.20E-11 ± 2.85E-10	
S	4.50E-02 ± 1.75E-01	1.72E-01 ± 4.09E-01	2.40E-03 ± 5.80E-03	
T	3.72E-10 ± 2.00E-05	4.04E-01 ± 6.31E-02	1.45E-03 ± 3.34E-03	
Q	5.97E+00 ± 1.70E+00	5.57E+00 ±2.76E+00	3.32E-02 ± 1.82E-01	
U	5.44E+00 ± 1.18E+00	4.19E+00 ± 2.11E+00	3.33E-02 pm 1.83E-01	
V	1.14E+02 ± 1.13E+01	2.64E-01 ± 3.73E-01	3.82E+01 ± 7.32E+01	

Tabla 5.8: Comparación entre las metaheurísticas segundo experimento

		Problema											
		E	*F*	*G*	*I*	*J*	*K*	*R*	*S*	*T*	*Q*	*U*	*V*
Función normailzada	*1.0*	5	5	7	5	2	5	10	1,7,10	8	8	8	5
	0.9			2			2						2
	0.8	4		5			4,6		2,6			2,5	6,8
	0.7		2	1,6,10					8				4
	0.6	6		3	2					5	5	3,4,6	
	0.5	1				4,5			3,4,5	2	2,4	1,7	
	0.4					3,6					1,7		1,3
	0.3	3		8			1,3		9	4	3,6	9	
	0.2	7		4		8	8			1	9,10	10	7
	0.1		10		4,10		7,11	2,5,8		6,10			9
	0.0	2,8,9,10,11	1,3,4,6,7,8,9,11	9,11	1,3,6,7,8,9,11	9,10,11	9,10	1,3,4,6,7,9,11	11	3,7,9,11	11	11	10,11

donde :

1 es PSO-W	4 es PSO-cf-local	7 es FIPS	10 es MMC-o
2 es PSO-cf	5 es UPSO	8 es CPSO-H	11 es MMC-v
3 es PSO-W-local	6 es FDR	9 es CLPSO	

Los resultados obtenidos por el segundo experimento, mostrados en las tablas 5.7 y 5.8, exhiben el buen comportamiento de MMC-o y MMC-v en especial con los problemas multimodales y multimodales rotados.

En los resultados de la prueba de Wilcoxon (ver Tabla 5.9), se observa que el MMC-o es significativamente diferente que los métodos PSO-cf, UPSO, CLPSO y MMC-v; en contraste, el MMC-v es significativamente diferente que los procedimientos PSO-W, PSO CF-,-CF PSO local, UPSO, FDR y MMC-o.

En la tabla 5.10 se muestran los tiempos de ejecución de MMC-o y MMC-v en las instancias consideradas en el segundo experimento.

Tabla 5.9: Resultados de la prueba de Wilcoxon segundo experimento.

	MMC-o		MMC-v	
	p	h	p	h
PSO-W	0.236584093	0	0.026229388	1
PSO-cf	0.035089116	1	0.002945646	1
PSO-W-local	0.839859973	0	0.140955219	0
PSO-cf-local	0.260236203	0	0.004264765	1
UPSO	0.030382822	1	0.002945646	1
FDR	0.340778614	0	0.022576063	1
FIPS	0.839859973	0	0.312321422	0
CPSO-H	0.583360467	0	0.214493808	0
CLPSO	0.034692915	1	0.36980179	0
MMC-o	1	0	0.012022825	1
MMC-v	0.012022825	1	1	0

5.4.3. Experimento 3

Los resultados obtenidos por el tercer experimento se muestran en las tablas 5.11 a la 5.14. La información de las tablas 5.11 y 5.12 evidencia que el MMC-o es 15.4 % mejor que los otros métodos metaheurísticos; mientras que, el MMC-v es 69.2 % mejor que los otros procedimientos. Ambos procedimientos presentan buenos resultados sobre instancias multimodales.

Tabla 5.10: Tiempo de ejecución segundo experimento (media ± desviación estandar).

Instancia	MMC-o	MMC-v
E	2.34E+01 ± 2.08E+00	2.12E+01 ± 1.20E+00
F	2.18E+01 ± 1.56E+00	2.43E+01 ± 1.92E+00
G	2.26E+01 ± 1.26E+00	2.26E+01 ± 5.95E-01
I	4.36E+01 ± 3.56E+00	4.32E+01 ± 1.04E+00
J	1.91E+01 ± 9.72E-02	2.39E+01 ± 2.22E+00
K	1.83E+01 ± 1.82E+01	2.15E+01 ± 2.09E+00
R	1.85E+01 ± 4.36E-01	2.11E+01 ± 1.44E+00
S	1.66E+01 ± 7.62E-01	1.93E+01 ± 1.07E+00
T	3.78E+01 ± 6.57E-01	4.02E+01 ± 2.27E+00
Q	1.91E+01 ± 6.50E-01	2.05E+01 ± 1.45E+00
U	2.30E+01 ± 1.64E+00	2.23E+01 ± 1.05E+00
V	2.30E+01 ± 1.32E+00	2.14E+01 ± 1.20E+00

Tabla 5.11: Resultados del tercer experimento (Media ± desviación estandar).

Problema	HS	IHS	GHS
A	1.72E-01 ±0.0729	1.10E+00 ±0.181	7.28E-02 ±0.114
C	4.23E+00 ±3.03	3.33E+00 ±2.2	$\underline{0.00E+00 \pm 0.00E+00}$
D	3.03E+01 ±1.20E+02	3.45E+01 ±1.04E+01	4.17E-02 ±5.04E-02
E	1.39E+00 ±8.24E-01	3.50E+00 ±1.83E+00	8.63E-03 ±1.53E-02
F	1.13E+00 ±4.07E-01	1.89E+00 ±3.15E-01	2.09E-02 ±2.17E-02
G	1.12E+00 ±4.12E-02	1.12E+00 ±4.09E-02	1.02E-01 ±1.76E-01
H	$\underline{-1.03E+00 \pm 0.00E+00}$	$\underline{-1.03E+00 \pm 0.00E+00}$	-1.03E+00 ±1.80E-05
L	4.30E+03 ±1.36E+03	4.31E+03 ±1.06E+03	5.15E+03 ±6.35E+03
M	-4.44E+02 ±2.78E+00	-4.39E+02 ±3.70E+00	1.35E+03 ±3.62E+02
N	3.89E+03 ±1.12E+03	3.32E+03 ±1.52E+03	1.84E+04 ±4.54E+03
O	3.79E+03 ±3.27E+03	5.75E+03 ±3.76E+03	3.50E+07 ±2.21E+07
P	-3.29E+02 ±8.09E-01	-3.28E+02 ±6.67E-01	-2.63E+02 ±9.36E+00
Q	-2.75E+02 ±1.29E+01	-2.71E+02 ±1.62E+01	-1.92E+02 ±1.86E+01

Problema	SHS	MMC-o	MMC-v
A	1.02E-04 ±0.000017	1.00E-01 ±0.0326	$\underline{1.42E-06 \pm 0.00000535}$
C	$\underline{0.00E+00 \pm 0.00E+00}$	5.33E-01 ±7.30E-01	7.57E+00 ±3.02E+00
D	$\underline{4.02E-03 \pm 6.24E-03}$	1.23E+00 ±7.25E-01	8.61E+02 ±2.36E+02
E	1.77E-02 ±6.75E-02	1.22E-01 ±1.67E-01	$\underline{2.13E-07 \pm 8.56E-07}$
F	4.84E-01 ±3.57E-01	1.20E-01 ±6.58E-02	$\underline{7.60E-06 \pm 4.13E-05}$
G	5.05E-02 ±3.54E-02	3.90E-01 ±1.95E-01	$\underline{8.73E-07 \pm 4.78E-06}$
H	$\underline{-1.03E+00 \pm 0.00E+00}$	-1.03E+00 ±3.26E-05	-1.03E+00 ±1.10E-03
L	1.18E+01 ±7.45E+00	1.15E+02 ±1.02E+02	$\underline{2.47E-04 \pm 1.28E-03}$
M	4.50E+02 ±0.00E+00	-4.49E+02 ±4.15E-01	$\underline{-4.50E+02 \pm 3.53E-01}$
N	-4.31E+02 ±1.73E+01	-3.28E+02 ±1.06E+02	-3.65E+02 ±1.29E+02
O	2.51E+03 ±3.97E+03	1.23E+03 ±5.30E+02	$\underline{4.43E+02 \pm 2.12E+01}$
P	-3.30E+02 ±3.50E-01	-3.30E+02 ±1.33E-01	$\underline{-3.01E+02 \pm 1.82E-01}$
Q	-2.32E+02 ±3.00E+01	$\underline{-2.95E+02 \pm 1.04E+01}$	-8.75E+01 ±3.39E+02

Tabla 5.12: Comparación entre las metaheurísticas tercer experimento

Función normalizada	Problema												
	A	C	D	E	F	G	H	L	M	N	O	P	Q
1.0	2	6	6	2	2	1,2	6	3	3	3	3	3	6
0.9													
0.8								1,2					
0.7													
0.6		1			1								3
0.5									4				
0.4		2		1								6	4
0.3					4	5							
0.2	1									1,2			1,2
0.1	3,5	5			5	3	3.5						5
0.0	4,6	3,4	1,2,3,4,5	3,4,5,6	3,6	4,6	1,2,4	4,5,6	1,2,5,6	4,5,6	1,2,4,5,6	1,2,4,5	

donde:

1 es HS 3 es GHS 3 es MMC-o

2 es IHS 4 es SHS 4 es MMC-v

La prueba Wilcoxon (ver tabla 5.13), muestra que el MMC-o y MMC-v presentan un comportamiento similar que los otros procedimientos utilizado en este experimento. En la tabla 5.14, muestra que no existe una diferencia significativa en los tiempos de ejecución para el MMC-o y MMC-v.

Tabla 5.13: Resultados de la prueba de Wilcoxon tercer experimento.

	MMC-o		MMC-v	
	p	h	p	h
HS	0.17408109	0	0.190901835	0
IHS	0.12380728	0	0.190901835	0
GHS	0.59019491	0	0.095524021	0
SHS	0.87768955	0	0.42660902	0
MMC-o	1	0	0.55529495	0
MMC-v	0.55529495	0	1	0

Tabla 5.14: Tiempo de ejecución tercer experimento (media ± desviación estandar).

Instancia	MMC-o	MMC-v
A	5.04E+01 ± 3.75E+00	4.40E+01 ± 2.85E+00
B	4.21E+01 ± 2.03E+00	4.49E+01 ± 3.24E+00
C	4.02E+01 ± 2.01E+00	4.61E+01 ± 4.10E+00
D	3.99E+01 ± 3.71E+00	4.37E+01 ± 5.10E+00
E	3.80E+01 ± 1.48E+00	3.85E+01 ± 7.34E-01
F	4.55E+01 ± 3.44E+00	4.47E+01 ± 4.90E+00
G	2.34E+01 ± 1.22E+00	2.89E+01 ± 1.49E+00
K	4.32E+01 ± 2.77E+00	5.13E+01 ± 4.99E+00
L	3.96E+01 ± 1.20E+00	4.94E+01 ± 2.69E+00
M	6.10E+01 ± 3.36E+00	7.76E+01 ± 8.77E+00
N	4.36E+01 ± 2.75E+00	4.56E+01 ± 2.48E+00
O	4.12E+01 ± 2.42E+00	4.52E+01 ± 5.98E+00
P	4.19E+01 ± 3.84E+00	4.73E+01 ± 1.97E+00

5.4.4. Experimento 4

Los resultados obtenidos por el cuarto experimento se muestran en las tablas 5.15 a la 5.18. Los datos de las tablas 5.15 y 5.16. muestran que el MMC-v es 53.8 % mejor que los otros procedimientos. Por otro lado, el MMC-o muestra un buen comportamiento en todos los casos de estudio, sin embargo, nunca fue el mejor.

La prueba Wilcoxon (véase Tabla 5.17) muestra que el MMC-o tiene un desempeños similar a los otros procedimientos; en contraste, MMC-v es significativamente diferente que HS, IHS y GHS. No hubo diferencias en los tiempos de ejecución entre MMC-o y MMC-v (ver tabla 5.18).

5.5. Análisis de resultados

Los resultados obtenidos en los cuatro experimentos muestran la capacidad del MMC (MMC-o y MMC-v) para explorar el espacio de búsqueda, a través de la interacción entre agentes, y generar buenas soluciones soluciones; en particular, para las instancias multimodales y multimodales rotadas probados.

También se puede decir, con base a la información obtenida, que los procedimientos MMC-o y MMC-v tienen comportamientos distintos; siendo la MMC-v la variante que ofrece mejores resultados. Por ende, se puede afirmar que cambios en las estrategias para generar un motivo tienen efectos significativos en la calidad de las soluciones encontradas por el MMC.

Con base en las prueba de Wilcoxon, se puede decir que el MMC es un procedimiento diferente a las metaheurísticas comparadas en estos experimentos (variantes de HS y PSO).

Tabla 5.15: Resultados del cuarto experimento (Media ± desviación estandar).

Problema	HS	IHS	GHS
A	8.29E+01 ± 6.72E+00	8.25E+01 ± 6.34E+00	1.92E+01 ± 5.09E+00
C	2.03E+04 ± 2.00E+03	2.08E+04 ± 2.18E+03	5.22E+03 ± 1.13E+03
D	7.96E+03 ± 5.72E+02	8.30E+03 ± 7.31E+02	1.27E+03 ± 3.95E+02
E	3.43E+02 ± 2.72E+01	3.43E+02 ± 2.51E+01	8.07E+01 ± 3.07E+01
F	1.39E+01 ± 2.85E-01	1.38E+01 ± 5.30E-01	8.77E+00 ± 8.80E-01
G	1.96E+02 ± 2.48E+01	2.04E+02 ± 1.92E+01	5.43E+01 ± 1.86E+01
L	2.15E+05 ± 2.83E+04	2.14E+05 ± 2.83E+04	3.22E+05 ± 3.96E+04
M	2.22E+04 ± 2.55E+03	2.30E+04 ± 2.30E+03	8.88E+04 ± 9.07E+03
N	2.72E+05 ± 3.85E+04	2.74E+05 ± 2.30E+03	4.97E+05 ± 5.19E+04
O	2.24E+09 ± 3.81E+08	2.21E+09 ± 3.59E+07	2.79E+10 ± 3.94E+09
P	3.62E+01 ± 2.56E+01	3.67E+01 ± 2.53E+01	5.09E+02 ± 4.52E+01
Q	1.89E+03 ± 1.25E+01	1.88E+03 ± 1.55E+01	1.83E+03 ± 3.35E+01

Problema	SHS	MMC-o	MMC-v
A	1.76E-02 ± 2.12E-02	5.34E+00 ± 5.91E-01	$\underline{5.95E-05 \pm 3.23E-04}$
C	$\underline{1.00E-01 \pm 3.05E-01}$	1.40E+02 ± 3.13E+01	3.56E+01 ± 8.34E+00
D	$\underline{3.57E+01 \pm 8.60E+01}$	6.13E+03 ± 5.30E+02	1.69E+04 ± 7.91E+02
E	1.24E+01 ± 2.64E+00	1.39E+02 ± 1.05E+01	$\underline{2.26E-06 \pm 1.23E-05}$
F	$\underline{0.00E+00 \pm 0.00E+00}$	2.82E+00 ± 1.58E-01	4.45E-04 ± 1.57E-03
G	2.79E-02 ± 9.21E-03	2.05E+00 ± 1.90E-01	$\underline{3.67E-05 \pm 2.01E-04}$
L	3.73E+04 ± 5.91E+03	2.06E+05 ± 5.29E+04	$\underline{1.57E+00 \pm 8.59E+00}$
M	$\underline{-4.50E+02 \pm 9.30E-06}$	-3.27E+02 ± 2.19E+01	-4.49E+02 ± 5.55E-01
N	6.33E+04 ± 1.24E+04	2.13E+05 ± 5.14E+04	$\underline{1.72E+03 \pm 2.63E+03}$
O	7.82E+02 ± 2.93E+02	1.62E+05 ± 6.62E+04	$\underline{7.25E+02 \pm 1.91E+02}$
P	$\underline{-3.17E+02 \pm 2.73E+00}$	-1.94E+02 ± 1.07E+01	-2.31E+02 ± 2.34E-02
Q	1.01E+03 ± 3.53E+01	-9.97E+01 ± 1.42E+01	$\underline{-1.79E+02 \pm 5.36E-01}$

Tabla 5.16: Comparación entre las metaheurísticas cuarto experimento

		Problema											
		A	C	D	E	F	G	L	M	N	O	P	Q
Función normailzada	*1.0*	1,2	1,2	6	1,2	1,2	1,2	3	3	3	3	3	1,2,3
	0.9												
	0.8												
	0.7							1,2					
	0.6					3		5		2			4
	0.5			1,2						1			
	0.4			5	5					5		1,2	
	0.3		3				3		1,2				
	0.2	3				5						5	
	0.1	5		3	3			4		4	1,2	6	5,6
	0.0	4,6	4,5,6	4	4,6	4,6	4,5,6	6	4,5,6	6	4,5,6	4	

donde:

1 es HS 3 es GHS 3 es MMC-o

2 es IHS 4 es SHS 4 es MMC-v

Tabla 5.17: Resultados de la prueba de Wilcoxon cuarto experimento..

	MMC-o		MMC-v	
	p	h	p	h
HS	0.06060197	0	0.002009505	1
IHS	0.06060197	0	0.002009505	1
GHS	0.099877403	0	0.003549839	1
SHS	0.470486422	0	0.43573066	0
MMC-o	1	0	0.193930852	0
MMC-v	0.193930852	0	1	0

Tabla 5.18: Tiempo de ejecución cuarto experimento (media ± desviacion estandar).

Instancia	MMC-o	MMC-v
A	8.00E+01 ± 3.37E+00	1.04E+02 ± 5.09E+00
C	8.87E+01 ± 6.84E+00	1.05E+02 ± 8.22E+00
D	8.65E+01 ± 5.52E+00	1.14E+02 ± 8.43E+00
E	8.89E+01 ± 8.84E+00	1.02E+02 ± 9.76E+00
F	8.67E+01 ± 1.04E+01	1.22E+02 ± 2.09E+01
G	1.01E+02 ± 6.78E+00	1.16E+02 ± 1.05E+01
L	1.05E+02 ± 7.18E+00	1.28E+02 ± 5.11E+00
M	7.93E+01 ± 2.92E+00	9.58E+01 ± 1.06E+01
N	2.83E+02 ± 2.44E+01	2.83E+02 ± 2.73E+01
O	9.67E+01 ± 5.56E+00	1.14E+02 ± 1.28E+01
P	9.17E+01 ± 8.78E+00	1.08E+02 ± 1.03E+01
Q	1.07E+02 ± 5.47E+00	1.18E+02 ± 9.81E+00

Capítulo 6

Aplicación del MMC en la solución de instancias PLN restrictas

Este capítulo tiene el propósito de mostrar la adaptación y aplicación del MMC para solucionar un conjunto de 11 instancias del problema PLN restricto. Además, se comparan los resultados obtenidos por este procedimiento con los encontrados por otras metaheurísticas sobre el mismo conjunto de problemas.

Por lo anterior, el presente capítulo se encuentra estructurado en cuatro secciones, las cuales son: marco de referencia; adaptación de MMC; experimentación y análisis de resultados.

6.1. Marco de referencia

En la sección 2.1.3.2 se describe y caracteriza al problema de PLN restricto y en la ecuación 2.1.6 se muestra el modelo general de este problema. Un gran número de problemas, de la vida real, se modelan como instancias del caso restricto PLN.

El PLN restricto se integra básicamente de: un conjunto de variables, una función objetivo y un conjunto de restricciones que delimitan el espacio factible para las variables de decisión [106]

Algunas instancias del PLN restricto han sido resueltas con métodos exactos, tales como: métodos primarios, métodos de penalización y barrera, métodos de Lagrange, métodos duales [113, 138, 230]; éstos procedimientos se analizaron en la sección 2.1.3. Por otro lado, algunas otras instancias del PLN restricto se han resuelto con procedimientos metaheurísticos, tales como: recocido simulado [115], búsqueda tabú [20],

estrategia evolutiva basada en asignaciones homomorfas [123], PSO [106], algoritmos culturales [38, 37], evolución cultural diferencial [126], sociedad y civilizaciones [195, 196], estrategias evolutivas [148, 149], ABC [116], HS [128], algoritmo anticultura [233], entre otros.

Ya que el MMC es un método evolutivo, en la siguiente subsección se analiza las técnicas evolutivas desarolladas para el manejo de restricciones.

6.1.1. Procedimientos para el manejo de restricciones

Algunas técnicas exactas explotan las restricciones que constituyen el caso PLN para encontrar la solución óptima. Por otro lado, en la mayoría de los metaheurísticos se realizan adecuaciones para el manejo de restricciones; lo cual ha provocado el surgimiento de una gran diversidad de métodos de manipulación de restricciones. Cada uno de los procedimientos y técnicas disponibles para el manejo de restricciones tienen ventajas y desventajas; su éxito depende en gran medida de las características de la instancia a resolver, así como de la información disponible sobre la misma.

En la literatura, hay varias clasificaciones sobre las técnicas evolutivas para manejo de restricciones. En este trabajo, se analizan las clasificaciones propuestas por: Michalewicz [152] y Coello [37].

Michalewicz [152] divide las técnicas para manejo de restricciones en cinco categorías básicas, las cuales son:

- **Métodos basados en la preservación de la factibiliad de la solución**. En estos se modifican a los individuos infactibles a fin de hacerlos factibles.
- **Métodos basados en una función de penalización**. En éstos se modifica la función de aptitud mediante la inclusión de un conjunto de elementos que castiguen a las soluciones infactibles; con lo cual, se transforma un problema con restricciones a uno sin restricciones. Algunas de las técnicas de penalización son: a)pena de muerte, b) penalizaciones estáticas, c) penalizaciones dinámicas, d) penalizaciones adaptativas, entre otros.
- **Métodos inspirados en optimización multiobjetivo**. En éstos se realiza una distinción entre las soluciones factibles e infactibles durante toda la ejecución del algoritmo.
- **Métodos basados en decodificadores**. En estas estrategias se transforma el espacio de búsqueda n-dimensional en un cubo$[-1, 1]^n$.
- **Métodos híbridos**. En estas estrategias se combinan algunas de las técnicas antes mencionadas, el objetivo de incorporar el conocimiento sobre las características del problema.

Por otra parte, Coello [37] divide a las técnicas para el manejo de restricciones en los siguientes grupos:

- **Métodos basados en función de penalización**. En estos procedimientos se utiliza alguna estrategia para castigar la infactibilidad de las soluciones.
- **Métodos basados en representaciones especiales**. En estos procedimientos se realizan cambios en la representación del problema a fin de simplificar el espacio de búsqueda.
- **Métodos que usan algoritmos de reparación**. Estos procedimientos utilizan alguna estrategia para hacer factible una solución infactible.
- **Métodos basados en la separación de los objetivos y las restricciones**. Estos procedimientos manejan a las restricciones y a los objetivos por separado.
- **Métodos híbridos**. En estas estrategias se combinan algunas de las técnicas antes mencionadas, el objetivo de incorporar el conocimiento sobre las características del problema

6.1.2. Técnicas evolutivas dearrolladas para resolver el PLN restricto

Koziel y Michalewicz en [123], desarrollaron un EA basado en homomorfismo $\mathcal{S} \rightarrow [-1,1]^n$ para la optimización. Este método es competitivo para la solución de optimización con restricciones; sin embargo, su implementación es compleja y requiere de un elevado número de evaluaciones de la aptitud.

Reynolds, Michalewicz Z. y Cavaretta desarrollaron un algoritmo cultural, el cual utiliza GENOCOP [1] para el manejo de restricciones. En este procedimeinto, el manejo de restricciones se asoció con el espacio de creencias [200].

En [31], los autores diseñaron un procedimiento que combina: un método de búsqueda débil con un esquema de representación del conocimiento. El espacio de creencias se formaba a partir del conocimiento producido por GENOCOP y programación evolutiva (PE) para resolver problemas de optimización no lineal con restricciones lineales.

La propuesta anterior fue modificada posteriormente en [112] al introducir un algoritmo cultural basado en programacón evolutiva. En esta variante, la información sobre la factibilidad de la solución se incluyó en el espacio de creencias. Otro aporte importante de este estudio, fue definir a una celda de la creencia como un esquema n dimensional del esquema regional de base. Una celda de creencia proporciona un mecanismo explícito para la adquisición, el almacenamiento y la integración de los conocimientos sobre las restricciones. Estos esquemas se utilizan para orientar la búsqueda hacia regiones prometedoras.

[1] Algoritmo genético para la optimización de problemas con restricciones; dicho método fue propuesto por Michalewicz en [153]

En [195] se presentó un EA para la optimización con restricciones, basado en la evaluación de posiciones dominantes de las soluciones y trabajando por separado, la función objetivo y las restricciones. Este algoritmo incorpora un esquema de clasificación de Pareto y una selección de la pareja para el apareamiento inteligente.

El algoritmo de la sociedad y la civilización (SCA), introducido en [196], implementa un esquema de multinivel de Pareto. El algoritmo SCA se probó sobre cuatro casos de optimización con restricciones, demostrando un buen comportamiento en comparación con otros metaheurísticos.

Coello y Landa desarrollaron el CAEP [37], el cual utiliza el conocimiento extraído durante la búsqueda con el fin de modificar el operador de mutación. Esta estrategia se basó en la técnica desarrollada por Jin y Reynolds[112]. Los mismos autores propusieron una variante del CAEP para resolver problemas de optimización multiobjetivo, la cual trata a los múltiples objetivos a través de una clasificación de Pareto y el elitismo [38].

Coello y Landa en [126] propusieron un híbrido entre evolución diferencial (DE) y algoritmos culturales, al cual denotaron como CDE. En este procedimiento las fuentes de conocimiento se modifican a través de los operadores de DE; además, las fuentes de conocimiento están contenidas en el espacio de creencias del algoritmo cultural. Este algoritmo implementa las siguientes reglas para el manejo de la selección entre dos soluciones:

- Un individuo factible es siempre mejor que uno infactible ,
- Si ambos individuos son factibles, se prefiere al individuo con el mejor valor de función objetivo,
- Si ambos individuos son infactibles, se prefiere al individuo con menor cantidad de restricciones violadas.

Finalmente, se desarrolló un algoritmo denominado de contra la cultura [233] (también llamado anti-cultural). En este algoritmo se establece un espacio triple cultural, el cual se genera por la posibilidad de que los individuos puedan desobedecer las directrices de la cultura; además, del mecanismo de doble herencia de los algoritmos culturales tradicionales. El efecto de la desobediencia de los individuos es similar a un operador de mutación. Adicionalmente, en este procedimento se utilizó un algoritmo genético en el espacio de la población.

6.2. Adaptación del método del *MMC* para la optimización con resticciones

Para el manejo de restricciones por el *MMC* se desarrolló un método híbrido, en el cual se combinaron procedimientos basados en métodos inspirados en optimización multiobjetivo con el método de decodificación propuesto por Koziel y Michalewicz [123]. El procedimiento desarrollado por Koziel se basa en la relación de homomorfismo ($\mathcal{H}$), entre $[-1,1]^n$ y $\mathcal{S}$.

6.2.1. Melodía y función de satisfacción

La melodía y función de satisfacción afectan directamente el funcionamiento del algoritmo MMC. Por ende, en esta sección se describen y caracterizan dichos elementos. Considerando, la propuesta de Koziel y Michalewicz, cada melodía ($x_{\star,\star,\star}$) se generó en un espacio contenido en el espacio$[-1,1]^n$ (veasé ecuación 6.2.1).

$$y_{i,q,\star} = [y_{i,q,1} y_{i,q,2} \dots y_{i,q,n}] \tag{6.2.1}$$

donde: $y_{i,q,\star}$ es la melodía q del $i-$ésimo compositor y $y_{i,q,l} \in [-1,1]$ es el $l-$ésimo motivo en la $q-$ésima melodía. Inicialmente, cada motivo se generó aleatoriamente considerando una distribución uniforme en el intervalo [-1,1].

En la ecuación 6.2.2 se muestra la relación de homomorfismo entre el espacio de búsqueda y el hipercubo $[-1,1]^n$.

Con base en las ecuaciones 6.2.2 y 6.2.1 se desarrollarón las ecuaciones 6.2.3 y 6.2.4, las cuales establecen la relación entre $x_{i,q,l}$ y $y_{i,q,l}$.

$$\mathcal{H}([y_{i,q,1}, y_{i,q,2}, \dots, y_{i,q,n}]) = [x_{i,q,1}, x_{i,q,2}, \dots, x_{i,q,n}] \tag{6.2.2}$$

donde: $x_{i,q,l}$ es el valor asignado a la $l-$ ésima variable de decisión asociada al motivo $y_{i,q,l}$

$$y_{i,q,l} = \frac{2 * x_{i,q,l} - (x_l^U + x_l^L)}{x_l^U - x_l^L} \quad \forall l = 1, \dots, n \tag{6.2.3}$$

$$x_{i,q,l} = y_{i,q,l} * \left(\frac{x_l^U - x_l^L}{2} \right) + \frac{x_l^U + x_l^L}{2} \quad \forall l = 1, \dots, n \tag{6.2.4}$$

donde: x_l^L y x_l^U son los límites superior e inferior de la $l-$ésima variable de decisión respectivamente [123].

Por otro lado, en la construcción de la función de satisfacción se utilizaron las reglas propuestas por Coello en [126] con las modificaciones siguientes:

- Una solución factible siempre se prefiere a una solución infactible.
- Si se comparan dos soluciones factibles se prefiere a aquella con mejor valor de función objetivo, en el caso de empate se prefiere a la más cercana al casco convexo definido por las restricciones.
- Si se comparan dos soluciones infactibles se prefiere a aquella que viole el menor número de restricciones en el caso de empate se prefiere a la más cercana al casco convexo.

6.2.2. Modificaciones del algoritmo

Se realizaron las siguientes modificaciones en el algoritmo MMC; para que se pudiese solucionar el caso PNL restrictos.

- En la fase de **inicializar el algoritmo** se modificó la estrutura de la partitura; ya que se redefinió a P como la unión de las matrices MT y MC, tal y como lo muestra la ecuación 6.2.5

$$P_{i,\star,\star} = \left(\begin{array}{c|c} MT_i & MC_i \end{array} \right) \tag{6.2.5}$$

donde: MT_i es la matriz de melodías (véase ecuación 6.2.6) y MC_i es la matriz de restricciones (véase ecuación 6.2.8)

$$MT_i = \left(\begin{array}{cccc|c} y_{i,1,1} & y_{i,1,2} & \cdots & y_{i,1,n} & \phi(x_{i,1,\star}) \\ y_{i,2,1} & y_{i,2,2} & \cdots & y_{i,2,n} & \phi(x_{i,2,\star}) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ y_{i,Ns,1} & y_{i,Ns,2} & \cdots & y_{i,Ns,n} & \phi x_{i,Ns,\star}) \end{array} \right) \tag{6.2.6}$$

donde: $\phi(tune_{i,q,\star})$ es la función de aptitud asociada con $x_{i,q,\star}$, la cual se define como:

$$\phi(tune_{i,q,\star}) = \begin{cases} f(x_{i,q,\star}) & \text{Si todas las restricciones} \\ & \text{son satisfechas por } x_{i,q,\star} \\ f(x_{i,worst}) & \text{Si al menos una} \\ & \text{restricción es violada por } x_{i,q,\star} \end{cases} \tag{6.2.7}$$

donde: f es la función objetivo, $x_{i,worst}$ es el peor valor de las soluciones factibles (en términos del valor de la función objetivo) para el compositor i.

$$MC_i = \left(\begin{array}{cccc|c|c} c_{i,1,1} & c_{i,1,2} & \dots & c_{i,1,m} & \mathcal{C}_{i,1,\star} & dif_{i,1} \\ c_{i,2,1} & c_{i,2,2} & \dots & c_{i,2,m} & \mathcal{C}_{i,2,\star} & dif_{i,2} \\ \vdots & \vdots & \dots & \vdots & \vdots & \vdots \\ c_{i,Ns,1} & c_{i,Ns,2} & \dots & c_{i,Ns,m} & \mathcal{C}_{i,Ns,\star} & dif_{i,Ns} \end{array}\right) \tag{6.2.8}$$

donde: $c_{i,q,j}$ indica si la $j-$ésima restricción es satisfecha o no por la $q-$ésima melodía (véase ecuación 6.2.9); $\mathcal{C}_{i,q,\star}$ es el total de resticciones violadas por la melodía q (véase ecuación 6.2.10); $dif_{i,q}$ se obtiene por sumar las diferencias del lado derecho e izquierdo (veasé ecuación 6.2.11) del conjunto de restricciones (véase ecuación 2.1.6), se denota con $g_j(x)$ al conjunto de resticciones del tipo desigualdad y se denota con $h_j(x)$ al conjunto de resticciones del tipo igualdad.

$$c_{i,q,j} = \begin{cases} 1 & \text{Si la } j-\text{ésima restrición es violada} \\ & \text{por la melodía } q \\ 0 & \text{Si la } j- \text{ ésima restrición es satisfecha} \\ & \text{por la melodía } q \end{cases} \tag{6.2.9}$$

$$\mathcal{C}_{i,q,\star} = \sum_{j=1}^{m} c_{i,q,j} \tag{6.2.10}$$

$$dif_{i,q} = \sum_{j=1}^{p} |h_j(x_{i,q,\star})| + \sum_{j=p+1}^{m} |g_j(x_{i,q,\star})| \tag{6.2.11}$$

- En la fase de **interacción entre agentes** se definió una estrategia auto-adaptativa para el parámetro $fcla$ a fin de emular la capacidad humana que continuamente organiza y reorganiza sus vínculos con los demás; a través del Algoritmo 47 cada compositor es capaz de actualizar sus vínculos independientemente de otros compositores. Inicialmente, se asignan a cada compositor un valor aleatorio de $fcla$; usando para ello una distribución uniforme, tal que $fcla \in [0, 1]$ y, posteriormente, este valor es recalculado antes de actualizar los vínculos entre los compositores.

- En la fase de **construcción de una nueva melodía**: después que el $i-$ésimo compositor ha adquirido la información de su medio, procede a evaluar la información en su conocimiento previo a través del Algoritmo 48.

  A continuación, se **construye una nueva melodía** a través del Algoritmo 49.

  En el Algoritmo 50, se muestra la rutina utilizada por el $i-$ésimo compositor para seleccionar los elementos base para generar una nueva melodía. En este Algoritmo, se considera la idea de solución

Algoritmo 47: Auto-adaptación del parámetro $fcla$ del $i-th$ compositor

Input: v,$fcla_i$,Nc,$P_{i,\star,\star}$, sociedad artificial previa

Output: $fcla_i$

if $v \geq 2$ **then**

  for $i = 1 : Nc$ **do**

    $x_{i,worst}^{v-1}$ es la melodía con peor valor de $\phi(x_{i,1,\star})$ generada por el $i-$ésimo compositor en el tiempo"t_{v-1}"

    $x_{i,worst}^{v-2}$ es la melodía con peor valor de $\phi(x_{i,1,\star})$ generada por el $i-$ésimo compositor en el tiempo"t_{v-2}"

    if $tune_{i,worst}^{v-2} = tune_{i,worst}^{v-1}$ **then**

      $\tau_0 = \frac{1}{\sqrt{Nc}}$

      $fcla_i = fcla_i * \exp^{N(0,\tau_0)}$

      Verificar que $fcla_i \in [0, 1]$

    end

  end

end

Algoritmo 48: Evaluación de la aptitud de las melodías en $KM_{i,\star,\star}$

Input: $KM_{i,\star,\star}$

Output: $tune_{i,new}$

$r \leftarrow$ número de melodías en $KM_{\star,\star,i}$ $a_i = \begin{cases} -1 & \text{Para problemas de maximización} \\ 1 & \text{Para problemas de minimización} \end{cases}$

for $j' = 1 : r$ **do**

  $fitness1(KM_{i,j',\star}) = \phi(tune_{i,j',\star}) + a_i * \left(\mathcal{C}_{i,j',\star} + \frac{dif_{i,j'}}{\delta_i}\right)$.

end

$fitness(KM_{i,worst})$ es la solución con peor valor de $fitness1(KM_{i,\star,\star})$

for $j' = 1 : r$ **do**

  $fitness(KM_{i,j',\star}) = |fitness1(KM_{i,j',\star}) - fitness(KM_{i,worst})| + \frac{1}{\sqrt{r}}$.

end

for $j' = 1 : r$ **do**

  $fitness(KM_{i,j',\star}) = \frac{fitness(KM_{i,j',\star})}{\sum_{j'=1}^{r} fitness(KM_{i,j',\star})}$.

end

dominada, generalmente, se dice que una solución A es dominada por otra solución B, si A satisface igual o menor número de restricciones que B.

- Para la **actualización de la partitura** se utilizó el Algoritmo 51

  En resumen el MMC fue modificado en las fases siguientes: inicializar el algoritmo; interacción entre agentes; contrucción de una nueva melodía con la finalidad de poder resolver instancias del PLN en el caso restricto.

6.3. Experimentación

6.3.1. Problemas referenciales

Se utilizó un conjunto de l2 instancias referenciales; a este conjunto de intancias, generalmente se le denomina *G-suite*; y ha sido ampliamente utilizado en el desarrollo, prueba y análisis de otras metaheurísticas [123, 152, 151]. A continuación, se muestran estas instancias.

Algoritmo 49: Generar una nueva melodía

Input: cfg, ifg,Nc,n

Output: Nueva melodía

for $i = 1:1:Nc$ **do**
  for $l = 1:n$ **do**
    $y^{max}_{i,\star,l} \leftarrow$ máximo valor de $y_{i,\star,l}$ en $KM_{i,\star,l}$.
    $y^{min}_{i,\star,l} \leftarrow$ mínimo valor de $y_{i,\star,l}$ en $KM_{i,\star,l}$.
  end
  if $rand < (1 - ifg)$ **then**
    Seleccionar las melodías base, por medio, del Algoritmo 50 ($Melodía\ base_{\star,\star}$).
    for $l = 1:1:n$ **do**
      if $rand < (1 - cfg)$ **then**
        $\overline{y_{i,\star,l}} \leftarrow$ media de los elementos $Melodía\ base_{\star,l}$.
        $\sigma_{y_{i,\star,l}} \leftarrow$ desviación estándar de los elementos $Melodía\ base_{\star,l}$.
        $y_{i,new,l} = \overline{y_{i,\star,l}} + (rand * \sigma_{y_{i,\star,l}})$.
      else
        $y_{i,new,l} = -1 + rand * (2)$.
      end
    end
  else
    if $rand < 0.5$ **then**
      for $l = 1:n$ **do**
        $y_{i,new,l} = -1 + rand * (2)$.
      end
    else
      for $l = 1:n$ **do**
        $y_{i,new,l} = y^{min}_{i,\star,l} + rand * (y^{max}_{i,\star,l} - y^{min}_{i,\star,l})$
      end
    end
  end
end

Algoritmo 50: Selección de las melodías base

Input: $KM_{i,\star,\star}$,Nc, mecanismo a través del cual se selecciona la tercera melodá base (cof, este parámetro sólo toma valores enteros)

Output: Melodías base para generar una nueva melodía

for $i = 1 : 1 : Nc$ **do**
- $KM_{i,best,\star} \leftarrow$ es la melodía con mejor valor de $fitness(KM_{i,\star,\star})$ en $KM_{i,\star,\star}$.
- $Melodía\ base_{1,\star} \leftarrow KM_{i,best,\star}$.
- Eliminar $KM_{i,best,\star}$ de $KM_{i,\star,\star}$.
- $KM'_{i,\star,\star} = \emptyset$ **if** $KM_{i,best,\star}$ *viola alguna resticciónn de la intancia a resolver* **then**
  - Eliminar todas las melodías de $KM_{i,\star,\star}$ dominadas por $KM_{i,best,\star}$.
- **end**
- Recalcular la calcular la aptitud de lo elementos de $KM_{i,\star,\star}$.
- $Melodía\ base_{2,\star} \leftarrow$ melodía seleccionada, probabilísticamente, de $KM_{i,\star,\star}$.
- **if** $cof_i = 1$ **then**
  - $Melodía\ base_{3,\star} \leftarrow$ seleccionar aleatoriamente un valor para cada motivo contenidos en $KM_{i,\star,\star}$.
- **else**
  - $Melodía\ base_{3,\star} \leftarrow$ es una melodía generada por los promedios de los valores para cada motivo en $KM_{i,\star,\star}$.
- **end**

end

Algoritmo 51: Actualizar partitura

Input: $P_{\star,\star,i}$ $y_{i,new,\star}$

Output: Partitura actualizada para el $i-$ésimo compositor

```
x_{i-worst} ← de la melodía con peor aptitud en P_{⋆,⋆,i}
. if y_{i,new,⋆} es factible then
    if y_{i-worst} es factible then
        if φ(y_{i,new,⋆}) es mejor que φ(y_{i-worst}) then
            Reemplazar y_{i-worst} por y_{i,new,⋆} en la partitura P_{⋆,⋆,i}.
        else
            if φ(y_{i,new,⋆}) es igual que φ(y_{i-worst}) then
                if dif(y_{i,new,⋆}) < dif(y_{i-worst}) then
                    Reemplazar y_{i-worst} por y_{i,new,⋆} en la partitura P_{⋆,⋆,i}.
                end
            end
        end
    else
        Reemplazar y_{i-worst} por y_{i,new,⋆} en la partitura P_{⋆,⋆,i}.
    end
else
    if x_{i-worst} es infactible then
        if C_{y_{i,new,⋆}} < C_{y_{i-worst}} then
            Remplazar y_{i-worst} por y_{i,new,⋆} en la partitura P_{⋆,⋆,i}.
        else
            if C_{y_{i,new,⋆}} = C_{y_{i-worst}} then
                if dif(y_{i,new,l}) < dif(y_{i-worst}) then
                    Reemplazar y_{i-worst} por y_{i,new,⋆} en la partitura P_{⋆,⋆,i}.
                end
            end
        end
    end
end
```

G1

$$
\begin{aligned}
&\text{mín } f(x) = 5x_1 + 5_2 + 5x_3 + 5x_4 - 5\sum_{l=1}^{4} x_l^2 - \sum_{l=5}^{13} x_l.\\
&\text{subject to}\\
&2x_1 + 2x_2 + x_{10} + x_{11} \leq 10\\
&-8x_1 + x_{10} \leq 0\\
&-2x_4 - x_5 + x_{10} \leq 0\\
&2x_1 + 2x_3 + x_{10} + x_{12} \leq 10\\
&-8x_2 + x_{11} \leq 0\\
&-2x_6 - x_7 + x_{11} \leq 0\\
&2x_2 + 2x_3 + x_{11} + x_{11} \leq 10\\
&-8x_3 + x_{11} \leq 0\\
&-2x_8 - x_9 + x_{12} \leq 0\\
&0 \leq x_l \leq 1 \ \forall l = 1, \ldots, 9\\
&0 \leq x_l \leq 100 \ \forall l = 10, \ldots, 12\\
&0 \leq x_{13} \leq 1\\
&\text{donde: } f(x^\star) = -15\\
&\text{con } x^\star = [1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 1]
\end{aligned}
\tag{6.3.1}
$$

G2

$$
\begin{aligned}
&\text{máx } f(x) = \left| \frac{\sum_{l=1}^{n} \cos^4(x_l) - 2\prod_{l=1}^{n} \cos^2(x_l)}{\sqrt{\sum_{l=1}^{n} l * \cos^2(x_l)}} \right|\\
&\text{subject to}\\
&\prod_{l=1}^{n} x_l \geq 0.75\\
&\prod_{l=1}^{n} x_l \leq 0.75n\\
&0 \leq x_l \leq 10 \ \forall l = 1, \ldots, n\\
&\text{Si } n = 20 \text{ entonces } f(x^\star) = 0.8036\\
&\text{Si } n = 50 \text{ entonces } f(x^\star) = 0.83
\end{aligned}
\tag{6.3.2}
$$

G3

$$
\begin{aligned}
&\text{máx } f(x) = (\sqrt{n})^n * \prod_{l=1}^{n} x_l\\
&\text{subject to}\\
&\sum_{l=1}^{n} x_l^2 = 10 \leq x_l \leq 1 \ \forall l = 1, \ldots, n\\
&\text{donde: } f(x^\star) = 1\\
&\text{con } x^\star = [\tfrac{1}{\sqrt{n}}, \tfrac{1}{\sqrt{n}}, \ldots, \tfrac{1}{\sqrt{n}}]
\end{aligned}
\tag{6.3.3}
$$

G4

$$
\begin{aligned}
&\text{mín } f(x) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141\\
&\text{subject to}\\
&85.334407 + 0.0056858x_2x_5 + 0.0006262x_lx_4 - 0.0022053x_3x_5 \geq 0\\
&85.334407 + 0.0056858x_2x_5 + 0.0006262x_lx_4 - 0.0022053x_3x_5 \leq 92\\
&80.51249 + 0.0071317x_2x_5 + 0.0029955x_lx_2 + 0.0021813x_3^2 \geq 90\\
&80.51249 + 0.0071317x_2x_5 + 0.0029955x_lx_2 + 0.0021813x_3^2 \leq 110\\
&9.300961 + 0.0047026x_3x_5 + 0.00l2547x_lx_3 + 0.0019085x_3x_4 \geq 20\\
&9.300961 + 0.0047026x_3x_5 + 0.00l2547x_lx_3 + 0.0019085x_3x_4 \leq 25\\
&78 \leq x_1 \leq 102\\
&33 \leq x_2 \leq 45\\
&27 \leq x_l \leq 45 \ \forall l = 3, 4, 5\\
&\text{where: } f(x^\star) = -30665.5\\
&\text{with } x^\star = [78, 33, 29.995, 45, 36.776]
\end{aligned}
\tag{6.3.4}
$$

G5

$$
\begin{aligned}
&\text{mín } f(x) = 3x_1 + 0.000001x_1^2 + 2x_2 + \tfrac{0.000002}{3x_2^3}\\
&\text{subject to}\\
&x_4 - x_3 + 0.55 \geq 0\\
&x_3 - x_4 + 0.55 \geq 0\\
&1000\sin(-x_3 - 0.25) + 1000\sin(-x_4 - 0.25) + 894.8 - x_1 = 0\\
&1000\sin(x_3 - 0.25) + 1000\sin(x_3 - x_4 - 0.25) + 894.8 - x_2 = 0\\
&1000\sin(x_4 - 0.25) + 1000\sin(x_4 - x_3 - 0.25) + 1294.8 = 0\\
&0 \leq x_l \leq 1200 \ \forall l = 1, 2\\
&-0.55 \leq x_i \leq 0.55 \ \forall l = 3, 4\\
&\text{donde: } f(x^\star) = 5126.4981\\
&\text{con } x^\star = [679.9453, 1026.067, 0.1188764, -0.3962336]
\end{aligned}
\tag{6.3.5}
$$

G6

$$
\begin{aligned}
&\text{mín } f(x) = (x_1 - 10)^3 + (x_2 - 20)^3 \\
&\text{subject to} \\
&(x_1 - 5)^2 + (x_2 - 5)^2 - 100 \geq 0 \\
&-(x_1 - 6)^2 - (x_2 - 5)^2 + 82.81 \geq 0 \\
&13 \leq x_1 \leq 100 \\
&0 \leq x_2 \leq 100 \\
&\text{donde: } f(x^\star) = -6961.81381 \\
&\text{con } x^\star = [14.095, 0.84296]
\end{aligned} \tag{6.3.6}
$$

G7

$$
\begin{aligned}
&\text{mín } f(x) = x_1^2 + x_2^2 + x_1 x_2 - 14x - 1 - 16x_2 \\
&+(x_3 - 10)^2 + 4(x4 - 5)^2 + (x_5 - 3)^2 + 2(x_6 - 1)^2 \\
&+5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45 \\
&\text{subject to} \\
&105 - 4x_1 - 5x_2 + 3x_7 - 9x_8 \geq 0 \\
&-3(x_1 - 2)^2 - 4(x_2 - 3)^2 - 2x_3^2 + 7x_4 + 120 \geq 0 \\
&-10x - 1 + 8x_2 + 17x_7 - 2x_8 \geq 0 \\
&-x_1^2 - 2(x_2 - 2)^2 + 2x_1 x_2 - 14x_5 + 6x_6 \geq 0 \\
&8x_1 - 2x_2 - 5x_9 + 2x_{10} + 12 \geq 0 \\
&-5x_1^2 - 8x_2 - (x_3 - 6)^2 + 2x_4 + 40 \geq 0 \\
&3x_1 - 6x_2 - 12(x_9 - 8)^2 + 7x_{10} \geq 0 \\
&-0.5(x_1 - 8)^2 - 2(x_2 - 4)^2 - 3x_5^2 + x_6 + 30 \geq 0 \\
&-10 \leq x_l \leq 10 \; \forall l = 1, \ldots, 10 \\
&\text{donde: } f(x^\star) = 24.3062091 \\
&\text{con } x^\star = [2.171996, 2.363683, 8.773926, 5.095984, \\
&0.99065048, 1.430574, 1.321644, 9.828726, 8.280092, 8.375927]
\end{aligned} \tag{6.3.7}
$$

G8

$$
\begin{aligned}
&\text{máx } f(x) = \frac{\sin^3(2\pi x_1)*\sin(2\pi x_2)}{x_1^3*(x_1+x_2)} \\
&\text{subject to} \\
&x_1^2 - x_2 + 1 \leq 0 \\
&1 - x_1 - (x_2 - 4)^2 \leq 0 \\
&0 \leq x_l \leq 10 \ \forall l = 1, 2 \\
&\text{donde: } f(x^\star) = 0.1
\end{aligned}
\tag{6.3.8}
$$

G9

$$
\begin{aligned}
&\text{mín } f(x) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 \\
&+10x_5^6 + 7x_7^4 - 4x_6x_7 - 10x_6 - 8x_7 \\
&\text{subject to} \\
&127 - x_1^2 - 3x_2^4 - x_3 - 4x_4^2 - 5x_5 \geq 0 \\
&196 - 23x_1 - x_2^2 - 2x_6^2 + 8x_1 \geq 0 \\
&282 - 7x_1 - 3x_2 - 10x_3^2 - x_4 + x_5 \geq 0 \\
&-4x_1^2 - x_2^2 + 3x_1x_2 - 5x_6 + 11x_7 \geq 0 \\
&-10 \leq x_l \leq 10 \ \forall l = 1, \ldots, 7 \\
&\text{donde: } f(x^\star) = 680.6300873 \\
&\text{con } x^\star = [2.330499, 1.951372, -0.4775414 \\
&\ 4.365726, -0.6244870, 1.038131, 1.594227]
\end{aligned}
\tag{6.3.9}
$$

G10

$$
\begin{aligned}
&\text{mín } f(x) = x_1 + x_2 + x_3 \\
&\text{subject to} \\
&1 - 0.0025(x_4 + x_6) \geq 0 \\
&1 - 0.01(x_8 - x_5) \geq 0 \\
&x_2 x_7 - 1250x_5 - x_2 x_7 \geq 0 \\
&1 - 0.0025(x_5 + x_7 - x_4) \geq 0 \\
&x_1 x_6 - 833.33252x_4 - 100x_1 + 83333.333 \geq 0 \\
&x_3 x_8 - 1250000 - x_3 x_5 + 2500x_5 \geq 0 \\
&100 \leq x_1 \leq 10000 \\
&1000 \leq x_l \leq 10000 \ \forall l = 2, 3 \\
&10 \leq x_l \leq 1000 \ \forall l = 4, \ldots, 8 \\
&\text{donde: } f(x^\star) = 7049.330923 \\
&\text{con } x^\star = [579.3167, 1359.943, 5110.071, \\
&\ 182.074, 295.5985, 217.9799, 286.4162, 395.5979]
\end{aligned}
\tag{6.3.10}
$$

G11

$$
\begin{aligned}
&\text{mín } f(x) = x_1^2 + (x_2 - 1)^2 \\
&\text{subject to} \\
&x_2 - x_1^2 = 0 \\
&-1 \leq x_l \leq 1 \ \forall l = 1, 2 \\
&\text{donde: } f(x^\star) = 0.75000455 \\
&\text{con } x^\star = [\pm 0.70711, 0.5]
\end{aligned}
\tag{6.3.11}
$$

G12

$$
\begin{aligned}
&\text{máx } f(x) = \tfrac{100 - (x_1 - 5)^2 - (x_2 - 5)^2 - (x_3 - 5)^2}{100} \\
&\text{subject to} \\
&(x_1 - \rho_1)^2 + (x_2 - \rho_2)^2 + (x_3 - \rho_3)^2 \leq 0.25 \\
&\text{for: } \rho_s = 1, 3, 5, 7, 9 \ \forall s = 1, 2, 3 \\
&0 \leq x_l \leq 10 \ \forall l = 1, 2, 3 \\
&\text{donde: } f(x^\star) = 1 \\
&\text{con } x^\star = [5, 5, 5]
\end{aligned}
\tag{6.3.12}
$$

El *G-suite* incluye problemas con funciones objetivo lineales y no lineales; y restricciones del tipo igualdad y desigualdad. En la tabla 6.1 se muestra la información relevante de cada una de las instancias.

Tabla 6.1: Resumen de información de los 12 casos de prueba [152]

Función	Número de variables	Tipo de variables de desición	Radio $\rho \star$	LI	NE	NI	α
G1	13	Cuadrática	0.0111 %	9	0	0	6
G2	k	No lineal	99.8474 %	0	0	2	1
G3	k	Polinomial	0.0000 %	0	1	0	1
G4	5	Cuadrática	51.1230 %	0	0	6	2
G5	4	Cúbica	0.0000 %	2	3	0	3
G6	2	Cuadrática	0.0066 %	0	0	2	2
G7	10	Cuadrática	0.0003 %	3	0	5	6
G8	2	No lineal	0.8560 %	0	0	2	0
G9	7	Polinomial	0.5121 %	0	0	4	2
G10	8	Lineal	0.0010 %	3	0	3	6
G11	2	Cuadrática	0.0000 %	0	1	0	1
G12	3	Cuadrática	-	0	0	25	0

$\star$ $\rho = \frac{\text{número de soluciones } \in \mathcal{F}}{\text{número de soluciones } \in \mathcal{S}}$;

LI número de restricciones del tipo ecuaciones lineales;

NE número de restricciones del tipo ecuaciones no-lineales;

NI número de restricciones del tipo inecuaciones no-lineales y

α número de restricciones activas.

6.3.2. Diseño de experimentos

Con fin de probar, analizar, evaluar la actuación del MMC para solucionar instancias del PLN restricto se probaron las instancias del *G-suite*, para cada una de las pruebas se realizaron 20 repeticiones independientes, de cada una de las repeticiones se registró la mejor solución encontrada. Posteriormente, los resultados encontrados con MMC se compararon contra las soluciones obtenidas por la metaheurístcas siguientes:

- EA basado en homorfismo (KM) [123], en el cual utilizaron 1,400,000 llamadas a la función objetivo.

- Sistema inmune artificial (AIS)[46] en el cual utilizaron 350,000 llamadas a la función objetivo.
- CDE [126] en el cual utilizarón 100,100 llamadas a la función objetivo
- CAEP [38] en el cual utilizarón 97,540 llamadas a la función objetivo
- Evolución diferencial con el uso dinámico de la primera variante APM [48], en el cual utilizaron 350,000 llamadas a la función objetivo.
- PSO [106], con 25,000 llamadas a la función objetivo.
- TSGA [233] con 100,000 evaluaciones de la función objetivo

Ya que, las técnicas seleccionadas para la comparación realizan en promedio 208,373.4 evaluaciones de la función objetivo con una desviación estándar de 4,916,458. se decidió fijar el número de evaluaciones a realizar por el MMC en 240,000. Sobre los resultados de MMC se aplicó la prueba Bootstrap[2] para determinar interválos de confianza sobre los resultados del experimento.

Con el objeto de facilitar la comparación entre los resultados obtenidos por las metaheurísticas, estos datos se normalizaron con la ecuación 5.3.23.

Además se usó la prueba de Wilcoxon en los resultados obtenidos por el MMC y generados por las otras metaheurísticas; a fin de determinar si el MMC producía resultados estadísticamente diferentes.

6.3.3. Configuración de parámetros

Debido alta heterogeneidad de las instancias de prueba, se decidió calibrar los parámetros para cada uno de ellos. Para este proceso de ajuste, se usó el procedimiento iterativo de fuerza bruta (veáse anexo F), realizando 15 pruebas [3] en cada una de las instancias a resolver; los resultados del proceso de ajuste se muestran en la tabla 6.2.

6.4. Resultados Numéricos

El algoritmo MMC se implemento en Matlab 7.10.0, y se ejecuto en una computadora DELL inspiron 1720.

[2]El Bootstrap es un método de remuestreo propuesto por Bradley Efron en 1979, el cual pemite estimar la precisión de muestras estadísticas (media, varianza,percentiles, entre otros) a través de tomar de forma aleatoria un conjunto de datos de la información disponible

[3]Se realizarón sólo 15 puebas debio a la gran cantidad de recursos empleados (tiempo computacional) en cada una de ellas.

Tabla 6.2: Configuración de parámetros del *MMC*

Parámetro	G1	G2	G3	G4	G5	G6
máx $\_arrangement$	48000	48000	12000	48000	48000	48000
ifg	0.005	0.005	0.001	0.005	0.005	0.005
cfg	0.02	0.02	0.0009	0.01	0.01	0.01
Nc	5	5	20	5	5	5
Ns	10	10	5	10	3	10
cof	1	1	2	1	1	1t

Parámetro	G7	G8	G9	G10	G11	G12
máx $\_arrangement$	24000	24000	24000	24000	20000	24000
ifg	0.005	0.001	0.04	0.095	0.005	0.005
cfg	0.006	0.05	0.072	0.1	0.005	0.0075
Nc	10	10	10	10	12	10
Ns	10	15	6	3	10	5
cof	1	1	1	2	1	1

En la tabla 6.3 se muestran los resultados obtenidos, su varianza, desviación estándar y *rho* (relación entre el número de soluciones factibles y el número de soluciones evaluadas 20). La poca distancia entre la mejor solución encontrada y el valor medio, así como la desviación estándar baja, exhiben la buena calidad de resultados obtenidos por *MMC*.

En la Tabla 6.4, se muestran los resultados de la prueba bootstrap considerando una certeza del 95 %.

Como se mencionó, los resutados obtenidos por el MMC se compararon con las soluciones encontradas por KM, AIS, CDE, CAEP, APMGA variante 1, PSO, TSCGA. En la Tabla 6.5 se muestran los mejores resultados encontrados por las metaheurísticas, la tabla 6.6 muestran los resultasos promedios y en la Tabla 6.7, se muestran los peores resultados. Los datos proporcionados para cada metaheurística se tomaron de las referencias originales.

En las Tablas 6.5, 6.6 y 6.7 se observa que el MMC produce en promedio en 33.33 % de los casos las mejores soluciones.

Tabla 6.3: Resultados numéricos del *MMC*.

Instancia	ρ	Mejor	Media	Peor	s^2	s
G1	1	-15	-14.9	-13	2.00E-01	4.47E-01
G2	1	0.8036	0.7864	0.7395	3.18E-04	1.78E-02
G3	0	0.9999	0.9839	0.7768	2.47E-03	4.97E-02
G4	1	-30664.9924	-30664.9913	-30664.9705	2.38E-05	4.88E-03
G5	0.8	4232.57159	4896.5759	5612.51901	1.48E+05	3.85E+02
G6	1	-6961.8138	-6961.8138	-6961.8138	0.00E00	0.00E00
G7	1	24.3281	24.4677	24.9870	3.27E-02	1.81E-01
G8	1	0.09582	0.09582	0.09582	4.61E-24	2.14E-12
G9	1	684.1806	684.1996	684.2519	2.78E-04	1.66E-02
G10	1	8648.2227	9286.4934	11745.217	7.39E+05	8.60E+02
G11	1	0.7501	0.7798	0.8801	1.25E-03	3.54E-02
G12	0	1	1	1	1.83E-18	1.35E-09

La prueba G2 se ejecutó con $n = 20$ y la prueba G3 con $n = 10$.

donde: s^2 es varianza y s es desviación estándar

Tabla 6.4: Intervalo con el 95 % confianza obtenido por el método de bootstrap

Instancia	Límite superior	Límite inferior
G1	-15	-14.7
G2	0.77871	0.79413
G3	0.95913	0.99755
G4	-30664.99236	-30664.98909
G5	4741.83877	5056.98263
G6	-6961.81388	-6961.81388
G7	24.39274	24.54882
G8	0.09582	0.09582
G9	684.19197	684.20662
G10	8944.15324	9683.31701
G11	0.76516	0.79470
G12	1.00000	1.00000

Tabla 6.5: Mejores resultados obtenidos por las metaheurísticas

Instancia	KM	AIS	CDE	CAEP	APMGA variante 1	PSO	TSCGA	MMC
G1	-14.7864	**-15.0000**	**-15.0000**	-	**-15.0000**	-14.9987	**-15.0000**	**-15.0000**
G2	0.7995	0.7703	**0.8036**	-	-	-	0.7908	**0.8036**
G3	0.9997	**1.0000**	0.9954	-	-	-	0.9999	0.9999
G4	-30664.5000	-30665.0815	-30665.5387	-30664.8000	-30665.5238	**-30665.500**0	-30665.3743	-30664.9924
G5	-	5126.6861	5126.5709	-	5127.3606	-	-	**4232.5716**
G6	-6952.1000	-6961.1792	**-6961.8139**	-	-6961.7961	-6943.5000	-6961.8092	-6961.8138
G7	24.6200	24.3324	**24.3062**	-	24.4163	24.6269	24.3361	24.3281
G8	**-0.0958**	**-0.0958**	**-0.0958**	**-0.0958**	**-0.0958**	-0.0959	-	**-0.0958**
G9	680.9100	680.8317	**680.6301**	-	680.6334	680.8370	-	684.1806
G10	7147.9000	7133.1280	**7049.2481**	-	7205.1436	7219.2500	-	8648.2227
G11	**0.7500**	**0.7500**	0.7499	0.7403	0.7524	**0.7500**	-	0.7501
G12	**1.0000**	-	**1.0000**	**1.0000**	-	**1.0000**	-	**1.0000**

Tabla 6.6: Resultados medios obtenidos por las metaheurísticas

Instancia	KM	AIS	CDE	CAEP	APMGA variante 1	PSO	TSCGA	MMC
G1	-14.7082	**-15.0000**	**-15.0000**	-	-14.9526	14.9880	**-15.0000**	-14.9000
G2	**0.7967**	0.7040	0.7249	-	-	-	0.7554	0.7864
G3	0.9989	0.9970	0.7886	-	-	-	**0.9995**	0.9839
G4	-30655.3000	-30648.1750	-30665.5387	-30611.1000	-30665.0947	**-30665.5000**	-30664.8926	-30664.9913
G5	-	5307.2016	5207.4107	-	5321.5714	-	-	**4896.5759**
G6	-6342.6000	-6959.5503	**-6961.8139**	-	-6742.3431	-6899.5000	-6961.7470	-6961.8138
G7	24.8260	31.5141	**24.3062**	-	26.4574	26.6590	24.4093	24.4677
G8	-0.0892	**-0.0958**	**-0.0958**	-0.0953	-0.0877	**-0.0958**	-	**-0.0958**
G9	681.1600	682.1937	**680.6301**	-	680.7851	681.5270	-	684.1996
G10	8163.6000	8158.9658	**7049.2483**	-	8392.4000	7558.7000	-	9286.4934
G11	**0.7500**	0.7505	0.7580	0.7930	0.8556	0.7500	-	0.7798
G12	0.9999	-	**1.0000**	0.9973	-	**1.0000**	-	**1.0000**

Tabla 6.7: Peores resultados obtenidos por las metaheurísticas

Instancia	KM	AIS	CDE	CAEP	APMGA variante 1	PSO	TSCGA	MMC
G1	-14.6154	**-15.0000**	**-15.0000**	-	-14.6374	-14.9579	**-15.0000**	-13.0000
G2	**0.7912**	0.5956	0.5909	-	-	-	0.7094	0.7395
G3	**0.9978**	0.9810	0.6399	-	-	-	0.9987	0.7768
G4	-30645.9000	-30613.4426	-30665.5387	-30466.8000	-30661.7610	-30665.4000	-30663.9483	**-30664.9705**
G5	-	5927.3671	**5327.3905**	-	5993.0113	-	-	5612.5190
G6	6952.1000	-6955.6034	-6961.8139	-	-1476.0028	-6821.1000	-6961.5063	**-6961.8138**
G7	25.0690	47.2142	**24.3062**	-	30.7900	31.9468	24.6403	24.9870
G8	-0.0291	**-0.0958**	**-0.0958**	-0.0901	-0.0258	**-0.0958**	-	**0.0958**
G9	683.1800	688.6037	**680.6301**	-	681.1733	686.6570	-	684.2519
G10	9659.3000	9493.8894	**7049.2485**	-	10349.9094	8020.1500	-	11745.2170
G11	**0.7500**	0.7516	0.7965	0.8380	0.9468	**0.7500**	-	0.8801
G12	0.9996	-	**1.0000**	0.9863	-	**1.0000**	-	**1.0000**

Tabla 6.8: Comparación de los mejores resultados

	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	G11	G12
1.0	1	2	3	1	5	6	1,6	1,2,3,4,5,6,8	8	8	4	1,2,3,4,5,6,7,8
0.9												
0.8												
0.7				4								
0.6												
0.5				8		1	5					
0.4		7		2								
0.3												
0.2					2						5	
0.1		1	1	7	3		2,7,8		1,2,6	1,2,5,6		
0.0	2,3,5,6,7,8	3,8	2,7,8	3,5,6	8	2,3,5,7,8	3		3,5	3	1,2,3,6,8	

donde:

1 es KM	3 es CDE	5 es APMGA	7 es TSCGA
2 es AIS	4 es CAEP	6 es PSO	8 es MMC

Tabla 6.9: Comparación de los resultados promedio

	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	G11	G12
1	6	2	3	4	5	1	2	1,2,3,4,5,6,8	8	8	5	1,2,3,4,5,6,7,8
0.9					2							
0.8		3										
0.7												
0.6										5		
0.5		7								1,2		
0.4					3	5			2		4	
0.3				2			5,8		6		8	
0.2		8		1						6		
0.1		1	8			6	1		1		3	
0	1,2,3,5,7,8		1,2,7	3,5,6,7,8	8	2,3,7,8	3,7,8		3,5	3	1,2,6	

donde:

1 es KM 3 es CDE 5 es APMGA 7 es TSCGA

2 es AIS 4 es CAEP 6 es PSO 8 es MMC

Tabla 6.10: Comparación de los peores resultados

	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	G11	G12
1	8	2,3	3	4	5	1	2	2,3,4,6	2	8	5	4
0.9					2							
0.8									6			
0.7								1		5	8	
0.6			8		8			5		1		
0.5									8	2		
0.4		7				5					4	
0.3		8		2			5,6		1			
0.2	1,5				3					6	3	
0.1		1	2	1					5			
0	2,3,6,7		1,7	3,5,6,7.8		2,3,6,7,8	1,3,7,8	8	3	3	1,2,6	1,3,6,8

donde:

1 es KM 3 es CDE 5 es APMGA 7 es TSCGA

2 es AIS 4 es CAEP 6 es PSO 8 es MMC

Tabla 6.11: Resutados de la prueba Wilcoxon

Algoritmo	Mejor		Promedio		Peor	
	p	h	p	h	p	h
KM	0.97380076	0	1	0	0.5993607	0
AIS	0.97380076	0	0.94764453	0	1	0
CDE	0.86240136	0	0.77278195	0	0.72897789	0
CAEP	0.97142857	0	1	0	0.88571429	0
APMGA variant 1	1	0	1	0	1	0
PSO	0.94965035	0	0.88103661	0	0.94965035	0
TSCGA	1	0	1	0	1	0
MMC	1	0	1	0	1	0

En las tablas 6.8, 6.9 y 6.10 se muestra la comparación entre la metaheurísticas. En ellas se puede ver el buen comportamiento del MMC para resolver instancias no lineales restrictas.

Con base en los resultados obtenidos de la prueba Wilcoxon (ver tabla 6.11), podemos decir que el MMC tiene un rendimiento similar a las otras metaheurísticas comparadas en esta prueba. En la tabla 6.12 se muestra el tiempo de ejecución del algoritmo.

6.5. Análisis de resultados

Con base en los resultados obtenidos, se puede decir que el MMC alcanza los mejores resultados en el 41,66 %, 25 % y 33 % de los casos para los mejores, promedios y peores resultados, respectivamente. Estos resultados ilustran que el MMC tiene una alta capacidad para resolver instancias restrictas del problema PLN. Cabe mencionar que los mejores resultados encontrado por el MMC corresponden a las instancias G1, G2, G5, G8 y G12; en contraste los perores resultados obtenidos por el MMC fueron sobre la instancia G10.

Tabla 6.12: Tiempo de ejecución del algoritmo MMC en segundos

Instancia	Media	Máximo	Mínimo	s^2	s
G1	300.57	311.83	294.64	21.44	4.63
G2	328.63	340.82	321.22	20.51	4.53
G3	423.89	434.61	412.17	35.87	5.99
G4	254.83	259.24	249.23	7.18	2.68
G5	256.15	336.23	219.87	949.45	30.81
G6	251.48	275.69	242.87	94.87	9.74
G7	310.66	334.88	302.11	60.24	7.76
G8	284.88	288.64	280.60	5.46	2.34
G9	259.94	269.94	251.40	23.92	4.89
G10	280.22	292.29	262.15	72.35	8.51
G11	315.26	339.41	298.38	128.03	11.32
G12	559.28	582.53	541.04	124.65	11.16

Capítulo 7

Problema de diseño de zonas

Este capítulo tiene el propósito de mostrar la adaptación y aplicación del MMC para solucionar una intancia del problema de diseño de zonas electorales; la instancia utilizada fue el Estado de México, ya que la zonificación de este estado es una de las más difíciles, ya que tiene el mayor número de unidades geográficas básicas y es en el que se requiere generar el mayor número de distritos a nivel nacional. Los resultados obtenidos con el MMC fueron comparados con los resultados encontrados por recocido simulado (estrategía utilizada por el IFE - Instituto Federal Electoral-).

Por lo anterior, el presente capítulo, se encuentra estructurado en las siguientes secciones: marco de referencia; adaptación de MMC; experimentación y análisis de resultados.

7.1. Marco de referencia

7.1.1. El problema de diseño de zonas

En términos generales, el problema de diseño de zonas es la partición o división de un espacio geográfico en zonas; tal que, se satisfagan un conjunto de criterios específicos; generalmente, en las ciencias geo espaciales el diseño de zonas se realiza por la agrupación o conglomerados de unidades geográficas básicas (UGB). Previamente, existentes en el espacio de interés [162]. Lo anterior, se ilustra a continuación.

Ejemplo 7.1

Considere una área geográfica cuadrada de 25 hectáreas (ha); la cual, se encuentra dividida en unidades cuadradas de una ha, tal como se muestra en la Figura 7.1. Se desea diseñar tres zonas, tal que toda unidad

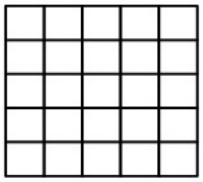

Figura 7.1: Área geográfica inicial

esté contenida en exactamente una zona; cada zona contenga entre ocho y nueve unidades; además cada zona sea conexa[1]*. En la Figura 7.2, se muestran varias zonificaciones que no satisfacen los criterios del problema; mientras que, en la Figura 7.3, se muestran algunas zonificaciones que satisfacen las restricciones del problema.*

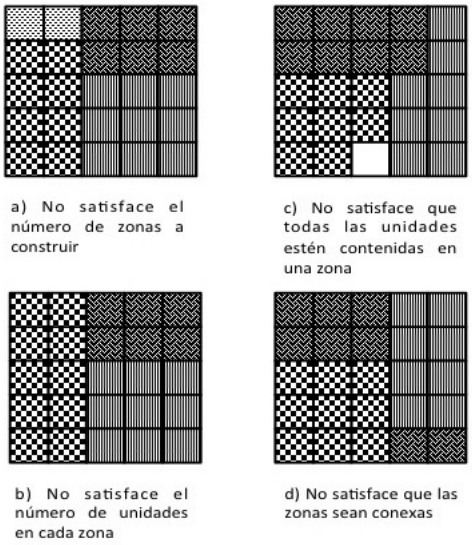

Figura 7.2: Ejemplo de zonas que no satisfacen los criterios del problema

El problema de diseño de zonas es un problema de optimización combinatoria; para el cual se ha demostrado que las siguientes afirmaciones son ciertas [8, 85, 204].

[1]Se dice que una zona X es conexa, si para cualquier par de unidades x_1 y $x_2 \in X$ existe un conjunto de unidades contenidas en X que conectan a x_1 con x_2

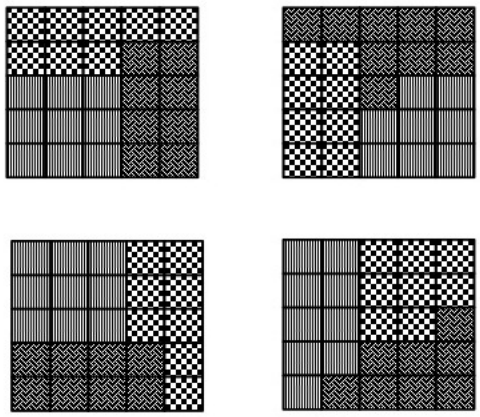

Figura 7.3: Ejemplo de zonas que satisfacen los criterios del problema

- El Diseño de zonas donde se minimice el equilibrio poblacional[2] es un problema NP-duro.
- El Diseño de zonas conexas minimizando el costo asociado a las zonas es un problema NP-duro.
- El Diseño de zonas conexas, minimizando el equilibrio poblacional es un problema NP-duro.
- El Diseño de zonas donde se maximice la compacidad geométrica [3] de las zonas generadas es un problema NP-duro.

7.1.2. Trabajos relacionados

El problema de diseño de zonas se presenta donde se requiere construir subregiones de una región determinada; por ejemplo: en la actividad forestal al configurar las unidades de manejo forestal (UNMAFOR); en la actividad educativa al construir los distritos educativos, entre otros. En la Figura 7.4, se muestran algunas de las aplicaciones más usuales del problema de zonificación.

Las características de algunas de las aplicaciones anteriores se muestran en las tablas 7.1 y 7.2; si se desea mayor información se puede consultar [203].

[2]Se busca que todas las zonas contengan, en promedio, la misma cantidad de habitantes; por lo cual, se penalizan las diferencias entre las cantidades de habitantes

[3]Se dice que una zona X es compacta si su forma es regular y homogénea dentro de un espacio limitado

Figura 7.4: Aplicaciones del problema de diseño de zonas, elaborado con base en [203]

Tabla 7.1: Aplicaciones del diseño de zonas. Elaborado con base en [203]

Aplicación	Unidades geográficas básicas	Características de las zonas	Número de zonas
Zonas de Ventas	Formada por los clientes con características similares (principalmente ubicación geográfica)	Zonas compactas y conexas; además se busca que las zonas contengan el mismo número de clientes; alguna aplicaciones incluyen el tiempo máximo de traslado	Determinado por el número de equipos de trabajo
Zonas Escolares	Formado por estudiantes con características similares (principalmente ubicación)	Zonas conexas, accesibles y compactas; además se busca que las zonas contengan el mismo número de estudiantes.	Determinado por el número de escuelas

Tabla 7.2: Aplicaciones del diseño de zonas

Aplicación	Unidades geográficas básicas	Características de las zonas	Número de zonas
Zonas electorales	Establecidas por la ley.	Las zonas deben ser compactas, conexas y además se busca que las zonas contengan el mínimo número de personas	Determinadas por ley.

En las tablas 7.3, 7.4, 7.5, 7.6 y 7.7, se muestran en orden cronológico algunas de las aplicaciones del problema diseño de zonas, desarrolladas en las últimas décadas.

Tabla 7.3: Aplicaciones sobre diseño de zonas

Año	Autor	Aplicación	Algoritmo
1963	Weaver y Hess	Zonas electorales	Location-allocation
1965	Nagel	Zonas electorales	
1965	Hess, Weaver, Siegfeldt, Whelan y Zitlau	Zonas electorales	Location-allocation
1966	Kaiser	Regionalización	Exacto (En este trabajo se propuso la inclusión de un criterio que combina en forma ponderada el equilibrio poblacional y la compacidad geométrica)
1967	Mills	Zonas electorales	Location-allocation

Tabla 7.4: Aplicaciones sobre diseño de zonas (continuación)

Año	Autor	Aplicación	Algoritmo
1970	Garfinkel y Nemhauser	Zonas electorales	Búsqueda exhaustiva
1971	Hess y Samuels	Zonas de ventas	Desarrollarón un modelo denominado GEOLINE, el cual usa una heurística de ubicación y asignación para resolver el problema de diseño de zonas. Esta heurística es iterativa y se compone de dos fases independientes, una localización y otra de asignación [114].
1972	Helbig et al	Zonas electorales	Programación entera combinada
1973	Openshaw	Regionalización	Algoritmo de Conglomerados y métodos de jerarquización
1976	Segal y Weinberger	Zonas de mantenimiento	
1982	Robertson	Zonas electorales	Location-allocation
1990	Ferland y Guénette	Escolar	
1990	Browdy	Redistribución de distritos	Recocido simulado
1991	Schoepfle y Church	Escolar	Exacto
1995	Opensaw y Rao	Reingeniería del diseño de zonas	Recocido simulado, Búsqueda Tabú

Tabla 7.5: Aplicaciones sobre diseño de zonas (continuación)

Año	Autor	Aplicación	Algoritmo
1996	IFE	Zonas electorales	Recocido simulado
1997	Ricca	Zonificación electoral	
1997	Ricca y Simeone	Zonificación electoral (Para cinco regiones de Italia)	Recocido simulado
1998	Mehrotra, Johnson y Nemhauser	Zonas electorales	Branch and price
2000	Cirincione, Darling y O'Rourke	Zonas electorales	Aleatoria sin función objetivo
2001	Macmillan	Zonas electorales	Recocido simulado
2002	Muyldermans, Cattrysse, Van Oudheusden y Lotan	Esparcimiento de sal	Rutas por arcos
2002	Aerts y Heuvelink	Uso de tierra	Recocido simulado
2003	Bozkaya, Erkut y Laporte	Zonas electorales	Búsqueda tabú
2003	Blais, Lapierre y Laporte	Servicio médico a domicilio	Búsqueda Tabú
2003	Bergey, Ragsdale y Hoscote	Zonas eléctricas	Algoritmos Genéticos
2003	Bergey, Ragsdale y Hoscote	Zonas eléctricas	Recocido simulado/Genéticos
2003	D'Amico, Wang, Batta y Rump	Distritos para vigilancia	Recocido simulado

Tabla 7.6: Aplicaciones sobre diseño de zonas (continuación)

Año	Autor	Aplicación	Algoritmo
2003	Forman y Yue	Zonificación electoral	Algoritmos genéticos
2003	Bergey et al	Zonificación electoral	Algoritmos genéticos
2003	Bozkaya et al.	Zonas electorales	Búsqueda tabú
2004	Duque	Regionalización	Algoritmo de regionalización con búsqueda selectiva
2004	Duque y Church	Regionalización	Heurística de concentración
2004	Caro, Shirabe, Guignard y Weintraub	Escolar	–
2005	Baçao, Lobo y Painho	Zonas electorales	Algoritmos genéticos
2005	Xiao	Zonas electorales	Algoritmos evolutivos
2005	Shortt, Moore, Coombes y Wymer	Servicio médicos	ERA por las siglas en inglés de *European Regionalisation Algorithm*
2005	Baçâo et al.	Zonificación electoral	Algoritmos genéticos
2005	Vargas-Suárez *et al*	Zonas de ventas	GRASP
2006	Bong y Wang	Zonas electorales	Búsqueda tabú/búsqueda dispersa/path relinking
2006	Chou y Li	Zonas electorales	Monte Carlo/recocido simulado
2006	DesJardins y Bulka	Escolar	Hill climbing
2006	Middleton	Regionalización	Heurística destilación (Heuristic Distillation)

Tabla 7.7: Aplicaciones sobre diseño de zonas (continuación)

Año	Autor	Aplicación	Algoritmo
2007	Chou y Li	Zonas electorales	Recocido simulado, algoritmos genéticos
2007	Ricca y Simeone	Zonas electorales	Búsqueda tabú, recocido simulado, descendent y old bachelor Acceptance
2007	Tavares-Pereira, Rui, Mousseau y Roy	Zonas tarifarias	Evolutivos
2008	Fernández, Kalcsics, Nickel y Ríos- Mercado	Centros de acopio para reciclado	GRASP
2008	Ricca, F., Scozzari, A. y Simeone, B.	Zonificación electoral	Diagramas de Vonoroi combinadas con técnicas de búsqueda local
2009	Ríos-Mercado y Fernández	Zonas de ventas	GRASP
2009	Rincón García y Gutiérrez Andrade	Zonas electorales	Recocido simulado
2009	Bernábe *et al*	Regionalización	Algoritmo de Conglomerados y recocido simulado
2011	Rincón-García *et al*	Zonas electorales	Recocido simulado y búsqueda de entornos variables
2012	Rincón-García *et al*	Zonas electorales	PSO-discretizado

7.1.3. Diseño de zonas electorales

Como se mecionó con anterioridad en este trabajo, se estudia y analiza la aplicación del diseño de zonas para la generación de áreas electorales; a esta aplicación se le denomina diseño de zonas electorales (DZE). Este problema es de gran importancia para las naciones democraticas; ya que la zonificación de un lugar puede influir en los resultados que se obtengan de una elección (véase [203, 69, 25])

El DZE busca garantizar un proceso electoral democrático, a través de ponderar distintos criterios, siendo los más usuales en la literatura consultada las condiciones de: equilibrio poblacional, conexidad y compacidad geométrica [204]. Generalmente, se incluyen los elementos de compacidad geométrica y equilibrio poblacional en la función objetivo del problema; mientras, que la conexidad conforma las restricciones del problema.

En la literatura no existe un concenso sobre la forma de determinar el equilibrio poblacional y la compacidad geométrica. En [203, 172], se revisan varios de los mecanismos disponibles para deteminar equilibrio poblacional y compacidad geométrica.

En este trabajo, se determina el equilibrio, a través de la ecuación 7.1.1, la cual fue utilizada por el IFE para la distritación del 2004 [55].

$$C_1(P) = \sum_{s \in S} \left(\frac{100 P_T}{d\left(\frac{P_N}{300}\right)} \right)^2 \left(\frac{P_s}{P_T} - \frac{1}{n} \right)^2 \tag{7.1.1}$$

donde: $C_1\,(P)$ es el costo de equilibrio poblacional asociado al plan de zonificación P.

P_N población nacional.

P_T población de la entidad.

P_s población de la zona s .

d porcentaje de desviación poblacional máxima aceptable por zona.

$S = \{1, 2, 3, ..., n\}$ es el número de zonas electorales que se deben generar en la entidad.

n número de zonas.

En este trabajo, se incluye también un conjunto de restricciones sobre el equilibrio poblacional de las zona, el cual se puede consultar en [205] ecuación 7.1.2.

$$\left(\frac{100 P_T}{d\left(\frac{P_N}{300}\right)} \right)^2 \left(\frac{P_s}{P_T} - \frac{1}{n} \right)^2 - 1 \leq 0 \ \forall s \in S \tag{7.1.2}$$

Para determinar la compacidad geométrica se utilizó la ecuación 7.1.3, La cual es la utilizada por el IFE para dichos propositos.

$$C_2(P) = \sum_{s \in S} \left(\frac{PC_s}{4\left((AC_s)^{0.5}\right)} \right) \tag{7.1.3}$$

$$\tag{7.1.4}$$

donde: PC_s es el perímetro de la zona s.

AC_s es el área de la zona s.

Con base en los elementos de equilibrio poblacional y compacidad geométrica se estructura la función objetivo del problema, tal y como lo muestra la ecuación 7.1.5

$$\text{mín}\, C(P) = \alpha_1 C_1(P) + \alpha_2 C_2(P) \tag{7.1.5}$$

donde: $Z_s = \{i : x_{is} = 1\}$ es el conjunto de unidades geográficas que forman la zona.

$P = \{Z_1, Z_2, Z_3, \ldots, Z_n\}$ es un plan de zonificación.

α_1 y α_2 son factores de ponderación. $\alpha_1 \in [0, 1]$ y $\alpha_2 = 1 - \alpha_1$.

La conexidad necesaria en las zonas, se incluyen como restricciones del problema; este modelo fue propuesto por Shirabe en [224], donde se establece una analogía entre conexidad de una zona y el fluido desde múltiples fuentes hasta un sumidero a partir de lo cual Shrirabe construyó las ecuaciones 7.1.6.

$$\begin{gathered}
\textstyle\sum_{j:(i,j)\in A} y_{ijs} - \sum_{j:(i,j)\in A} y_{jis} \geq x_{is} - M w_{is}\ \forall i \in I\ \forall s \in S \\
\textstyle\sum_{i \in I} w_{is} = 1 \forall s \in S \\
\textstyle\sum_{j:(i,j)\in A} y_{ijs} \leq (M-1) x_{is}\ \forall i \in I\ \forall s \in S \\
x_{is} \in \{0,1\}\ \forall i \in I\ \forall s \in S \\
w_{is} \in \{0,1\}\ \forall i \in I\ \forall s \in S \\
y_{ijs} \geq 0 \forall (i,j) \in A
\end{gathered} \tag{7.1.6}$$

Donde: $I = 1, 2, 3, \ldots, n$ n es el nÃºmero de UGB consideradas en el problema.

$R = 1, 2, 3, \ldots, r$ r es el nÃºmero de zonas que se deben generar.

$w_{is} = \begin{cases} 1 & \text{si la UGB i de la zona s es un sumidero} i \in I\ s \in R \\ 0 & enotrocaso \end{cases}$

$A = (i, j)$: las UGB i, j son contiguas

y_{ijs} es una variable continua no negativa que representa la cantidad de flujo desde UGB i hasta la UGB j en la zona s. M es un número mayor que la cantidad de variables asociadas a las UGB.

$$x_{is} = \begin{cases} 1 & \text{si la UGB i pertenece a la zona si} \in I \; s \in R \\ 0 & enotrocaso \end{cases}$$

Con base a lo anterior, al conjuntar 7.1.5 y 7.1.6, se construye el modelo mostrado en la ecuación 7.1.7.

$$\begin{gathered} \text{mín}\, C(P) = \alpha_1 C_1(P) + \alpha_2 C_2(P) \\ \text{sujeto a:} \\ \textstyle\sum_{i=1}^{n} x_{is} \geq 1 \forall \; s \in S \\ \textstyle\sum_{s=1}^{r} x_{is} = 1 \; \forall i \in I \\ \textstyle\sum_{j:(i,j)\in A} y_{ijs} - \sum_{j:(i,j)\in A} y_{jis} \geq x_{is} - M w_{is} \; \forall i \in I \; \forall s \in S \\ \textstyle\sum_{i\in I} w_{is} = 1 \forall s \in S \\ \textstyle\sum_{j:(i,j)\in A} y_{ijs} \leq (M-1) x_{is} \; \forall i \in I \; \forall s \in S \\ x_{is} \in \{0,1\} \; \forall i \in I \; \forall s \in S \\ w_{is} \in \{0,1\} \; \forall i \in I \; \forall s \in S \\ y_{ijs} \geq 0 \forall (i,j) \in A \end{gathered} \tag{7.1.7}$$

En este modelo la función objetivo buscará minimizar la desviación poblacional y maximizar la compacidad geométrica [204]. Mientras que al satisfacer las restricciones se grarantiza que las zonas generadas sean conexas.

Al conjuntar los modelos 7.1.2 y 7.1.7 se genera el siguiente modelo:

$$\begin{gathered} \text{mín}\, C(P) = \alpha_1 C_1(P) + \alpha_2 C_2(P) \\ \text{sujeto a:} \\ \textstyle\sum_{i=1}^{n} x_{is} \geq 1 \forall \;\; s \in S \\ \textstyle\sum_{s=1}^{r} x_{is} = 1 \;\; \forall i \in I \\ \textstyle\sum_{j:(i,j)\in A} y_{ijs} - \sum_{j:(i,j)\in A} y_{jis} \geq x_{is} - M w_{is} \;\; \forall i \in I \; \forall s \in S \\ \textstyle\sum_{i\in I} w_{is} = 1 \forall s \in S \\ \textstyle\sum_{j:(i,j)\in A} y_{ijs} \leq (M-1) x_{is} \; \forall i \in I \; \forall s \in S \\ \left(\frac{100 P_T}{d\left(\frac{P_N}{300}\right)}\right)^2 \left(\frac{P_s}{P_T} - \frac{1}{n}\right)^2 - 1 \leq 0 \; \forall s \in S \\ x_{is} \in \{0,1\} \;\; \forall i \in I \; \forall s \in S \\ w_{is} \in \{0,1\} \;\; \forall i \in I \;\; \forall s \in S \\ y_{ijs} \geq 0 \forall (i,j) \in A \end{gathered} \tag{7.1.8}$$

7.1.4. Lineamientos para la construcción de los distritos electorales en México

El artículo 53 de la constitución política de los Estados Unidos Méxicanos establece que se divide en 300 distritos electorales y se establece la forma para determinar el número de distritos para cada uno de los estados.

En el acuerdo CG-104-2004 del Consejo General del IFE [55], se establece que para los estudios y proyectos conduncentes para distritación de la República Mexicana se deben tener en cuenta los siguientes criterios y consideraciones operativas.

1 Los distritos se integrarán con territorio de una sola entidad federativa.

2 Para la determinación del número de distritos para cada entidad federativa, se seguiráa lo dispuesto en el artículo 53 de la Constitución Política de los Estados Unidos Mexicanos.

3 Se aplicará el equilibrio demográfico en la determinación de los distritos partiendo de la premisa de que la diferencia de población de cada distrito, en relación con la media poblacional debe ser lo más cercano a cero.

4 Se debe procurar en la conformación de distritos electorales con mayoría de población indígena. En todo caso se preservará la integridad territorial de las comunidades indígenas.

5 Los distritos tendrán continuidad geográfica tomando en consideración los límites político-administrativos y los accidentes geográficos.

6 En la delimitación de los distritos se procurará obtener la mayor compacidad, de tal forma que el perímetro de los distritos tenga una forma geométrica lo más cercana a un polígono regular. Ningún distrito podrá rodear integramente a otro.

7 Para la integración de distritos se utilizará la distribución municipal y seccional vigentes. La unidad de agregación mínima será la sección.

8 Los distritos se constituirán preferentemente con municipios completos.

9 Para establecer las cabeceras distritales se considerarán los siguientes parámetros: mayor población, vías de comunicación y servicios públicos. En caso de existir dos o más localidades semejantes, y una de ellas sea, en la actualidad, cabecera distrital, prevalecerá esta última.

10 En la conformación de los distritos, se procurará optimizar los tiempos de traslado entre los recorridos a su interior, considerando su tamaño, su extensión y la distribución geográfica de sus localidades.

Los criterios señalados anteriormente se dividen jerárquicamente, de la siguiente manera: primer nivel (criterios del 1 al 3); segundo nivel (criterio 4); tercer nivel (criterios 5 y 6); cuarto nivel (criterio 7); quinto nivel (criterio 8); sexto nivel (criterio 9) y séptimo nivel (criterio 10.)

Adicionalmente, en el CG-104-2004 se establece que el IFE utilizará para la construcción de los distritos electorales un algoritmo de optimización basado en recocido simulado; el cual, se construyó tras considerar los siguientes criterios:

- La ponderación de las variables que integren la función de costo del mencionado algoritmo, contendrá factores de valoración de acuerdo a la jeraquía de los criterios enunciados.
- El algoritmo contemplará los criterios de equilibrio poblacional, compacidad geográfica, reducción de tiempos de traslado y barreras geográficas.
- En todo caso, la construcción del algoritmo, función de costo y programación complementaria, será hecha del conocimiento de los representantes de los partidos políticos ante el IFE;
- Aquellas variables que no sea posible integrar en el algoritmo, serán consideradas en forma externa durante la etapa de análisis y ajustes

En [203, 206, 205] se utiliza la técnica de recocido simulado para la distritación de algunos estados de la República Mexicana. A continuación, se muestran las adaptaciones realizadas al algoritmo propuesto a fin de resolver instancias del DZE.

7.2. Adaptación del método del MMC para el problema DZE

Dada la naturaleza discreta del problema DZE, se realizaron modificaciones al MMC con el objeto de poder trabajar problemas combinatorios.

7.2.1. Melodía y función de satisfacción

Una melodía para resolver el problema DZE, se construye como se muestra en la ecuación 7.2.1.

$$\begin{aligned} x_{i,q,\star} &= [x_{i,q,1}, x_{i,q,2}, \ldots, x_{i,q,N_{UB}}] \\ x_{i,q,l} &\in \{1, N_Z\} \\ x_{i,q,l} &\in \mathbb{Z}^+ \end{aligned} \tag{7.2.1}$$

donde:

$x_{i,q,\star}$ es la $q-$ésima melodía del $i-$ésimo compositor

N_{UB} es el número de unidades geográficas básicas en la región.

$x_{i,q,l}$ es la $l-$ésima unidad geográfica

N_Z es el número de zonas en la región.

Por otro lado, en la construcción de la función de satisfacción se utilizaron las reglas propuestas por Coello y Lamda [126] con las modificaciones utilizadas para resolver el problema restricto no lineal (veáse capítulo 6).

7.2.2. Modificaciones al algoritmo MMC

Se realizaron las siguientes modificaciones en el algoritmo MMC; para solucionar el problema DZE.

- En la fase de **inicializar el algoritmo**, se incluyeron mecanismos para el manejo de información de la instancia a resolver (número de unidades básicas totales en el estado; número total de zonas; número de procesos [4]; número de zonas por proceso, número y ubicación de las unidades básicas incluidas en cada uno de los procesos). Debido a los procesos y a que un estado se debe distritar con máx $\_arrangement$ arreglos, se utilizó la ecuación 7.2.2 a fin de determinar el número de arreglos para cada uno de los procesos.

$$iteraciones = \begin{cases} \text{Si } up_i > 1 & \lceil (\text{máx}\_arrangement - pcuu) * \frac{N_{Z_p}}{N_Z} \rceil \\ \text{Si } up_i = 1 & 1 \end{cases} \tag{7.2.2}$$

  donde: up_i número de distritos (zonas) en el proceso i; $pcuu$ número de procesos que contienen una sola zona; N_{Z_p} número de zonas en el proceso y N_Z número total de zonas en el estado.

  Adicionalmente, en la subfase de generación de las partituras iniciales, se incluyó un método basado en la preservación de la factibilidad de la solución, a fin de volver conexa una solución no conexa. En cada paso de este método, se seleccionó una unidad básica, para moverla a una zona adyacente hasta alcanzar zonas conexas; las cuales, satisfacen las restricciones mostradas en la ecuación 7.1.6. Para ello, el MMC utiliza la rutina mostrada en el algoritmo 52

  Cabe resaltar, que la rutina 52, no busca satisfacer las restricciones 7.1.2; por lo cual, se generan soluciones conexas que no necesariamente satisfagas dichas inecuaciones. Para manejar estas restricciones

[4] Un proceso es una subregión del estado, la cual se debe distritar de forma independiente; es decir cada proceso representa un problema de DZE independiente

Algoritmo 52: Factibilización de una solución infactible

```
1 while x_{i,new,l} sea no conexa do
2 |   Seleccionar una unidad básica de su zona actual
3 |   Mover la unidad seleccionada a una zona adyacente a ella.
4 end
```

se utilizó la misma estrategia que la usada para las instancias restrictas del problema no lineal.

En la determinación del grado de satisfacción alcanzada por cada una de las melodías se implicaron las ecuaciones 6.2.7,6.2.9 y 6.2.11 utilizadas en el capítulo anterior para el manejo de restricciones.

Por último, se utilizó la estrategia auto-adaptativa del parámetro $fcal$, explicado en el capítulo 6

- En la fase de **interacción entre agentes**, se utilizaron dos políticas para la adquisición de conocimiento. La primera política utilizada fue la propuesta originalmente para el MMC, la cual establece: "el $i-$ésimo compositor adquirirá información de otro compositor k, si y sólo si la peor melodía en $P_{i,\star,\star}$ es peor que la peor melodía en $P_{k,\star,\star}$". Al algoritmo que utilizo esta política se le denominó MMC-o (veáse el algoritmo 53).

  En contraste, la segunda política utilizó el criterio de metrópolis para decidir; el $i-$ compositor adquirirá información de otro compositor k, a través de la rutina mostrada en el algoritmo 54. Se le denominó MMC-v a la versión del algoritmo que uso esta política.

- En la fase de **generación de una nueva melodía**, se utilizó el algoritmo 55 para **construir una nueva melodía**.

  En el algoritmo 56, se muestra como el $i-$ésimo compositor selecciona los elementos base para generar una nueva melodía. Este algoritmo, implica la idea de solución dominada - generalmente, se dice que una solución A es dominada por otra solución B, si A satisface igual o menor número de restricciones que B-. El $fitness(KM_{i,\star,\star})$ de una melodía se determina a través de la misma manera que lo realizado para el caso restricto del problema no lineal.

- Para la **actualización de la partitura** se utilizó la rutina mostrada en el algoritmo 57

En resumen el MMC se modifico en las siguientes fases: inicializar el algoritmo; interacción entre agentes; generación de una melodía y actualización de una partitura para la solución de instancias del problema DZE.

Algoritmo 53: Política para la adquisición de conocimiento MMC-o.

```
for i = 1 : 1 : Nc do
    x_{i-worst} ← de la melodía con peor aptitud en P_{i,⋆,⋆}
    for i = 1 : 1 : Nc do
        if Existe un vínculo entre los compositores i y k then
            x_{k-worst} ← de la melodía con peor aptitud en P_{k,⋆,⋆}
            if fitness(x_{i-worst}) es peor que fitness(x_{k-worst}) then
                El i− compositor adquiere información del k−ésimo compositor
            end
        end
    end
end
```

Algoritmo 54: Política para la adquisición de conocimiento MMC-v.

```
x_{i-worst} ← de la melodía con peor aptitud en P_{i,⋆,⋆}
x_{k-worst} ← de la melodía con peor aptitud en P_{k,⋆,⋆}
a_2 = exp^{(fitness(x_{i-worst}) − fitness(x_{k-worst})) / fitness(x_{k-worst})}
if fitness(x_{i-worst}) es peor que fitness(x_{k-worst}) then
    El i− compositor adquiere información del k−ésimo compositor
else
    if rand < a_2 then
        El i− compositor adquire información del k−ésimo compositor
    end
end
```

Algoritmo 55: Generar una nueva melodía

```
for i = 1 : 1 : Nc do
    if rand < (1 − ifg) then
        Seleccionar las melodías base, por medio, del algoritmo 56 (Melodía base_{⋆,⋆} ).
        for l = 1 : 1 : n do
            if rand < (1 − cfg) then
                a_1 ← rand
                if a_1 < 1/3 then
                    x_{i,new,l} = Melodía base_{1,l}
                else
                    if a_1 < 2/3 then
                        x_{i,new,l} = Melodía base_{2,l}
                    else
                        x_{i,new,l} = Melodía base_{3,l}
                    end
                end
            else
                x_{i,new,l} = 1 + rand ∗ (N_{Z_p} − 1).
            end
        end
    else
        Generar x_{i,new,l} aleatoriamente
    end
    if x_{i,new,l} es infactible then
        Emplear el algoritmo 52
    end
    Evaluar la solución encontrada
end
```

Algoritmo 56: Selección de las melodías base

Input: $KM_{i,\star,\star}$,Nc, mecanismo a través del cual se selecciona la tercera melodía base (cof, este parámetro sólo toma valores enteros)

Output: melodías base para generar una nueva melodía

for $i = 1 : 1 : Nc$ **do**
  $KM_{i,best,\star} \leftarrow$ es la melodía con mejor valor de $fitness(KM_{i,\star,\star})$ en $KM_{i,\star,\star}$.
  $Melodía\ base_{1,\star} \leftarrow KM_{i,best,\star}$.
  Eliminar $KM_{i,best,\star}$ de $KM_{i,\star,\star}$.
  $KM'_{i,\star,\star} = \emptyset$
  if $KM_{i,best,\star}$ *viola alguna restricción de la instancia a resolver* **then**
    Eliminar todas las melodías de $KM_{i,\star,\star}$ dominadas por $KM_{i,best,\star}$.
  end
  Recalcular la aptitud de los elementos de $KM_{i,\star,\star}$.
  $Melodía\ base_{2,\star} \leftarrow$ melodía seleccionada, probabílisticamente, de $KM_{i,\star,\star}$.
  if $cof_i = 1$ **then**
    $Melodía\ base_{3,\star} \leftarrow$ seleccionar aleatoriamente un valor para cada motivo contenido en $KM_{i,\star,\star}$.
  else
    $Melodía\ base_{3,\star} \leftarrow$ es una melodía generada por los promedios de los valores para cada motivo en $KM_{i,\star,\star}$.
  end
end

Algoritmo 57: Actualizar partitura

```
1  x_{i-worst} ← de la melodía con peor aptitud en P_{i,⋆,⋆}
2  . if C(x_{i,new,⋆}) ≤ C(x_{i-worst}) then
3      if C(x_{i,new,⋆}) < Cx_{i-worst}) then
4          Reemplazar x_{i-worst} por x_{i,new,⋆} dentro de P_{i,⋆,⋆}.
5      else
6          if φ(x_{i,new,⋆}) ≤ φ(x_{i-worst}) then
7              if φ(x_{i,new,⋆}) < φ(x_{i-worst}) then
8                  Reemplazar x_{i-worst} por x_{i,new,⋆} dentro de P_{i,⋆,⋆}.
9              else
10                 if diff(x_{i,new,⋆}) < diff(x_{i-worst}) then
11                     Reemplazar x_{i-worst} por x_{i,new,⋆} dentro de P_{i,⋆,⋆}.
12                 end
13             end
14         end
15     end
16 end
```

7.3. Experimentación

7.3.1. Caso de estudio

7.3.1.1. Generalidades del Estado de México

En esta sección se análizan las características consideradas en la distritación del Estado de México; ya que es el caso de estudio.

El Estado de México se localiza en la zona central de la República; en la parte oriental de la meseta de Anáhuac. Colinda al norte con los estados de Querétaro e Hidalgo; y al sur con Guerrero y Morelos; al este con Puebla y Tlaxcala; y al oeste con Guerrero y Michoacán, así como con el Distrito Federal, al que rodea al norte, este y oeste [54].

El Estado de México tiene una superficie de 22,356.80 km^2 [108], y de acuerdo al censo del 2010 la problación total del estado es de 15,175,862 habitantes; de los cuales, 7,778,876 mujeres y el resto hombres [107, 108]. El Edo. Mex. se conforma por 4,786 localidades; de las cuales, un 35.2 por ciento son pequeñas, con menos de 99 habitantes, distribuidas en 122 municipios .

7.3.1.2. Elementos de distritación Estado de México

En el Estado de México existen 836 unidades geográficas (véase Figura 7.5) a partir de las cuales se deben generar 40 distritos electorales agrupados en trece procesos, tal y como lo muestra la tabla 7.8. Se puede ver a cada proceso como un problema de zonificación independiente de los otros procesos.

7.3.2. Diseño de experimentos

Para la distritación del Estado de México se utilizaron los datos ocupados por el IFE, para la generación de zonas electorales en el 2004. Estos datos se muestran en la tabla 7.9.

Los valores de α_1 utilizados fueron $0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8$ y 0.9 y se realizaron 10 pruebas independientes para cada valor de α_1. Considerando los datos anteriores, se implementaron el MMC-o y el MMC-v para generar una distritación del Estado de México. Posteriormente, se deteminó la mejor, la peor y la solución promedio para cada uno de los valores de α_1. Después, las soluciones obtenidas se filtrarón a fin de establecer el conjunto de soluciones no dominadas, en términos generales, se dice que una solución domina a otra si es mejor o igual en todos los objetivos y mejor en al menos uno de ellos. En la definición 72, se formaliza el concepto de solucion dominada

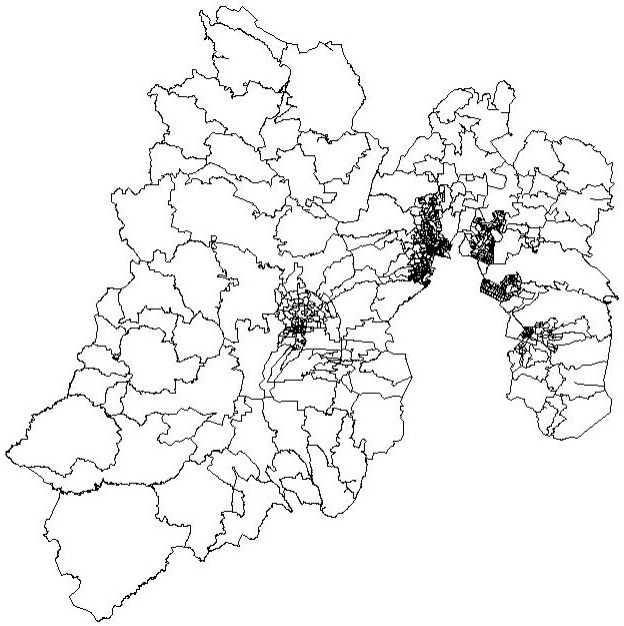

Figura 7.5: Unidades geográficas del Estado de México

Definición 72 *Se dice que un vector solución a domina a otro b , denotado como* $a\pi = b$*, si y sólo si*

$$\forall i \in \{1, 2, \ldots, n\} | f_i(a) \geq f_i(b) \wedge \exists j \in \{1, 2, \ldots, n\} f_i(a) > f_j(b)$$

Debido a la naturaleza multiobjetivo del problema, se determinó la calidad de los conjuntos de soluciones no dominadas generadas por cada solución, identificando el número de soluciones de cada algoritmo que están en el frente de Pareto y la distancia generacional de los algoritmos MMC-o el MMC-v y recocido simulado. La distancia genaracional se determinó mediante la ecuación 7.3.1

$$G = \frac{\sqrt{\Sigma_{i=1}^{n} d_i^2}}{n} \tag{7.3.1}$$

Además, se determinó la métrica de conjunto de cubierto a través de la ecuación 7.3.2

$$C(A_1, A_2) = \frac{|s_2 \in \mathcal{PF}_2; \exists s_1 \in \mathcal{PF}_1 : s_1 \succ s_2|}{|\mathcal{PF}_2|} \tag{7.3.2}$$

donde: $\mathcal{PF}_1$ y $\mathcal{PF}_2$ son los conjuntos de soluciones no dominadas obtenidos por los algoritmos de A_1 y A_2, respectivamente. Si $C(A_1, A_2)$ es 1, indica que toda solución obtenida por A_2 está dominada por soluciones producidas por A_1; en contraste si $C(A_1, A_2)$ es 0 indica que ninguna solución obtenida porA_2 está dominada por soluciones producidas por A_1

Tabla 7.8: Procesos y zonas del Edo. Mex.

Proceso	Número de Zonas
1	13
2	1
3	2
4	2
5	5
6	3
7	4
8	3
9	2
10	1
11	1
12	1
13	2

Tabla 7.9: Datos para la zonificación del Edo. Mex.

Factor	Valor
Población nacional	97,483,412
Población estatal	13,096,686
Número de distritos	40
Número de unidades geográficas	836
Porcentaje de desviación aceptado	15 %
Factores de ponderación	α_1 y α_2

7.3.3. Configuración de parámetros

La configuración de los parámetros del MMC-o y el MMC-v se realizó a través del método de fuerza bruta, (veáse F). Cada variante se calibro de manera independiente utilizando 15 pruebas, sin embargo

como se observa en la Tabla 7.10 los resultados fuerón similares. Para esta actividad se utilizaron los valores extemos de α_1; la configuración de parámetros encontrada, para estas estrategias, se muestra en la Tabla 7.10.

Tabla 7.10: Configuración de parámetros

	Estrategia	
Parámetro	MMC-o	MMC-v
máx_$arrangement$	20000	20000
ifg	0.001	0.001
cfg	0.05	0.05
Nc	5	5
Ns	7	7

7.3.4. Resultados Numéricos

Los algoritmos MMC-o y MMC-v se implementaron en Matlab 7.10.0, y se ejecutaron en una computadora MacBookAir.

En las tablas 7.11 y 7.12, se muestran la mejor, la peor y la solución promedio encontrada por el MMC-o y el MMC-v para cada uno de los valores de α_1 [5]. Adicionalmente, en las Tablas 7.13 y 7.14 se muestran los conjuntos de soluciones no dominadas generadas por el MMC-o y el MMC-v, respectivamente.

[5]Recuerde que $\alpha_2 = 1 - \alpha_1$

Tabla 7.11: Resultados obtenidos con el MMC-o

Valor de α_1	Peor solución			Solución promedio			Mejor solución		
	Equilibrio poblacional	Compacidad	Función objetivo	Equilibrio poblacional	Compacidad	Función objetivo	Equilibrio poblacional	Compacidad	Función objetivo
0.2	13.420	66.234	79.653	12.936	64.812	77.748	12.182	63.936	76.118
0.3	14.115	65.493	79.608	13.540	64.541	78.081	12.492	63.697	76.189
0.4	15.098	65.078	80.176	12.616	65.142	77.759	11.811	64.222	76.033
0.5	14.122	66.758	80.880	12.239	66.356	78.595	10.957	65.732	76.689
0.6	13.473	67.277	80.749	12.493	66.325	78.818	11.995	64.176	76.171
0.7	12.762	68.032	80.794	11.894	66.919	78.813	10.769	65.854	76.623
0.8	12.754	68.174	80.928	11.353	67.901	79.253	11.190	65.877	77.066
0.9	11.626	70.208	81.834	11.549	68.170	79.719	11.577	66.660	78.237

Tabla 7.12: Resultados obtenidos con el MMC-v

Valor de α_1	Peor solución			Solución promedio			Mejor solución		
	Equilibrio poblacional	Compacidad	Función objetivo	Equilibrio poblacional	Compacidad	Función objetivo	Equilibrio poblacional	Compacidad	Función objetivo
0.2	13.600	66.253	79.853	13.569	65.496	79.064	13.351	65.307	78.658
0.3	16.898	65.114	82.012	13.423	65.354	78.777	12.291	64.409	76.700
0.4	13.936	66.253	80.189	12.414	65.353	77.767	11.842	63.632	75.474
0.5	13.358	67.651	81.009	12.340	66.598	78.938	11.119	64.345	75.464
0.6	13.254	67.235	80.489	12.044	66.482	78.526	11.111	65.612	76.723
0.7	12.087	69.525	81.612	11.869	67.015	78.884	10.694	66.125	76.819
0.8	11.797	69.780	81.576	12.272	67.397	79.669	11.724	64.810	76.534
0.9	11.823	70.333	82.156	11.185	69.257	80.442	11.616	66.499	78.115

Cabe mencionar que una solución factible x está dominada por una segunda solución factible x' si y solo si $f(x') \leq f(x)$

Tabla 7.13: Soluciones no dominadas obtenidas con el MMC-o

Compacidad	Equilibrio poblacional	Función objetivo
63.697	12.492	76.189
63.982	12.48	76.462
64.176	11.995	76.171
64.222	11.811	76.033
65.732	10.957	76.689
65.854	10.769	76.623
68.633	10.558	79.191
68.849	10.324	79.173
70.078	10.095	80.174

Tabla 7.14: Soluciones no dominadas obtenidas con el MMC-v

Compacidad	Equilibrio poblacional	Función objetivo
63.632	11.842	75.474
64.345	11.119	75.464
65.612	11.111	76.723
65.747	11.082	76.829
65.952	10.989	76.941
66.125	10.694	76.819
69.22	10.653	79.874
69.58	10.352	79.932
70.783	10.348	81.132

En la tabla 7.15 se muestra el tiempo de ejecución empleado por el MMC-o y el MMC-v en la distritación del Edo. Mex. con los diferentes valores de α_1.

Tabla 7.15: Tiempos de ejecución

	MMC-o			MMC-v		
α	Tiempo promedio	Varianza	Desviación estándar	Tiempo promedio	Varianza	Desviación estándar
0.2	6510.566667	223078.1987	472.3115483	6832.8	670847.656	819.0529018
0.3	9394.525	1743686.576	1320.487249	6958.325	333128.9421	577.1732341
0.4	7515.322222	294491.9719	542.6711453	7019.7125	701089.627	837.31095
0.5	8058.4375	2475373.628	1573.332015	7365.014286	370376.5981	608.585736
0.6	6926.47	335141.5334	578.9140985	6667.7	378725.32	615.4066298
0.7	7230.344444	1262620.84	1123.664025	6845.8375	219276.7827	468.2699891
0.8	6876.26	388844.1493	623.573692	7115.544444	474063.3928	688.522616
0.9	6624.325	782826.0593	884.7745811	7911.25	1141940.837	1068.616319

En la Figura 7.6, se muestra un comparativo entre los conjuntos de soluciones no dominadas obtenidas por los algoritmos MMC-o, MMC-v y SA. En las tabla 7.16 se muestra el conjunto de soluciones no dominadas, de los resultados comparados.

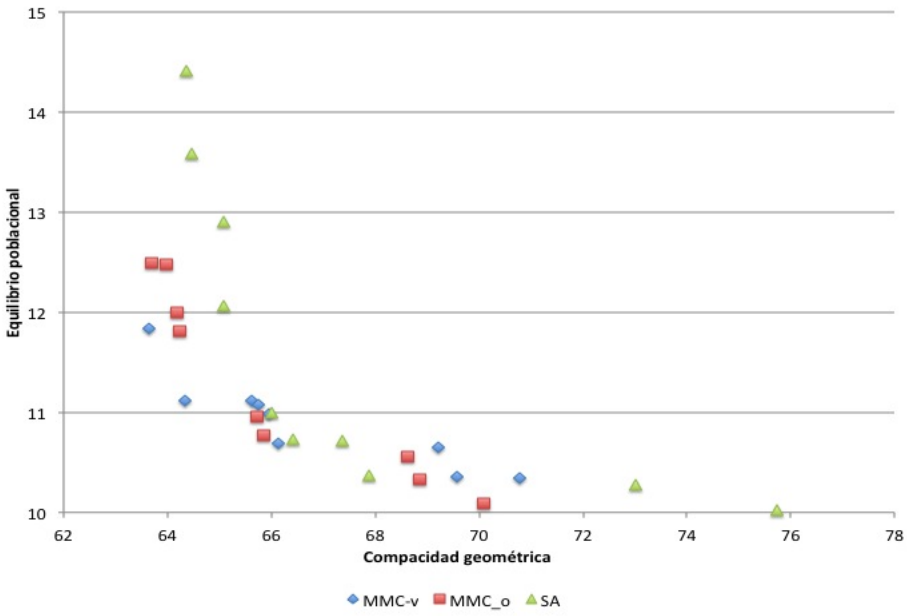

Figura 7.6: Comparativo entre el conjunto de soluciones no dominadas

En la tabla 7.16, se muestra que tanto el MMC-v como el MMC-o generan 4 de las once soluciones que conforman el conjunto de soluciones no dominadas; mientras que el SA sólo aporta 3. En la tabla 7.17 se muestran los resultados de calcular la cobertura de conjunto sobre los algoritmos comparados.

7.4. Análisis de resultados

Los resultados obtenidos (véase Tabla 7.17) muestran la capacidad del MMC (MMC-o y MMC-v) para generar buenas soluciones en la distritación del estado, ya que el MMC-o domina en 72 % a las soluciones generadas por SA; mientras que el MMC-v domina el 63 % a las soluciones generadas por SA.

También se puede decir, con base a la información obtenida, que los procedimientos MMC-o y MMC-v tienen comportamientos distintos; siendo la MMC-o la variante que ofrece mejores resultados, ya que éste domina al MMC-v en 44 % de las soluciones.

Tabla 7.16: Soluciones no dominadas

Compacidad	Equilibrio poblacional	Función objetivo	Algoritmo
63.632	11.842	75.474	MMC-v
64.345	11.119	75.464	MMC-v
65.612	11.111	76.723	MMC-v
65.732	10.957	76.689	MMC-o
65.854	10.769	76.623	MMC-o
66.125	10.694	76.819	MMC-v
67.889	10.374	78.263	SA
68.849	10.324	79.173	MMC-o
70.078	10.095	80.174	MMC-o
75.744	10.024	85.767	SA
76.088	9.977	86.064525	SA

Tabla 7.17: Cobertura de conjuntos

		C_2		
		MMC-o	MMC-v	SA
C_1	MMC-o		0.333	0.727
	MMC-v	0.444		0.636
	SA	0.111	0.111	

Parte IV

Conclusiones y trabajos futuros

Capítulo 8

Conclusiones y trabajos futuros

En este trabajo se diseñó y desarrolló una nueva metaheurística denominada Método de Composición Musical (MMC); la cual, imita el comportamiento creativo del ser humano en un sistema socio-cultural, involucrado en la generación de nuevas obras musicales (véase capítulo 4). El algoritmo MMC es por lo tanto una metaheurística social, la cual posee tres escalas de tiempo que son: a) **primera escala cambios rápidos**. En esta escala cada compositor genera un cambio en su partitura tratando de alcanzar su mayor grado de satisfacción, dicho cambio puede implicar una modificación en el metapatrón de las características de las melodías que integran la partitura, b) **segunda escala, cambios medios**. En esta escala emerge un metapatrón con las características comunes de las obras generadas localmente, además, en ella los compositores pueden modificar su red social y c) **tercera escala cambios lentos**. En esta escala surge un sistema cultural que afecta el comportamiento de los agentes dentro de la sociedad.

La estructura general del MMC es la siguiente: Inicialmente se genera una sociedad artificial de Nc compositores y se definen las reglas de interacción entre los agentes que la conforman. Entonces, para cada uno de los compositores en la sociedad, aleatoriamente, se crea un conjunto de Ns temas que se registran en la partitura asociada a ese compositor($P_{\star,\star,i}$) -la partitura servirá como la memoria del compositor- Posteriormente y hasta satisfacer el criterio de paro, se realiza lo siguiente: a) se actualizan los vínculos entre los compositores de la sociedad; b) los compositores interactúan entre sí; cada uno de ellos analiza la información recibida de los demás compositores y selecciona los datos que tomará de su entorno, a este conjunto de datos se le denomina ideas adquiridas socialmente ($ISC_{\star,\star,i}$); c) el $i-$ésimo compositor construye su conocimiento ($KM_{\star,\star,i}$), con base en su $P_{\star,\star,i}$ y las ideas adquiridas socialmente; en términos

generales, $KM_{\star,\star,i} = P_{\star,\star,i} \cup ISC_{\star,\star,i}$; d) cada compositor genera una nueva melodía ($x_{\star,new}$), a partir de su conocimiento y a los posibles destellos de genialidad; después, este compositor determina el grado de satisfacción alcanzado con $x_{\star,new}$ y e) finalmente, el $i-$ésimo compositor, con base en el grado de satisfacción, debe decidir si $x_{\star,new}$ remplazará a algún elemento de su partitura.

Posteriormente, se adaptó sucesivamente el MMC para evaluar su desempeño sobre varias clases de problemas de optimización: la resolución de instancias de programación no-lineal irrestricta se presentó en el capítulo 1; mientras que los resultados obtenidos sobre instancias de optimización no-lineal restricta y de optimización combinatoria (diseño de zonas electorales) se comentaron en los capítulos 6 y 7, respectivamente. El análisis de resultados obtenidos por el MMC y su posterior comparación contra los resultados obtenidos por otras mataheurísticas mostraron lo siguiente:

- El MMC resolvió eficazmente las instancias referenciales de los problemas PLN (restricto e irrestricto) y DZE.
- En el problema PLN irrestricto, los resultados obtenidos por el MMC superaron a los obtenidos por las heurísticas comparadas en los casos de intancias multimodales rotadas. Cabe resaltar que este resultado no se debe al carácter estocástico del algoritmo, ya que las pruebas de Wilcoxon realizadas confirman que existe, para estas instancias, una superioridad significativa del desempeño del MMC sobre sus competidores.
- La adaptación del MMC para resolver el PLN restricto implicó el diseño y desarrollo de estrategias para el manejo de restricciones. Los resultados promedio obtenidos por el MMC superaron a los resultados promedios obtenidos por las heurísticas comparadas en las instancias G5 y G8. Estas instancias involucran funciones tracendentes en la función objetivo o bien en las restricciones. Además, los resultados de la prueba de Wilcoxon muestran que el MMC tiene un comportamiento similar al de otras heurśticas comparadas.
- En el problema de diseño de zonas electorales para una instancia específica (Estado de México) el MMC fue capaz de generar un amplico conjunto de soluciones no dominan, las cuales dominan alrededor del 70 % las soluciones generadas por SA, en contraste SA domina sólo en un 11 % a las soluciones generadas por el MMC.

Con base a lo anterior, se puede afirmar que el MMC es una alternativa competitiva y eficiente para resolver instancias de los problemas PLN (restricto e irrestricto) y DZE.

Como productos de la investigación doctoral se obtuvieron, además del presente documento de tesis, publicaciones ([156], [157]) y presentaciones en congresos [160].

Finalmente, como perspectivas de apliación del trabajo presentado en este documento existen algunas de las oportunidades de mejoras y futuros trabajos, como por ejemplo:

- Implementar una calibración más fina y rigurosa de los parámetros de operación del algoritmo, basada en técnicas de diseño de experimentos aplicadas sobre instancias consideradas como críticas para la resolución. Por ende, un trabajo futuro emanado de esta investigación es utilzar de parámetros auto-adaptativos en la solución de problemas.
- Diseñar nuevas políticas de interacción entre los agentes (alternas a la utilizada en adaptaciones del MMC) y sus implicaciones en la solución de problemas; por ejemplo, usar el valor promedio de la función de satisfacción obtenida por las melodías en la partitura del $i-$ésimo comositor para comparar, evaluar y adquirir información.
- Desarrollar una versión del MMC adaptada para el tratamiento de problemas de optimización multi-objetivo; ya que muchos problemas del mundo real requieren de una consideración de objetivos múltiples. Las técnicas actuales de manejo de varios objetivos, particularmente en el maro de Algoritmos Evolutivos, permiten alcanzar conjuntos de soluciones eficientes, representando compromisos o balances entre los diferentes criterios de calidad. Parece, por lo tanto, relevante ampliar el marco de aplicación del MMC a esta clase de problemas, basándose en estrategias existentes o más novedosas. Cabe mencionar que existen bancos de problemas usados en la literatura especializada que permitirían evaluar el desempeño del nuevo algoritmo.
- Adaptar e implementar el MMC para la resolución de otros problemas de optimización algunos de los cuales son: alineamiento múltiple de secuencias, coloración robusta, composición musical.

Bibliografía

[1] R. Abbasian, M. Mouhoub, and A. Jula. Solving graph coloring problems using cultural algorithms. In *Twenty-Fourth International FLAIRS Conference*, 2011.

[2] R. L. Ackoff. *Planificación de la empresa del futuro.* LIMUSA, 1983.

[3] R. L. Ackoff. *Rediseñando el futuro.* LIMUSA, 1993.

[4] B. Adenso-Díaz and M. Laguna. Fine-tuning of algorithms using fractional experimental designs and local search. *Operation Research*, 2006:99–114, 54.

[5] M. M. Ali, C. Khompatraporn, and Z. B. Zabinsky. A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems. *Journal of Global Optimization*, 31:635–672, 2005.

[6] J. A. Almánzar and A. M. Alarcón, editors. *Hombre Y Sociedad.* Intec, 1987.

[7] P. Alsina and F. Sese. *La música y su evolución historia de la música con propuestas didácticas y 49 audiciones.* GRAO, 2003.

[8] M. Altman. Is automation the answer: The computational complexity of automated redistricting. *Rutgers Computer and Law Technology Journal*, 23:81–141, 1997.

[9] C. Álvarez, J. Soria, and R. García. *Investigación Operativa: Modelos y técnicas de Optimización.* Proyecto de Innovación Educativa. Universidad Politécnica de Valencia, 2002.

[10] E. L. Atashpaz-Gargari. Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competition. In *IEEE Congress on Evolutionary Computation*, volume 7., pages 4661–4666, 2007.

[11] M. Avriel. *Nonlinear Programming: Analysis and Methods.* Dover Books on Computer Science Series. Dover Publications, 2003.

[12] J. Barnes. Class and committees in norwegian island parish. *Human Relations*, 7:39–58, 1954.

[13] R. Battiti, M. Brunato, and F. Mascias. *Reactive search and intelligent optimization*. Springer, USA, 2010.

[14] M. S. Bazara, J. J. Jarvis, and H. D. Sherali. *Programación lineal y flujo en redes*. Limusa. Noriega Editores, 1998.

[15] B. Belaire. *Programación matemática para la economía y la empresa*. Educació. Laboratori de materials. Universitat de València, 2009.

[16] T. Bertin-Mahieux, R. J. Weiss, and D. P. W. Ellis. Clustering beat-chroma patterns in a large music database, 2010.

[17] D. Bertsekas. *Nonlinear programming*. Athena scientific optimization and computation series. Athena Scientific, 1999.

[18] J. A. Biles. Genjam: A genetic algorithm for generating jazz solos. In *International Computer Music Conference. Aarhus, Denmark: International Computer Music Association*, pages 131–137, 1994.

[19] M. Birattari. *Tuning metaheuristics: A machine learning perspective*. Springer, 2009.

[20] J. Bland. Nonlinear optimization of constrained functions using tabu search. *International Journal of Mathematical Education in Science and Technology*, 24(5):741–747, 1993.

[21] F. Blasco-Vercher and V. Sanjosé-Huguet. *Los instrumentos musicales*. Universidad de Valencia, 1994.

[22] H. M. Botee and E. Bonabeau. Evolving ant colony optimization. In *ADVANCES IN COMPLEX SYSTEMS*, pages 149–159, 1998.

[23] J. Brito-Santana, C. Campos-Rodríguez, F. C. García-López, M. García-Torres, B. Melián-Batista, J. A. Moreno-Pérez, and J. M. Moreno-Vega. Metaheurísticas: una revisión actualizada. Grupo de computación inteligente. Departamento de estadística, investigación operativa y computación., junio 2004.

[24] Z. Cai and Y. Wang. A multiobjective optimization-based evolutionary algorithm for constrained optimization. *IEEE Transactions on evolutionary computation.*, 10:658 – 675, 2006.

[25] J. L. Carson, M. H. Crespin, C. J. Finocchiaro, and D. W. Rohde. Redistricting and party polarization in the u.s. house of representatives. *American Politics Research*, 35:878–905, 2007.

[26] E. Castillo, A. J. Conejo, P. Pedregal, R. García, and N. Alguacil. Formulación y resolución de modelos de programación matemática en ingeniería y ciencia, 2002.

[27] B. C. Castro Souza. Creativity and problem solving: elements for a model of creativity.

[28] J. Cerdá and U. de Barcelona. *Análisis real*. Col.lecció UB. Universitat de Barcelona, 1996.

[29] R. Chelouaha and P. Siarry. Tabu search applied to global optimization. *European Journal of Operational Research*, 23:256–270, 2000.

[30] N. Christakis and J. Fowler. *Conectados: El Sorprendente Poder de las Redes Sociales y Como Nos Afectan*. Pensamiento (Taurus (Firm))). Aguilar, Altea, Taurus, Alfaguara, S.A. de C.V, 2010.

[31] C.-J. Chung and R. G. Reynolds. A testbed for solving optimization problems using cultural algorithms, 1996.

[32] M. Clerc and J. Kennedy. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *IEEE Trans. Evol. Comput.*, 6:58–73, 2002.

[33] C. Coello-Coello. Búsqueda tabú: Evitando lo prohibido. *Soluciones Avanzadas. Tecnologías de Información y Estrategias de Negocios*, 5(49):72–80, septiembre 1997.

[34] C. Coello-Coello. Introducción a la computación evolutiva, mayo 2011.

[35] C. Coello-Coello. *Learning and Intelligent Optimization: 5th International Conference, LION 5, Rome, Italy, January 17-21, 2011, Selected Papers*. Lecture Notes in Computer Science. Springer, 2011.

[36] C. A. Coello-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(11-12):1245–1287, Jan. 2002.

[37] C. A. Coello-Coello and R. Landa-Becerra. Constrained optimization using an evolutionary programming-based cultural algorithm, 2002.

[38] C. A. Coello-Coello and R. Landa-Becerra. Evolutionary multiobjective optimization using a cultural algorithm. In *Swarm Intelligence Symposium, 2003. SIS '03. Proceedings of the 2003 IEEE*, 2003.

[39] S. Cook. The p versus np problem,, 1992.

[40] W. Cook, W. Cunningham, W. Pulleyblank, and A. Schrijver. *Combinatorial Optimization*. Wiley Series in Discrete Mathematics and Optimization. Wiley, 2011.

[41] D. Cope. *The algorithmic composer*. A-R Editions Inc, Wisconsin USA, 2000.

[42] D. Cope. *Computer modelof musical creativity*. MIT Press, London England, 2005.

[43] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*. MacGraw Hill Book company and The MIT Press, 2001.

[44] T. G. Crainic and G. Laporte. *Fleet management and logistics*. Springer, 1998.

[45] J. A. G. Cristobal. *Método de síntesis dimensional óptima de sistemas multicuerpo con restricciones dinámicas . Aplicació al diseño de mecanismos planos.* PhD thesis, Universidad de La Rioja, 2003.

[46] N. Cruz-Cortés. *Sistema inmune artificial para solucionar problemas de optimización.* PhD thesis, CINVESTAV. Instituto Politecnico Nacional, 2004.

[47] Z. J. Czech and P. Czarnas. Parallel simulated annealing for the vehicle routing problem with time windows. In *10th Euromicro Workshop on parallel. Distributed and network based processing. Canary IslandsSpain*, pages 376–383, 2002.

[48] E. da Silva, H. Barbosa, and A. Lemonge. An adaptive constraint handling technique for¸differential evolution with dynamic use of variants in¸engineering optimization. *Optimization and Engineering*, 12:31–54, 2011. 10.1007/s11081-010-9114-2.

[49] E. de Bono. *El pensamiento práctico.* Editorial Paidos, 1993.

[50] K. A. De-Jong. *Analysis of the behavior of a class of genetic adaptive systems.* PhD thesis, University of Michigan, 1975.

[51] S. G. de los Cobos Silva, J. G. Close, M. A. G. Andrade, and A. E. M. Licona. *Búsqueda y exploración estocástica.* Universidad Autónoma Metropolitana, México, 2010.

[52] A. de Vries. *Quantum Computation.* Books on Demand, 2012.

[53] H. Deitel and P. Deitel. *Cómo programar en C, C++ y Java.* Pearson Educación, 2004.

[54] G. del Estado de México. Estado, geografía y estadística.

[55] C. G. del Instituto Federal Electoral. Cg104/2004. Electronico, 2004.

[56] F. Dobslaw. A parameter-tuning framework for metaheuristics based on design of experiments and artificial neural networks. *World academy of science, engineering and technology*, 64:213 –219, 2010.

[57] A. Dolado and L. Arana. *Metodología de programación. Principios y aplicaciones.* Editorial Club Universitario, 2004.

[58] P. Domínguez and U. de Sevilla. *Educación Social: Análisis de Recursos Comunitarios.* Serie Ciencias de la Educación/Universidad de Sevilla Series. Universidad de Sevilla, 2002.

[59] Y. Donoso and R. Fabregat. *Multi-Objective Optimization in Computer Networks Using Metaheuristics.* Auerbach Publications, 2007.

[60] M. Dorigo, V. Maniezzo, and A. Colorni. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics*, 26:29 – 41, 1996.

[61] J. Dréo, A. Pétrowski, P. Siarry, and E. Taillard. *Metaheuristics for hard optimization: methods and case studies*. Springer, 2006.

[62] S. Edelkamp and S. Schroedl. *Heuristic Search: Theory and Applications*. Morgan Kaufmann. Elsevier Science, 2011.

[63] A. Eiben and S. Smit. Parameter tuning for configuring and analyzing evolutionary algorithms.

[64] A. E. Eiben, R. Hinterding, and Z. Michalewicz. Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 3:124 – 141, 1999.

[65] F. Escolano. *Inteligencia artificial: Modelos, técnicas y áreas de aplicación*. Thomson, 2003.

[66] T. Feo and M. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6(2):109–133, 1995.

[67] D. Floreano and C. Mattiussi. *Bio-inspired artificial intelligence: theories, methods, and technologies*. Intelligent robotics and autonomous agents. MIT Press, 2008.

[68] D. Fogel. An introduction to simulated evolutionary optimization. *IEEE Transactions on Neural Networks*, 5(1):3–14, January 1994.

[69] R. Forgette and G. Platt. Redistricting principles and incumbency protection in the u.s. congress. *Political Geography*, 24,:934–951, 2005.

[70] B. Furht, editor. *Handbook of Social Network Technologies and Applications*. Springer, 2010.

[71] J. García-Cabello. *Cálculo diferencial de las ciencias económicas*. Delta Publicaciones, 2006.

[72] N. García-Canclinini. *La producción simbólica teoría y metódo en sociología del arte*. Siglo veitiuno editores, 1979.

[73] D. García-Castaó. *Las rutas de los mercaderes y el alborear de la matemática*. Editorial Club Universitario, 2009.

[74] M. R. Garey and D. S. Jhonson. *Computers and intractability. A guide to the theory of NP-Completeness.* Freeman and company, 1979.

[75] R. Garfinkle and D. Garfinkle. *El universo en tres pasos*. Editorial Critica., 2010.

[76] Z. W. Geem. Optimal cost design of water distribution networks using harmony search. *Engineering Optimization*, 38:259–280, 2006.

[77] Z. W. Geem. *Recent Advances in Harmony Search Algorithm.* Springer, 2009.

[78] Z. W. Geem. *Music-Inspired Harmony Search Algorithm.* Springer, USA, 2010.

[79] Z. W. Geem and J.-Y. Choi. Music composition using harmony search algorithm. *Lecture Notes in Computer Science,*, 4448:593–600, 2007.

[80] Z. W. Geem, K. J.-H., and L. G.V. A new heuristic optimization algorithm: harmony search. *Simulation*, 76:2:60–68, 2001.

[81] Z. W. Geem, K. S. Lee, and Y. Park. Application of harmony search to vehicle routing. *American Journal of Applied Science*, 2(12):1552–1557, 2005.

[82] C. Gershenson. Computing networks: A general framework to contrast neural and swarm architectures. *CoRR*, abs/1001.5244, 2010.

[83] N. Gessler. Evolving cultural things-that-think. In *omputational Synthesis: From Basic Building Blocks to High Level Functionality. Papers from the 2003 AAAI Spring Symposium, Technical Report SS-03-02. Menlo Park, AAAI Press.*, pages 75–81, 2003.

[84] N. Gessler. Fostering creative emergences in artificial cultures. In *In Artificial Life XII - Proceedings of the Twelfth International Conference on the Synthesis and Simulation of Living Systems*, pages 669–676. MIT Press, 2010.

[85] K. C. Gilbert, D. D. Holmes, and R. E. Rosenthal. A multiobjective discrete optimization model for land allocation. *Management Science*, 31(12):1509–1522, 1985.

[86] R. Giudici. *Introducción a la teoría de grafos.* Equinoccio, 1997.

[87] F. Glover. Tabu search-part 1. *ORSA J Computing*, 1(3):190–206, 1989.

[88] F. Glover. Tabu search–part ii. *INFORMS Journal on Computing*, 2(1):4–32, 1990.

[89] F. Glover. *Tabu search and adaptive memory programming – advances, applications, and challenges.* Kluwer Academic Publishers, Dordrecht, Massachusetts, USA, 1996.

[90] F. Glover and G. Kochenberger. *Handbook of metaheuristics.* International series in operations research & management science. Kluwer Academic Publishers, 2003.

[91] F. Glover and M. Laguna. *Tabu Search.* Number v. 1 in Tabu Search. Kluwer Academic Publishers, 1998.

[92] D. Goldin and P. Wegner. The church-turing thesis: Breaking the myth. *New Computational Paradigms*, pages 152–168, 2005.

[93] L. Gómez. *Temas de filosofía del derecho*. Facultad de Derecho, Universidad Católica Andrés Bello, 1988.

[94] T. González. *Handbook of approximation algorithms and metaheurististics*. Chapman & Hall/CRC computer and information science series. Chapman & Hall/CRC, 2007.

[95] J. J. Grefenstette. Optimization of control parameters for genetic algorithms. *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS*, 16:122–128, 1986.

[96] R. P. Grimaldi. *Matemáticas discretas y combinatorias. Una introducción con aplicaciones*. Pearson Printice Hall, 1998.

[97] E. Gurovich. *Introducción a la teoría de la computación: Autómatas y lenguajes formales*. Prensas de ciencias. Universidad Nacional Autónoma de México Facultad de Ciencias, 2008.

[98] C. Gutiérrez and U.E.D.C.R.I.A.D.B.F.O.D.C.R.C. *Epistemología e Informática*. Euned, 1993.

[99] A. D. Hall. *Ingeniería de sietemas*. C.E.C.S.A., 1983.

[100] P. Hansen and N. Mladenović. *Developments of variable neighborhood sea*. Essays and Surveys in Metaheuristics, 2002.

[101] A.-R. Hedar and M. Fukushima. Hybrid simulated annealing and direct search method for nonlinear unconstrained global optimization. *Optimization Methods and Software*, 17(5):891–912, 2002.

[102] K. Heller, F. Mönks, M. Csikszentmihalyi, and R. Wolfe. *The international handbook of giftedness and talent*. Elsevier, New York USA, 2000.

[103] F. S. Hillier and G. J. Lieberman. *Introducción a la investigación de operaciones*. Mc Graw Hill, 5¡ë edition, 1991.

[104] J. H. Holland. Genetic algorithms. *Scientific American*, Second edition:66–72, 1992.

[105] A. Horner and D. E. Goldberg. Genetic algorithms and computer assisted music composition. In *Music Composition. ICMC91 Proceedings, San Francisco: International Computer Music Association*, pages 479–482, 1991.

[106] X. Hu and R. Eberhart. Solving constrained nonlinear optimization problems with particle swarm optimization. In *6th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2002)*, pages 203–206, 2002.

[107] INEGI. Cuentame... información por entridad estado de méxico.

[108] INEGI. México en cifras. información nacional, por entidad federativa y municipios. méxico.

[109] B. Jacob. Composing with genetic algorithms. In *International Computer Music Association*, pages 452–455, 1995.

[110] B. L. Jacob. Algorithmic composition as a model of creativity. *Organised Sound (Cambridge University Press)*, 1:157–165, 1996.

[111] M. Jiao and J. Tang. A novel particle swarm optimization for constrained engineering optimization problems. In L. Kang, Z. Cai, X. Yan, and Y. Liu, editors, *Advances in Computation and Intelligence*, volume 5370 of *Lecture Notes in Computer Science*, pages 79–88. Springer Berlin / Heidelberg, 2008. 10.1007/978-3-540-92137-0_9.

[112] X. Jin and R. G. Reynolds. Using knowledge-based evolutionary computation to solve nonlinear constraint optimization problems: a cultural algorithm approach. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, 1999.

[113] M. C. Joshi and K. M. Moudgalya. *Optimization: Theory and Practice*. Alpha Science International, Ltd, 2004.

[114] J. Kalcsics, S. Nickel, and M. Schröder. Towards a unified territorial design approach applications, algorithms and gis integration. *TOP: An Official Journal of the Spanish Society of Statistics and Operations Research*, 13(1):1–56, 2005.

[115] J. kao Hao and J. Pannier. Simulated annealing and tabu search for constraint solving. In *Proceedings of the fifth international*, 1998.

[116] D. Karaboga and B. Basturk. On the performance of artificial bee colony (abc) algorithm. *Applied Soft Computing*, 8:687–697, 2008.

[117] N. Karmarkar. A new polynomial-timen algorithm for linear programming. *Combinatorica*, 4:373–395, 1984.

[118] J. Kennedy and R. Mendes. Population structure and particle swarm performance. In *Proc. IEEE Congr. Evol. Comput*, pages 1671–1676, 2002.

[119] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[120] G. Komarasamy and A. Wahi. An optimized k-means clustering technique using bat algorithm. *European Journal of Scientific Research*, 84(2):263–273, 2012.

[121] B. Korte and J. Vygen. *Combinatorial optimization. Theory and algorithms*. Springer, 2000.

[122] C. Kosniowski and M. Solanas. *Topología algebraica*. Reverté, 1992.

[123] S. Koziel and Z. Michalewicz. Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization. *Evolutionary Computation*, 7:pp, 1999.

[124] O. Kramer. *Self-Adaptive Heuristics for Evolutionary Computation*. Studies in Computational Intelligence. Springer, 182, 2008.

[125] J. Lagarias and M. Todd. *Mathematical Developments Arising from Linear Programming: Proceedings of a Joint Summer Research Conference Held at Bowdoin College, June 25-July 1, 1988*. Contemporary Mathematics. American Mathematical Society, 1990.

[126] R. Landa-Becerra and C. A. Coello-Coello. Optimization with constraints using a cultured differential evolution approach. In *In Proceedings of the GECCO Conference*, page 34, 2005.

[127] K. S. Lee and Z. W. Geem. A new structural optimization method based on the harmony search algorithm. *Computers & structures*, 82:781–798, 2004.

[128] K. S. Lee and Z. W. Geem. A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. *Computer methods in applied mechanics and engineering*, 194:3902–3933, 2005.

[129] R. C. T. Lee, S. S. Tseng, and R. C. Chang. *Introducción al diseño y análisis de algoritmos. Un enfoque estratégico*. Mc Graw-Hill, 2007.

[130] C. Lévi-Strauss. *The raw and the cooked. Mythologiques Volume One*. University of Chicago Press, 1983.

[131] N. Lezaun and N. Ursua. *Cerebro y conocimiento: un enfoque evolucionista*. Nueva ciencia. Anthropos, 1993.

[132] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Transaction of evolutionary computation*, 10:281–295, 2006.

[133] S.-H. Liu, M. Mernik, and B. R. Bryant. Entropy-driven parameter control for evolutionary algorithms. *Informatica*, 31:41–50, 2007.

[134] Y.-T. Liu. Creativity or novelty?: Cognitive-computational versus social-cultural. *Design Studies*, 23:261–276, 2000.

[135] A. Lodi, K. Allemand, and T. M. Liebling. An evolutionary heuristic for quadratic 0-1 programming. *European Journal of Operational Research*, 119(3):662 – 670, 1999.

[136] J. Lombraña. *Historia de la Lógica*. Universidad de Oviedo, 1989.

[137] J. C. López Garciá. Educación básica. algoritmos y programación. guía para docentes. segunda edición. eduteka, noviembre 2009.

[138] D. G. Luenberger. *Linear and Nonlinear Programming*. Addison-Wesley, 1984.

[139] W. G. Macready and D. H. Wolpert. What makes an optimization problem hard? *Complexity*, 5:40–46, 1996.

[140] G. Magaril-Il'yaev and V. Tikhomirov. *Convex Analysis: Theory and Applications*. Translations of Mathematical Monographs. American Mathematical Society, 2003.

[141] M. Mahdavia, M. Fesangharyb, and E. Damangirb. An improved harmony search algorithm for solving optimization problems. *Applied Mathematics and Computation*, pages 1537–1579, 2007.

[142] R. Martí. Procedimientos metaheurísticos en optimizaciín combinatoria. *Matemátiques*, 1(1):3–62, 2003.

[143] C. Martínez-Priego. *Quiero ser community manager: 10 profesionales y 5 compañías analizan una nueva realidad*. Esic Editorial, 2012.

[144] J. R. McDonnell, R. G. Reynolds, and D. B. Fogel. *Evolutionary programming 4*. MIT Press, 1995.

[145] B. Melián, J. A. M. Pérez, and J. M. M. Vega. Metaheuristics: A global view. *Revista Iberoamericana de Inteligencia Artificial*, 19:7–28, 2003.

[146] R. Mendes, K. J., and N. J. The fully informed particleswarm: Simpler, maybe better. *IEEE Trans. Evol. Comput*, pages 204–210, 2004.

[147] E. Mezura-Montes, O. Cetina-Domínguez, and B. H.-O. na. *Mecatrónica*, chapter Nuevas Heurísticas Inspiradas en la Naturaleza para Optimización Numérica,, pages 249–272. Editorial IPN, 2010.

[148] E. Mezura-Montes and C. A. C. Coello. A simple multimembered evolution strategy to solve constrained optimization problems. *IEEE Transactions on Evolutionary computation*, 9:1–17, 2003.

[149] E. Mezura-Montes and C. A. Coello-Coello. Fuentes de dificultad en optimización global con restricciones usando algoritmos evolutivos. In *In F. Padilla et. al editors Proceedings of the 2nd Mexican Conference on Evolutionary Computation (COMCEV 2005)*, pages 57–62, 2005.

[150] Z. Michalewicz. Genetic algorithms, numerical optimization, and constraints. In *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 151–158, July 1995.

[151] Z. Michalewicz, K. Deb, M. Schmidtz, and T. Stidsenx. Test-case generator for constrained parameter optimization techniques. *IEEE Transactions on Evolutionary Computation*, 4:197–215, 2000.

[152] Z. Michalewicz and D. B. Fogel. *How to solve it: modern heuristics*. Springer, 1998.

[153] Z. Michalewicz and C. Z. Janikow. Genocop: a genetic algorithm for numerical optimization problems with linear constraints. *Communications of the ACM - Electronic supplement to the December*, 39:article no. 175, 1996.

[154] D. J. Michelini, J. S. Martín, and J. Wester. *Etica discurso conflictividad*. Universidad Nacional de Río Cuarto, 1995.

[155] L. Montejano-Peimbert. *Cuerpos de Ancho Constante*. Fondo de Cultura Economica, 1998.

[156] R. Mora-Gutiérrez, J. Ramírez-Rodríguez, and E. Rincón-García. An optimization algorithm inspired by musical composition. *Artificial Intelligence Review. Springer Netherlands.*, pages 1–15, 2012. 10.1007/s10462-011-9309-8.

[157] R. Mora-Gutiérrez, J. Ramírez-Rodríguez, E. Rincón-García, A. Ponsich, and O. Herrera. An optimization algorithm inspired by social creativity systems. *Computing*, pages 1–28, Agosto 2012. 10.1007/s00607-012-0205-0.

[158] R. A. Mora-Gutiérrez. Desarrollo de un procedimiento para solucionar el problema de alineameinto múltiple de secuencias. Master's thesis, UNAM, 2009.

[159] R. A. Mora-Gutiérrez, J. Ramírez-Rodríguez, and M. Elizondo-Cortes. Heurística para solucionar el problema de alineameinto múltiple de secuencias. *Revista de matemáticas: teoría y aplicaciones*, 18(1):121–136, 2011.

[160] R. A. Mora-Gutiérrez, J. Ramírez-Rodríguez, E. A. Ricardo García, O. Herrera, A. Ponsich, and L.-V. Pedro. An optimization algorithm inspired by musical composition in constrained optimization problem. In *XVIII Simposio Internacional de Métodos Matemáticos Aplicados a las Ciencias*, 2012.

[161] E. Morales and J. González. Capitulo 11. optimización basada en colonia de hormigas.

[162] P. Moreno Regidor and J. García López de Lacalle. Estado del arte en procesos de zonificación. *GeoFocus*, 11:155–181, 2011.

[163] V. Muniswamy. *Design And Analysis Of Algorithms*. I.K. International Publishing House Pvt. Ltd., 2009.

[164] R. Myers, E. R. Hancock, and L. Labelling. Empirical modeling of genetic algorithms. *Evolutionary Computation*, 9:461–493, 2001.

[165] N. N and N. Marroquín. *Tras Los Pasos de Un... Hacker*. CreateSpace, 2010.

[166] V. Nannen and A. E. Eiben. Relevance estimation and value calibration of evolutionary algorithm parameters. In *Proceedings of the 20th international joint conference on Artifical intelligence*, IJCAI'07, pages 975–980, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[167] G. Navarro. Teoría de la computacón (lenguajes formales, computabilidad y complejidad), septimbre 2011.

[168] N. Navarro-Fernández. *Caracterización y cuantificación de la influencia de la música como agente físico sobre el comportamiento de células madre neurales embrionarias en cultivo*. PhD thesis, Facultad de Medicina. Univesidad de Valladolod., 2010.

[169] A. Neme and S. Hernández. Algorithms inspired in social phenomena. In R. Chiong, editor, *Nature-Inspired Algorithms for Optimisation*, volume 193 of *Studies in Computational Intelligence*, pages 369–387. Springer Berlin / Heidelberg, 2009. 10.1007/978-3-642-00267-0-13.

[170] M. N. nez. Nuevas tendencias de investigación en programación lineal. primera parte: Métodos de punto interir. *Tiempo Compartido*, 2:3–6, 1991.

[171] M. Nielsen and I. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.

[172] R. G. Niemi, B. Grofman, C. Carlucci, and H. T. Measuring compactness and the role orf a compactness standard in a test for partisian and racial gerrymandering. *Journal of Politics*, 52 No. 4:1155–1181, 1990.

[173] A. D.-M. noz, J. J. Pantrigo-Fernández, and M. Gallego-Carrillo. *Metaheurísticas*. Librería-Editorial Dykinson, 2007.

[174] M. Omran and M. Mahdavi. Global-best harmony search. *Applied Mathematics and Computation*, pages 643–656, 2008.

[175] Y. S. Ong and A. Keane. Meta-lamarckian learning in memetic algorithms. *Evolutionary Computation, IEEE Transactions on*, 8(2):99 – 110, april 2004.

[176] C. Orozco, R. Llanos, O. García, and S. K. *Redes sociales: Infancia, familia y comunidad*. Universidad del Norte, 2003.

[177] J. I. Palacios-Sanz. El concepto de musicoterapia a través de la historia. *Revista Inteuniversitaria de formación de profesorado*, 42:19–31, 2001.

[178] Q.-K. Pan, P. N. Suganthan, M. F. Tasgetiren, and J. J. Liang. A self-adaptive global best harmony search algorithm for continuous optimization problems. *Applied Mathematics and Computation*, 216:830–848, 2010.

[179] C. H. Papadimitrou and K. Steiglitz. *Combinatorial Optimization Algorithms and Complexity*. Dover publications inc., USA, 1982.

[180] P. M. Pardalos and M. G. C. Resende. *Handbook of applied optimization*. Oxford University Press, 2002.

[181] K. E. Parsopoulos and M. N. Vrahatis. Upso-a unified particle swarm optimization scheme.

[182] P. Pascual-Mejía. *Didáctica de la Música*. Pearson - Prentice Hall, España, 2006.

[183] J. Pasqual. *La evaluación de políticas y proyectos: criterios de valoración económicos y sociales*. Icaria Editorial, 1999.

[184] K. M. Passino. Biomimicry of bacterial foraging for distributed optimization and control. *IEEE Control Systems Magazine, .*, 22:52–67, 2002.

[185] K. M. Passino. Bacterial foraging optimization. *International Journal of Swarm Intelligence Research*, 1:1–16, 2010.

[186] D. A. Pelta. *Algoritmos heuristicos en bioinformatica*. PhD thesis, Universidad de Granada, 2002.

[187] D. A. Pelta and J. L. Verdegay. Un método de búsqueda por entornos, adaptativo y difuso, 2003.

[188] A. Peña. *Mapas Conceptuales: Una Técnica para Aprender*. Educación Hoy. Narcea, 1994.

[189] T. Peram, K. Veeramachaneni, and M. C. K. Fitness-distance-ratio based particle swarm optimization. In *Proc. Swarm Intelligence Symp*, pages 174–181, 2003.

[190] M. Pilski and F. Seredyński. Function optimization using metaheuristics, 2006.

[191] D. Pojular. *Programación lineal y optimización en redes: ejercicios resueltos de investigación operativa*, volume 95 of *Col'alecció Material Series*. Universidad Autónoma de Barcelona, 2000.

[192] G. Polya. *How to Solve It: A New Aspect of Mathematical Method*. Princeton Science Library. Princeton University Press, 2004.

[193] J. Prawda. *Métodos y modelos de investigación de operaciones*. Editorial Limusa, 1976.

[194] A. Ramos, P. Sánchez, J. M. Ferrer, J. Barquén, and P. Linares. Modelos matemáticos de optimización.

[195] T. Ray, T. Kang, and S. K. Chye. An evolutionary algorithm for constrained optimization. In *GECCO'00*, pages 771–777, 2000.

[196] T. Ray and K. M. Liew. Society and civilitation: an optimization algorithm based on the simulation of social behavior. *IEEE Transaction on evolutionary computation,*, 7:386–396, 2003.

[197] T. A. Regelski. Taking the .ªrt" of music for granted: a critical sociology of the sesthetic philosophy of music. In *Proceedings of the second international symposium on the philosophy of music education. Critical reflections on music education*, pages 23–58. Canadian Music Education Research Centre, University of Toronto, 1994.

[198] G. Restrepo. *Teoría de la integración*. Colección Ciencias Físicas, Naturales y Exactas. Programa Editorial Universidad del Valle, 2004.

[199] R. Reynolds. An introduction to cultural algorithms. In *in Proceedings of the 3rd Annual Conference on Evolutionary Programming, World Scientific Publishing*, pages 131–139, 1997.

[200] R. Reynolds, Z. Michalewicz, and M. Cavaretta. *Evolutionary programming 4*, chapter Using cultural algorithms for constraint handling in GENOCOP, pages 298–305. Complex adaptive systems. MIT Press, 1995.

[201] R. G. Reynolds. An introduction to cultural algorithms. In *in Proceedings of the 3rd Annual Conference on Evolutionary Programming,World Scientific Publishing*, pages 131–139, 1994.

[202] E. Ridge and D. Kudenko. Screening the parameters affecting heuristic performance. In *In Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 2007.

[203] E. Rincón-García. *Diseño de zonas geométricamente compactas utilizando celdas cuadradas*. PhD thesis, Universidad Nacional Autónoma de México. Programa de Maestría y Doctorado en Ingeniería. Facultad de Ingeniería, 2010.

[204] E. Rincón-García and M. a. Gutiérrez-Andrade. Compacidad en celdas aplicada al diseño de zonas electorales. *EconoQuantum*, 5:73–96, 2009.

[205] E. Rincón-García, M. A. Gutiérrez-Andrade, S. G. de los Cobos Silva, and P. Lara-Velázquez. Nuevas medidas de compacidad geométrica para el diseño de zonas. *Revista de Matemática: Teoría y Aplicaciones*, 17(1):1–10, 2010.

[206] E. A. Rincón-García, M. A. Gutiérrez-Andrade, S. G. de-los Cobos-Silva, P. Lara-Velázquez, and R. A. Mora-Gutiérrez. Recocido simulado y búsqueda de entornos variables para diseñar zonas electorales. TLAIO4, Noviembre 2011.

[207] M. Rizo-García. Redes. una aproximación al concepto.

[208] J. Román. *Lógica matemática y computabilidad.* Díaz de Santos, 1990.

[209] Rosing and ReVelle. Heuristic concentration: Two stage solution construction. *European Journal of Operational Research*, 97:75–86, 1997.

[210] F. Rothlauf. *Design of modern heuristics. Principles and aplication.* Natural computing. Springer, 1 st edition, 2011.

[211] G. Rubiano. *Topología general.* Universidad Nacional de Colombia, 2002.

[212] A. Rubio-Largo, M. A. Vega-Rodríguez, J. A. Gómez-Pulido, and J. M. Sánchez-Pérez. Algoritmo multiobjetivo inspirado en el comportamiento de las luciérnagas para resolver el problema rwa. In *Actas del VIII Congreso Español sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados*, 2012.

[213] H. Ruíz. *Estructura socioeconomica de Mexico/ Socio-Economic Structure of Mexico*. Cengage Learning Latin America, 2005.

[214] M. P. Saka. Optimum geometry design of geodesic domes using harmony search algorithm. *Advances in Structural Engineering*, 10:595–606, 2007.

[215] J. J. Salazar-González. *Programación matemática.* Ediciones Díaz de Santos, 425, 2001.

[216] R. Salomon. Reevaluating genetic algorithm performance under coordinate rotation of benchmark functions; a survey of some theoretical and practical aspects of genetic algorithms. *BioSystems*, 39:263–278, 1996.

[217] E. Salort. *Métodos Cuantitativos.* Colección Libro Docente/Universidad Politécnica de Valencia Series. Universidad Politécnica de Valencia. Servicio de Publicaciones, 1997.

[218] C. Salto. *Metaheurísticas híbridas paralelas pasa problema industriales de corte, empaquetado y otros relacionados.* PhD thesis, Universidad Nacional de San Luis, 2009.

[219] L. Santana-Quintero and A. Coello-Coello. Una introducción a la computación evolutiva y algunas de sus aplicaciones en economía y finanzas. *Revista de Métodos Cuantitativos para la Economía y la Empresa*, 2:3–36, diciembre 2006.

[220] R. Saunders and J. S. Gero. Artificial creativity: A synthetic approach to the study of creative behaviour. In J, editor, *Computational and Cognitive Models of Creative Design V, Key Centre of Design Computing and Cognition, University of Sydney, Sydney*, pages 113–139, 2001.

[221] A. Schoenberg. *Fundamentos de la composición musical.* Real Musical, Madrid, 1994.

[222] C. E. Shalley. Effects of productivity goals, creativity goals, and personal discretion on individual creativity. *Journal of Applied Psychology*, 76(2):179–185, 1991.

[223] Y. Shi and R. C. Eberhart. A modified particle swarm optimizer. In *Proc. IEEE Congr. Evol. Comput*, pages 69–73, 1998.

[224] T. Shirabe. A model of contiguity for spatial unit allocation. *Geographical Analysis*, 37:2–16, 2005.

[225] S. Sinha. *Mathematical Programming: Theory and Methods*. Elsevier Science, 2006.

[226] R. Sivarethinamohan. *Operations Research*. McGraw-Hill Education (India) Pvt Ltd, 2008.

[227] C. Souza and B. Carvalho. Creativity and problem solving: elements for a model of creativity, septiembre 2007.

[228] R. J. Sternberg. *Handbook of creativity*. Cambridge University Press, 1999.

[229] A. Storr. *La música y la mente: el fenómeno auditivo y el porqué de las pasiones*. Editorial Paidos, 2007.

[230] W. Sun, J. Han, and J. Sun. Global convergence of nonmonotone descent methods for unconstrained optimization problems. *Journal of Computational and Applied Mathematics*, 146(1):8 –98, 2002. Papers presented at the 1st Sino-Japan Optimization Meeting, 26-28 October 2000, Hong Kong, China.

[231] G. Taguchi and Y. Yokoyama. *Taguchi methods: design of experiments*. TAGUCHI METHODS SERIES. ASI Press, 1993.

[232] H. A. Taha. *Investigación de Operaciones*. Pearson - Prentice Hall, 7¡ë edition, 2004.

[233] W. Tang and Y. Li. Constrained optimization using triple spaces cultured genetic algorithm. *International Conference on Natural Computation*, 6:589–593, 2008.

[234] G. Thomas, M. Weir, J. Hass, and F. Giordano. *Cálculo: una variable*. Cálculo. Pearson Educación, 2005.

[235] J. M. Tjensvold. Generic distributed exact cover solver, Diciembre 2007.

[236] J. T.Palma-Méndez and R. Marín-Morales. *Inteligencia artificial. Técnicas, métodos y aplicaciones*. Mc Graw Hill, 2008.

[237] M. Uría, B. Gladish, and S. García. *Optimización dinámica: Teoría del control óptimo*. Universidad de Oviedo. Servicio de Publicaciones, 1998.

[238] F. van den Bergh and A. P. Engelbrecht. A cooperative approach to particle swarm optimization. *IEEE Trans. Evol. Comput.*, 8:225–239, 2004.

[239] J. Vázquez and D. Vázquez. *La desafección social hacia los partidos políticos*. Juan Vázquez Yebra, 2011.

[240] M. Vela and B. Cano. *Historia filosófica de la sociedad humana,: parte primera*. en la imprenta de don Benito Cano, 1797.

[241] P. Viñuela, P. Borrajo, P. Fernández, and D. Millán. *Lenguajes, gramáticas y autómatas: un enfoque práctico*. Addison-Wesley, 1997.

[242] B. Wang, X. Jin, and B. Cheng. Lion pride optimizer: An optimization algorithm inspired by lion pride behavior. *Science China*, 55:2369–2389, 2012.

[243] C.-M. Wang and Y.-F. Huang. Self-adaptive harmony search algorithm for optimization. *Expert Systems with applications*, 37:2826–2837, 2010.

[244] K. Wayne. Polynomial-time reductions, 2001.

[245] T. Weise. *Global Optimization Algorithms Theory and Application*. it-weise.de (self-published): Germany, 2009.

[246] S. R. White. Concepts of scale in simulated annealing. *AIP Conference Proceedings*, 122(1):261–270, 1984.

[247] C. Williams. *Explorations in Quantum Computing*. Texts in Computer Science. Springer, 2011.

[248] W. L. Winston. *Investigación de operaciones. Aplicaciones y algoritmos*. Thomson, 4 edition, 2005.

[249] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transaction of evolutionary computation,*, 1:67–82, 1997.

[250] S.-X. Yang. Firey algorithms for multimodal optimization. stochastic algorithms foundations and applications,. *Lecture Notes in Computer Sciences*, 5792:169–178, 2009.

[251] X.-S. Yang. Test problems in optimization. In *Engineering Optimization: An Introduction with Metaheuristic Applications*. John Wiley & Sons,, 2010.

[252] C. Zhang and H.-P. Wang. Mixed-discrete nonlinear optimization with simulated annealing. *Engineering Optimization*, 21(4):277–291, 1993.

Apéndice A

Modelación de problemas de optimización

A.1. Los modelos usados por la programación matemática

Estos modelos, generalmente, contienen los siguientes elementos: [248, 232, 103]:

- **Alternativas o variables de decisión**: Son n decisiones cuantificables, cuyo valor afecta el desempeño del sistema.
- **Restricciones**: Representan un conjunto de m relaciones o condiciones (expresadas como ecuaciones e inecuaciones) que un subconjunto de variables están obligadas a satisfacer.
- **Función objetivo (o funciones objetivo)**: Es una medida cuantitativa sobre la calidad de las soluciones de un problema. Se expresa como una función matemática de las variables de decisión.

Modelar es un proceso creativo-intelectual para la generación de modelos, el cual debe ser sistemático, racional y teóricamente guiado, su objetivo es analizar y resolver problemas. La modelación de problemas de optimización ha sido abordada, estudiada y sistematizada por la investigación de operaciones (véase Figura A.1). La modelación es de crucial importancia, ya que permite generar un instrumento para describir, estudiar, analizar y comprender el comportamiento del sistema. Debido a su importancia se analiza brevemente esta proceso intelectual.

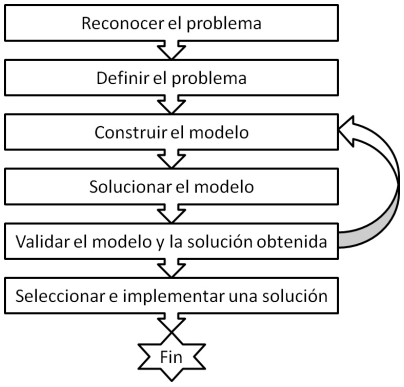

Figura A.1: Fases de la modelación de un problema de optimización

A.2. El proceso de construcción de modelos de optimización

De la modelación de un problema se obtienen algunos beneficios implícitos y explícitos, como: a) el modelo del sistema; b) favorece el intercambio de opiniones y conocimiento entre los actores involucrados; c) organización, sistematización y explotación de la información disponible sobre el sistema; d) el análisis de los resultados obtenidos servirá como guía para la toma de decisiones, entre otros.

Generalmente, se utilizan los principios de parsimonia –también, conocido como principio de sencillez, Navaja de Occam u Ockham–, y el de no contradicción como guas en la modelación. El "principio de parsimonia" (Guillermo de Ockham, 1280-1349), establece no emplear más conceptos, ideas u objetos teóricos que los estrictamente necesarios para generar una explicación que sea satisfactoria del fenómeno (o fenómenos) de interés [75, 73, 183].

Principio A.1 (Principio de parsimonia) ***Entia non sunt multiplicanda praeter necessitatem*** *(el número de entes no debe ser multiplicado sin necesidad.)[154]*

El "principio de no contradicción", establece que dos juicios contradictorios sobre un objeto u evento no pueden ser simultáneamente validos; y por lo tanto, basta con reconocer la validez de uno de ellos para poder negar formalmente el otro. En otras palabras: no se puede atribuir al mismo concepto dos cualidades opuestas en las mismas condiciones y en el mismo instante. A continuación, se expresa este principio en la lógica proposicional:

Principio A.2 (Principio de no contradicción) $\neg(p \wedge \neg p)$

A.2.1. Reconocer del problema

En esta fase se detectan, definen y plantean los problemas presentes en el sistema; para lo cual: se crean, comparan y analizan un modelo ideal y otro real del sistema, las actividades que conforman esta fase son:

- Determinar las fronteras del sistema.
- Determinar y caracterizar el medio ambiente en el que se desenvuelve el sistema.
- Identificar, caracterizar y analizar la función [1], los fines [2] y los medios [3] del sistema.
- Evaluar la situación actual del sistema.
- Crear la situación ideal del sistema.
- Detectar las posibles causas de los problemas.
- Identificar las posibles consecuencias de los problemas.
- Plantear los problemas.

La información obtenida de esta fase será utilizada en la etapa siguiente.

A.2.2. Definir el problema

En la fase critica se establecen los límites del problema a resolver; por ende, esta etapa afecta en forma significativa los resultados y conclusiones obtenidos de la modelación ; ya que generalmente es difícil obtener una respuesta "correcta" de un problema "equivocado" [103]. En esta fase, se tomará la decisión sobre cual de los problemas presentes en el sistema se debe solucionar, para lo que es conveniente tomar como referencia los fines del sistema. Las actividades que se realizan en esta fase son:

- Definir el problema que va a ser investigado.
- Especificar los supuestos bajo los cuales el sistema será modelado.
- Identificar y describir las alternativas de solución.
- Determinar los objetivos del estudio.
- Recolectar información sobre el problema.
- Descripción verbal del problema.

[1] Es la acción principal que realiza el sistema.

[2] Son los resultados que persiguen las acciones del sistema, pueden ser usados como una gua en la toma de decisiones.

[3] Son el conjunto de elementos que interactúan para alcanzar los fines del sistema.

A.2.3. Construir el modelo matemático

Consiste en el reemplazo del objeto cognitivo por su imagen matemática. Durante la escritura matemática, se deben definir: las características de las variables de decisión, estructurar las ecuaciones o inecuaciones que representen correctamente las relaciones existentes entre las variables de decisión en el problema real, la función objetivo (funciones objetivos), y los parámetros necesarios. Esta es una fase creativa, en la cual se debe prestar atención a la precisión de la formulación.

La construcción de un modelo adecuado para reproducir la realidad, es una etapa crucial para obtener una solución satisfactoria del problema real. Los modelos de programación matemática pueden ser clasificados de acuerdo a las características y propiedades de sus elementos en [232, 248]:

a) Modelos continuos vs. discretos.

   Si todas las variables de decisión del modelo pueden asumir cualquier valor $\mathbb{R}$ entonces se dice que el modelo de optimización es un **modelo continuo** ; en contraste, cuando al menos una variable de decisión debe asumir valores en: $\mathbb{Z}$ o $\mathbb{N}$; entonces, se dice que el modelo de optimización es **discreto** -si todas las variables asumen valores en $\mathbb{Z}$ o $\mathbb{N}$, se dice que el modelo es discreto puro. En contraste, si existe alguna variable continua, se dice que el modelo es mixto-. En el ejemplo A.1, se muestran algunos problemas de optimización como ejemplos de estos modelos.

Ejemplo A.1

Modelo continuo

$$\text{mín} \sum_{j=1}^{n} c_i * x_i$$

sujeto a:

$$\sum_{j=1}^{n} a_{i,j} * x_{i,j} \leq, \geq \ o \ = b_j \ \forall \ i = 1, 2, \ldots, m$$

$$x_i \geq 0$$

$$x \in \mathbb{R}^n, c \in \mathbb{R}^n$$

Modelo discreto

$$\text{máx} \sum_{i=1}^{n} c_i * x_i$$

sujeto a:

$$\sum_{j=1}^{n} a_{i,j} * x_{i,j} \leq, \geq \ o \ = b_j \ \forall \ i = 1, 2, \ldots, m$$

$$x_i \geq 0$$

$$x \in \mathbb{Z}^n$$

- Modelos lineales vs. no lineales.

  Un **modelo lineal** implica que todas las restricciones y el conjunto de funciones objetivo cumplen con los principios de proporcionalidad [4] y superposición [5], pero si alguna restricción o alguna de las funciones objetivo no cumplen con dichos principios entonces se trata de un **modelo no lineal**. En el ejemplo A.2, se muestran algunos problemas de optimización como ejemplos de estos modelos.

  Ejemplo A.2

  Modelo lineal

$$\text{mín} \sum_{i=1}^{n} c_i * x_i$$

$$\textit{sujeto a:}$$

$$a_{i,j} * x_i \leq, \geq \ o \ = b_j \ \forall \ j = 1, \ldots, m$$

$$x_i \geq 0$$

$$x \in \mathbb{Z}^n \wedge x \in [0, 1]$$

  Modelo no-lineal

$$\text{mín} \sum_{i=1}^{n} c_i * x_i^k$$

$$\textit{sujeto a:}$$

$$a_{i,j} * x_i \leq, \geq \ o \ = b_j \ \forall \ j = 1, \ldots, m$$

$$x \in \mathbb{R}^n$$

$$k \neq 1$$

- Modelos mono objetivos o multiobjetivos.

  En **modelo multiobjetivo** se plantean un conjunto de funciones objetivos –dos o más– habitualmente en conflicto entre si. La existencia de múltiples funciones objetivo plantea una diferencia fundamental con un **modelo mono objetivo**: no existirá una única solución al problema, sino un conjunto de soluciones que plantearan diferentes compromisos entre los valores de las funciones a optimizar. En el ejemplo A.3, se muestran algunos problemas de optimización como ejemplos de estos modelos.

  Ejemplo A.3

  Modelo mono objetivo

[4] Implica satisfacer la propiedad: $f(\alpha * x) = \alpha * f(x)$

[5] Se satisface la propiedad $f(x + y) = f(x) + f(y)$

$$\text{mín} \sum_{i=1}^{n} c_i * x_i$$

sujeto a:

$$a_{i,j} * x_i \leq, \geq \ o \ = b_j \ \forall \ j = 1, \ldots, m$$

$$x_i \geq 0$$

$$x \in \mathbb{Z}^n \wedge x \in [0, 1]$$

Modelo multiobjetivo

$$\text{mín}\{f_1(x), f_2(x), \ldots, f_k(x)\}$$

sujeto a:

$$a_{i,j} * x_i \leq, \geq \ o \ = b_j \ \textit{para todo} \ j = 1, \ldots, m$$

$$x_i \geq 0$$

$$x \in \mathbb{R}^n, c \in \mathbb{R}^n, A \in \mathbb{R}^{n \times m}, b \in \mathbb{R}^n$$

- Modelos determinísticos vs. estocásticos.

  Un **modelo determinista** es aquel donde se supone que todos los datos pertinentes se conocen con certeza; es decir, para cualquier conjunto de valores de las variables de decisión se conoce con seguridad si las restricciones se cumplen o no además del valor de la función objetivo (o funciones objetivo). En contraste, en los **modelos estocásticos** –también conocidos como probabilísticos– se presupone que algunas variables son aleatorias, y por lo tanto, no se conocerá su valor con exactitud hasta tomar las decisiones correspondientes, tal desconocimiento debe ser incorporado al modelo. En el ejemplo A.4, se muestran algunos problemas de optimización como ejemplos de estos modelos.

Ejemplo A.4

Modelo determinístico

$$\text{mín} \sum_{i=1}^{n} c_i * x_i$$

sujeto a:

$$a_{i,j} * x_i \leq, \geq \ o \ = b_j \ \forall \ j = 1, \ldots, m$$

$$x_i \geq 0$$

$$x \in \mathbb{Z}^n \wedge x \in [0, 1]$$

Modelo estocástico

$$\text{mín} \sum_{i=1}^{r} G(i)$$

sujeto a:

$$T \geq 0 \wedge T \in \mathbb{Z}^n$$

$$x(1) \geq 0$$

$$x(i+1) \geq x(i),\ i = 1, \ldots, T$$

$$x(N) \leq \textit{horizonte de planeación}$$

$$x(i) \geq 0 \wedge x(i) \in \mathbb{Z}^n$$

- Modelos estáticos vs. dinámicos.

  Un **modelo estático** se utiliza para analizar un sistema en un instante en el tiempo, por lo cual en su formulación no se considera el avance del tiempo. Por el contrario, en un **modelo dinámico** se considera que al menos un elemento de decisión evoluciona o cambia con respecto del tiempo; por ende, en el modelo se describen a las variables de decisión como funciones del tiempo, describiendo trayectorias temporales. En el ejemplo A.5, se muestran algunos problemas de optimización como ejemplos de estos modelos.

Ejemplo A.5

Modelo estático

$$\text{mín} \sum_{i=1}^{n} c_i * x_i$$

sujeto a:

$$a_{i,j} * x_i \leq, \geq \ o \ = b_j \ \forall \ j = 1, \ldots, m$$

$$x_i \geq 0$$

$$x \in \mathbb{Z}^n \wedge x \in [0, 1]$$

Modelo dinámico

$$\text{mín} \int_0^T \left(c_1 [x'(t)]^2 + c_2 x(t) \right) dt$$

sujeto a:

$$x(0) = 0$$

$$x(T) = B$$

$$x'(t) \geq 0$$

A.2.4. Solucionar el modelo

Después de construir el modelo, se utilizan mecanismos, generalmente algoritmo[6], con el objeto de encontrar la solución óptima (o cuasi-óptima) del problema en cuestión. El método de solución a usar en la resolución de cualquier problema está en función de las características y propiedades de su modelo matemático (véase sección 1.1.1). Durante esta fase es posible el desarrollo o adecuación de algún (os) algoritmo (s) para resolver un modelo de programación matemática.

A.2.5. Validar el modelo y la solución obtenida

Tiene como propósito perfeccionar el modelo propuesto de tal forma que, dicho modelo sea una herramienta adecuada para analizar y predecir el comportamiento del sistema de interés. Para alcanzar el propósito anterior se suele utilizar recursivamente el proceso de modelación desde la fase de construcción del modelo hasta la fase de validación del mismo, y se termina hasta alcanzar una predicción razonable del sistema; es decir, todo modelo diseñado debe probarse de manera exhaustiva con el objeto de encontrar y corregir la mayor cantidad de defectos como sea posible. Lo anterior es para lograr que el modelo represente de forma adecuada al sistema, aunque existe la posibilidad de que queden algunas fallas ocultas en el modelo (que quizá nunca se detecten), pero se habrán eliminado suficientes errores importantes como para que sea confiable utilizarlo.

A.2.6. Seleccionar e implementar una solución

Se interpretan los resultados numéricos obtenidos por el modelo: como un conjunto de instrucciones de operación, tareas o decisiones emitidas en forma clara y comprensible para los involucrados en el sistema. El éxito de la puesta en práctica, depende en gran parte del apoyo que proporcionen los involucrados en el sistema. Es recomendable implicar a los directamente involucrados en el sistema durante todo el proceso de modelación; además de señalar con claridad los beneficios derivados de implementar dichas recomendaciones, la buena comunicación ayuda a asegurar que el estudio se concluya con éxito.

[6]Un algoritmo es un procedimiento con un número finito de pasos bien definidos para realizar una tarea.

Apéndice B

Conceptos básicos de topología

El espacio topológico es una estructura matemática, el cual se puede definir de manera informal como: dado un conjunto X es una familia de subconjuntos suyos que contienen a los subconjuntos triviales, [1] y además son cerrados para la unión y la inserción infinita. A continuación, se da la definición formal de espacio topológico.

Definición 73 *Un espacio topológico es un par (X, U), donde X es un conjunto y U es una familia de subconjuntos de X con las siguientes propiedades:*

1. $X \in U$ *y* $\varnothing \in U$
2. *Si* $\{V_{i \in I}\} \subset U$ *entonces* $\bigcup_{i \in I} V_i \in U$
3. *Si* $V_i \in U \; \forall i = 1, 2, \ldots, n$ *entonces* $\bigcap_{i=1}^{n} V_i \in U$

[122, 28]

U se denomina **topología** de (X, U); mientras que a los elementos de U se les llama los conjuntos abiertos de (X, U) o simplemente **conjuntos abiertos** de X (si se sobreentiende la topología); mientras que sus complementos se llaman conjuntos cerrados. A los elementos en X se les nombra puntos de U [122]. Si se denota por $\mathfrak{F}(X)$ al conjunto de todos los subconjuntos de X, una topología en X no es más que una elección de $(X) \subseteq \mathfrak{F}$ que satisfaga las condiciones de un espacio topológico. En la Definición 73, la primera propiedad implica que en cualquier familia de subconjuntos U debe contener a los subconjuntos X y $\varnothing$. La segunda

[1] Los subconjuntos triviales son el conjunto vaco y el propio conjunto, ya que para todo conjunto X se cumple $X \subseteq X$ y $\varnothing \subseteq X$

propiedad implica que la unión de toda colección de conjuntos de U este también en U, se puede probar que si $\{V_1, V_2, \ldots, V_n\}$ entonces $V_1 \cup V_2 \cup \ldots \cup V_n \in U$. Finalmente, la tercera propiedad implica que la intersección de cualquier colección finita de conjuntos de U este también en U; ya que, si $\{V_1, V_2, \ldots, V_n\}$, entonces $V_1 \cap V_2 \cap \ldots \cap V_n \in U$.

Teorema 30 *Para cualquier universo $\mathcal{U}$ y cualesquiera conjuntos $A, B \subseteq \mathcal{U}$, las siguientes proposiciones son equivalentes:*

$$A)\ A \subseteq B \qquad B)\ A \cap B = A$$
$$C)\ A \cup B = B \qquad D)\ \overline{B} \subseteq \overline{A}$$

[96]

En los ejemplos B.1,B.2,B.3, se muestran algunos espacios topológico.

Ejemplo B.1

Dado un conjunto X y U= $\{\varnothing, X\}$, se desea determinar si X junto con U es un espacio topológico.

- *Puesto que U contiene a los conjuntos triviales, se cumple la primera propiedad de los espacios topológicos.*
- *Si se une un número arbitrario de miembros de U los resultados posibles son X y $\varnothing$ (con base en las propiedades del neutro [2] y en las propiedades idempotentes [3]), dado que $\varnothing \in U$ se cumple la segunda propiedad de espacios topológicos.*
- *Si se intersectan un número finito de miembros de U el resultado posible es $\varnothing$ (basado en las propiedades de dominación de los conjuntos [4]), dado que $\varnothing \in U$ se cumple la tercera propiedad de espacios topológicos.*

Por lo tanto, (X, U) es un espacio topológico denominado espacio topológico trivial y U. es la topología trivial de X.

Ejemplo B.2

Dado un conjunto X= $\{a, b\}$ y U= $\{\varnothing, \{a\}, X\}$, se desea determinar si X junto con U es un espacio topológico.

- *Puesto que U contiene a los conjuntos triviales se cumple la primera propiedad de los espacios topológicos.*

[2] $A \cup \varnothing = A$
[3] $A \cup A = A$
[4] $A \cap \varnothing = \varnothing$

- *Si se une un número finito de miembros de U los resultados posibles son* $\varnothing, \{a\}, X$ *(con base en el teorema 30, en las propiedades de dominación e idempotentes), dado que* $\varnothing, \{a\}, X \in U$ *se cumple la segunda propiedad de espacios topológicos.*
- *Si se intersecan un número finito de miembros de U los resultados posibles son* $\varnothing, \{a\}, X$ *(basado en el teorema 30, en las propiedades de dominación e idempotentes), dado que* $\varnothing, \{a\}, X \in U$ *se cumple la tercera propiedad de espacios topológicos.*

Por lo tanto, (X, U) *es un espacio topológico y U. es la topología Sierpinski en X.*

Ejemplo B.3

Dado un conjunto X= $\{a, b, c\}$ *y U=* $\{\varnothing, \{a\}, \{b\}, X\}$*, se desea determinar si X junto con U es un espacio topológico.*

- *Puesto que U contiene a los conjuntos triviales se cumple la primera propiedad de los espacios topológicos.*
- *Si se une un número finito de miembros de U los resultados posibles son* $\varnothing, \{a\}, \{b\}, \{a, b\}, X$ *(con base en el teorema 30, en las propiedades de dominación e idempotentes), dado que* $\{a, b\} \notin U$*, NO se cumple la segunda propiedad de espacios topológicos.*
- *Si se intersectan un número finito de miembros de U los resultados posibles son* $\varnothing, \{a\}, \{b\}, X$ *(basado en el teorema 30, en las propiedades de dominación e idempotentes), dado que* $\varnothing, \{a\}, \{b\}, X \in U$ *se cumple la tercera propiedad de espacios topológicos.*

Por lo tanto, (X, U) *NO es un espacio topológico.*

Los conceptos de conjuntos abierto y cerrado son importantes para definir y caracterizar a los espacios topológicos.

Definición 74 *Sea* (X, U) *un espacio topológico y* $U_1 \subset X$*. Se dice que* U_1 *es un conjunto cerrado si* $X \setminus U_1$ *es un conjunto abierto. Es otras palabras, dado una* (X, U)*,el conjunto* U_1 *es cerrado si y solo si* $(X \setminus U_1) \in U$*.*

Las propiedades de los conjuntos cerrados, son: a)los conjuntos vacío y X son cerrados en X; b) la intersección de conjuntos cerrados es cerrada; c) cualquier unión finita de conjuntos cerrados es cerrada. Las propiedades anteriores se pueden deducir al considerar la definición 74 y las leyes de Morgan [5] de teoría de conjuntos. En la proposición 8 se formalizan las propiedades de los conjuntos cerrados [211].

[5] $\overline{A \cup B} = \overline{A} \cap \overline{B}$ y $\overline{A \cap B} = \overline{A} \cup \overline{B}$

Proposición 8 *Sea un espacio topológico* (X, U)*. Entonces*

- *Los conjuntos* $\varnothing$ *y* X *son cerrados.*
- *Si* $\{U_i\}_{i\in I} \subset U$ *entonces* $\bigcap_{i\in I} U_i \in U$
- *Si* $\{U_i\}_{i\in I} \subset U$ *entonces* $\bigcup_{i\in I} U_i \in U$*, tal que* $|I| < \infty$

Lema 5 *El entorno de un punto* $x \in X$ *es todo conjunto abierto que contenga a* x*.*

Lema 6 U_1 *es un conjunto abierto si y sólo si es un entorno de cada* $x \in U_1$*.*

Puesto que los conceptos de conjuntos abierto y cerrado son en cierto sentido duales, se puede conocer la topología de un espacio topológico a partir de conocer a los conjuntos de cerrados.

Definición 75 *Dados un conjunto* X *y una familia de subconjuntos* U *de* X*, si* U *satisface las propiedades de la proposición 8, entonces, existe una única topología* U_i *en* X*.*

Un conjunto X puede contener más de dos topologías. Para comparar dos topologías U_1 y U_2 posibles en un conjunto X, generalmente se hace referencia a si U_1 es más o es menos fina que U_2.

Definición 76 *Dados un conjunto* X *y dos topologías* U_1 *y* U_2 *en* X*, si* $U_1 \supset U_2$ *entonces se dice que* U_1 *es más fina que* U_2*. En caso contrario se dice que* U_1 *es menos fina que* U_2*.*

Lema 7 *No todas las topologías son comparables.*

Se dice que un espacio topológico (X, U) es separado, o Hausdorff, si en él para dos puntos distintos existen sendos entornos disjuntos [28]. En tal caso un punto constituye un conjunto cerrado. Debido a la complejidad involucrada en establecer una topología en un espacio X se recurre a establecer la base de una topología X.

Definición 77 *Una base de una topología* U *en* X *es una subfamilia de conjuntos abiertos* β *tal que todo conjunto abierto es resultado de la unión de elementos de* β*, es decir, para todo* $V \in U$ *existe un conjunto de índices* I*, tal que:*

$$V = \bigcup B_i$$

$$B_i \in \beta$$

Proposición 9 *Dado un espacio topológico* (X, U) *y* β *una base de la topología, entonces:*

- $X = \bigcup_{B\in\beta} B$

- *Si* $B_1, B_2 \in \beta$ *y* $x \in B_1 \cap B_2$ *entonces existe un* $B_3 \in \beta$ *tal que* $x \in B_3 \subset B_1 \cap B_2$

Se denomina como subbase de un espacio topológico (X, U) a una subfamilia β_{sub} de U tal que la colección de todas las intersecciones finitas de elementos de β_{sub} forman una base para la topología U.

Teorema 31 *Si se consideran un conjunto* X *y* β *una familia de subconjuntos de* X *que satisface las propiedades:*

- $X = \bigcup_{B \in \beta} B$
- *Si* $B_1, B_2 \in \beta$ *y* $x \in B_1 \cap B_2$ *entonces existe un* $B_3 \in \beta$ *tal que* $x \in B_3 \subset B_1 \cap B_2$

Entonces existe una única topología U *que tiene como base a* β*, se dice que la topología* U *es generada por* β *y se denota como* $U=U(\beta)$*.*

Un conjunto abierto puede ser definido mediante el concepto de base como lo muestra la siguiente definición:

Definición 78 *Sea* β *una base del espacio topológico* (X, U) *y* $A \subset X$*. Entonces* A *es un conjunto abierto si y solo si para cada* $x \in S$*, existe un* $B \in \beta$ *tal que* $x \in B \subset A$

Dado un (X, U) se dice que $\beta(x) \subseteq P(x)$ es una base de vecindades en $x \in X$ si cada $B \in \beta(x)$ es una vecindad de x y para cada vecindad V de x existe una $B \in \beta(x)$ tal que $B \subseteq V$.

Definición 79 *Dado un* $X \neq \varnothing$*. Una función* $f : x \to \beta(x)$ *es una base topológica de vecindades si satisface las siguientes condiciones:*

- $x \in B$ *para todo* $B \in \beta(x)$*.*
- *Si* $B_1 \in \beta(x)$ *y* $B_2 \in \beta(x)$*, entonces existe un* $B \in \beta(x)$ *tal que* $B_1 \cap B_2 \supseteq B$
- *Si* $B \in \beta(x)$*, entonces existe* $A \in \beta(x)$ *tal que* A *es* $\beta-$*abierto* $A \subseteq \beta$*.*

Sea $f : X \to Y$ una función entre dos espacios topológicos. f es una **función continua** en $x \in X$ si para todo entorno V de $f(x)$ en Y, su anti imagen $f^{-1}(V)$ es entorno de $x \in X$, en otras palabras una función continua no "rompe" lo que esta unido y no "pega" lo que esta separado.

Definición 80 *Una función* $f : X \to Y$ *entre dos espacios topológicos es continua si para cualquier conjunto abierto* G *en* Y*, el conjunto* $f^{-1}(G) = \{x \in X : f(x) \in G\}$ *es un conjunto abierto de* X*.*

La función $f : x \to \beta(x)$ es una base de vecindades, la que genera la topología producto de la familia finita $(U_i)_{1 \leq i \leq n}$ de topologías. El producto topológico de la familia $(X_i, U_i)_{1 \leq i \leq n}$ de espacios topológicos

es el espacio topológico $\left(\prod_{i=1}^{n} X_i; U\right)$ donde U es la topología producto. Cabe mencionar que la topología natural de $\mathbb{R}^n$ es la topología producto si se considera de cada uno de los factores la topología natural en $\mathbb{R}$ [198] Ahora bien, dado un conjunto X y dos topologías U_1 y U_2 generadas de las bases topológicas β_1 y β_2 respectivamente; se dice que β_1 y β_2 son equivalentes si ambas generan la misma topología ($U_1 = U_2$). Si dos espacios tienen exactamente las mismas propiedades topológicas entonces se dice que dichos espacios son homeomorfos.

Definición 81 *Dados X y Y; si existen las funciones continuas $f : X \to Y$ y $g : Y \to X$ que son inversas una de la otra, entonces se dice que X y Y son homeomorfos y se denota por $X \cong Y$. Además, se dice que f y g son homeomorfismos entre espacios topológicos X y Y.*

Apéndice C

Métricas de distancia

C.1. Distancia Euclídiana

Dados los puntos $x = (x_1, x_2, \ldots, x_n)$ e $y = (y_1, y_2, \ldots, y_n)$ la distancia Euclídina es:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{C.1.1}$$

C.2. Distancia Euclídiana Ponderada

Dados los puntos $x = (x_1, x_2, \ldots, x_n)$ e $y = (y_1, y_2, \ldots, y_n)$ la distancia Euclídina ponderada es:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} w_i (x_i - y_i)^2} \tag{C.2.1}$$

donde:w:i son los coeficientes de ponderación o "peso".

C.3. Distancia Euclídiana al cuadrado

Dados los puntos $x = (x_1, x_2, \ldots, x_n)$ e $y = (y_1, y_2, \ldots, y_n)$ la distancia Euclídina ponderada es:

$$d(x, y) = \sum_{i=1}^{n} (x_i - y_i)^2 \tag{C.3.1}$$

C.4. Distancia Manhattan

Dados los puntos $x = (x_1, x_2, \ldots, x_n)$ e $y = (y_1, y_2, \ldots, y_n)$ la distancia manhattan es:

$$d(x, y) = \sum_{i=1}^{n} |x_i - y_i| \tag{C.4.1}$$

A esta distancia también se le conoce como distancia City-Block

C.5. Distancia de Chebychev

Dados los puntos $x = (x_1, x_2, \ldots, x_n)$ e $y = (y_1, y_2, \ldots, y_n)$ la distancia de Chebychev es:

$$d(x, y) = \text{máx}\, |x_i - y_i| \tag{C.5.1}$$

C.6. Distancia de Minkowski

Dados los puntos $x = (x_1, x_2, \ldots, x_n)$ e $y = (y_1, y_2, \ldots, y_n)$ la distancia de Minkowski es:

$$d(x, y) = \left(\sum (|x_i - y_i|)^p \right)^{\frac{1}{p}} \tag{C.6.1}$$

C.7. Distancia de Clark

Dados los puntos $x = (x_1, x_2, \ldots, x_n)$ e $y = (y_1, y_2, \ldots, y_n)$ la distancia de Clark es:

$$d(x, y) = \sum_{i=1}^{n} \frac{|x_i + y_i|^2}{|y_i - x_i|^2} \tag{C.7.1}$$

C.8. Distancia de Canberra

Dados los puntos $x = (x_1, x_2, \ldots, x_n)$ e $y = (y_1, y_2, \ldots, y_n)$ la distancia de Canberra es:

$$d(x, y) = \sum_{i=1}^{n} \frac{|x_i - y_i|^2}{|y_i + x_i|^2} \tag{C.8.1}$$

C.9. Distancia Ji-cuadrada

Ya que generalmente las frecuencias observadas ($x = (x_1, x_2, \ldots, x_n)$) y esperadas ($y = (y_1, y_2, \ldots, y_n)$) son distintas, la distancia Ji-cuadrada (también conocida como chi-cuadrada o Phi-cuadrada) indica si la

diferencia entre x e y son lo suficientemente grandes como para contradecir la hipótesis de que los vetores x e y son homogéneos, o bien las discrepancias entre las frecuencias son causadas sólo al azar.

$$d(x,y) = \sqrt{\sum_{i=1}^{n} \left(\frac{(x_i - y_i)^2}{y_i} \right)} \tag{C.9.1}$$

C.10. Distancia de Hamming

Métrica desarrollada por Richard Hamming en 1951. En términos genérales la distancia de Hamming cuantifica el número de posiciones donde las secuencias son diferentes. A continuación se define de manera formal.

$$\begin{gathered} d(x,y) = \textstyle\sum_{i=1}^{n} z_i \\ \text{donde:} \\ z_i = \begin{cases} 0 & \text{si } x_i = y_i \\ 1 & \text{si } x_i \neq y_i \end{cases} \end{gathered} \tag{C.10.1}$$

donde x, y, son vectores de la misma longitud n

Es quizá la forma más sencilla de distancia y prácticamente cualquier otra forma para la determinación de distancias la contiene. La distancia de Hamming sólo permite el remplazo de caracteres asociado a un costo, sin embargo su inconveninte es estar solamente definida para la comparación de secuencias con la misma longitud.

C.11. Distancia de Disimilitud

La distancia de disimilitud (también conocida como Tversky) contabiliza la parte común y la diferencias entre dos cadenas de caracteres a través de las siguiente equación:

$$d(x,y) = \frac{\alpha(comun)}{(\alpha(comun) + \beta(diferente))} \tag{C.11.1}$$

donde α y β son "pesos".

C.12. Distancia de Levenshtein

En la literatura la distancia de Levenshtein es sinónimo de la distancia de edición y se le considera una generalización de la distancia de Hamming. Es una métrica de distancia que se encuentra definida para secuencia de longitud igual o diferente.

La distancia de Levenshtein es el costo mínimo asociado con las operaciones necesarias para transformar la cadena $(x = (x_1, x_2, \ldots, x_n))$ en la cadena$(y = (y_1, y_2, \ldots, y_m))$ [129]. Las cadenas pueden ser de longitud diferente.

Las operaciones permitidas para transformar una secuencia en otra son:

1. Sustituir
2. Eliminar
3. Insertar

La distancia de Levenshtein se fundamenta en el concepto de Homemorfismo que permite transformar un conjunto x en un conjunto y.

C.13. Distancia de Indel

Es una variante de la distancia de Levenshtein en la que sólo se permiten las operaciones inserción y eliminación de caracteres, ya que se utiliza el supuesto de que la operación de sustitución es equivalente a insertar y eliminar un carácter en un punto específico de la secuencia. Por lo que, la función de distancia queda expresada solo en términos de dos operaciones básicas de edición.

C.14. Distancia de Damerau

Se le considera como la generalización del procedimiento de Levenshtein, ya que su única diferencia es la incorporación de la operación de transposición de caracteres adyacentes.

Su objetivo es encontrar el número mínimo de operaciones básicas (trasponer, sustituir, insertar y eliminar) para transformar una cadena la cadena $(x = (x_1, x_2, \ldots, x_n))$ en la cadena$(y = (y_1, y_2, \ldots, y_m))$.

Apéndice D

Implementaciones especiales del método simplex

D.1. Método de la gran M y Método de las dos fases

Estos métodos se implementan ante la presencia de **variables artificiales**[1] en el modelo a solucionar. Estos métodos estan estrechamente relacionados [232].

El método de la gran "M" -también conocido como método de penalización o técnica M de Charnes [225, 226]- asigna una penalización, de valor M^2, en la función objetivo a cada una de las variables artificiales. La penalización tiene como objeto consguir que las varibles artificiales asuman un valor de cero en la solución final. El procedimento del método de la gran "M" se muestra en el algoritmo 58

El método de las dos fases, como su nombre lo indica, consiste resolver la instancia de interes en dos

[1] Una variable artificial desempeña una función similar a variable de holgura; ya que proporciona una variable básica inicial para aquellas resticciones del tipo = y $\geq$. Cabe mencionar, que las varibales artificiales no tienen sentido fisíco; por ende sólo se utilizan para generar una solución inicial y se busca que tengan un valor cero en la solución final, de lo contrario la solución resultante será no factible.

[2] M es un número positivo suficientemente grande, teóricamente se requiere que $M \to \infty$. Se asigna un coeficiente M a las variables artificiales en un problema de minimización; en contraste, se asigma un coeficiente $-M$ a las variables artificiales en un problema de maximización

Algoritmo 58: Método de la gran "M"

```
1  Expresar la instancia de PL en su forma estándar.
2  Introducir una variable artificial en la i−ésima restricción si no tiene una variable de holgura.
3  Incluir las artificiales con su correspondiente coeficiente en la función objetivo.
4  LLevar a cero los coeficientes de las variables artificiales del renglón asociado a la función
   objetivo a través de operaciones elementales.
5  Aplicar el algoritmo simplex para resolver el problema.
6  if Todas las variables artificiales son igual a cero en la solución óptima then
7  |  Se ha encontrado la solución óptima del problema original
8  else
9  |  Se ha encontrado que el problema original es no factible
10 end
```

etapas o fases. **En la primera fase** se busca minimizar la suma de las variables artificiales (mín $\sum x_a$); sujeto a: $Ax + x_a = b$, $x \geq 0$ y $x_a \geq 0$; si el valor óptimo para este problea es cero y todas las variable artificiales toman un valor igual a cero entonces la instancia original tiene una solución factible, por lo que se procede con la segunda fase; en contraste, si al resolver este problema se determina que alguna variable artificial es positiva entonces el problema original no tiene solución, por lo que se da por teminado el agoritmo. **En la segunda fase** se inicia con base en el tablero final de la primera fase, se retoma la función objetivo de la instancia original, haciendo todas las variables artificiales iguales a cero y se eliminan de las restricciones.

D.2. Método simplex revisado

Es un procedimento sistemático, para implementar el algoritmos simplex en un arreglo más pequeño, con lo cual se ahorra espacio de almacenamiento [14]. Otra de las ventajas de éste método es reducir errores por redondeo dentro de límites razonables determinando la inversa de la base en cualquier iteración [9].

La diferencia de entre el método simplex y el revisado radica en el elemento básico para realizar los cálculos; ya que el primero utiliza renglones de la matriz A, en contraste el segundo utiliza a la matriz

báasica“B” y su inversa. En el algoritmo simplex revisado se utiliza el arreglo tabular mostrado en la tabla D.1

Tabla D.1: Estructura Tabular del método simplex revisado

Báse inversa	Lado derecho
w	$c_B\bar{b}$
B^{-1}	$\bar{b}$

El procedimento del algoritmo simplex revisado se muestra en el Algoritmo 59

D.3. El algoritmo de descomposición

El algoritmo de descomposición, basado en el principio de descomposición, consiste en operar sobre dos programas lineales separados uno sobre el conjunto de las restricciones complicadas (complicantes) y otro sobre el conjunto de restricciones especiales [14].

Al problema lineal sobre las restricciones generales se llama **programa maestro**; en contraste, se denomina **subproblema** al problema lineal sobre las restricciones especiales.

Como se mencionó, en el capítulo 1, se dice que un problema Π a es “separable” , si y sĺo si Π se puede dividir en p problemas más pequeños y que la solución de Π se obtiene a través de la solución de los p problemas. Dado que, el conjunto poliédrico $x \in \mathbb{R}^n$ del problema de PL, puede ser representado a través de la combinación lineal de los puntos extremos de x; el problema de optimización puede ser expresado como lo muestra la ecuación D.3.1.

Algunos autores consideran que el algoritmo de descoposición puede ser visto como una mejora del algoritmo simplex revisado [193]. El algoritmo descomposición utiliza un arreglo tabular como se muestra en la Tabla D.3

$$\begin{gathered} \text{mín} \sum_{j=1}^{p} \lambda_j (cx_j) \\ \text{sujeto a:} \\ \sum_{j=1}^{p} (Ax_j)\lambda_j = b \\ \sum \lambda_j = 1 \\ \lambda_j \geq 0 \ j = 1, 2, \ldots, p \end{gathered} \tag{D.3.1}$$

Algoritmo 59: Algoritmo Simplex revisado

```
1  Determinar una solución básica fatibe inicial con base inversa B^{-1}
2  Calcular w = c_B B^{-1}, b̄ = B^{-1} b
3  Configurar un tablero para el método simplex revisado (véase D.1)
4  óptimo= 0
5  no acotado= 0
6  while óptimo= 0 y no acotado= 0 do
7      Calcular z_j − c_j = w a_j − c_j, para cada variable no básica.
8      Realizar prueba optimalidad
9      if La solución es óptima then
10         óptimo= 1
11     else
12         Determinar qué vector a_k entrará la base
13         Calcular y_k = B^{-1} a_k
14         if y_k ≤ 0 then
15             no acotado= 1
16         else
17             Calcular el radio y_0 / y_k para determinar qué vector sale de la base
18             Actualizar B^{-1}
19             Recalcular w = c_B B^{-1}, b̄ = B^{-1} b
20         end
21     end
22 end
```

Algoritmo 60: Algoritmo de descomposición

óptimo = 0

no acotado = 0

Reformular el problema de optimización de interés (véase ecuación D.3.1)

Encontra una solución básica factible del problema anterior.

Contruir el arreglo tabular correspondiente con B

while *óptimo* $= 0$ *y no acotado* $= 0$ **do**

Resolver el subproblema

$$\text{máx}(wA - c)x + \alpha$$

$$\text{suejto a:} x \in X$$

Determinar el valor de $z_k - \hat{c}_k$

if $z_k - \hat{c}_k = 0$ **then**

óptimo = 1

% La solución básica factible del problema maestro es la solución óptima del problema original

else

Determinar $y_k = B^{-1} \begin{bmatrix} Ax_k \\ 1 \end{bmatrix}$ Actulizar la columna $\begin{bmatrix} Ax_k \\ 1 \end{bmatrix}$ en el arreglo tabular del problema maestro.

if *Al meno un* sy_{ik} *es mayor que cero* **then**

Determinar el valor de r a través de:

$$\frac{\bar{b}_r}{y_{rk}} = \min_{1 \leq i \leq m+1} \left\{ \frac{b_i}{y_{ik}} : y_{ik} > 0 \right\}$$

Pivotear sobre y_{rk}

Eliminar la columna λ_k

else

no acotado= 1

end

end

end

Tabla D.2: Estructura Tabular del método de descomposición

Báse inversa	Lado derecho
(w, α)	$\hat{c}_B \bar{b}$
B^{-1}	$\bar{b}$

donde: $\hat{c}_j = cx_j$, $w = \hat{c}_B B^{-1}$, $barb = B^{-1} \begin{bmatrix} b \\ 1 \end{bmatrix}$

D.4. Simplex dual

En términos generales, se dice que el método Simplex dual consiste en la solución de una instancia de PL desde el punto de vista dual; a través de trabajar desde la tabla primal del problema. Este método es aplicable cuando exista un lado derecho negativo en alguna de las restricciones y no se satisfaga el criterio de paro [248]. Es decir, que la solución sea no factible, pero si óptima.

El procedimento del simplex dual se muestra en el Algoritmo 61

Existen otras variantes del método simplex, como: simplex para variables acotadas, simplex para redes, entre otros. Para mayor información véase [248, 14, 179, 121].

Algoritmo 61: Algoritmo simplex dual

óptimo = 0 no acotado = 0 **while** *óptimo* = 0 *e infactible* = 0 **do**
    if *El lado derecho de todas las restricciones es no negativo* **then**
        óptimo = 1
    else
        Elegir la variable la básica más negativa como la que saldrá de la báse.
        if *Todos los coeficientes de reemplazo con las variables no básicas son no negativos* **then**
            no acotado = 1
        else
            Determinar la columna pivote, a través

$$\max_{j \text{tal que: } x_{rj}} \frac{x_{0j}}{x_{rj}} []$$

            Aplicar la operación de pivoteo para generar una nueva tabla
        end
    end
end

Apéndice E

Características básicas de las redes sociales

- **Tamaño**. Es el número de actores que participan en la red social.
- **Densidad** Describe, en general, el nivel de unión entre los nodos de una gráfica.
- **Centralidad** Es un atributo estructural de los nodos en un gráfo, para su descripción se asocia un valor numérico cada nodo en función de su posición estructural en la gráfica. Las medidas más importantes de centralizad son: a) grado de centralizado (número de vínculos que posee un nodo con otros), b) cercanía (se basa en la distancia de un nodo a un conjunto de nodos) c) intemediación (frecuencia con la que un nodo aparece en la ruta más corto que conecta otros dos nodos)
- **Recorrido** Es una sucesión finita alternada de vértices y aristas; esta sucesión inicia en un vértice v_0 y termina en un vértice v_n
- **Ruta** Es un recorrido en el cual ninguna arista se repite.
- **Camino** Es un recorrido en el cual ningún vértice se repite.
- **Longitud de una ruta** Es número de aristas en la ruta.
- **Distancia entre un par de nodos** Es la ruta más corta que conecta a dichos vértices.
- **Dispersión** Describe, en general, el nivel de unión entre los nodos de una gráfica y en particular, sirve para definr la distancia entre los miembros de la red [176]

- **Coeficiente de agrupamiento** Probabilidad de que dos personas vinculadas a un nodo se asocien entre ellas.
- **Homogenidad o heterogeneidad** Es el grado de semejanza o diferencia entre los integrantes de una red.
- **Atributos y Vínculos especfícos** Tales como el compromiso y carga de la relación; durabilidad, historia en común.
- **Tipos de funciones**: cumplidas por cada vínculo y por el conjunto.

Apéndice F

Ajuste de parámetros

Un balance adecuado entre las fases de diversificación e intensificación de una metaheurística depende primordialmente de una apropiada determinación del valor de los parámetros [133]. Por lo tanto, el ajuste del valor de los parámetros es una actividad crucial para la implementación de una metaheuristica, pues las desiciones emanadas de esta actividad, influiran en gran medida en los resultados obtenidos para algún problema. A manera de ejemplo: considérese que se está ejecutando un algoritmo genético (AG) para resolver un problema, a medida que la solución encontrada por el AG se aproxima a la solución optima se esperaría que el factor de mutación decreciera y con ello se intensificara la búsqueda sobre la regiones prometedoras.

En el ajuste de los parámetros en los algoritmos evolutivos, pueden distinguirse principalmente: a) **la afinación de parámetro**s (o calibración de parámetros) y b) el **control de parámetros**. La calibración de parámetros involucra determinar un valor estático (a través de la información disponible y experimentación) antes de la ejecución del método; en contraste el control involucra cambios dinámicos en el valor de un parámetro durante la ejecución del algoritmo.

Generalmente, se requiere un adecuado manejo de ambas técnicas para alcanzar solución satisfactoria a un problema. Lo anterior se esquematiza en la Figura F.1.

El ajuste de parámetros es una tarea ardua; que involucra la toma de decisiones sobre los siguentes aspectos:

- Definir el espacio de soluciones.
- Representación de individuos
- Definir la estructura de vecindarios

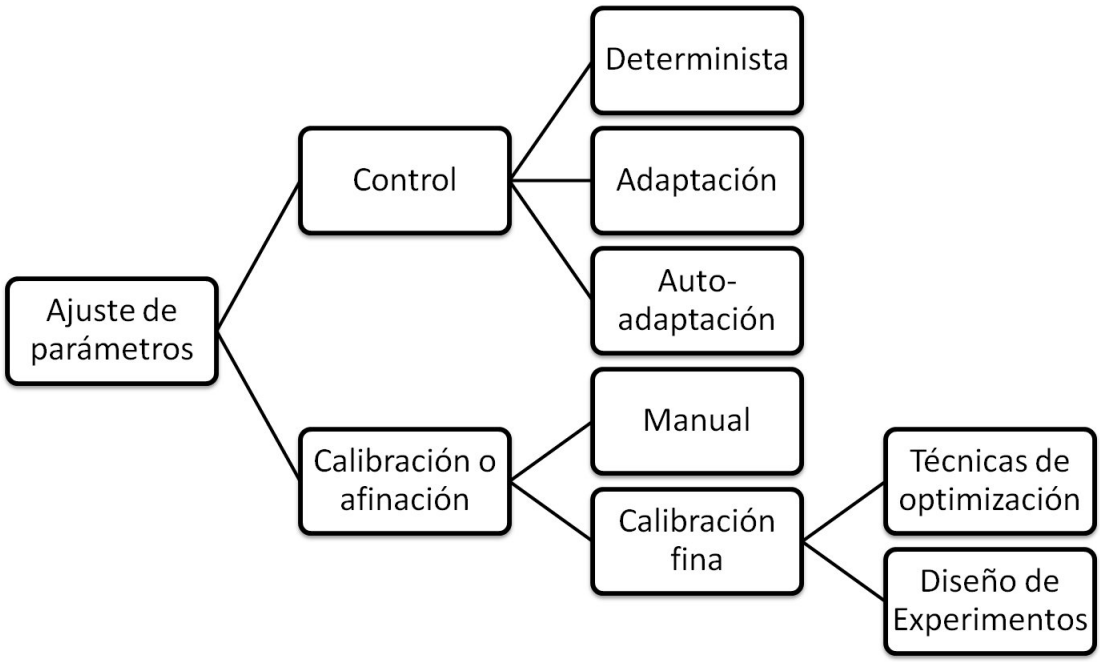

Figura F.1: Clasificación de los métodos de elección de parámetros.

- Función objetivo (función de aptitud)
- Valor de los parámetros y asociación de probabilidades

Se debe tener en cuenta que las desiciones tomadas tendran un efecto muy significativo en la calidad de la solución final alcanzada; pues una apropiada configuración de parámetros tendrá un impacto substancial en el proceso de solución (en las fases de intensificación y diversificación) y en la calidad de la solución encontrada [56].

Los parámetros de una metahuritica se pueden clasificar en los siguientes tipos:

- Categóricos, por ejemplo: La estrategia de selección en los AG, esquema de cooperación en recocido simulado paralelo, etc.
- Cualitativos
  - Discretos, por ejemplo: Tamaño de la población en AG, número de hormigas; tamaño de la memoria tabú en búsqueda tabú, tamaño de la menoría harmónica en búsqueda armónica, etc.
  - Continuos, por ejemplo: Probabilidad de cruza en AG; temperatura inicial en recocido simulado, parámetro de ajuste del ritmo en búsqueda harmonica, etc.

En las siguientes secciones se analizan los métodos susados en el ajuste de parámetros.

F.1. Afinacción de parámetros

Cada una de las metaheurísticas actuales establece un conjunto de parámetros[1]. Por lo tanto, antes de ejecutar a cualquier procedimiento metaheurístico para resolver algún problema, es necesario establecer una configuración del conjunto de argumentos, la cual puede ser estática (con lleva a una afinación de parámetros), o dinámica (implica un control de parámetros), considérese el ejemplo F.1:

Ejemplo F.1

Un tomador de decisiones TD_1 debe resolver el problema de la función de Rastrigin en dos dimensiones (su formulación matemática se muestra en F.1.1), para ello TD_1 decide utilizar un algoritmo genético simple, para lo cual el científico deberá asignar los siguientes parámetros: a)tamaño de la población, b)número de generaciones c)probabilidad de cruce d)probabilidad de mutación, antes de ejecutar el AG.

$$\begin{gathered} \text{mín } f(x) = \sum_{l=1}^{2} x_l^2 - 10\cos(2\pi * x_l) + 10 \\ -5.12 \leq x_l \leq 5.12 \text{ for all } l = 1, 2 \end{gathered} \tag{F.1.1}$$

El problema de calibración de parámetros puede ser descrito de manera general como: dada una metahuristica a, determinar la mejor configuración θ con base en un criterio $\mathcal{C}$ dentro en un tiempo T. A continuacón se formaliza el problema general de afinación de parámetros [19]:

Definición 82 *El problema de calibración de parámetros formalmente se determina por una tupla $(\Theta, I, P_I, P_C, t, \mathcal{C}, T)$ una solución de este problema involucra encontrar una configuración $\bar{\theta}$ tal que:*

$$\bar{\theta} = \arg\min_{\theta} \mathcal{C}(\theta)$$

donde:

Θ *Es el conjunto de configuraciones candidatas $\Theta : \{\theta_1, \theta_2, \ldots, \theta_k\}$. I es generalmente un conjunto infinito de instancias. P_I es una medida de probabilidad sobre el conjunto I de instancias, de forma que $P_I(i)$ es la probabilidad de que la instancia i sea seleccionada para ser resuelta. $t : I \to \mathbb{R}$ es una función asociada a cada una de las instancias y el tiempo asociado para su computo. $\rfloor$ es una variable aleatoria que representa el costo de la mejor solución encontrada por utilizar la configuración θ sobre la instancia i para el tiempo $t(i)$. $C \subseteq \mathbb{R}$ es el rango de $\rfloor$ que es el posible valor de la mejor solución encontrada por una configuración $\theta \in \Theta$ sobre una instancia $i \in I$. P_C es una probabilidad medida sobre el conjunto C, tal que $P_C(\rfloor|\theta, i)$ indica la probabilidad que $\rfloor$ sea el costo de la mejor solución encontrada por una corrida en el tiempo $t(i)$ con una*

[1] Un parámetro es factor (variable o argumento) necesario para analizar o valorar una situación, en otras palabras un par.

configuración θ sobre una instancia i. $\mathcal{C}(\theta) = \mathcal{C}(\theta|\Theta, I, P_I, P_C, t)$ es el criterio que debe ser optimizado con respecto de θ. Formalmente, se define el citerio de $\mathcal{C}$ como: $\mathcal{C}(\theta) = E_{I,C}(\mathcal{C}) = \int \mathcal{C} dP_C(\mathcal{C}|\theta, i) dP_I(i)$ que es la esperanza con respecto a P_C y P_I y la integración es el sentido de la integral de Lebesgue [2] T es el total de tiempo disponible para la experimentación, antes de seleccionar la configuración.

Las propiedades y los valores obtenidos en configuración de parámetros depende de los siguientes factores:

- Caracteriticas del citerio $\mathcal{C}$ seleccionado.
- Caracteristicas de la metaheuritica para la cual se búsca la configuración de parámetros.
- Conjunto de instancias de prueba.
- Mecanismo utilizado para la calibración.

La calibración de parámetros tiene los siguientes propósitos:

- Obtener un particularización de la metaheurística, la cual tenga un buen comportamiento con base a alguna criterio dado.
- Obtener una particularización de la metaheurística, la cual sea robusta a cambios específicos en las instancias a resolver.
- Obtener una particularización de la metaheurística, la cual se robusta al efectos aleatorios (ruido) durante su ejecución.
- Analizar la robustez de la metaheurística dada a cambios en los valores de la configuración.

F.1.1. Calibración Manual

La calibración manual depende de la experiencia y conocimiento adquirido del usuario. Esta práctica involucra alta subjetividad, por ende, se tiene poca confiabilidad de que la configuración encontrada de los parámetros sea la mejor posible.

F.1.1.1. Tomar una configuración disponible en literatura

Esta práctica consiste en asignar a los parámetros de una metaheurística "a" aquellos valores reportados en la literatura para intancias similares resueltas de manera exitosa con "a" (o alguna de sus variantes).

[2] La integral de Lebesgue es una contrucción matemática que permite exter el concepto de la integral de Riemman de tal forma que se pueden integrar funciones más generales, tratar simultaneamente funciones acotadas y no acotadas y replazar el intervali $[a, b]$ por conjuntos más generales.

En [4] se indica que esta práctica es común en las implementaciones de recocido simulado y los algoritmos genéticos.

Por ejemplo, para recocido simulado, White [246] sugiere como una estrategia para determinar T_0 calcular la desviación estándar de la función objetivo o energía ($\sigma(f(x))$) de un conjunto aleatorio de soluciones factibles; pues ($\sigma(f(x))$) de la distribución de energía define la máxima temperatura de modo tal que $T_0 \approx \sigma(f(x))$ [13] (en [210] $T_0 \approx \sigma(f(x))...2(f(x))$). Adicionalmente White propone que el mínimo cambio de energía observado sobre el conjunto aleatorio define la temperatura final [13]. Algunos trabajos han utilizado esta información sin evluar si esta resulta adecuada dado el contexto de la investigación.

No existe una configuración general de los parámetros de alguna metaheurística; pues la información necesaria para una calibración adecuada de los parámetros cambia de problemas a problema. Con base en lo anterior, y el teorema de "No Free Lunch" (NFL), es evidente que asignar los parámetros de la metaheurística "*a*" conciderando solamente la información disponible en literatura sobre casos semejantes (resueltos exitosamente por "*a*" o alguna de sus variantes) a la instancia a resolver generará un arreglo de valores inadecuado; lo que repercutirá en el proceso de solución del problema de interés, considérese el ejemplo F.2

Ejemplo F.2

Continuando con el ejemplo F.1. Con el fin de establecer una configuración adecuada de los parámetros el investigador I_1 realiza una revisión sobre los trabajos relacionados con algoritmos genéticos usados para resolver instancias de problemas de optimización no lineal (PLN). Al cabo de un tiempo I_1 da con el [50], en el cual se determina la mejor configuración de parámetros para un algoritmo genético simple usado para resolver un conjunto de instancias de problemas PLN (llamado Environment E), en la Tabla F.1 se muestra la configuración encontrada por De Jong, que genera los mejores resultados para el conjunto de problemas de prueba [50, 64].

El tomador de decisiones TD_1 nota que en Environment E no se incluye la función de Rastrigin en dos dimensiones. Por lo anterior, TD_1 debe decidir si tomar la asignación propuesta por De Jong para un conjunto de problemas similares al que él desea resolver o bien utilizar algún otro procedimiento a fin de encontrar una asignación adecuada a los parámetros del AG sobre el caso de interés. Para tomar su decisión TD_1 consulta varias fuentes siendo una de ellas [249], en dicho artículo se indica que debido al estrecho rango de instancias consideradas en un estudio, se debe ser muy cuidadoso al tratar de generalizar los resultados. Por esta razón TD_1 decide no utilizar la configuración propuesta por De Jong.

Tabla F.1: Configuración de parámetros de un AG para resolver Environment E

Parámetro	Valor
Tamaño de la población	50
Probabilidad de cruza	0.6
Probabilidad de mutación	0.001
Brecha generacional	100 %
Estrategía de selección	Elitismo

F.1.1.2. Prueba y error

En la mayoría de las invetigaciones, los paráteros de las metaheurísticas son calibrados a través de un procedimento de prueba y error [19].Esta práctica depende en gran medida del tomador de decisiones y consiste cambiar la configuración de los parámetros θ sobre I_M hasta que los resultados obtenidos sean satisfactorios para el decisor. Lo anterior, se muestra en el Algoritmo 62.

Algoritmo 62: Pseudocodigo del método de prueba y error

```
1 Implementar la metahurística con una configuración θ de los parámetros.
2 while Alcanzar resultados satisfactorios do
3 |   Evaluar la metahurística con una configuración θ.
4 |   Modificar la configuración θ.
5 end
```

Ajustar los parámetros mediante este método conlleva a los siguientes inconvenientes:

- Se requiere una cantidad considerable de recursos (computacionales y humanos) a fin de alcanzar una adecuada aproximación de los parámetros.
- Es necesario que el encargado de la calibración posea los conocimientos (sobre la metaheurística el problema, la implementación. entre otros) suficientes a fin de tomar la mejor decisión posible.
- Generalmente los resultados no son los mejores posibles, pese a utilizar una gran cantidad de recursos.

Laguna se señala que la mayoría de las investigaciones desarrolladas a través de esta práctica hacen poca

referencia al proceso implicado en la asignación de los valores a los parámetros, así como un escaso análisis de sensibilidad sobre dichos valores [4].

A causa de alta subjetividad implicada en este proceso, se tiene poca confiabilidad de que la configuración de parámetros obtenida sea la mejor posible; así mismo, se conjetura que el desempeño de la metaheurística al resolver alguna intancia usando dicha asignación de valores tampoco será la mejor. Es decir, dado que la calibración de los parámetros depende del usuario, la reproductibilidad de una metaheurística resulta difícil; así mismo la comparación entre diferentes algoritmos paramétricos [210], considerese el ejemplo F.3

Ejemplo F.3

Se pide de manera independiente a los investigadores I_1 e I_2 calibrar los parámentos de los algoritmos A_1 y A_2 para resolver una instancia ϕ de un problema de optimización. Cuando ambos investigadores terminan el ajuste de los parámetros resulta que A_1 tiene un mejor comportamiento que A_2 al resolver ϕ al usar los valores determinados por I_1, en contraste A_1 tiene un mejor comportamiento que A_2 sobre ϕ si fue ajustado por I_1.

Algunos trabajos desarrollados con esta practica son:

- Pilski and Franciszek Seredyński, 2006 [190], comparan el comportamiento de las metaheurísticas: a) optimización por nubes de partículas (PSO), b) sistema inmune artificial (AIS) y c) algoritmo genético (AG) sobre un conjunto de cinco instancias de optimalización global. [3] .

  El conjunto de prueba se formo por intancias de: a)la función esfera, b) la función Griewank, c) la función Rastrigin d) la función Rosenbrock y e) la función Schwefel en 5, 10, 20 y 30 dimensiones. En las Tablas F.2, F.3, F.4 se muestra la configuración de los parámetros usados.

  Los resultados experimentales muestran que PSO y AG se comportan mejor que AIS al resolver instancias de funciones multivariables. Las conclusiones no son contundentes pues los autores expresan la necesidad de realizar otras pruebas a fin de definir correctamente la eficiencia lineal de los métodos comparados.

- Czech y Czarnas en [47] adaptan el recocido simulado paralelo para resolver el problema de ruteo de vehículos con ventanas de tiempo (VRPTW por las siglas en inglés de *vehicle routing problems*

[3] Global optimización es una rama de las matemáticas y análisis numérico [245], la cual tiene por objeto encontrar el optimo global da una función (o conjunto de funciones) o bien probar que dicho punto no existe.

Tabla F.2: Configuración de parámetros PSO

Parámetro	Valor
Tamaño de la población	50 (sólo en Rastrigin 100)
p_1 y p_2	cercano a 2

Tabla F.3: Configuración de parámetros GA

Parámetro	Valor
Tamaño de la población	40 a 90
Coeficiente de mutación	0.01
Coeficiente de cruza	0.8 (sólo en Rosenbrock 0.95)

with time windows). Ellos obtuvieron buenos resultados mediante el recocido simulado paralelo sobre un conjunto de instancias benchmark (propuestas por Solomon en 1987) utilizando la siguiente configuración:

- La temperatura inicial del alineamiento es $T_0 = \gamma * f(x_0)$ donde: $\gamma \in [0.001, 1]$ y $f(x_0)$ es el valor de la función objetivo evaluada en la solución inicial ($f(x) = d + \sigma(cn * e_{min})$).
- Un esquema de enfriamiento geométrico $T_{i+1} = T_i * \beta$ donde $\beta = 0.94$.
- El número de pasos ejecutados por el alineamiento en cada temperatura osilla entre n^2 y $10 * n^2$ con $n = 100$.
- El criterio de paro es dirigido por el equilibrio del sistema, si la mejor solución permanece sin cambio por un periodo τ de iteraciones ($\tau = 20, .., 40$).
- La importancia relativa del número de rutas con respecto a la distancia de viaje. Para lo cual, se

Tabla F.4: Configuración de parámetros AIS

Parámetro	Valor
Tamaño de la población	100
Número de anticuerpos	100 (sólo en Schwefel 70)
Factor de multiplicación	Esfera y Griewank 9, Rastrigin 10, Schwefel 7 y Rosenbrock 6

guía la configuración de las rutas generadas por medio de minimizar el valor de $cn * e_{min}$ donde c es el número de rutas en la solución y e_{min} es el número de cliente en la ruta más pequeña. Además, se debe considerar que la distancia del viaje debe satisfacer $\sigma(cn * e_{min}) \gg d$ para algún $\sigma \in [0.5, 5]$

- Esquema de cooperación: $V^1_{r+1} = P_T(V^1_r)$, $V^j_{r+1} = P_T(V^j_r)$ para toda $j \neq 1$ y si $r + 1 \neq u * w$; $V^j_{uw} = P_T(V^j_{uw-1})$ si $cost(P_T(V^j_{uw-1})) \leq cost(V^j_{uw-1})$; $V^j_{uw} = V^{j-1}_{uw}$ en cualquier otro caso; donde $w = n = 100$.
- Frecuencia de iteración durante el número de pasos ejecutados en una temperatura se asume que el sistema interactuó alrededor de w veces.
- Número de cooperaciones del proceso $p = 5$

F.1.2. Calibración fina

La calibración fina de los parámetros es un proceso sistematizado para la asignación de los valores alos parámetros, a través, de utilizar técnicas y procediemientos del diseño experimentales y/o la optimización. Independientemente del método de afinación de parámetros usado, debe hacerse notar que un conjunto único de valores no se pueden generalizar a todo los problemas. Es decir, por medio de este procedimiento se encuentran un conjunto *ad-hoc* para un subconjunto de instancias especifico.

Las tecnicas de calibración fina se utilizan en el estudio de la relación entre la respuesta y la configuración de los parámetros; estas técnicas tienen por objeto encontrar una configuración que genere la mejor respuesta posible, considerando el contexto de la invetigación y los recursos disponibles.

F.1.2.1. Diseño experimental

Esta práctica consiste en realizar un conjunto de experimentos (típicamente se calibra uno en uno a los parámetros); posteriormente, se analiza la información obtenida utilizando herramientas estadísticas. Es decir, una alternativa para la calibración es recurrir al diseño de experimentos (DoE) [4]; pues la DoE es un marco de referencia para la conducción de experimentos representativos [56]. Laguna y Adenso-Díaz manifiestan que el diseño experiental involucra las reglas y los procedimientos mediante los cuales se asignaran y seran tratadas la unidades experimentales [4].

[4]Un experimento diseñado es una prueba o serie de pruebas en las cuales de inducen cambios deliberados en las variables de entrada de un proceso o sistema, de manera que sea posible observar e identificar las causas de los cambios en la respuesta de salida

El DoE ofrece una manera objetiva y practica para determinar una configuración óptima de los parámetros [124]. De manera general, esta práctica involucra las siguientes actividades:

- Determinar los objetivos de la experimentación.
- Seleccionar el conjunto de parámetros a analizar.
- Elegir el procedimiento que se utilizara para buscar la apropiada configuración de parámetros.
- Ejecutar el procedimiento seleccionado.
- Analizar los resultados experimentales.
- Asignar a los parámetros la configuración determinada a partir de los resultados obtenidos.

La calibración de parámetros por medio de la experimentación conlleva a los siguientes inconvenientes [152, 64]:

- Los parámetros no son independientes por lo que se debe analizar todas combinaciones posibles, sin embargo, el costo asociado puede ser prohibitivo.
- Aúnque no se consideraran las interacciones entre los parámetros y se calibrarán uno a uno los parámetros, dicho proceso requeriría un tiempo considerable.
- Existe la posibilidad que para un problema dado, la configuración obtenida no sea la óptima, pese a que utilizar una gran cantidad de recursos.

Los procedimientos basados en DoE se clasifican como: A) Métodos muéstrales y B) Métodos muéstrales iterativos. A continuación, se describen cada uno de estos grupos; además, de mecionar alguinos de los trabajos realizados en cada uno de ellos.

A) Métodos muéstrales.

Son aquellos procedimientos que reducen el espacio de búsqueda a través de eliminar el número de parámetros a probar, con respecto al total requerido un diseño experimental factorial [63]. En un DoE factorial se seleccionan k factores ($k \geq 2$) que afectan el comportamiento del sistema, cada uno con determinados "niveles", se debe satisfacer que las unidades experimentales cubren todas las posibles combinaciones entre todos los niveles de los factores, considérese el ejemplo F.4.

Ejemplo F.4

En un sistema se tienen 3 factores (A, B, C)*, cada uno de los factores tiene 2 niveles; por lo tanto hay ocho combinaciones posibles; pues* $2^k = 2^3 = 8$

Sin embargo,en la calibración de parámetros generalmente el número de combinaciones en un diseño factorial completo es demasiado grande para ser procesado; por lo cual, se recurre a un DoE factorial fraccional (métodos muestrales), estos diseños utilizán una conjunto de experimentos tomados estrategicamente del total de experimentos posibles en un DoE factorial [4].

Los métodos muéstrales más comúnmente utilizados en la configuración de parámetros son: **cuadro latino** [5] y las **matrices ortogonales Taguch**i [63]. El cuadro latino fue ocupado por primera vez por Myers y Hancock en la configuración de parámetros de un AG [164]. Las matrices ortogonales Taguchi fueron introducidos por Taguchi como una alternativa para el DoE fraccional [231, 4, 63] .

Algunos trabajos desarrollados por medio de esta practica son:

- Las primeras ideas sobre la calibración de parámetros en la computación evolutiva pueden encontrarse en los trabajos de Rechenberg y Schwefel [19].
- De Jong utilizo un diseño experimental para determinar la mejor configuración de valores para los parámetros de un AG al resolver un conjunto de instancias de prueba, denominado "Environment E". Dicho conjunto de prueba se integraba por una colección de instancias representativas de los problemas de optimización no lineal; las cuales fueron: F_1 función esfera en 3 dimensiones, F_2 función Rosenbrock en 2 dimensiones, F_3 función paso en 5 dimenciones F_4 función cuadratica con ruido en 30 dimensiones y F_5 función de Shekel en 2 dimensiones [50]. Los resultados encontrados por De Jong son mostrados en la Tabla F.1.
- Greffenstette experimentó con un AG sobre los problemas de prueba de "Enviroment E" untilizando dos diseños experimentales diferentes [95]. La mejor configuración de valores encontrados por Greffenstette se muestra en la Tabla F.5.
- Xu,Chiu y Glover, en el contexto del problema de Steiner Tree-star, realizaron dos pruebas estadísticas sobre un pequeño número de experimentos; con base en los resultados, se genero una configuración adecuada de cinco parámetros de búsqueda tabú; además, el diseño experimental propuesto en este trabajo es ocupado actualmente en la afinación de los parámetros para las metaheurísticas usadas para resolver instancias del problema de ruteo de vehículos (VRP), pues dicho diseño experimental genera resultados adecuados [44].
- Xu y Kelly identificaron la contribución relativa de cinco diferentes componentes de búsqueda tabú en siete instancias de VRP con base a los resultados obtenidos en esta investigación ellos concluyeron que

[5] Arreglo matricial de las unidades experimentales

Tabla F.5: Configuración de parámetros experimentos de Greffenstette

Parámetro	Valor primer exprimento	Valor segundo experimento
Tamaño de la población	30	80
Probabilidad de cruza	0.95	0.45
Probabilidad de mutación	0.01	0.1
Brecha generacional	100 %	90 %
Estrategía de selección	Elitismo	No elitismo

la memoria tabú y las estrategias de recomenzar y recuperar le ayudan a TS para encontrar buenas soluciones en el problema VRP [44, 19].

B) Métodos muéstrales iterativos.

Los métodos muéstrales iterativos involucran una experimentación secuencial, pues la aproximación a la configuracón de interes se realiza de forma iterativa utilizando esperimentaciones y concideren la información obtenida en las etapas anteriores. En los ultimos años se han desarrollado varios métodos muestrales iterativos, algunos son:

- El método CALIBRA fue desarrollado por Adenso-Díaz y Laguna, 2006 [4], en este método se combina el diseño experimental (matrices ortogonales de Taguchi) y la búsqueda local. En la Figura F.2 se esquematiza el método CALIBRA.
- El Calibración de parámetros a través de un análisis experimental fue dearrollado por Ridge y Kudenko, 2007 [202], en este trabajo se utilizo un DoE factorial fraccional (2_{IV}^{12-5} con 8 replicas y 24 puntos centro) para calibrar los parámetros de Sistema de colonia de hormigas (ACS) sobre instancias del problema del agente viajero (TSP); además, se utilizo un ANOVA para el análisis de datos.
- El método REVAC (Estimación de Relevancia de la Calibración de Valores) fue propuesto por Nannen y Eiben, 2007 [166] para la calibración de parámetros de algoritmos evolutivos de manera sitematica y semi-automatica. Este método se basa en la teoría de medición de relevancia de los parámetros. El proceso de calibración por REVAC implica determinar la distribución de probabilidad por parámetro. De manera inicial, todos los parámetros se iniciarón con una distribución uniforme; en cada iteracion se fue acualizando la probabilidad en función de la información obtenida de la función de entropia de Shannon ($H(A) = -\sum_{i=1}^{n} p_i \log_2 p_i$).

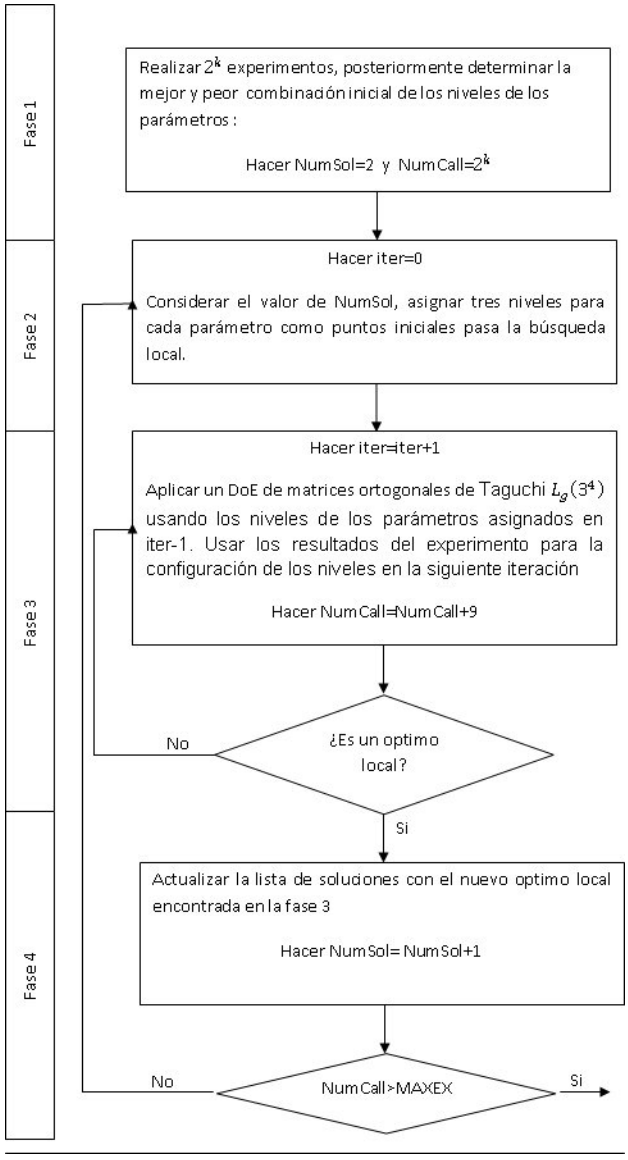

Figura F.2: Diagrama de fujo CALIBRA [4]

F.1.2.2. Técnicas de optimización

Buscar la configuración de parámetros mediante la optimización involucra modelar un problema complejo (las características y propiedades de las variables necesarias así como la interacción entre las mismas, un criterio de optimización no lineal, entre otros), resolverlo y tomar decisiones. Sin embargo, después de la modelación se obtiene una instancia de optimización difícil, la cual es similar a los casos donde se justifica el uso de técnicas heurísticas.

La idea central de esta práctica es utilizar una metaheurística (frecuentemente un algoritmo evolutivo) para buscar una configuración de los parámetros de alta calidad, que será utilizada por otra metaheurística.

Algunos de los métodos desarrollados con la idea anterior son:

- Aproximación por fuerza bruta.

  Sí un experimento requiere t unidades de tiepo para ejecutarse, entonces el número total de experimentos que pueden ser realizados en un tiempo T es $M = \lfloor \frac{T}{t} \rfloor$, bajo este enfoque el poder de cálculo requerido para cada una de las configuraciones candidatas se estima constante, y por tanto, el número de configuraciones muestreaadas es $N = \lfloor \frac{M}{\Theta} \rfloor$ [19]. En el Algoritmo 63, se muestra el pseudocodigo de una aproximación por fuerza bruta [19]:

- Familia de algoritmos Racing, fueron intoducidos por Maron y Moor para resolver el problema de aprendizaje de maquinas. En el Algoritmo 64, se muestra el pseudocodigo [19]:

- Botee y Bonabeau, 1998 proponen utilizar un AG simple para determinar la mejor configuración de cinco parámetros de un OCA sobre instancias del TSP. Los autores expresan que los resultados preliminares encontrados fueron promisorios [22].

F.2. Control de parámetros

Generalmente, se tiene poca probabilidad de determinar *a priori* un conjuto de valores adecuado psrs los parámetros de una metaheurística aplicada sobre cualquier instancia. Para usar métodos dimanicos es necesarío plantearse las siguientes cuestiones antes de ejecutar una metaheurística al reolver alguna instancia

- ¿Cúal componente o elemento se va a cambiar o se ajustar?

  - Representación.
  - Función de aptitud.
  - Método de selección.

Algoritmo 63: Algoritmo Brutus

Input: Caracteristicas del conjunto de parámetros a optimizar.

Output: $\tilde{\theta}$

```
N = ⌊M/Θ⌋
A =Asignar la serie(|Θ|)
for k = 1; k ≤ N; k++ do
    i =muestra de las instancias()
    for cada θ en Θ do
        s =ejecutar el experimento(θ, i)
        c =evaluar la solución(s)
        A[θ] = (A[θ]*(k−1)+c)/k
    end
end
θ̃ =tomar el mínimo(A)
```

- Control de parámetros.
- Cambio en los operadores.

▪ ¿Cómo se va realizar el cambio o ajuste?

- Determinístico.
- Adaptativo.
- Auto-adaptativo.

▪ ¿A qué nivel se va realizar el ajuste?

- Entorno.
- Población.
- Individuo.
- Componentes.

Algoritmo 64: Algoritmo Racing

Input: Caracteristicas del conjunto de parámetros a optimizar.

Output: $\tilde{\theta}$

C =Asignar ls serie(máxima instancia,$|\Theta|$)

while $experiments\_soFar + |S| \leq M \wedge experiments\_soFar + 1 \leq max\_instances$ **do**

  i =muestra de las instancias()

  experiments_soFar =experiments_so$Far + 1$

  for *cada* θ *en* Θ **do**

    $s =$ ejecutar el experimento(θ, i)

    experiments_soFar =experiments_so$Far + 1$

    C[experiments_so Far, θ]=evaluar_solución(S)

  end

  S =drop_candidates$(S, C$,use_prueba)

end

$\tilde{\theta}$ =seleccionar_mejor_valor(S, C)

F.2.1. Control determinista

El control de los parámetros determinista es una estrategia que consiste en alterar el parámetro a través de alguna regla determinista. Dicha regla se activa en momentos fijos provocando un cambio predefinido en las variables sin necesidad de utilizar otra información.

F.2.2. Control adaptativo

Adaptarse significa cambiar de comportamiento para afrontar nuevas circunstancias, en términos generales, el control adaptivo es diseñar y aplicar un mecanismo que permita el cambio en la configuración de parámetros. El control adaptativo ocurre cuando hay algún tipo de mecanismo de retroalimentación; el cual, tiene el proposito de servir como estrategia para decidir la dirección o la magnitud del cambio en el parámetro. La asignación de los valores de los parámetros implican considerar la información anterior.

F.2.3. Control auto-adaptativo

La auto-adaptación de parámetros de control surgé de la idea de la evolución. Este mecanismo de control implica que los parámetros que se desean adaptar deben ser codiciados dentro de los individuos de la población; de tal manera, que a medida que los individuos evolucionan y se adaptan también lo hagan los parámetros que utilizan.

Algunos trabajos son:

1. Método SAGA (Algoritmo Genético Auto Adaptativo) propuesto por Hinterding, Michalewicz, y Peachey, en este trabajo se describe un esquema de adaptación del tamaño de la población (se utiliza el fitness de cada población como criterio de ajuste del tamaño de la población).
2. El método PRoFIGA (Algoritmo Genético de Cambio de tamaño de población sobre mejoramiento del fitness), posee la capacidad de auto regular el tamaño de la población. La población crece cuando existe una mejora en el mejor fitness de la población, o bien, cuando no se ha mejorado en un largo periodo de tiempo, y decrece cuando no ocurre ninguno de los dos casos anteriores (la disminución es en un porcentaje 1-5 %).