CENTRO DE INVESTIGACIONES EN ÓPTICA, A.C.

ALGORITMOS DE VISIÓN E INTELIGENCIA COMPUTACIONAL PARA EL ACOMODO DE PATRONES DE CORTE EN MATERIALES

Como requisito para obtener el grado de:
Maestra en Optomecatrónica (VERSIÓN FINAL)

Asesor:
Dr. Francisco Javier Cuevas de la Rosa

Estudiante:
Lic. Anabel Rodríguez Rodríguez

Julio de 2019
León, Guanajuato, México

Dedicado a mi familia
con todo mi respeto y cariño.

Agradecimientos

Agradezco a Dios y a todos los seres de luz que han guiado mi camino para llegar hasta aquí.

Agradezco al gobierno mexicano, al CONACYT y al CIO por darme la oportunidad de estudiar un posgrado de excelencia como es la Maestría en Optomecatrónica.

Agradezco el apoyo otorgado mediante el proyecto FORDECYT 296737 CONSORCIO EN INTELIGENCIA ARTIFICIAL para el desarrollo de la presente tesis.

Especial agradecimiento al Doctor Francisco Javier Cuevas de la Rosa por permitirme ser parte de su grupo de estudiantes, por darme la asesoría necesaria y corregirme en los momento indicados, por ser mi guía y ejemplo apoyando mi labor científica.

Me siento eternamente agradecida con cada uno de los profesores que aportaron un granito de arena a mi formación profesional y por ser una inspiración para llegar al final de esta investigación.

A mi familia, porque sin ellos no hubiese sido posible llegar a un país extranjero a cumplir mis sueños, por su apoyo a la distancia y por confiar siempre en mí. Por impulsarme y por demostrarme que no hay imposibles, que los sueños sí se cumplen y que hoy se vuelve realidad.

A mis amigos, que muchos se han convertido en familia, gracias por confiar en mí, por ayudarme incondicionalmente en todo momento, por no dejarme caer siempre que lo necesité.

A mi novio que en el último año estuvo todo el tiempo pendiente de mí y colaborando para poder cumplir mis metas. Por su paciencia y su cariño.

A todos y cada uno de los que han hecho posible mi sueño,

Muchas Gracias.

Resumen

La presente investigación resuelve el problema de acomodo de patrones de corte en materiales finitos utilizando técnicas de Visión e Inteligencia Artificial y Algoritmos Genéticos (AGs).

Se implementó un algoritmo que permite seleccionar las imágenes del material y los patrones de corte y procesarlas digitalmente para realizar el acomodo. Posteriormente se diseñó una estructura cromosómica para codificar la información genética, en la que se realiza una combinación de todos los patrones que intervienen en el acomodo. El cromosoma es de tamaño variable ya que contiene un bit de colocación para cada patrón y de éste depende la cantidad de patrones que intervienen en el acomodo. Se genera aleatoriamente la población inicial y se evalúa a cada individuo mediante la función objetivo.

La función objetivo o de aptitud contiene la información necesaria para minimizar el desperdicio en el material teniendo en cuenta que no hayan patrones ni píxeles traslapados y que no estén fuera del área válida de acomodo, además que contenga la mayor cantidad de patrones, lo que permite maximizar el área ocupada en el material de acomodo. Cada generación evoluciona aplicando los operadores de selección, cruzamiento y mutación. Cada individuo es evaluado por la función de aptitud y se compara el mejor de cada generación con el de la anterior para encontrar la mejor solución que tenga el menor desperdicio y la mayor cantidad de patrones acomodados.

Al término de las iteraciones se realiza un proceso de reparación del mejor cromosoma con el propósito de corregir si queda algún patrón traslapado o fuera del área de acomodo y se muestra esta solución.

El algoritmo implementado fue sometido a pruebas con imágenes hechas digitalmente obteniendo resultados satisfactorios en cuanto al porcentaje de desperdicio que es bajo en comparación con otras investigaciones.

Este trabajo se puede aplicar en cualquier industria que requiera del proceso de acomodo ya que puede adaptarse a cualquier tipo de material.

Índice general

Índice de figuras

Índice de cuadros

CAPÍTULO 1

Introducción

La visión es uno de los mecanismos sensoriales más importantes de percepción del ser humano, debido a que, mediante el uso de éste los seres humanos pueden reconocer objetos y personas, aproximar distancias a un objeto, admirar el mundo que lo rodea y poder sobrevivir e interaccionar con el mundo circundante. Debido a esto, surge la Visión Artificial o Visión por Computador, como una rama de la Inteligencia Artificial (IA), que tiene por objetivo modelar matemáticamente los procesos de percepción visual en los seres vivos y generar programas que permitan simular estas capacidades visuales por computadora. Tal y como los humanos usamos nuestros ojos y cerebros para comprender el mundo que nos rodea, la visión por computador trata de producir el mismo efecto con el propósito de que las computadoras puedan percibir y comprender una imagen o secuencia de imágenes y actuar según convenga en una determinada situación.

El Procesamiento Digital de Imágenes (PDI) en la visión artificial implica la manipulación de las imágenes vistas como señales digitales capturada por sensores ópticos para extraer la información elemental subyacente [3]. El análisis de imágenes está encaminado a determinar ciertas estructuras tales como bordes o regiones, formas de figuras, así como relaciones similares entre ellas. Todas estas implicaciones se fusionan para ver el resultado en las aplicaciones reales en el sector automotriz, manufacturero, entre otros.

1.1. Problema de Acomodo de Patrones

El acomodo de patrones de corte es una fase del proceso productivo en las industrias donde se hace necesario cortar material con el fin de obtener piezas o patrones en los tamaños y formas requeridas. Constituye un proceso clave a resolver con técnicas de Visión e Inteligencia Computacional en un importante número de industrias manufactureras: industria metalúrgica, industria del papel, industria textil, industria del calzado, industria del vidrio, en las cuales se hace indispensable el acomodo de patrones para maximizar el área de utilización y minimizar el desperdicio.

La optimización del acomodo de los patrones de corte es de interés para la industria ya que conlleva a un ahorro del material y a reducir los costos de producción. Cortar el material de la manera más eficaz generalmente lleva a un incentivo financiero. Si una compañía puede reducir la cantidad de material desperdiciado, esto significará un ahorro considerable de la materia prima y al mismo tiempo, este ahorro puede hacer posible que el cliente se beneficie, ya que si el empresario reduce sus precios de producción hace a la compañía más competitiva en el mercado.

El problema de patrones de corte es un problema de gran complejidad, catalogado como un problema NP-completo, tanto por las características y variables que involucra como por las técnicas que se utilizan para abordarlo, además, el espacio de búsqueda de la solución puede ser muy grande, lo que requiere un tiempo exponencial para resolverlo.

Existen diferentes estrategias de solución para este problema de optimización como son los métodos heurísticos [4] [5]: recocido simulado [6], redes neuronales, enjambre de partículas [7] [8], algoritmos genéticos [9] [10] [11] [1] y métodos de programación lineal, sin embargo, aún no existe un método global dada la complejidad del problema [12]. Es una temática en constante evolución y en la actualidad existen muchas investigaciones enfocadas en la solución de este problema. El interés en este problema puede ser sustentado por su aplicación práctica y el reto que representa para la academia; pues en general, es computacionalmente difícil de resolver.

1.2. Objetivos Generales

Desarrollar un sistema de visión para la colocación de patrones de corte en materiales finitos mediante la implementación de un arreglo optomecatrónico y el uso de metaheurísticas. Se pretende obtener un diseño de metrología óptica enfatizado principal-

mente en la industria manufacturera principalmente en la industria del calzado.

1.2.1. Objetivos Específicos

- Revisar las técnicas implementadas hasta el momento y realizar un análisis comparativo.
- Desarrollar algoritmos computacionales de procesamiento de imágenes para preprocesar las imágenes que intervienen en el problema.
- Implementar algunas metaheurísticas con la finalidad de disminuir el desperdicio de material y lograr una mejor distribución de los patrones.
- Implementar una aplicación computacional capaz de procesar la información obtenida en el arreglo experimental y tomar decisiones.

1.3. Planteamiento del Problema

Aunque se ha investigado mucho para resolver el problema y se han automatizado los procesos industriales, muchas empresas siguen llevando a cabo el proceso de adaptación de patrones de forma manual, por lo que el uso óptimo del material depende de la experiencia del operador encargado del proceso, lo que afecta al tiempo de corte y al desperdicio del material.

A partir de las investigaciones previas de este problema y los métodos de soluciones propuestos por investigadores para resolver el problema del acomodo óptimo de patrones de corte en materiales [13] [14] [15], se consideró primordialmente la implementación de los Algoritmos Genéticos (AGs), debido a que son técnicas meta-heurísticas que trabajan con un conjunto amplio de soluciones, que a partir de la función objetivo son capaces de encontrar la mejor solución teniendo en cuenta la información suficiente que interviene en el proceso de optimización; es por ello que para la solución del problema en cuestión se utilizaron los AGs como principal herramienta.

Partiendo de la digitalización de imágenes, representadas como matrices, se realizó la simulación del acomodo de patrones de corte, definiendo los parámetros iniciales, generando una población inicial, se calcula la función de aptitud u objetivo a cada uno de los individuos para medir la competitividad y determinar cuáles individuos se eliminan y cuáles se van a reproducir y formar una nueva población aplicando los operadores genéti-

cos hasta encontrar la mejor solución.

La modelación empleada en el presente trabajo, considera maximizar el número de formas acomodadas, minimizando el material desperdiciado tomando en cuenta las restricciones que se derivan de este problema:

- Evitar traslapes entre los patrones.
- Acomodo dentro de los límites del material.
- Los daños del material deben tomarse como regiones no válidas del corte.
- Permitir la rotación u orientación de los patrones.

Para poner a prueba el software, se tuvieron en cuenta materiales finitos que comprueban la optimalidad del software variando los parámetros y aplicando las diferentes técnicas de selección que fueron programadas, además de comparar las pruebas realizadas con investigaciones anteriores para corroborar los resultados de esta investigación.

1.4. Contribuciones de la Investigación

Se pretende ampliar el estudio del arte en el ámbito científico con la finalidad de aportar nuevos métodos que permitan obtener una solución global para el problema de Acomodo de Patrones de Corte en Materiales.

1.5. Antecedentes

El Problema de patrones de corte (CSP) fue formulado por primera vez en 1939 por el economista ruso Kantorovich [16], conocido por su teoría y desarrollo de técnicas para la asignación óptima de recursos y fue el primero que introdujo el término de la programación lineal. En 1961 Gilmore and Gomory [14] propusieron un modelo de resolución a este tipo de problemas en una sola dimensión. Desde entonces se han propuesto disímiles soluciones para resolver el problema del acomodo de patrones.

Algunas investigaciones más recientes que han sido estudiadas como base para la presente investigación son:

Tesis de Yamily Zusuki Marín [17] donde se realiza una simulación del acomodo de patrones utilizando algoritmos genéticos, en el cual utiliza la codificación genética generando puntos de posicionamiento a partir del tamaño del material de acomodo e ir combinando en cada posición qué patrón poner.

En la investigación de Francisco Cuevas et al. [1], se utilizó un algoritmo genético para el acomodo de patrones, donde la función objetivo maneja 4 variables y se tratan los patrones anidados y no superpuestos en una misma variable, a diferencia del presente trabajo que separa estos conceptos y a cada uno le da un peso. Los resultados de esta investigación se muestran en la Figura 1.1.

Figura 1.1: Resultado obtenido utilizando un algoritmo genético por Francisco Cuevas et al. [1].

Otro algoritmo genético enfocado en el problema del material de corte por Godfrey C. Onwubolu y Michael Mutingi [18], en el cual se realiza el arreglo de patrones de corte rectangulares.

La solución propuesta por Pipatpong Poshyanonda y Cihan H. Dagli es el neuro-néster genético [19], donde las redes neuronales y los algoritmos genéticos (GAs) se integran para acomodar patrones regulares o irregulares y utilizar un método heurístico conocido como patrón de deslizamiento.

Otra solución propuesta por Meghdad HMA Jahromi et al. [20], utilizando Simulated Annealing (SA) y Tabu Search (TS), donde intentan resolver el problema de las existencias de corte unidimensionales (1D).

Otros algoritmos metaheurísticos que se han propuesto para el problema de acomodo son: Eunis López Camacho y otros [21] describen una metodología hiperheurística para generar un algoritmo determinista.

Terashima Marín et al. [22] proponen una algoritmo genético para resolver problemas bidimensionales (2D) de empaque de contenedores regulares e irregulares.

Por otro lado, Aline M. Del Valle et al. [23] presentaron una heurística basada en el procedimiento de búsqueda adaptativa aleatoria codiciosa (GRASP) para resolver los problemas de corte en 2D con patrones irregulares.

Entre muchas otras investigaciones recientes [24] [25] [26] [27] [28] que han sido revisadas para dar solución a esta investigación.

1.6. Organización de la Tesis

Esta tesis se divide en 6 capítulos los cuales reflejan la investigación realizada, a continuación, se menciona una breve descripción de su contenido:

- Capítulo 1. Introducción.
  Breve introducción para poner en contexto al lector con la descripción y la justificación del proyecto.
- Capítulo 2. Conceptos Básicos.
  Se detallan los fundamentos teóricos sobre el Procesamiento de las Imágenes, así como los métodos de optimización que serán utilizados.
- Capítulo 3. Algoritmos Genéticos.
  Se describe la base teórica para comprender los algoritmos genéticos, los cuales constituyen la herramienta de optimización para la solución del problema de investigación.
- Capítulo 4. Análisis e Implementación.
  Se realiza un minucioso análisis del problema, todas las restricciones que se tienen en cuenta y la implementación para la solución del problema.
- Capítulo 5. Pruebas y Resultados.
  Se muestran las pruebas que se realizaron durante el proceso de implementación del algoritmo, variando los patrones de corte y el material de acomodo, y se presentan los resultados obtenidos en las pruebas.

- Capítulo 6. Conclusiones.
  Se realiza un análisis de los resultados teniendo en cuenta el cumplimiento de los objetivos, así como las ventajas y desventajas de la solución propuesta. Finalmente se mencionan los trabajos a futuros que se derivan de la presente investigación.

1.7. Conclusiones Parciales

Se realiza un bosquejo de los antecedentes y del planteamiento del problema y se propone la solución que se abordará en la presente investigación.

CAPÍTULO 2

Conceptos Básicos

2.1. Procesamiento Digital de Imágenes

El Procesamiento Digital de Imágenes (PDI) puede ser visto como una transformación de una imagen en otra imagen, es decir, a partir de una imagen se obtiene otra imagen modificada [29]. El PDI es el conjunto de técnicas que se aplican a las imágenes a través de una computadora, con el objetivo de mejorar la calidad y el aspecto visual de ciertos elementos estructurales para el analista y extraer la mayor información posible de éstas, que proporcionen elementos para su interpretación.

Las técnicas de PDI, además de permitir analizar una escena en diferentes regiones del espectro electromagnético, también posibilitan la integración de varios tipos de datos, debidamente registrados. Estas técnicas son de gran importancia para la presente investigación, en la obtención de patrones y material (mediante la digitalización), además permiten manipular la información de las imágenes y poder aplicar las técnicas de mejoraemiento, ya que por lo general cuando se digitaliza una imagen, ésta contiene ruido.

2.1.1. Representación de una Imagen Digital

Una imagen digital se puede considerar como un arreglo matricial con un determinado número de filas (j) y columnas (i), donde cada elemento de la matriz (i, j) es un punto en la imagen conocido como píxel y su valor representa la intensidad o el nivel de gris de la imagen en ese punto (Figura 2.1).

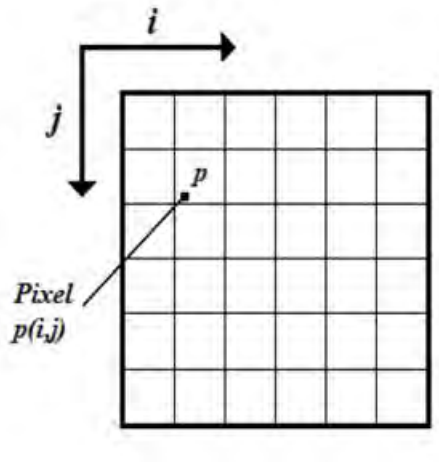


Figura 2.1: Representación en píxeles de una imagen digital.

2.1.2. Etapas del PDI

El tratamiento digital de imágenes comprende un amplio rango de hardware, software y recursos teóricos y abarca varias etapas como muestra la Figura 2.2. Es necesario tener en cuenta que el conocimiento sobre un dominio del problema está codificado en un sistema de procesamiento de imágenes como una base de conocimientos.

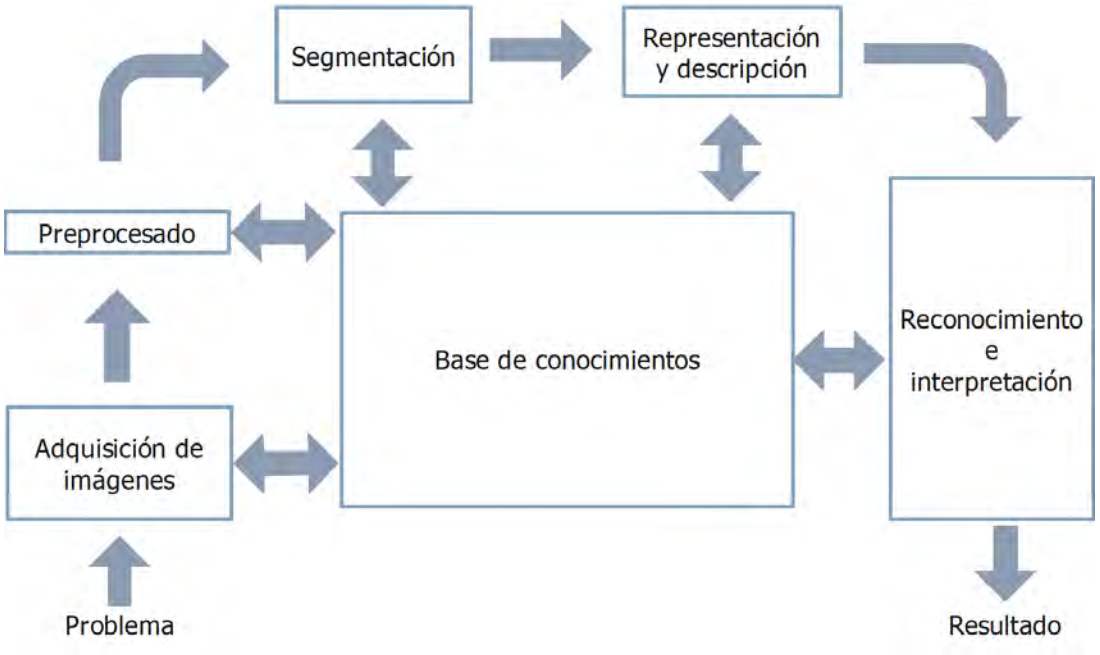


Figura 2.2: Etapas fundamentales del procesamiento digital de imágenes [2].

A continuación, se explica brevemente las etapas fundamentales del PDI:

1. Adquisición de la imagen: se adquiere y digitaliza la imagen.

2. Pre-procesamiento de la imagen: una vez obtenida la imagen, se pre-procesa la imagen aplicando técnicas de mejora de contraste, eliminar el ruido y aislar regiones que no sean de interés.

3. Segmentación: consiste en partir una imagen de entrada en sus partes constituyentes u objetos. Ejemplo de ello, en el reconocimiento de caracteres, el objetivo de la segmentación es el de extraer caracteres individuales y palabras de fondo.

4. Representación y descripción. La representación es solo una parte de la solución para transformar los datos del píxel en bruto a una forma adecuada para luego ser tratada por la computadora. La descripción o selección de rasgos, consiste en extraer rasgos con alguna información cuantitativa de interés o que sean fundamentales para diferenciar una clase de objetos de otra.

5. Reconocimiento e Interpretación. El reconocimiento es el proceso que asigna una etiqueta a un objeto basándose en la información proporcionada por sus descriptores. La interpretación implica asignar significado a un conjunto de objetos reconocidos o entidades etiquetadas.

2.1.3. Mejora de la Imagen

Como se explicó anteriormente, en general, cuando se digitaliza una imagen que contiene ruido, es necesario aplicar técnicas de PDI para mejorar su calidad. Existen diferentes técnicas de mejora que tienen como objetivo principal procesar una imagen de forma que resulte la más adecuada para una aplicación específica [2].

2.1.3.1. Métodos de Dominio Espacial

Para mejorar la imagen se utilizan los métodos de dominio espacial, que se refiere al propio plano de la imagen, es decir, se manipulan directamente sus píxeles. La función de procesamiento de dominio espacial se expresa en la Ecuación 2.1.

$$g(x,y) = T[f(x,y)], \tag{2.1}$$

donde, $f(x,y)$ y $g(x,y)$ representan la imagen de entrada y procesada, T es un operador definido en un entorno (x,y) que actúa sobre f y además puede operar sobre un conjunto de imágenes de entrada.

Algunas técnicas de procesamiento bastante simples, pero potentes, pueden formularse sobre la base de transformaciones del nivel de gris. Debido a que la mejora de cada

punto de una imagen depende sólo del nivel de gris en ese punto, estas técnicas se definen como procesamiento de puntos (Figura 2.3).

(a) (b)

Figura 2.3: (a) Imagen original en escala de grises. (b)Imagen binarizada utilizando el operador umbral.

Otras técnicas permiten una variedad de funciones de tratamiento que actúan sobre un entorno predefinido de (x,y), este tipo de formulación se basa en el empleo de máscaras (plantillas, ventanas o filtros), por ejemplo, de 3×3, en la cual los valores de los coeficientes determinan la naturaleza del proceso, como la acentuación de bordes. Estas técnicas basadas en este tipo de aproximación se conocen como procesamiento por máscaras o filtrados.

2.1.3.2. Métodos de Dominio de Frecuencia

El procesamiento en el dominio de la frecuencia está basado en la transformada de Fourier de una imagen. La base de las técnicas en el dominio de frecuencia es el teorema de convolución, formulado en la Ecuación 2.2.

$$g(x,y) = h(x,y) * f(x,y), \tag{2.2}$$

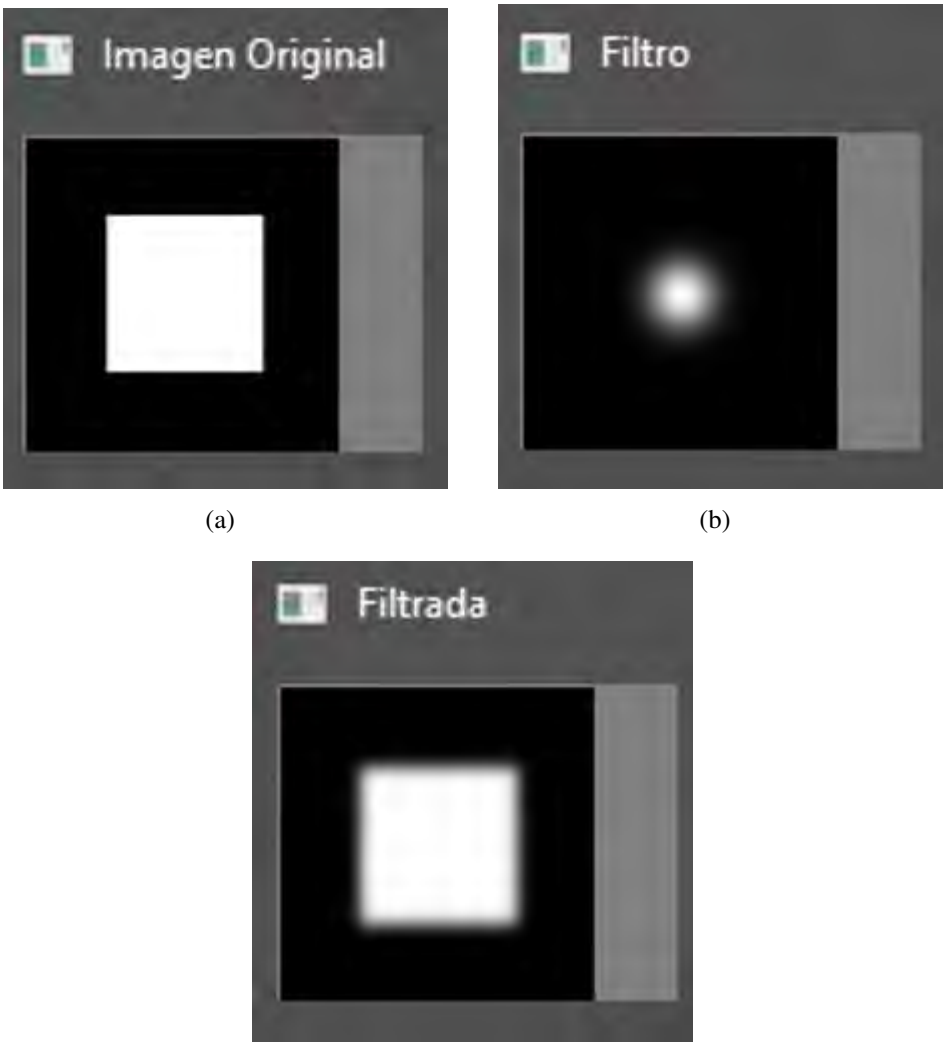
donde, $g(x,y)$ imagen formada por la convolución (siendo $*$ el operador de convolución), $f(x,y)$ imagen original y $h(x,y)$ operador lineal invariante de posición. Entonces por el teorema de convolución se cumple que, la relación en el dominio de frecuencia es el

producto de las transformadas de Fourier como se muestra en la Ecuación 2.3.

$$G(u,v) = H(u,v) \cdot F(u,v), \tag{2.3}$$

donde, $G(u,v)$ es la transformada de Fourier de $g(x,y)$, $F(u,v)$ es la transformada de Fourier de $f(x,y)$ y $H(u,v)$ es la transformada de Fourier de $h(x,y)$.

La Figura 2.4 muestra un ejemplo en el que se realiza la operación de convolución a la transformada de Fourier de la imagen original con el filtro Gaussiano pasabajas, obteniendo la imagen resultante aplicando la transformada inversa de Fourier a la matriz de convolución.



(a) (b)

(c)

Figura 2.4: (a) Imagen original. (b) Filtro Gaussiano pasabajas. (c) Imagen obtenida aplicando convolución de la imagen (a) con (b).

2.1.4. Eliminación de Ruido

El ruido en una imagen es la información no deseada que contamina la imagen, que se obtiene tanto en el proceso de adquisición de la imagen o debido a interferencia en el canal de transmisión. Existen técnicas para eliminación de ruido como son los filtros en el dominio del espacio: filtro de la media, filtro gaussiano y filtro de la mediana, este último es el que más se destaca porque da mejores resultados.

2.1.4.1. Filtro de la Mediana

Este método consiste en visitar cada píxel de la imagen y reemplazarlo por la mediana de los píxeles vecinos. Se toma una máscara o ventana en la imagen, que debe ser cuadrada (por ejemplo de 3 x 3, se tienen 9 valores), se extraen los valores de la ventana y se calcula la mediana de dichos valores, ordenando los valores de los píxeles vecinos y seleccionado el que queda en medio (tomando el valor central sería el valor 5) y este se sustituye por el valor central de la ventana en la imagen original (Figura 2.5).

(a) (b)

Figura 2.5: (a) Imagen original con ruido sal y pimienta. (b) Imagen obtenida después de aplicar la convolución con el filtro de la mediana.

2.1.5. Detección de Contornos

El contorno en una imagen representa un cambio de la intensidad de los niveles de gris presentes en ella. La detección de contornos en una imagen permite simplificar el análisis de ésta, ya que realiza una reducción drástica de la cantidad de datos a ser procesados, definiendo los datos en dos regiones de niveles de gris significativamente distintos; el

paso de nivel oscuro a brillante o viceversa, determinan un contorno (Figura 2.6). Al

(a) (b)

Figura 2.6: (a) Imagen original. (b) Imagen obtenida después de aplicar la técnica del gradiente en X e Y.

mismo tiempo, los contornos en una imagen suministran una valiosa información sobre los límites de los objetos y puede ser utilizada para segmentar la imagen, reconocer objetos, etc.

2.2. Métodos de Optimización

La optimización se refiere al método que se utiliza para determinar los valores de las variables que intervienen en un proceso o sistema para que el resultado sea el mejor posible. Los problemas de optimización pueden ser encontrados en ramas como las matemáticas, estadísticas, ciencias empíricas, ciencia de la computación o economía, donde existe una gran variedad de situaciones que necesariamente han de resolverse aplicando la optimización.

Entre los casos más simples de un problema de optimización se encuentra el de minimizar o maximizar una función real, donde a partir de valores de entrada en un espacio de búsqueda definido, se computa la función y se encuentra el mínimo o el máximo valor de dicha función respectivamente. En general, la optimización permite descubrir los ”mejores valores”de alguna función objetivo dado un dominio definido, incluyendo una variedad de diferentes tipos de funciones objetivo y diferentes tipos de dominios.

2.2.1. Complejidad Computacional de un Problema

La teoría de la complejidad computacional es una rama de la teoría de la computación que se centra en la clasificación de los problemas computacionales de acuerdo con su dificultad inherente, y en la relación entre dichas clases de complejidad.

La teoría de la complejidad computacional trata de clasificar los problemas que pueden o no pueden ser resueltos con una cantidad determinada de recursos. A su vez, la imposición de restricciones sobre estos recursos es lo que la distingue de la teoría de la computabilidad, la cual se preocupa por qué tipo de problemas pueden ser resueltos de manera algorítmica.

2.2.2. Conceptos Determinista y No Determinista

El término determinista significa que, a partir de los datos de entrada, el algoritmo comienza en un estado inicial y a partir de aquí realiza la ejecución de la secuencia de estados predeterminados, es decir, solo hay una cosa que puede hacer a continuación sin importar lo que haga el algoritmo, el paso siguiente se determina por los pasos anteriores [30].

2.2.2.1. Clase de Complejidad NP

La complejidad NP (non-deterministic polynomial time) es el conjunto de problemas que pueden resolverse en un tiempo polinomial por una computadora no determinista. La clase de complejidad NP contiene muchos problemas de búsqueda y de optimización para los que se desea saber si existe una cierta solución o si existe una mejor solución que las conocidas.

La clase NP contiene todos los problemas que pertenecen a las clases P y NP-completo (Figura 2.7).

2.2.2.2. Clases de Complejidad P y NP-Completo

La Clase P es el subconjunto de problemas de decisión de la clase NP, que pueden ser resueltos en tiempo polinómico calculado por una computadora determinista.

La Clase NP-Completo es el subconjunto de los problemas de decisión en NP, que no pueden resolverse dentro de los problemas en P. Los problemas NP-completo son los más difíciles dentro de la clase NP, ya que son resueltos en computadoras no deterministas. Un

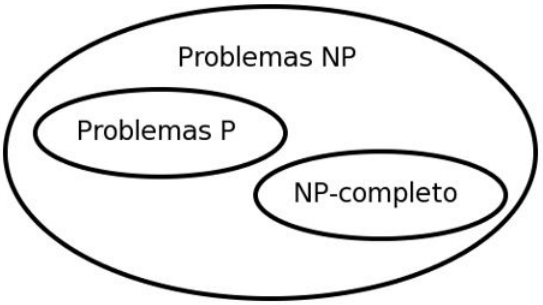


Figura 2.7: Clase de complejidad NP que contiene las clases P y NP-completo.

problema pertenece a esta clase si todos los algoritmos requieren de tiempo exponencial en el peor de los casos. Un ejemplo clásico que pertenece a esta clase de complejidad es el problema del viajero, consiste en encontrar una permutación que represente el recorrido de una serie de ciudades de tal forma que todas sean visitadas una sola vez, minimizando la distancia total viajada [30].

El problema de acomodo óptimo de patrones entra en la clase NP-completo, ya que requiere elegir entre varias soluciones la mejor, lo que hace que sea no determinista. Para la solución de este tipo de problemas existe una gran variedad de métodos heurísticos, que serán revisados más adelante.

2.3. Métodos Clásicos de Optimización

Existen muchas técnicas clásicas para resolver problemas de búsqueda y optimización que presentan ciertas características específicas, como es el caso de las funciones lineales con una o varias variables.

Para la optimización lineal, el método convencional es el Método Simplex, el cual fue creado en el año de 1947 por el matemático estadounidense George Bernard Dantzig, con el ánimo de crear un algoritmo capaz de solucionar problemas de m restricciones y n variables, se basa en un método iterativo que permite ir mejorando la solución en cada paso.

Para la optimización no lineal, existen métodos directos y no directos, dentro de la primera categoría se encuentra la búsqueda aleatoria, este método evalúa la función objetivo repetidamente en los valores de las variables independientes seleccionados al azar. Si se realizan un número suficiente de evaluaciones de la función objetivo, eventualmente, se localizará el óptimo, sin embargo, suele ser ineficiente, pero se puede utilizar como punto de partida para otros métodos.

Entre los métodos indirectos se encuentran: el método del gradiente, del gradiente conjugado, método de Newton y método de la Secante, los cuales también parten de un punto inicial, pero necesitan la primera derivada (los dos primeros) y la segunda derivada (los dos últimos) de la función objetivo para guiar la búsqueda o el tamaño del paso para encontrar el óptimo.

Los métodos anteriormente expuestos tienen en común que la búsqueda finaliza cuando se cumple cierto criterio de convergencia. La diferencia entre ellos radica en la regla mediante la cual se selecciona la dirección de movimiento en cada paso del algoritmo, sin embargo, uno de los grandes problemas de estas técnicas clásicas de optimización es que suelen requerir información que no siempre está disponible.

2.3.1. Métodos Heurísticos

Cuando los problemas de optimización requieren de espacios de búsquedas grandes y que, además, necesitan de un tiempo exponencial para ser resueltos, como es el caso del problema del viajero, los métodos clásicos de optimización no son suficientes, por tanto, es necesario recurrir a las técnicas heurísticas. En la actualidad, muchas aplicaciones prácticas comprenden problemas de este tipo, como es el estudio de la presente investigación acomodo de patrones de corte en materiales.

2.3.2. Heurísticas

La palabra heurística se deriva del griego heuriskein, que significa “encontrar” o “descubrir”. Una heurística es un método basado en la experiencia, es una técnica que busca soluciones buenas a un costo computacional razonable, aunque no garantiza la factibilidad u optimalidad de estas. En algunos casos, ni siquiera puede determinar qué tan cerca del óptimo se encuentra una solución factible en particular [31].

Las heurísticas son técnicas para resolver problemas de gran dificultad basados en principios generales de Inteligencia Artificial. Mediante el uso de heurísticas, es posible resolver más rápidamente problemas conocidos o similares a otros conocidos. Entre los métodos heurísticos más reconocidos se encuentra el método Escalando la Colina (*Hill Climbing*).

2.3.2.1. Escalando la Colina

La técnica escalando la colina se aplica a un punto a la vez. A partir de un punto, se generan varios estados posibles y se selecciona el mejor de ellos. El algoritmo no tiene retrocesos ni almacena ningún registro histórico.

2.3.3. Metaheurísticas

La palabra metaheurística combina el prefijo griego "meta" que significa "más allá" con el sentido de "nivel superior" y heurístico que significa "encontrar" o "descubrir"; se sitúan conceptualmente por encima de los heurísticos en el sentido de que guían el diseño de éstos e introducen mayor nivel de inteligencia al proceso heurístico y por lo tanto, pretenden la obtención de mejores resultados.

Sobre las metaheurística se introduce la siguiente definición [4]:

"Los procedimientos Metaheurísticos son una clase de métodos aproximados que están diseñados para resolver problemas difíciles de optimización combinatoria, en los que los heurísticos clásicos no son efectivos. Los Metaheurísticos proporcionan un marco general para crear nuevos algoritmos híbridos combinando diferentes conceptos derivados de la inteligencia artificial, la evolución biológica y los mecanismos estadísticos".

Algunos ejemplos de métodos metaheurísticos son:

- Búsqueda Tabú.
- Recocido Simulado.
- Enjambre de Partículas.
- Algoritmos Genéticos..

2.3.3.1. Búsqueda Tabú

La búsqueda tabú (*Tabu Search* o TS) es un método de optimización matemática, propuesto por el matemático estadounidense Fred Glover: "la búsqueda tabú guía un procedimiento de búsqueda local para explorar el espacio de soluciones más allá del óptimo local". Es un procedimiento que debe acoplarse a otra técnica, ya que no funciona por sí sola. La búsqueda tabú usa una "memoria" para guiar la búsqueda, de tal forma que algunas soluciones examinadas recientemente son "memorizadas" y se vuelven tabú (prohibi-

das) al tomar decisiones acerca del siguiente punto de búsqueda; es determinista, aunque se le pueden agregar elementos probabilísticos [32].

2.3.3.2. Recocido Simulado

La técnica de Recocido Simulado (*Simulated Annealing* o RS) es un algoritmo de búsqueda para problemas de optimización global, propuesta por Scott Kirkpatrick, C. Daniel Gelatt y Mario P. Vecchi en 1983 [33], tiene como objetivo encontrar una buena aproximación al valor óptimo de una función en espacios de búsquedas grandes. EL RS es un enfoque de búsqueda estocástica capaz de escapar de óptimos locales mediante una modificación a la solución actual, basado en una solución inicial que se trasforma en mejores soluciones conforme avanza la búsqueda.

Básicamente, en física el método de recocido se utiliza en la industria para obtener materiales más resistentes, en otras palabras, para mejorar las cualidades de un material. El proceso consiste en calentar el material a muy alta temperatura. En esa situación, los átomos adquieren una distribución aleatoria dentro de la estructura del material y la energía del sistema es máxima. Luego se hace descender la temperatura muy lentamente por etapas, dejando que en cada una de esas etapas los átomos queden en equilibrio (es decir, que los átomos alcancen una configuración óptima para esa temperatura). Al final del proceso, los átomos forman una estructura cristalina altamente regular, el material alcanza así una máxima resistencia y la energía del sistema es mínima.

Esta técnica ha demostrado su alta eficiencia para resolver problemas de optimización combinatoria aplicados a la ciencia y a la ingeniería.

2.3.3.3. Enjambre de Partículas

La optimización por enjambre de partículas (*Particle Swarm Optimization* o PSO) es un método propuesto por los investigadores James Kennedy, Eberhart y Shi [34, 35]; es una técnica basada en el comportamiento de las partículas en la naturaleza, por ejemplo, así como las abejas en busca de alimento tratan de localizar la región del espacio con mayor densidad de flores, ya que es allí donde presumiblemente existe más cantidad de polen. Cada abeja vuela de modo errático por el espacio, recordando en todo momento cuál es la región donde ha visto más flores. A su vez, el enjambre sabe colectivamente cuál es la región del espacio, de entre todas las exploradas, donde se han encontrado más flores. Cada abeja variará individualmente su movimiento con arreglo a estas dos direcciones, volando hacia algún lugar intermedio. Es posible que la abeja durante ese sobrevuelo

encuentre una región con más densidad de flores que la conocida hasta entonces (óptimo local), o incluso que la conocida por el enjambre (óptimo global); en este último caso, todo el enjambre orientará la búsqueda hacia esa nueva dirección. Pasado un tiempo, si se descubre otra región con mayor densidad floral, el enjambre reorientará nuevamente la búsqueda hacia allí, y así sucesivamente.

Basado en este principio, los métodos PSO permiten optimizar un problema a partir de una población de soluciones candidatas, denotadas como ”partículas”, moviendo éstas por todo el espacio de búsqueda según reglas matemáticas que tienen en cuenta la posición y la velocidad de las partículas. El movimiento de cada partícula se ve influido por su mejor posición local hallada hasta el momento, así como por las mejores posiciones globales encontradas por otras partículas a medida que recorren el espacio de búsqueda. El fundamento teórico de esto es hacer que la nube de partículas converja rápidamente hacia las mejores soluciones.

2.3.3.4. Algoritmos Genéticos

Los Algoritmos Genéticos (*Genetic Algorithm* o AGs) es una de las técnicas más populares entre los algoritmos evolutivos que están inspirados en la Teoría de la Evolución de Darwin [10] basada en el principio de la selección natural (Epígrafe 2.3.4). Los AGs fueron implementados por John Holland investigador de la Universidad de Michigan a finales de 1960 [36].

A grandes rasgos los AGs constituyen estrategias de búsqueda estocástica basados en el mecanismo de selección natural y se involucran aspectos de genética natural, imitando a la evolución biológica como estrategia de búsqueda para la resolución de problemas, proporcionando soluciones en problemas complejos con un gran número de parámetros [37].

Los AGs difieren de las estrategias de búsqueda convencionales en que éstos trabajan sobre un conjunto de soluciones potenciales, llamada población. A su vez, la población está compuesta por soluciones que se llaman individuos y cada individuo está constituido por una cadena de datos (genes) que representan cada variable asociada al problema de optimización. Para obtener nuevas poblaciones sucesivas mejoradas, en cada iteración se aplica sobre los individuos de la población los operadores comunes de los AGs: selección, cruce y mutación. Y en cada ciclo, se calcula la función objetivo (fitness) de cada individuo y se compara en cada iteración con el fin de obtener la solución con mejor adaptación.

2.3.4. Selección Natural

La teoría de la evolución de Darwin recae en la idea principal de que toda la vida está relacionada y que ha descendido de un ancestro común; la selección natural es el proceso por el cual los organismos cambian con el tiempo, como resultado de cambios físicos o de comportamiento heredables. Los cambios que permiten que un organismo pueda adaptarse mejor a su entorno, ayudándolo a sobrevivir y a tener mayor descendencia. Según Darwin: ”No es la especie más fuerte la que sobrevive, ni la más inteligente, sino la que responde mejor al cambio” [10].

La selección natural puede causar pequeños cambios en una especie, por ejemplo, en el color o el tamaño de una población a lo largo de varias generaciones. Esto se llama ”microevolución”. Pero la selección natural también es capaz de crear especies totalmente nuevas, lo que se conoce como ”macroevolución”. La evolución por selección natural es una de las teorías mejor fundamentadas en la historia de la ciencia, apoyada por la evidencia de una amplia variedad de disciplinas científicas, incluyendo la paleontología, la geología, la genética y la biología del desarrollo.

La selección natural puede ser expresada como la siguiente ley general, tomada de la conclusión del libro El origen de las especies, escrito por Charles Darwin [10].

“Existen organismos que se reproducen y la progenie hereda características de sus progenitores, existen variaciones de características si el medio ambiente no admite a todos los miembros de una población en crecimiento. Entonces aquellos miembros de la población con características menos adaptadas (según lo determine su medio ambiente) morirán con mayor probabilidad. Entonces aquellos miembros con características mejor adaptadas sobrevivirán más probablemente”.

2.4. Conclusiones Parciales

En este capítulo se estudiaron los conceptos básicos para el desarrollo de la presente investigación. Se exhibieron las diferencias entre las técnicas clásicas de optimización y las heurísticas, así como la gran importancia que han cobrado las metaheurísticas en la actualidad, lo cual ha impulsado al desarrollo de procedimientos eficientes para encontrar soluciones aproximadas, donde la rapidez del proceso es tan importante como la calidad de éstas.

En los últimos años, la comunidad científica internacional ha mostrado un creciente interés en los métodos de computación evolutiva, trayendo consigo el incremento del número de aplicaciones de los algoritmos evolutivos en problemas más complejos de búsqueda y optimización, que surgen en la ingeniería, los campos científicos e industrias. Por esta razón cabe destacar las ventajas de los AGs, por su capacidad de resolver problemas para los cuales no se conoce solución alguna en espacios de búsqueda muy grande.

El capítulo 2 describe minuciosamente cómo trabajan los Algoritmos Genéticos, así como las principales características, operadores, ventajas y aplicaciones; debido a que es la técnica aplicada a la solución de la presente investigación.

CAPÍTULO 3

Algoritmos Genéticos

Un algoritmo es una serie de pasos organizados que describe el proceso que se debe seguir para dar solución a un problema específico. En la actualidad, son muy utilizados los algoritmos evolutivos que son métodos de optimización y búsqueda de soluciones basados en la evolución biológica. Este tipo de algoritmos se utilizan en problemas con espacios de búsquedas muy grandes y no lineales, donde métodos tradicionales no son capaces de resolverlos. Entre los paradigmas fundamentales de los algoritmos evolutivos se encuentran los algoritmos genéticos que será el tema fundamental del presente capítulo.

3.1. Algoritmos Genéticos

Los Algoritmos Genéticos (*Genetic Algorithm* o AGs) son métodos adaptativos que pueden usarse para resolver problemas de búsqueda y optimización. Su estructura se asemeja a la teoría biológica de la evolución y se basa en la supervivencia del más apto. A lo largo de las generaciones, las poblaciones evolucionan en la naturaleza de acorde con los principios de la selección natural y la supervivencia de los más fuertes.

A partir de una población de individuos se hace evolucionar ésta, sometiéndola a acciones aleatorias semejantes a las que actúan en la evolución biológica (mutaciones y recombinaciones genéticas), así como también a una selección de acuerdo con algún criterio, en función del cual se decide cuáles son los individuos más adaptados que sobreviven y cuáles los menos aptos que son descartados.

Los AGs funcionan entre el conjunto de soluciones de un problema llamado fenotipo y el conjunto de individuos de una población natural, codificando la información de cada solución en una cadena, generalmente binaria, llamada cromosoma. Los símbolos que forman la cadena son llamados genes. Cuando la representación de los cromosomas se hace con cadenas de dígitos binarios se le conoce como genotipo. Los cromosomas evolucionan a través de iteraciones, llamadas generaciones. En cada generación, los cromosomas son evaluados usando alguna medida de aptitud (fitness). Las siguientes generaciones (nuevos cromosomas), son generadas aplicando los operadores genéticos repetidamente, siendo éstos los operadores de selección, cruzamiento, mutación y reemplazo (Figura 3.1).

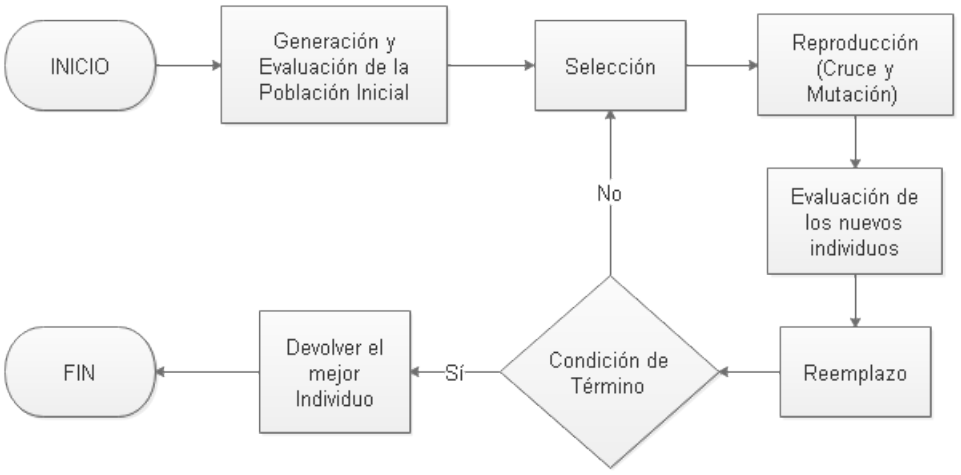


Figura 3.1: Diagrama de flujo de un AG.

Características generales de los AGs

- Se utilizan parámetros codificados como una cadena de longitud finita sobre un alfabeto específico.
- Son algoritmos de funcionamiento paralelo.
- Usan operadores probabilísticos.
- Están menos restringidos por continuidad, derivadas y unimodalidad.

Aplicaciones de los AGs

- Optimización.
- Aprendizaje de Máquinas.

- Bases de Datos.
- Reconocimiento de Patrones.
- Generación de Gramáticas.

Debido a que los AGs son clasificados entre las técnicas heurísticas, éstos poseen una función de aptitud que permite guiar la búsqueda teniendo en cuenta las variables que intervienen en el proceso de optimización, con el propósito de encontrar una buena solución, las cuales se van creando aleatoriamente a lo largo del proceso evolutivo.

3.2. Representación del Cromosoma

La representación tradicional de un cromosoma usando un AG para codificar un conjunto de soluciones del espacio de búsqueda, es una cadena de símbolos (de un alfabeto definido), donde cada elemento o conjunto de elementos de la cadena se le conoce como genes y el valor que puede tomar cada gen se le conoce como alelo (Figura 3.2).

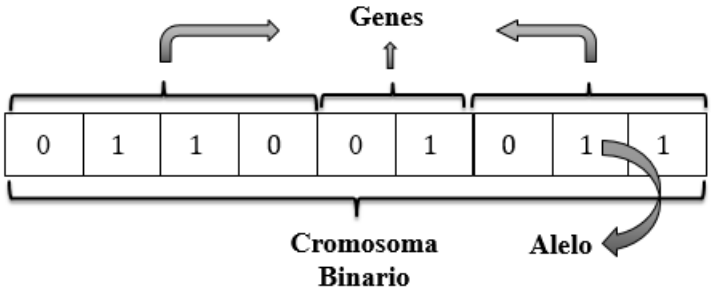


Figura 3.2: Codificación de un cromosoma binario.

Existen otras codificaciones donde los alelos pueden tomar valores enteros, reales o puntos flotantes, que son usados comúnmente para el desarrollo de operadores genéticos más específicos en dependencia de los requerimientos de cada problema.

3.3. Generación de la Población Inicial

La mayoría de las metaheurísticas parten de un conjunto de soluciones conocidas como población inicial. En los AGs, la población inicial es generada aleatoriamente o a partir de soluciones conocidas por el conocimiento a priori que se tiene del problema, aunque en la mayoría de los casos no se tiene esta información debido a que los problemas son complejos e involucran varias variables en el proceso.

La ventaja que tiene generar la población inicial aleatoriamente está dada por la gran diversidad en las soluciones iniciales, que permite explorar todas las zonas del espacio de búsqueda.

3.4. Función Objetivo

La función objetivo o de aptitud involucra todas las medidas o variables que intervienen en la optimización del problema, es decir, permite evaluar las soluciones de cada generación y asignarle un valor de aptitud (fitness) a cada individuo. La función objetivo debe contener la información suficiente para conducir la presión de solución del algoritmo evolutivo y de este modo, guiar el proceso de búsqueda con el fin de encontrar la mejor solución.

Los valores que arrojen la función objetivo deben ser valores reales en un intervalo no negativo (Ecuación 3.1):

$$f_i \longrightarrow R_+. \tag{3.1}$$

3.5. Métodos de Selección

Conocida la aptitud de cada individuo, se realiza el proceso de selección, que consiste en elegir de acuerdo a un criterio determinado, cuáles son los mejores padres para realizar el cruce y formar una nueva población. Entre los principales métodos de selección se pueden mencionar:

- Selección por Ruleta
- Escalamiento de Aptitud
- Escalamiento Sigma
- Método de Rankeo
- Método por Torneo
- Selección de Boltzmann

3.5.1. Selección por Ruleta

El método de selección proporcional o de la Ruleta fue uno de los primeros métodos utilizados por DeJong [38] desde que fueron establecidos los principios básicos de los

AGs por Holland [11]. La ruleta como el nombre lo indica consiste en que a cada individuo de la población se le asigna una parte proporcional de la ruleta de acuerdo a su valor de aptitud, de forma tal que la suma de todos los porcentajes sea la unidad. La probabilidad de selección de cada individuo está dada por la ecuación (Ecuación 3.2).

$$p_i = \frac{f_i}{\hat{f}}, \tag{3.2}$$

donde, p_i corresponde a la probabilidad de selección del individuo i-ésimo, f_i es el valor de aptitud de cada individuo y $\hat{f}$ es la suma de las aptitudes de toda la población o aptitud promedio poblacional. Para seleccionar un individuo se genera un número aleatorio entre 0 y 1 y se devuelve el individuo situado en esa posición de la ruleta. Esta posición se obtiene recorriendo los individuos de la población y acumulando sus proporciones de ruleta hasta que la suma exceda el valor obtenido.

3.5.2. Escalamiento de Aptitud

Se realiza un escalamiento de la solución en un rango seleccionado por el usuario, en otras palabras, esto significa que se realiza una especie de normalización de los valores de aptitud en una escala conveniente para el proceso de selección (Ecuación 3.3).

$$f'_i = \frac{f'_{max} - f'_{min}}{f_{max} - f_{min}} \cdot (f_i - f_{min}) + f'_{min}, \tag{3.3}$$

donde f'_i corresponde al valor de aptitud esperado del individuo i-ésimo, f'_{min} y f'_{max} corresponden a los valores de la escala, f_{min} y f_{max} son los valores de aptitud mínimo y máximo de la generación que se está evaluando y f_i es el valor de aptitud de cada individuo. Al igual que en la ruleta se genera un número aleatorio para seleccionar el individuo que alcance esa probabilidad en dependencia de su nueva aptitud.

3.5.3. Escalamiento Sigma

Es una técnica tratada como truncación Sigma [3] que mapea la actitud original de un individuo con su valor esperado y el valor de la desviación estándar de la población actual, con el propósito de evitar la convergencia prematura en el AG. Este algoritmo permite mantener relativamente constante la presión de selección a lo largo del proceso evolutivo, o lo que es lo mismo, el grado en el que se permite a los individuos reproducirse. Usando esta técnica el valor esperado de un individuo se encuentra en función de su aptitud, la media y la desviación estándar de la población como muestra la Ecuación 3.4.

$$f(x) = \begin{cases} (f_i - (\hat{f} - c \cdot \sigma(t))) & \text{si } f_i > \hat{f} - c \cdot \sigma(t) \\ 0 & \text{en otro caso} \end{cases}, \tag{3.4}$$

donde f_i' corresponde al valor de aptitud del individuo i-ésimo, $\hat{f}$ es la aptitud promedio poblacional, **c** es la constante que modula la presión de selección que debe tomar valores entre 2 y 4, y $\sigma(t)$ es el valor de la desviación estándar en el tiempo t como muestra la Ecuación 3.5, donde N es el tamaño de la población.

$$\sigma(t) = \sqrt{\frac{\sum_{i=0}^{n-1}(f_i - \hat{f})^2}{N}}. \tag{3.5}$$

La ventaja de este método está en que permite controlar la presión de selección. Al inicio del algoritmo el valor de c debe ser 4, esto es, 4 veces la desviación estándar que inicialmente será alta y por tanto el valor esperado de cada individuo es relativamente igual para toda la población y permite que tanto las buenas como las malas soluciones tengan la misma probabilidad de selección, sin embargo, a medida que avanzan las iteraciones, el valor de c debe disminuir al igual que la desviación estándar, lo que permite que los mejores individuos obtengan un valor esperado superior a los peores, por lo que los mejores individuos tendrán una mayor probabilidad de selección.

3.5.4. Método por Ranking o Jerarquías

La selección basada en el método de Ranking fue propuesta por Baker [39] con la finalidad de prevenir la convergencia prematura. En este método el valor esperado depende del valor por jerarquía que se le asigne según su aptitud. El valor esperado de cada individuo está dado por la Ecuación 3.6.

$$\begin{array}{c} V(i,t) = Min + (Max - Min) \cdot \frac{Rango_i - 1}{N-1} \\ 1 \leq Max \leq 2 \\ Min = 2 - Max \end{array}, \tag{3.6}$$

donde $V(i,t)$ es el valor esperado o la cantidad de hijos que se espera de ese individuo, Min y Max son los valores del rango para controlar la presión de selección donde se eligen los individuos más aptos para la reproducción, $Rango_i$ representa la posición del individuo i-ésimo después que se realiza el ordenamiento ascendente de la población y N es el tamaño de la población. Este método garantiza que haya un equilibrio entre las soluciones que tengan peor y mejor aptitud para ser seleccionadas.

La desventaja de esta técnica es que necesita de un algoritmo de ordenamiento, si la población es muy grande mayor tiempo de cómputo requiere el algoritmo para encontrar la solución.

3.5.5. Método por Torneo

La selección por torneo es un método propuesto por Goldberg y Deb [40] que previene la convergencia prematura en cierto grado dependiendo de un umbral y tiene la ventaja de disminuir el tiempo de cómputo del algoritmo. Está basado en la selección aleatoria de dos individuos y comparar sus aptitudes, para esto se genera un valor r que toma valores aleatorios entre 0 y 1, si este valor es menor que el umbral, se selecciona el mejor de los individuos, en caso contrario es elegido el peor. El diagrama de flujo del método por Torneo se muestra en la Figura 3.3.

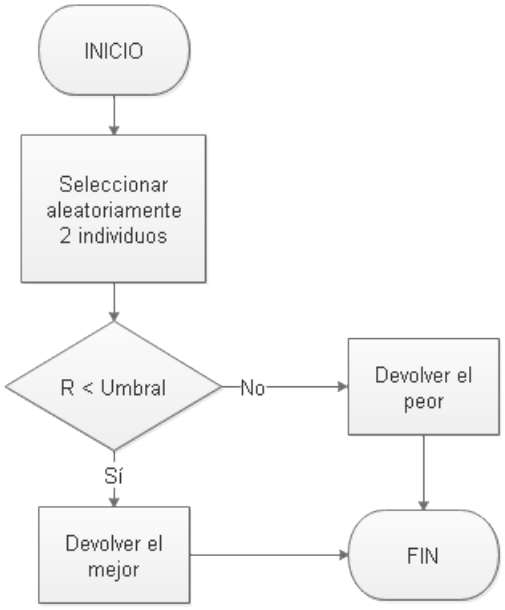


Figura 3.3: Diagrama de flujo de la selección por Torneo.

Este método es de fácil implementación y la presión de selección depende del valor del umbral. Si el umbral es 0.5 ambos individuos tendrán la misma probabilidad de ser seleccionados. Se recomienda variar el valor del umbral a medida que se incrementa el número de iteraciones, por ejemplo, inicialmente el umbral podría ser 0.6 e ir incrementando hasta llegar a 0.9, ya que si toma valor 1 se hará el algoritmo totalmente elitista, ya que siempre se elige al mejor y convergerá prematuramente el proceso evolutivo.

3.5.6. Selección de Boltzmann

El método de selección de Boltzman es uno de los más utilizados debido a los resultados satisfactorios que arroja durante el proceso evolutivo. Fue propuesto por Goldberg [41] en 1990, el cual utiliza una función de variación de temperatura para controlar la presión de selección. Este método permite mantener la presión de selección uniforme al principio del algoritmo y más elitista al final, en otras palabras, al principio todos los individuos tienen prácticamente la misma probabilidad de selección, lo que beneficia a los menos aptos que tendrán igual oportunidad que los mejores de ser seleccionados, lo que permite que se realice una exploración en todo el espacio de búsqueda. Sin embargo a medida que desciende la temperatura, los más aptos tendrán mejor probabilidad de selección que permite que el proceso evolutivo converja en las últimas iteraciones.

La Ecuación 3.7 muestra cómo calcular el valor esperado para cada individuo:

$$V(i,t) = \frac{e^{\frac{f_i}{T}}}{\langle e^{\frac{f_i}{T}} \rangle^t}, \tag{3.7}$$

donde $V(i,t)$ es el valor esperado del individuo i-ésimo en el tiempo t, f_i es el valor de aptitud de cada individuo, $\langle e^{\frac{f_i}{T}} \rangle^t$ es el promedio poblacional de la generación t y **T** es el valor de la temperatura, el cual al inicio debe tomar un valor elevado e ir disminuyendo gradualmente a medida que avancen las iteraciones utilizando una función de variación de temperatura adecuada.

3.6. Métodos de Recombinación o Cruzamiento

La recombinación es el principal operador genético, representa la reproducción sexual, opera sobre dos o más cromosomas a la vez para generar sus descendientes donde se combinan las características de los cromosomas padres. Entre los métodos de cruce se encuentran:

- Cruce en 1 Punto
- Cruce en 2 Puntos
- Cruce en N Puntos
- Cruza Aritmética

3.6.1. Cruce en 1 Punto

El cruce en 1 punto es la técnica de cruce más sencilla implementada en los AGs, se realiza entre dos cromosomas que son elegidos aleatoriamente de la población, se lanza un número aleatorio que es el punto de cruce, que debe ser mayor que 0 y menor que el tamaño del cromosoma, a partir de ese punto se intercambia la información genética de los padres para crear los nuevos individuos (Figura 3.4).

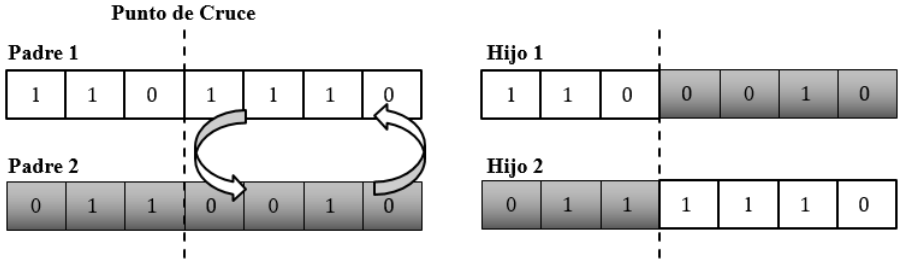


Figura 3.4: Cruce en 1 Punto.

Este método generalmente se utiliza para cromosomas binarios, sin embargo, puede ser utilizado con otra codificación en dependencia de la aplicación.

3.6.2. Cruce en 2 Puntos

EL cruce en 2 puntos se realiza entre dos cromosomas seleccionados al azar, se generan dos puntos de cruce y se intercambia la información genética de los padres que se encuentra entre ambos puntos como muestra la Figura 3.5.

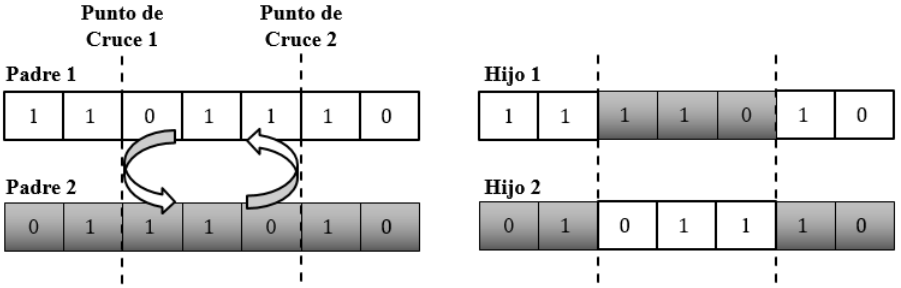


Figura 3.5: Cruce en 2 Puntos.

Este método es menos disruptivo o destructivo que el cruce en un punto, ya que solo cambia una parte de los genes en los sucesores, permitiendo que se herede mayor información de los padres a la nueva generación. El cruce en dos puntos es más común utilizarlo cuando la codificación es binaria.

3.6.3. Cruce en N Puntos

Como el nombre lo indica, se realiza la cruza en *N* puntos, se escogen aleatoriamente dos cromosomas y se generan *N* puntos de cruce. La recombinación de los nuevos individuos se realiza intercambiando los segmentos genéticos alternando los padres en cada punto, un ejemplo se muestra en la Figura 3.6.

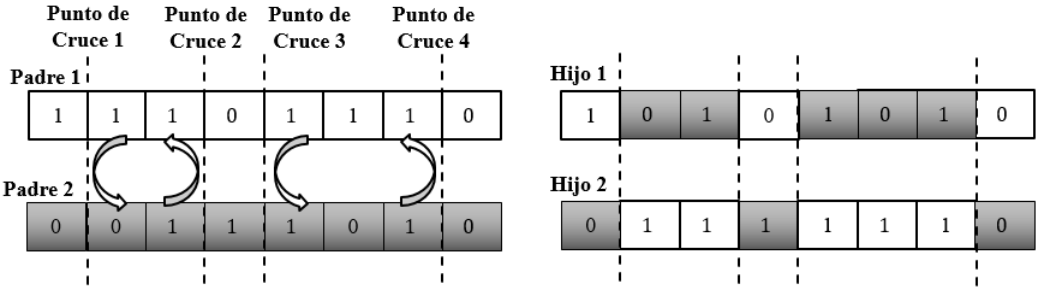


Figura 3.6: Cruce en *N* Puntos.

De Jong [38] fue el primero en implementar este método, el cual minimiza los efectos disruptivos de la cruza. El valor ideal para *N* es 2, a menos que existan muchas restricciones no debe incrementarse el valor de *N* por que tiende a hacerse más destructivo el cruce, sin embargo, el añadir más puntos de cruce posibilita que se explore mucho más el espacio de búsqueda.

3.6.4. Cruza Aritmética

La cruza aritmética se utiliza entre cromosomas con codificación real. Se puede realizar entre dos o más individuos seleccionados aleatoriamente, se escoge el alelo con mejor

aptitud para crear mejores hijos y se aplica la Ecuación 3.8 entre los valores del alelo seleccionado.

$$x_i' = \alpha_1 \cdot x_{1i} + \alpha_2 \cdot x_{2i} + ... + \alpha_n \cdot x_{ni} \quad con \sum_{i=1}^{n} \alpha_i = 1 \tag{3.8}$$

La Figura 3.7 muestra un ejemplo de la Cruza Aritmética entre tres cromosomas tomando $\alpha_1 = 0{,}3$, $\alpha_2 = 0{,}5$ y $\alpha_3 = 0{,}2$.

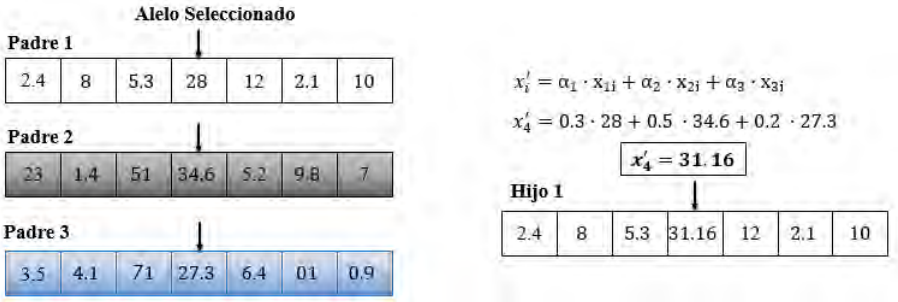


Figura 3.7: Cruza Aritmética aplicada a un alelo.

3.7. Métodos de Mutación

La mutación es un operador básico en los AGs que proporciona una pequeña porción de aleatoriedad en los individuos de la población. Al igual que en el proceso evolutivo, en ocasiones se presentan características que nunca antes habían aparecido en generaciones anteriores, lo que significa un cambio, muchas veces no trascendental, entre padres e hijos. Las mutaciones ocurren con muy baja probabilidad, generalmente menor al 1 % del total de genes de la población.

En los AGs se utiliza la mutación para mantener la diversidad en la población y para escapar de mínimos locales, ya que permite alcanzar zonas del espacio de búsqueda que no estaban cubiertas por los individuos de la población actual. El procedimiento para aplicar este método es muy simple, para cada alelo se genera un número aleatorio entre 0 y 1, se compara con la probabilidad de mutación, y si cumple la condición se modifica, de lo contrario permanece igual. En la Figura 3.8 se muestra un ejemplo de un cromosoma binario y el resultado de aplicar mutación, se cambia 1 por 0 y viceversa.

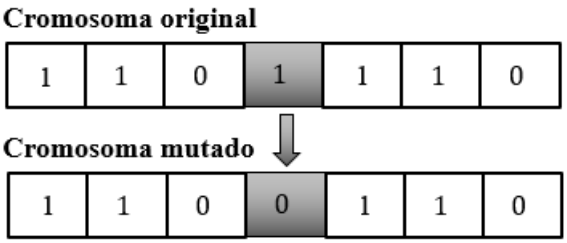


Figura 3.8: Mutación de un cromosoma binario.

3.8. Reemplazo

Una vez aplicados los operadores genéticos, de acuerdo al criterio de selección, se eligen los individuos que conformarán la nueva población que pasan a la siguiente etapa del proceso evolutivo.

3.9. Condición de Término

El AG se deberá detener cuando se alcance la solución óptima, pero ésta generalmente se desconoce y se utilizan otros criterios de detención. Normalmente se usan dos criterios: repetir el AG un número máximo de iteraciones o detenerlo cuando no haya cambios en la población, es decir, cuando la desviación estándar en la población no tenga cambios.

3.10. Conclusiones Parciales

Se describió el funcionamiento de un Algoritmo Genético, los operadores genéticos al igual que los diferentes métodos que son utilizados para su implementación.

El próximo capítulo explica detalladamente la implementación y desarrollo del problema en cuestión para obtener los resultados.

CAPÍTULO 4

Desarrollo e Implementación

En el presente capítulo se describe el desarrollo y la implementación del algoritmo genético para resolver el problema del acomodo de patrones de corte en materiales finitos. Se explica la codificación del cromosoma, que representan las combinaciones de los patrones en cada generación. Se analiza la función objetivo para evaluar la aptitud de cada cromosoma y los operadores genéticos que se utilizaron en la aplicación.

4.1. Interfaz gráfica

Se diseño una interfaz gráfica para seleccionar las imágenes del material de acomodo y de los patrones de corte, este proceso es el primer paso antes de comenzar el algoritmo genético.

4.1.1. Configuración del Material del Acomodo

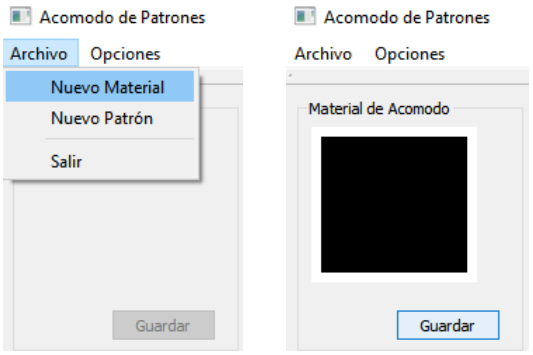
La configuración del Material de Acomodo se almacena en una estructura de datos (Figura 4.1), donde se guardan las propiedades fundamentales del material donde se deben acomodar los patrones.

```
struct Material
 {
     Mat m;
     int area;
 }st_material;
```

Figura 4.1: Estructura de datos que almacena las propiedades del material de acomodo.

Donde *m* es de tipo de dato *Mat* de la librería de OPENCV donde se almacena la imagen del material de acomodo; *area* es de tipo *int*, representa el área válida de acomodo visto como la cantidad de píxeles donde se pueden acomodar los patrones; *st_material* es la referencia de la estructura.

Las imágenes del material de acomodo están almacenadas en una carpeta del ordenador. La imagen es seleccionada por el usuario a través de la interfaz gráfica en el menú **Archivo/Nuevo Material de Acomodo**, una vez seleccionada la imagen, ésta se muestra en el panel **Material de Acomodo** y se guardan sus propiedades presionando el botón **Guardar** (Figura 4.2).



(a) (b)

Figura 4.2: (a) Menú nuevo material de acomodo en la interfaz de usuario. (b) Se muestra la imagen del material de acomodo una vez que ha sido seleccionada.

4.1.2. Configuración de los Patrones de Corte

La configuración de los patrones de corte se almacena en una estructura de datos (Figura 4.3), donde se guardan las propiedades de los patrones de corte que serán acomodados.

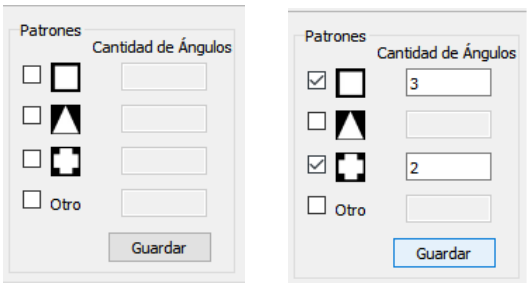
```
struct Patrones
{
    Mat p;
    int cx;
    int cy;
    int area;
    int cantidad_angulos;
    int bit_color;
}st_patrones[100], arrayPat[100];
```

Figura 4.3: Estructura de datos que almacena las propiedades de los patrones de corte.

Donde *p* es la imagen del patrón; *cx* y *cy* son valores enteros que corresponden al

centroide en x e y del patrón respectivamente; *area* es un valor entero que almacena el área del patrón equivalente a la cantidad de píxeles que son acomodados en el material; $cantidad\_angulos$ es un valor entero que almacena el número de ángulos que se rota cada patrón; bit_color es el nivel de gris que toma el patrón en el acomodo final y $st\_patrones$ y *arrayPat* son arreglos que hacen referencia a dicha estructura donde se pueden almacenar hasta 100 patrones diferentes.

Las imágenes de los patrones se encuentran almacenadas en una carpeta del ordenador. Los patrones son seleccionados por el usuario en el menú **Archivo/Nuevo Patrón** similar a la selección del material como muestra la Figura 4.2 (a) o seleccionando los patrones del panel **Patrones** como se muestra en la Figura 4.4 (a), en ambos casos, es necesario introducir la cantidad de ángulos que se rota cada patrón seleccionado. Después de realizar la selección de los patrones, éstos se guardan presionando el botón **Guardar** (Figura 4.4 (b)).



(a) (b)

Figura 4.4: (a) Se muestra el panel de selección de los patrones. (b) Se activa el botón para guardar los datos después que han sido seleccionados.

4.2. Digitalización de las Imágenes

Cuando las imágenes son seleccionadas se realiza el tratamiento de éstas para realizar el acomodo. Se cargan las imágenes en escala de grises y se realiza la binarización de éstas.

4.2.1. Binarización

La binarización es una técnica básica del Procesamiento Digital de imágenes que se encarga de convertir la imagen recibida en una imagen binaria con el propósito de separar el objeto que se quiere analizar del fondo de la imagen. Para realizar este proceso se utiliza normalmente un umbral de la escala de grises que permite distinguir qué píxeles

pertenecen al objeto y cuáles al fondo.

El proceso de encontrar el umbral se realiza con el método de Otsu propuesto por Nobuyuki Otsu en 1979 [42] el cual utiliza la varianza como técnica estadística para medir la dispersión entre los niveles de gris de una imagen. El método de Otsu calcula el valor umbral de forma que la dispersión dentro de cada segmento sea lo más pequeña posible, pero al mismo tiempo la dispersión sea lo más alta posible entre segmentos diferentes. Para ello se calcula el cociente entre ambas variancias y se busca un valor umbral para el que este cociente sea máximo.

Encontrado el umbral se realiza el proceso de Binarización, el cual se describe el pseudocódigo en el Algoritmo 1. Los valores que se asignan a las imágenes en el proceso de binarización, se tomaron a conveniencia para el desarrollo del algoritmo.

Algoritmo 1 Binarización de Imágenes

```
1:  procedure BINARIZAR
2:      Entrada:
3:      M → Imagen Original
4:      B → Imagen Binarizada
5:      U → Umbral
6:      id → Identificador: 0 = Imagen del Material, 1 = Imagen de los Patrones
7:      if id == 0 then
8:          for i = 0; i <= M.rows; i++ do
9:              for j = 0; j <= M.cols; j++ do
10:                 pixel ← m(i, j)
11:                 if pixel > U then
12:                     B(i, j) ← 10
13:                 else
14:                     B(i, j) ← 0
15:                 end if
16:             end for
17:         end for
18:     else
19:         for i = 0; i <= M.rows; i++ do
20:             for j = 0; j <= M.cols; j++ do
21:                 pixel ← m(i, j)
22:                 if pixel > U then
23:                     B(i, j) ← 1
24:                 else
25:                     B(i, j) ← 0
26:                 end if
27:             end for
28:         end for
29:     end if
30:     Output: B
31: end procedure
```

Las imágenes quedan binarizadas como muestra la Figura 4.5.

El proceso de binarizado es vital para el desarrollo de este trabajo, debido a que el acomodo se realiza a nivel de píxeles, lo mismo el cálculo de los patrones y píxeles que están traslapados como los que estén fuera del área de acomodo. Por lo que constituye un paso fundamental al inicio del algoritmo, lo cual se abordará más adelante.

El requerimiento es que el material tenga el área de acomodo oscura y los bordes claros y los patrones el área sea clara y el fondo oscuro. De modo que al sumar los píxeles del patrón con el fondo del material tengan valor 1, si hay traslape será mayor que 1, pero

(a) (b)

Figura 4.5: (a) Imagen Binarizada del Material. (b) Imagen binarizada del Patrón.

menor que 10 ya que si es mayor que 10 se debe a que el patrón quedó fuera del área de acomodo.

4.3. Codificación del Algoritmo Genético

Para la codificación genética se tuvo en cuenta un conjunto de parámetros que define al cromosoma, conocido como fenotipo, el cual contiene toda la información requerida para construir el cromosoma. El fenotipo está definido por una estructura de datos que contiene ocho parámetros como muestra la Figura 4.6.

```
struct Cromosomas
{
    Mat acomodo;
    int Genes[10000];
    int Fitness;
    int traslapes;
    int fuera_area;
    int num_traslapes;
    int num_fuera_area;
    int area_ocupada;
    int num_Patrones;
}pob_ini[100], pob_temp[100];
```

Figura 4.6: Estructura de datos que almacena las propiedades del cromosoma.

El primer elemento es *acomodo*, de tipo *Mat* de la librería OPENCV, que almacena la imagen del acomodo de los patrones que contiene el cromosoma.

El segundo es el cromosoma, es un arreglo de tipo entero que se denomina *Genes* donde se almacenan consecutivamente el identificador de cada patrón, el ángulo de rotación, la ubicación en filas y columnas dentro de la matriz de acomodo y un bit de colocación que representa si el patrón está incluido o no en el acomodo.

Desde el tercero al octavo parámetro de la estructura son valores enteros donde se guardan los valores siguientes respectivamente:

Fitness es el valor de la función objetivo o aptitud del cromosoma,

traslapes es la cantidad de píxeles traslapados dentro del área válida de acomodo,

fuera_area es el número de píxeles que se encuentran fuera del área válida de acomodo,

num_traslapados es el número de patrones traslapados,

num_fuera_area es el número de patrones fuera del área de acomodo,

area_ocupada es la cantidad de píxeles de los patrones que están bien acomodados dentro del área válida de acomodo,

num_pat es el número de patrones que intervienen en el acomodo.

pob_ini y *pob_temp* son referencias de esta estructura, son los arreglos que almacenan los individuos de la población inicial que se genera al inicio del algoritmo y la población temporal que cambia a partir de los operadores genéticos cuando se crean las nuevas generaciones.

4.3.1. Codificación del Cromosoma

El cromosoma es un arreglo de enteros, el cual es inicializado aleatoriamente obteniendo una combinación de todos los patrones que serán acomodados. Cada patrón posee un identificador, el ángulo de rotación entre 0° y 360°, la posición en filas y columnas dentro del material de acomodo y un valor 0 o 1 que indica si el patrón será o no acomodado (Figura 4.7). El cromosoma es de tamaño variable, ya que depende de cuántos patrones intervienen en el acomodo.

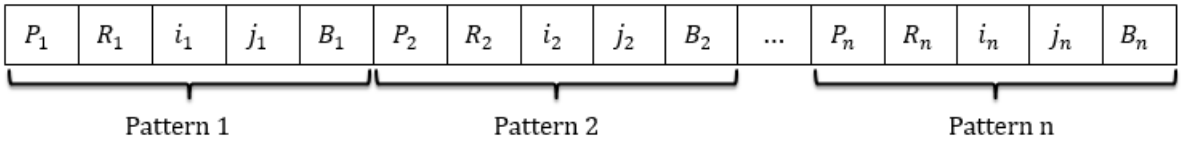


Figura 4.7: Codificación del Cromosoma.

Donde n es el número de patrones, P_k es el identificador del patrón, R_k es el ángulo de rotación, i_k y j_k son las posiciones de la fila y la columna en el material de acomodo respectivamente. B_k es el valor que indica si el patrón interviene o no en el acomodo, k enumera los patrones desde 1 hasta n.

4.3.2. Generación de la Población Inicial

La población inicial se genera aleatoriamente teniendo en cuenta los siguientes aspectos:

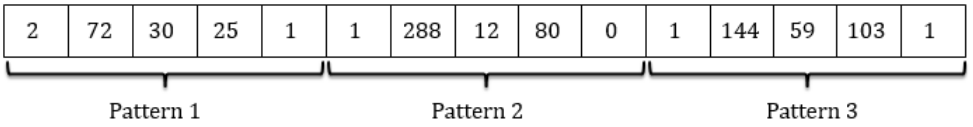
- La cantidad de patrones que fueron seleccionados: a cada patrón se le asigna un identificador, es decir, el primer patrón es 1, el segundo patrón es 2 y así sucesivamente hasta alcanzar el total de patrones seleccionados.
- El número de rotaciones de cada patrón: a cada patrón del cromosoma se le asigna aleatoriamente una ángulo de rotación. Inicialmente cada patrón tiene una cantidad de ángulos que el usuario le asigna. El algoritmo divide 360° entre el número de ángulos y guarda el patrón rotado en los diferentes ángulos, para luego ser utilizado según corresponda.
- Posiciones de la fila y columna en la imagen del material de acomodo: se generan aleatoriamente las posiciones (i, j) donde será acomodado cada patrón dentro del material teniendo en cuenta que el valor del píxel en esa posición sea 0, lo que significa que el patrón será acomodado dentro del área válida de acomodo.
- El bit de acomodo: se genera aleatoriamente para cada patrón del cromosoma un número 0 o 1. Si es 0 significa que el patrón no se tendrá en cuenta para acomodar y si es 1 el patrón si es acomodado.

Una vez que se genera la población inicial, se calcula la aptitud de cada cromosoma y todas las propiedades que lo conforman (Figura 4.6).

4.3.2.1. Ejemplo de un cromosoma y su acomodo correspondiente

Supongamos que se ha generado el cromosoma de la figura 4.8 (a) que contiene tres patrones. El primer patrón tiene identificador 2, con un ángulo de rotación de 72°, posicionado en la fila 30 y columna 25 y el bit de acomodo es 1 por tanto ese patrón sí interviene en el acomodo como se muestra en la Figura 4.8 (b). El segundo patrón que contiene el cromosoma tiene el identificador 1, el ángulo de rotación es 288°, está posicionado en la fila 12 y columna 80, sin embargo, el bit de acomodo es 0 por lo que no es acomodado en el material. A diferencia del tercer patrón con identificador 1 contenido en el cromosoma,

con un ángulo de rotación de 144°, en la fila 59 y columna 103 y sí es acomodado como se muestra en la Figura 4.8 (b).



(a)

(b)

Figura 4.8: (a) Ejemplo de un cromosoma que contiene 3 patrones. (b) Imagen del acomodo.

4.4. Función Objetivo

Para alcanzar el acomodo óptimo de los patrones, es necesario minimizar la pérdida del material teniendo en cuenta que no hayan patrones traslapados y que los patrones no estén fuera del área válida de acomodo, lo que conlleva a maximizar el área ocupada por los patrones. Además se tiene en cuenta la mayor cantidad de patrones que puedan ser acomodados y que el área ocupada por ellos sea máxima.

La función objetivo se muestra en la Ecuación 4.1.

$$\begin{aligned} F = W_P \cdot (P_T - P_A) + W_w \cdot (A_T - A_{OC}) + W_{NO} \cdot Num_{Ov} \\ + W_{NOA} \cdot Num_{OA} + W_{OP} \cdot Overlaps + W_{OA} \cdot Outside\_Area \end{aligned}, \tag{4.1}$$

donde,

W_P es el peso asignado a la cantidad de patrones,

$(P_T - P_A)$ es la cantidad de patrones que no son acomodados, se refiere a la diferencia entre la cantidad de patrones que almacena el cromosoma y la cantidad de patrones que son acomodados, mientras menor sea la diferencia significa que hay mayor cantidad de patrones en el acomodo,

W_W es el peso asignado al desperdicio,

$(A_T - A_{OC})$ es el área de desperdicio, se refiere a la diferencia entre el área total del material y el área ocupada por los patrones,

W_{NO} es el peso asignado al número de patrones traslapados,

Num_{Ov} es el número de patrones traslapados,

W_{NOA} es el peso asignado al número de patrones fuera del área válida de acomodo,

Num_{OA} es el número de patrones que están fuera del área válida de acomodo,

W_{OP}es el peso asignado al número de píxeles traslapados,

$Overlaps$ es el número de píxeles traslapados,

W_{OA} es el peso asignado al número de píxeles que están fuera del área válida de acomodo,

$Outside\_Area$ es el número de píxeles que están fuera del área válida de acomodo.

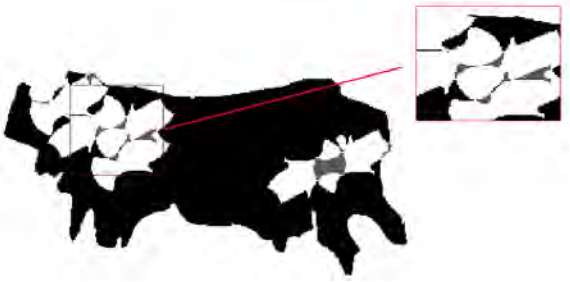
4.4.1. Patrones y Píxeles con Traslapes y Fuera del Área de Acomodo

En la función objetivo se tienen en cuenta tanto la cantidad de píxeles como la cantidad de patrones que se traslapan o quedan fuera del área válida de acomodo, ya que es necesario considerar que cuando se traslapan dos patrones pueden contener más píxeles traslapados, que cuando se traslapan más de dos patrones la cantidad de píxeles puede ser menor. Cuanto menos píxeles se traslapen menor es el desperdicio y mayor el área ocupada.

La Figura 4.9 muestra un ejemplo de traslapes, en el caso (a) hay dos patrones traslapados y en el caso (b) hay tres, sin embargo, la cantidad de píxeles traslapados en (a) es mayor que en (b). Es por esto que se tienen en cuenta tanto los píxeles como los patrones a la hora de calcular la aptitud de cada individuo de la población.

Con los patrones y píxeles que quedan fuera del área de acomodo ocurre exactamente la misma situación, mientras menos patrones y píxeles queden fuera del área de acomodo

(a)



(b)

Figura 4.9: (a) Traslape entre dos patrones. (b) Traslape entre tres patrones.

mayor es el aprovechamiento del material.

Para calcular la cantidad de patrones y píxeles que se traslapan y que quedan fuera del área de acomodo de cada individuo, se utilizó el procedimiento que se muestra en el Algoritmo 2 en forma de pseudocódigo.

A partir de la función *Traslape_Fuera_de_Area* se calculan todas las variables que utiliza la función objetivo para asignar un valor de aptitud a cada individuo.

4.4.2. Penalización de la Función Objetivo

Los pesos en la función objetivo son los que se encargan de penalizar a los individuos de acuerdo a su adaptación en la población. Estos son seleccionados por el usuario al inicio del proceso de configuración.

Es recomendable que los pesos tengan un valor elevado, ya que permite diferenciar entre una buena y una mala solución. A medida que los pesos son más grandes, el valor de la función objetivo será mayor, lo cual significa que es una mala solución y estas soluciones tiene menor probabilidad de ser seleccionadas en el proceso de selección del

Algoritmo 2 Patrones y Píxeles Traslapados y Fuera de Área

```
1: procedure TRASLAPE_FUERA_DE_AREA
2:     Entrada:
3:     Cromosoma → Cromosoma Actual
4:     for n = 0; n < Tamaño_del_Cromosoma; n+ = 5 do
5:         if Cromosoma.Genes[n+4] == 1 then
6:             Num_Patrones+ = 1
7:             rotada ← Patron rotado
8:             t, f ← false
9:             for i = 0, k = Cromosoma.Genes[n+2]; i <= rotada.rows; i++ do
10:                for j = 0, l = Cromosoma.Genes[n+3]; j <= rotada.cols; j++ do
11:                    m ← Cromosoma.acomodo(k,l)
12:                    p ← rotada(i, j)
13:                    if m == 0 & p == 1 then
14:                        Area_de_Acomodo+ = 1;
15:                    else
16:                        if m >= 1 & m < 10 & p == 1 then
17:                            Pixeles_traslapados+ = 1; t ← true
18:                        else
19:                            if m >= 10 & p == 1 then
20:                                Pixeles_fuera_de_area+ = 1; f ← true
21:                            end if
22:                        end if
23:                    end if
24:                    Cromosoma.acomodo(k,l) = m + p
25:                end for
26:            end for
27:            if t then
28:                Patrones_traslapados+ = 1
29:            end if
30:            if f then
31:                Patrones_fuera_de_area+ = 1
32:            end if
33:        end if
34:    end for
35:    Cromosoma.traslape ← Pixeles_traslapados
36:    Cromosoma.fuera_area ← Pixeles_fuera_area
37:    Cromosoma.num_traslape ← Patrones_traslapados
38:    Cromosoma.num_fuera_area ← Patrones_fuera_de_area
39:    Cromosoma.Area_de_Acomodo ← Area_de_Acomodo
40:    Cromosoma.num_patrones ← Num_Patrones
41:    Output: Cromosoma
42: end procedure
```

AG.

4.5. Operadores Genéticos

Los operadores genéticos explicados en el Capítulo 3: selección, cruce y mutación constituye el pilar fundamental de los AGs para generar las nuevas poblaciones y permitir que estas evolucionen con el incremento de las iteraciones.

4.5.1. Método de Selección de Boltzmann

El método de selección de Boltzman es un método de enfriamiento que requiere una función de variación de temperatura que permita que este valor descienda al mismo tiempo que se incrementa el número de iteraciones. Con este fin, se implementaron varias funciones que permiten este comportamiento.

4.5.1.1. Enfriamiento Lineal

La Ecuación 4.2 muestra cómo se calcula la temperatura en cada iteración, para que tenga un comportamiento lineal dependiendo de las temperaturas inicial y final así como el número total de iteraciones.

$$T_i = (\frac{T_{fin} - T_{ini}}{ITER}) \cdot i + T_{ini}, \tag{4.2}$$

donde T_i es la temperatura en la iteración i, T_{ini} y T_{fin} son los valores que el usuario asigna en la configuración inicial a las temperaturas inicial y final respectivamente, $ITER$ es el número total de iteraciones. La gráfica de la función se muestra en la Figura 4.10, con $T_{ini} = 10^4$, $T_{fin} = 0{,}1$ e $ITER = 5000$.

4.5.1.2. Enfriamiento Geométrico

Se utiliza una función que tiene un comportamiento geométrico [43] que permite disminuir la temperatura gradualmente a medida que aumenta el número de iteraciones como muestra la Ecuación 4.3.

$$T_i = \alpha \cdot T_{i-1}, \tag{4.3}$$

donde T_i es el valor de la temperatura en la iteración i, T_{i-1} es el valor de temperatura en la iteración anterior $(i-1)$ y el valor de α permite moderar la velocidad de descenso de la temperatura, los valores recomendados para α se encuentran entre 0,9 y 0,99.

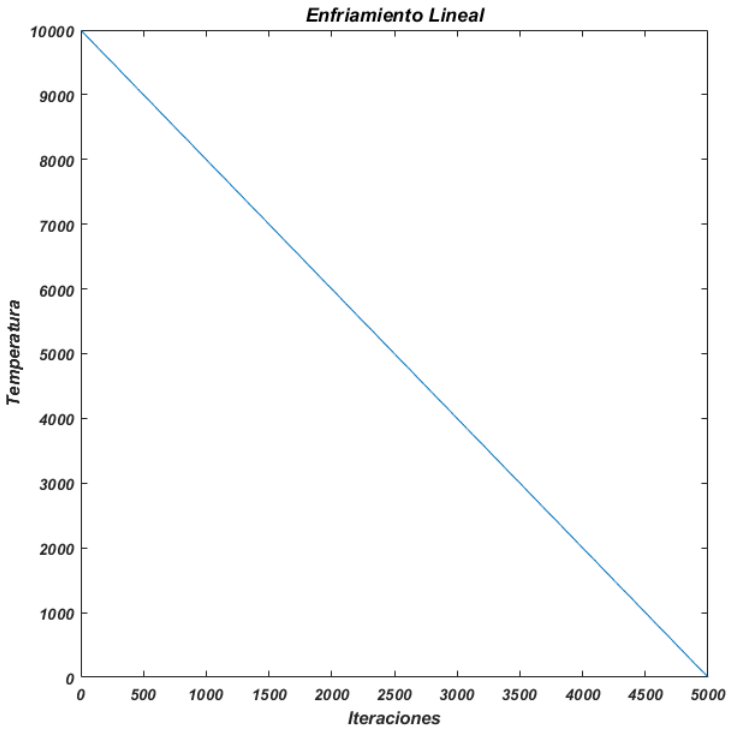


Figura 4.10: Comportamiento del enfriamiento lineal.

Tomando los siguientes valores $T_{ini} = 9000$, $T_{fin} = 1$, $ITER = 1000$ y $0{,}98 < \alpha < 0{,}996$ se obtienen la gráfica de la Figura 4.11.

4.5.1.3. Enfriamiento Exponencial

Otra función diseñada con la finalidad de variar la temperatura gradualmente es una función exponencial que permite variar la temperatura de una forma más suave y moderada que con la función geométrica, como muestra la Ecuación 4.4.

$$T_i = T_{ini} \cdot \left(\sqrt[N]{\frac{T_{fin}}{T_{ini}}} \right)^{i \cdot \alpha}, \tag{4.4}$$

donde T_i es el valor de la temperatura en la iteración i, T_{ini} y T_{fin} son las temperaturas inicial y final respectivamente, N es el número de iteraciones y α es un factor que varía en el intervalo $[0{,}5; 1]$ que modifica la velocidad con la que decae la temperatura. Todos estos valores pueden ser seleccionados por el usuario en la configuración inicial del programa. El comportamiento de esta función se muestra en la Figura 4.12, tomando los valores siguientes: $T_{ini} = 9000$ y $T_{fin} = 2$ y $N = 3000$.

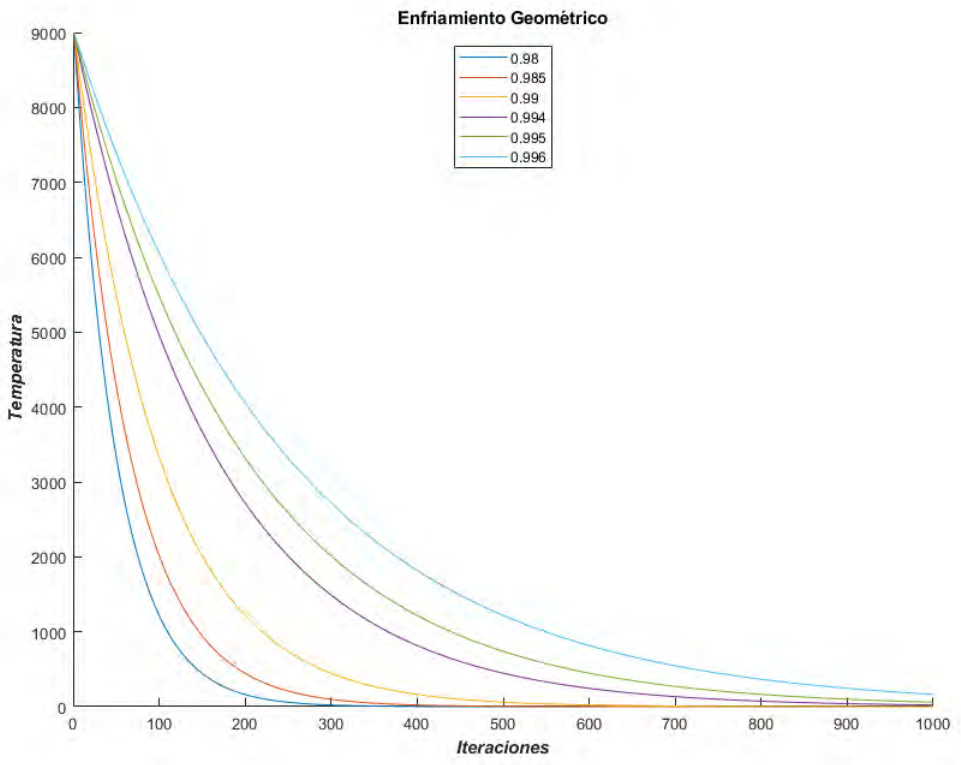


Figura 4.11: Comportamiento del enfriamiento geométrico.

4.5.2. Cruce en 2 Puntos

Para obtener mejores resultados el método más usado fue la cruza en 2 puntos, ya que es menos destructivo por lo que permite heredar mayor cantidad de información genética de los progenitores y de esta manera evolucionar con mejores resultados.

El algoritmo 3 muestra el pseudocódigo de este método.

4.5.3. Mutación

La método de mutación permite mantener la diversidad aún en las últimas iteraciones, con el propósito de realizar una mejor exploración del espacio de búsqueda. La mutación logarítmica permite realizar una mutación más efectiva dependiendo de la probabilidad de mutación como se muestra en la Ecuación 4.5.

$$\begin{aligned} t &= 1 + \frac{\ln(1-u)}{\ln(1-P_m)} \\ alelo\_mutado &+= t \end{aligned}, \tag{4.5}$$

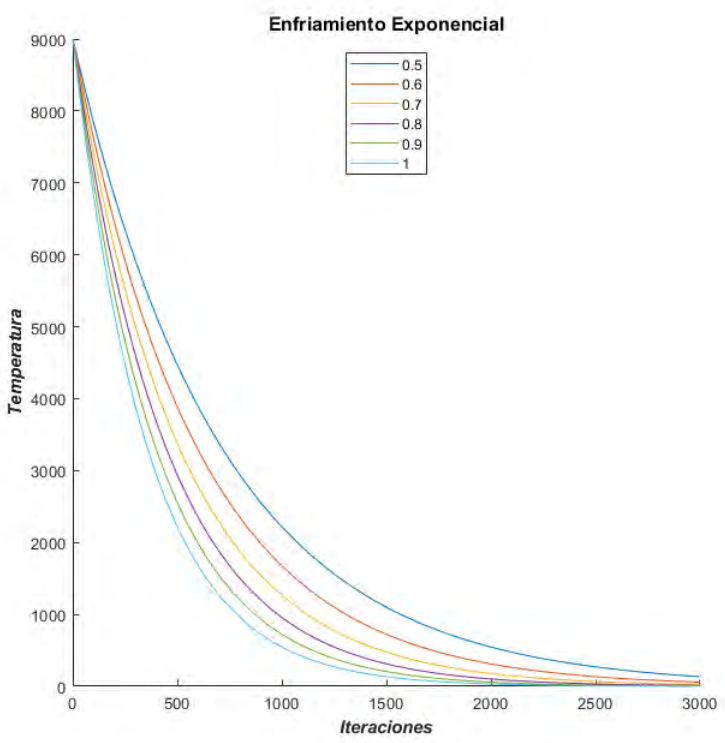


Figura 4.12: Comportamiento del enfriamiento exponencial.

donde t es el incremento de la posición en el cromosoma que va a mutar, u es un número aleatorio en el intervalo $[0;1]$, P_m es la probabilidad de mutación que selecciona el usuario al principio del algoritmo y $alelo\_mutado$ es la posición del alelo que va a ser mutado, inicialmente toma el valor de t y se va incrementando mientras sea menor que el tamaño del cromosoma, en caso que sea mayor, el offset (lo que sobra) es utilizado para el siguiente individuo.

4.6. Reparación del Cromosoma

Cuando se obtiene la mejor solución al final del proceso evolutivo, se realiza una reparación al cromosoma de modo que si quedó algún patrón traslapado o fuera de área, se pueda corregir. Para esto se utilizó una función que recorre el cromosoma y si encuentra algún traslape, cambia el bit de acomodo a 0, y así hasta recorrer todos los patrones, con el objetivo de quitar los patrones que estaban mal acomodados. Finalmente intenta volver a acomodar en las áreas de acomodo que quedaron libres. El pseudocódigo se muestra en en el Algoritmo 4.

Algoritmo 3 Cruce en 2 Puntos

```
1:  procedure CRUCE EN 2 PUNTOS
2:      Entrada:
3:      Padre1 → Cromosoma del Padre 1
4:      Padre2 → Cromosoma del Padre 2
5:      pto1 → Punto de Cruce 1
6:      pto2 → Punto de Cruce 2
7:      Hijo1 → Cromosoma del Hijo 1
8:      Hijo2 → Cromosoma del Hijo 2
9:      for i = 0; i < Tamaño_del_Cromosoma; i+ = 1 do
10:         if i < pto1 || i >= pto2 then
11:             Hijo1.Genes[i] = Padre1.Genes[i]
12:             Hijo2.Genes[i] = Padre2.Genes[i]
13:         else
14:             Hijo1.Genes[i] = Padre2.Genes[i]
15:             Hijo2.Genes[i] = Padre1.Genes[i]
16:         end if
17:     end for
18:     Output: Hijo1, Hijo2
19: end procedure
```

4.7. Generalización del Algoritmo

EL algoritmo para el acomodo de patrones abarca tres etapas:

1. Configuración de los parámetros, selección de las imágenes y su digitalización.

2. Proceso Evolutivo que comprende todas las etapas del AG.

3. Reparación de la Solución Final.

La Figura 4.13 muestra un diagrama de Flujo de todo el proceso.

Algoritmo 4 Reparación del Cromosoma

```
1:  procedure REPARAR CROMOSOMA
2:      Entrada:
3:      Cromosoma → Mejor Cromosoma
4:      Reparado → Solución Final
5:      Reparado ← Cromosomas.clone()
6:      for n = 0; n < Tamaño_del_Cromosoma; n+ = 5 do
7:          if Cromosoma.Genes[n+4] == 1 then
8:              Num_Patrones+ = 1
9:              rotada ← Patron rotado
10:             t, f ← false
11:             for i = 0, k = Cromosoma.Genes[n+2]; i <= rotada.rows; i++ do
12:                 for j = 0, l = Cromosoma.Genes[n+3]; j <= rotada.cols; j++ do
13:                     m ← Cromosoma.acomodo(k,l)
14:                     p ← rotada(i, j)
15:                     if m >= 1 & m < 10 & p == 1 then
16:                         t ← true; break;
17:                     else
18:                         if m >= 10 & p == 1 then
19:                             f ← true; break;
20:                         end if
21:                     end if
22:                     Reparado.acomodo(k,l) = m + p
23:                 end for
24:             end for
25:             if t||f then
26:                 Reparado.Genes[n+4] = 0
27:             end if
28:         end if
29:     end for
30:     Output: Reparado
31: end procedure
```

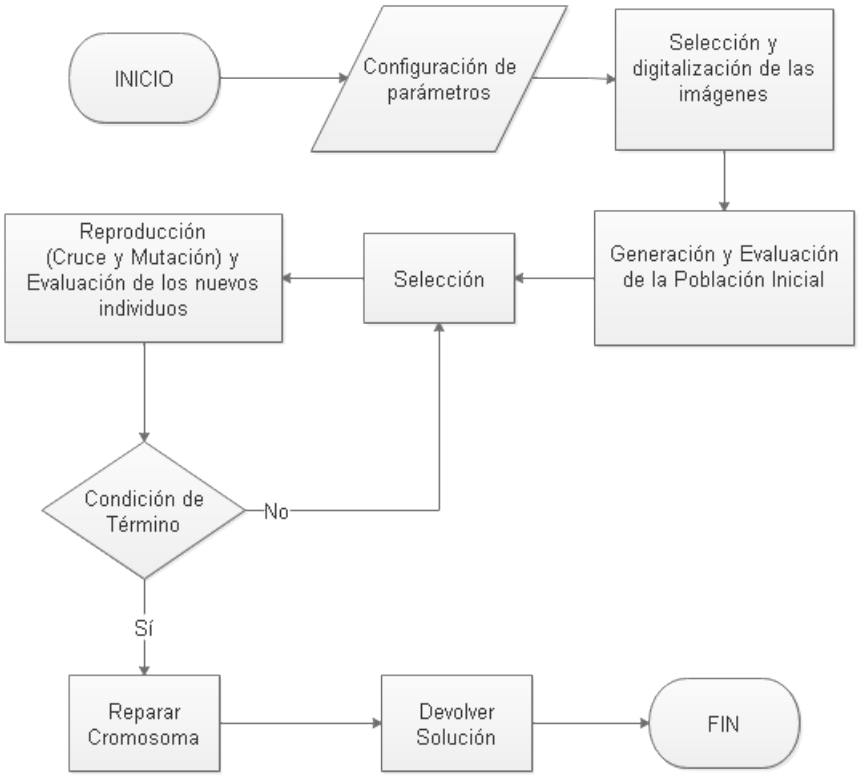


Figura 4.13: Diagrama de Flujo para el Acomodo de Patrones.

4.8. Conclusiones Parciales

En este capítulo se abordó todo lo relacionado con el análisis y la implementación utilizada para la solución del Acomodo de Patrones.

Se describió el proceso de binarización y la relevancia que tiene para el desarrollo de este problema. Además se explicó detalladamente la codificación cromosómica utilizada, la función objetivo que se diseñó que incluye toda la información necesaria que interviene en la minimización del desperdicio. Se explicaron los métodos implementados para llevar a cabo el proceso evolutivo hasta encontrar la mejor solución.

En el capítulo 5 se muestran los resultados de la investigación variando los parámetros y los operadores genéticos que han sido implementados para alcanzar los resultados satisfactorios.

CAPÍTULO 5

Resultados de la Investigación

En el presente capítulo se muestran los resultados obtenidos en esta investigación, utilizando todos los operadores genéticos implementados. Se realizaron pruebas con imágenes digitales y luego con imágenes reales y se comparan con resultados de investigaciones previas.

5.1. Parámetros Iniciales

En la configuración inicial son necesarios los parámetros iniciales que permiten ajustar el algoritmo para obtener una mejor convergencia. En el Cuadro 5.1 se muestran todos los parámetros y los valores que permiten sensibilizar el algoritmo.

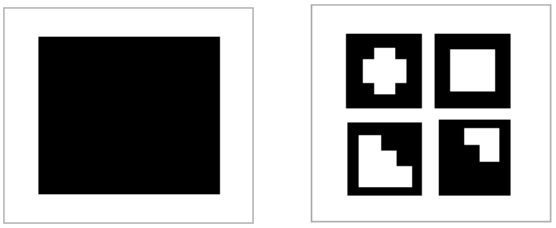
5.2. Material y Patrones de Prueba

Inicialmente para probar el algoritmo se utilizaron las imágenes mostradas en la Figura 5.1. Son patrones tomados de figuras geométricas, que sus cortes se complementan entre sí.

El tamaño de las imágenes de prueba para obtener resultados se manejó desde 16x15 píxeles hasta 260x200 píxeles. El tamaño de los patrones de corte utilizados se manejaron desde 5x5 píxeles hasta 60x70 píxeles. Las dimensiones de las imágenes en la Figura 5.1 fueron escaladas a conveniencia en el documento para visualizar mejor las formas de éstas.

Parámetro	Significado	Valores	Observaciones
L_{PaT}	Límite de los Patrones Traslapados	0	A partir de ese límite se penaliza los patrones traslapados en la función objetivo.
L_{PiT}	Límite de los Píxeles Traslapados	0	A partir de ese límite se penalizan los píxeles traslapados en la función objetivo.
L_{PaF}	Límite de los Patrones Fuera del Área de Acomodo	0	A partir de ese límite se penalizan los patrones que se encuentran fuera de área en la función objetivo.
L_{PiF}	Límite de los Píxeles Fuera del Área de Acomodo	0	A partir de ese límite se penalizan los píxeles que se encuentran fuera del área de acomodo en la función objetivo.
L_W	Límite del Desperdicio del Material	0	A partir de este valor se penaliza el factor desperdicio en la función objetivo.
W_{PaT}	Peso de los patrones traslapados	> 500	Valor con el que se penalizan los patrones traslapados en la función objetivo si no cumple el límite establecido.
W_{PiT}	Peso de los píxeles traslapados	> 500	Valor con el que se penalizan los píxeles traslapados en la función objetivo si no cumple el límite establecido.
W_{PaF}	Peso de los patrones que se encuentran fuera del área de acomodo	> 500	Valor con el que se penalizan los patrones que se encuentran fuera del área de acomodo en la función objetivo si no cumple el límite establecido.
W_{PiF}	Peso de los píxeles que se encuentran fuera del área de acomodo	> 500	Valor con el que se penalizan los píxeles que se encuentran fuera del área de acomodo en la función objetivo si no cumple el límite establecido.
W_W	Peso del Desperdicio en el Material	> 500	Valor con el que se penaliza el factor Desperdicio en la función objetivo si no cumple el límite establecido.
P_m	Probabilidad de Mutación	$<= 0{,}001$	Valor que se utiliza para realizar la mutación en el algoritmo genético y que haya variabilidad en las generaciones.

Cuadro 5.1: Sensibilidad de los parámetros.



(a) (b)

Figura 5.1: (a) Imagen del material de acomodo. (b) Imágenes de los patrones regulares.

5.2.1. Resultados de Acomodo Aplicando la Cruza en 1 Punto

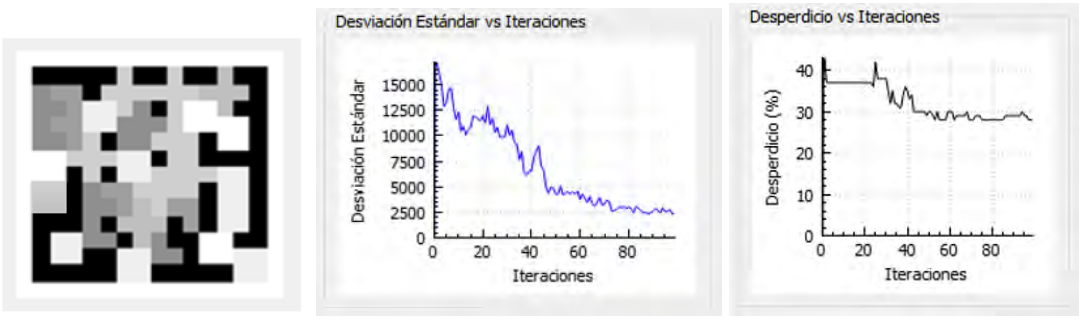
5.2.1.1. Corridas con 100 Iteraciones

Se realizaron pruebas variando los métodos de selección aplicando 100 iteraciones. Los resultados se muestran en el Cuadro 5.2.

Métodos de Selección	Mejor Iteración	Número de Patrones	Tiempo (s)	Desperdicio (%)	Aptitud
Selección por Ruleta	29	20	1	58	178000
Escalamiento de Aptitud	5	22	1	52	153000
Escalamiento Sigma	19	22	1	54	158000
Selección por Ranking	1	20	2	52	166000
Selección por Torneo	34	24	2	52	162000
Boltzmann. Enfriamiento Lineal	99	21	1	51	163000
Boltzmann. Enfriamiento Geométrico	100	29	1s	37	130000
Boltzmann. Enfriamiento Exponencial	100	31	1	35	125000

Cuadro 5.2: Resultados obtenidos aplicando el Cruce en 1 Punto con 100 iteraciones.

El mejor resultado se obtuvo con el enfriamiento exponencial aplicando el método de selección de Boltzman con temperaturas de $T_{ini} = 90000$ y $T_{fin} = 1$, con $\alpha = 0{,}4$ y una probabilidad de mutación de $P_m = 0{,}001$ como muestra la Figura 5.2.



(a) (b) (c)

Figura 5.2: (a) Resultado del acomodo obtenido con el Cruce en 1 Punto con 100 iteraciones. (b) Gráfica de la desviación estándar. (c) Gráfica del desperdicio durante el proceso evolutivo.

5.2.1.2. Corridas con 1000 Iteraciones

Se realizó el método de Cruce en 1 Punto variando los métodos de Selección como muestra el Cuadro 5.3. Se utilizaron 1000 iteraciones para las corridas de los diferentes métodos de Selección, usando una probabilidad de mutación $P_m = 0{,}0001$.

Es notable que los mejores resultados se obtienen con los métodos de Boltzmann ya que la presión de selección permite encontrar las mejores soluciones en las últimas iteraciones, a diferencia de los tres primeros métodos mostrados que convergen prematuramente con valores de aptitud mucho mayores a la mejor solución que se encuentra sombreada en la tabla 5.3.

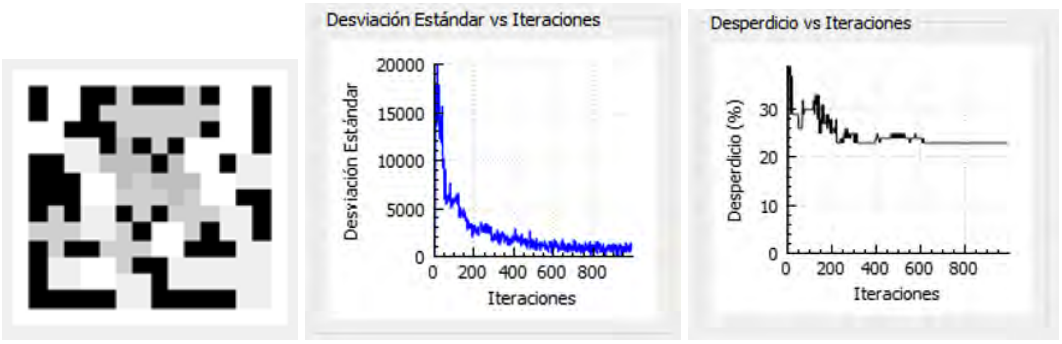
La Figura 5.3 muestra el acomodo obtenido aplicando la selección de Boltzmann con el Enfriamiento Geométrico, utilizando valores de temperatura: $T_{ini} = 90000$ y $T_{fin} = 1$, y con $\alpha = 0{,}995$, con una probabilidad de mutación de $P_m = 0{,}0001$, con los cuales se obtuvo el mejor resultado, ya que arrojó el menor valor de aptitud con la Cruza en 1 Punto. Además se muestran las gráficas de la desviación estándar y el desperdicio respectivamente en correspondencia con el número de iteraciones.

5.2.2. Resultados de Acomodo Aplicando la Cruza en 2 Puntos

Utilizando el Cruce en 2 Puntos para el procesos de recombinación de los cromosomas se obtienen mejores resultados, por lo que se utilizó este método variando los métodos de selección y el número de iteraciones.

Métodos de Selección	**Mejor Iteración**	**Número de Patrones**	**Tiempo (s)**	**Desperdicio (%)**	**Aptitud**
Selección por Ruleta	0	19	18	55	175000
Escalamiento de Aptitud	56	20	16	50	165000
Escalamiento Sigma	0	18	39	51	169000
Selección por Ranking	980	20	27	46	159000
Selección por Torneo	750	23	35	48	159000
Boltzmann. Enfriamiento Lineal	1000	22	49	49	161000
Boltzmann. Enfriamiento Geométrico	609	28	20	33	124000
Boltzmann. Enfriamiento Exponencial	605	26	20	36	131000

Cuadro 5.3: Resultados obtenidos aplicando el Cruce en 1 Punto con 1000 iteraciones.



(a) (b) (c)

Figura 5.3: (a) Resultado del acomodo obtenido con el Cruce en 1 Punto con 1000 iteraciones. (b) Gráfica de la desviación estándar. (c) Gráfica del desperdicio durante el proceso evolutivo.

5.2.2.1. Corridas con 100 iteraciones

La ventaja de realizar corridas de 100 iteraciones es que el tiempo de ejecución es mínimo y en sistemas en tiempo real es fundamental que la respuesta de los algoritmos sea rápida. El Cuadro 5.4 muestra una comparación cuantitativa de 100 iteraciones variando los métodos de selección.

En la Figura 5.4 se muestra el mejor resultado obtenido en las corridas de 100 iteraciones que se encuentra sombreado en el Cuadro 5.4, una vez más el método de selección de Boltzmann arroja los mejores resultados, aplicando el Enfriamiento Geométrico con una temperatura inicial de $T_{ini} = 90000$ y final de $T_{fin} = 1$, con $\alpha = 0{,}95$ y una probabilidad de mutación de $P_m = 0{,}95$.

(a) (b) (c)

Figura 5.4: (a) Resultado del acomodo obtenido aplicando el Cruce en 2 Puntos con 100 iteraciones. (b) Gráfica de la desviación estándar. (c) Gráfica del desperdicio durante el proceso evolutivo.

Métodos de Selección	**Mejor Iteración**	**Número de Patrones**	**Tiempo (s)**	**Desperdicio (%)**	**Aptitud**
Selección por Ruleta	16	20	1	54	170000
Escalamiento de Aptitud	8	21	2	54	168000
Escalamiento Sigma	0	20	2	57	176000
Selección por Ranking	14	22	2	50	159000
Selección por Torneo	12	22	2	51	162000
Boltzmann. Enfriamiento Lineal	63	25	2	45	149000
Boltzmann. Enfriamiento Geométrico	100	29	1	34	127000
Boltzmann. Enfriamiento Exponencial	100	29	1	36	128000

Cuadro 5.4: Resultados obtenidos aplicando el Cruce en 2 Puntos con 100 iteraciones.

5.2.2.2. Corridas con 1000 iteraciones

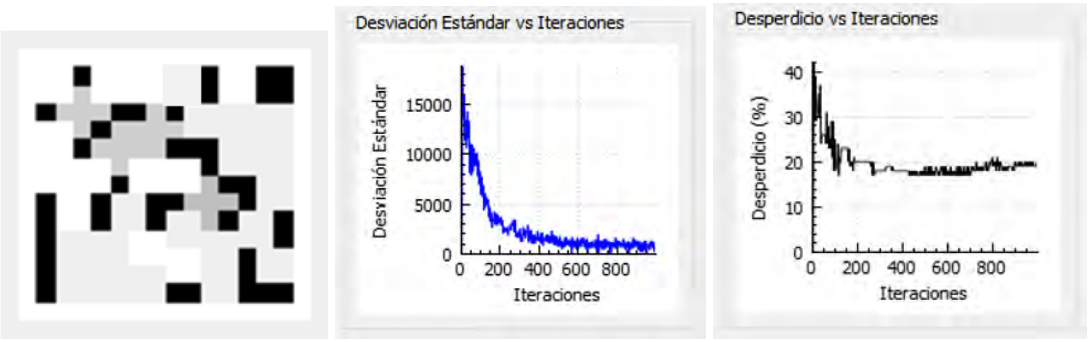
El Cuadro 5.5 muestra el cruce en 2 Puntos con 1000 iteraciones, donde se pudo observar que las peores soluciones son la obtenidas en la Ruleta, Escalamiento de Aptitud y Sigma, los cuales son métodos que tienen a converger prematuramente y el desperdicio queda por encima del 50 %. Los métodos que arrojan mejores resultados son los de Bolztman, en este caso la mejor solución se obtuvo con el Enfriamiento Exponencial utilizando valores de temperatura: $T_{ini} = 90000$ y $T_{fin} = 1$, y con $\alpha = 0{,}5$, con una probabilidad de mutación de $P_m = 0{,}0001$. El acomodo obtenido con el Método de Boltzmann aplicando el enfriamiento exponencial se muestra en la Figura 5.5

Métodos de Selección	Mejor Iteración	Número de Patrones	Tiempo (s)	Desperdicio (%)	Aptitud
Selección por Ruleta	0	18	52	57	180000
Escalamiento de Aptitud	30	19	31	51	170000
Escalamiento Sigma	51	18	17	53	172000
Selección por Ranking	994	19	14	49	165000
Selección por Torneo	640	21	15	49	164000
Boltzmann. Enfriamiento Lineal	1000	24	15	42	147000
Boltzmann. Enfriamiento Geométrico	1000	29	17	30	118000
Boltzmann. Enfriamiento Exponencial	1000	34	17	25	104000

Cuadro 5.5: Resultados obtenidos aplicando el Cruce en 2 Puntos con 1000 iteraciones.

5.2.2.3. Corridas con Mayor Número de iteraciones

También se hicieron pruebas aumentando el número de iteraciones a 3000 y 5000, la ventaja de hacer un gran número de iteraciones está en que se puede explotar más el espacio de búsqueda creando nuevas soluciones, sin embargo, el tiempo computacional



(a) (b) (c)

Figura 5.5: (a) Resultado del acomodo en 1000 iteraciones. (b) Gráfica de la desviación estándar. (c) Gráfica del desperdicio durante el proceso evolutivo.

se eleva considerablemente, algo que no es factible cuando se trata de llevar a la industria una aplicación.

5.3. Conclusiones Parciales

Los resultados obtenidos dan muestra de la versatilidad y robustez del algoritmo genético para resolver este tipo de problema, se corroboró la potencialidad del Método de Selección de Boltzmann ya que en todos los casos arrojó los mejores resultados. Cabe notar que el operador de cruce en 2 Puntos arrojó mejores resultados que el cruce en 1 Punto porque logró que se pudieran acomodar más patrones y a su vez que disminuyera el desperdicio.

Es importante aclarar que los resultados se obtienen en escala de grises, ya que al inicio de la configuración a cada patrón se le asigna un nivel de gris para diferenciarlo de los demás. Las imágenes que tienen bloques de un mismo color significa que existe adyacencia entre patrones de un mismo tipo, lo cual da muestra que el algoritmo fue lo suficiente inteligente para acomodar juntos patrones que poseen cortes semejantes, que a su vez permite que el acomodo quede más compacto y se ahorre más el material.

CAPÍTULO 6

Conclusiones y Trabajos a Futuro

6.1. Conclusiones

Como resultado de esta investigación se demostró la solidez y capacidad del AG implementado para resolver el problema de acomodo de patrones de corte en materiales finitos.

La codificación cromosómica propuesta es robusta y eficaz ya que permite de manera sencilla generar las combinaciones de los patrones de corte que intervienen en el acomodo. La función objetivo es eficiente porque posibilita guiar la búsqueda para encontrar una solución con el mayor aprovechamiento del material y a su vez que tenga el mínimo desperdicio.

Cabe notar que las mejores soluciones se obtuvieron con el método de selección de Boltzmann aplicando las funciones de enfriamiento que se diseñaron con el objetivo de disminuir la temperatura a medida que aumentan las iteraciones, lo que permitió explorar más el espacio de búsqueda y obtener mejores soluciones.

Otra de las ventajas del algoritmo implementado es que permite asociar patrones que tengan formas semejantes, además que el desperdicio sea menor comparado con otras investigaciones, generalmente los valores de desperdicio que se manejan están por debajo del 40 % y en la presente investigación se llegan a valores del 25 %.

En general, la desventaja que tienen los AGs es que cuando trabajan con poblaciones muy grandes tienden a ser muy lentos y por ende aumenta el tiempo de ejecución del proceso, lo que repercute negativamente en sistemas de tiempo real, sin embargo, paradójicamente una de las ventajas de los AGs es que permite manejar gran cantidad de datos y explotar el espacio de búsqueda considerando buenas y malas soluciones que evolucionan para encontrar una mejor durante el proceso evolutivo.

La investigación realizada puede ser aplicada en cualquier industria que precise del proceso de acomodo, ya que este algoritmo puede adaptarse a cualquier tipo de material que pueda ser digitalizado.

6.2. Trabajos a Futuro

Las principales metas para trabajos a futuro son:

- Detectar los niveles de calidad del material para evitar acomodar patrones en dichas áreas.
- Considerar el acomodo de los patrones en las diferentes áreas del material según la calidad de éste.
- Mejorar el tiempo de ejecución del algoritmo implementando técnicas de paralelización del algoritmo genético.
- Implementar otras técnicas metaheurísticas para combinarlas o adaptarlas al problema de acomodo, con el fin de obtener un algoritmo global que ofrezca mayores bondades a este problema de alta complejidad.

APÉNDICE A

Función de Binarización

```
Mat Binarizar(Mat m, int U, int id)
    {
        int i, j;
        Mat b = Mat::zeros(m.rows, m.cols, CV_8UC1);
        if(id == 0)      //Binarizar Material
        {
            for (i = 0; i < m.rows; i++)
                for (j = 0; j < m.cols; j++)
                {
                    uchar pixel = m.at<uchar>(i, j);
                    if ((int)pixel > U)
                        b.at<uchar>(i, j) = 10;
                    else
                        b.at<uchar>(i, j) = 0;
                }
        }else
            if(id==1)    //Binarizar Patr n
            {
                for (i = 0; i < m.rows; i++)
                    for (j = 0; j < m.cols; j++)
                    {
                        uchar pixel = m.at<uchar>(i,
                           j);
                        if ((int)pixel > U)
                            b.at<uchar>(i, j) = 1;
                        else
                            b.at<uchar>(i, j) = 0;
                    }
            }
        return b;
    }
```

APÉNDICE B

Método de Otsu

```
int OTSU(Mat m)
    {
        int t, histo[256], wB, wF, TH,  np;
        float  sum, sumB, sumF, varmax, sB, uB, uF;
        sum = sumB = sumF = uB = uF = wB = wF =
           varmax = TH = sB = 0;
        np = m.cols * m.rows;

        for (t = 0; t < 256; t++)
            histo[t] = 0;

        histograma(m, histo);

        for (t = 0; t < 256; t++)
            sum += (t * histo[t]) + 0.0;

        for (t = 0; t < 256; t++)
        {
            wB += histo[t];
            if (wB == 0) continue;

            wF = np - wB;
            if (wF == 0) break;

            sumB += (t * histo[t]) + 0.0;
            uB = (sumB / wB) + 0.0;
            uF = ((sum - sumB) / wF) + 0.0;

            sB = (wB * wF * pow(uB - uF, 2)) + 0.0;
            if (sB > varmax)
            {
                varmax = sB;
                TH = t;
            }
        }
```

```
        return TH;
    }
```

APÉNDICE C

Inicialización de la Población Inicial

```
void InicializarPoblacion()
  {
       int i, k, j, nP, x, y, angulo, valores[6];
       valores[0] = 0; valores[1] = 0; valores[2] = 0;
          valores[3] = 0; valores[4] = 0; valores[5] =
          0;

       for(i = 0; i < Poblacion; i++)
          {
              for(j = 0; j < 5 * Total_de_Patrones; j
                 +=5)
               {
                   //genero el identificador del
                      patr n
                   nP = rand() % (numPatrones);
                   pob_ini[i].Genes[j] = nP;

                   //Angulo aleatorio para cada patr n
                   k = rand() % (st_patrones[nP].
                      cantidad_angulos);
                   angulo = (360 / (st_patrones[nP].
                      cantidad_angulos)) * k;
                   pob_ini[i].Genes[j+1] = angulo;

                   //Posiciones x = fila,y = columna
                      del material de acomodo
                   x = rand() % (pob_ini[i].acomodo.
                      rows);
                   y = rand() % (pob_ini[i].acomodo.
                      cols);

                   while(true)
                   {
                       int filas_patron = st_patrones[
                          pob_ini[i].Genes[j]].p.rows;
```

```
                        int columnas_patron =
                           st_patrones[pob_ini[i].Genes[
                           j]].p.cols;
                        if(((x + filas_patron) < pob_ini
                           [i].acomodo.rows) && ((y +
                           columnas_patron) < pob_ini[i
                           ].acomodo.cols))
                        {
                            pob_ini[i].Genes[j+2] = x;
                            pob_ini[i].Genes[j+3] = y;
                            break;
                        }
                        else
                        {
                            x = rand() % (pob_ini[i].
                               acomodo.rows);
                            y = rand() % (pob_ini[i].
                               acomodo.cols);
                        }
                    }

                    //1 si est  dentro del acomodo y 0
                       si no est
                    k = rand() % 2;
                    pob_ini[i].Genes[j+4] = k;
                }

                 Traslapes_FueraArea(pob_ini[i],valores);
                 pob_ini[i].traslapes = valores[0];
                 pob_ini[i].fuera_area = valores[1];
                 pob_ini[i].num_traslapes = valores[2];
                 pob_ini[i].num_fuera_area = valores[3];
                 pob_ini[i].area_ocupada = valores[4];
                 pob_ini[i].num_Patrones =  valores[5];
                 pob_ini[i].Fitness = Fitness(pob_ini[i])
                    ;
            }
            return;
}
```

APÉNDICE D

Función Objetivo

```
int Fitness(Cromosomas pob)
{
    int FF = 0, desperdicio = 0, patrones = 0;
    patrones = Total_de_Patrones - pob.num_Patrones;
    desperdicio = st_material.area - pob.area_ocupada
       ;

    if(patrones <= 0 && desperdicio <= 0 && pob.
       traslapes <= limite_pixeles && pob.
       num_traslapes <= limite_patrones && pob.
       num_fuera_area <= limite_pat_fuera_area && pob
       .fuera_area <= limite_pix_fuera_area)
         FF = patrones + desperdicio + pob.traslapes
            + pob.fuera_area + pob.num_traslapes +
            pob.num_fuera_area;
    else
        FF = 1000 * patrones + w_waste * desperdicio
           + w_pix * pob.traslapes + w_pix_fa * pob.
           fuera_area + w_pat * pob.num_traslapes +
           w_pat_fa * pob.num_fuera_area;

    return FF;
}
```

APÉNDICE E

Patrones y Píxeles Traslapados y Fuera de Área

```
void Traslapes_FueraArea(Cromosomas pob, int array
   [6])
  {
      int i, j, k = 0, l= 0, m = 0, p = 0, n,
         traslape = 0, fuera_area = 0, Area_patrones
         = 0, tr = 0, fa = 0;
      bool t, f;
      int pos = 0, cant_ang = 0, cant_pat = 0;

      for(n = 0; n < Total_de_Patrones * 5; n+=5)
      {
          Mat rotada = Mat::zeros(st_patrones[pob.
             Genes[n]].p.rows,st_patrones[pob.Genes[n
             ]].p.cols, CV_8UC1);
          cant_ang = 0;
          if(pob.Genes[n+4] == 1)   // Incluyo el
             patr n
           {
              cant_pat ++;

              pos = (pob.Genes[n+1] * st_patrones[pob
                 .Genes[n]].cantidad_angulos) / 360;
              for(int r = 0; r < pob.Genes[n]; r++)
                     cant_ang += st_patrones[r].
                        cantidad_angulos;

              if(pob.Genes[n] == 0)
                    rotada = arrayPat[pos].p.clone();
              else
                    rotada = arrayPat[pos + cant_ang
                       ].p.clone();

               t = false;
               f = false;

               for(i = 0, k = pob.Genes[n+2]; i <
```

```
                        rotada.rows; i++, k++)
                         for(j = 0, l = pob.Genes[n+3]; j <
                             rotada.cols; j++, l++)
                         {
                              m = pob.acomodo.at<uchar>(k, l
                                 );
                              p = rotada.at<uchar>(i,j);
                              if(m == 0 && p == 1)
                                   Area_patrones++;
                              else
                                   if(m >= 1 && m < 10 && p
                                      == 1)
                                   {
                                        traslape++;
                                        t = true;
                                   }
                                   else
                                        if(m >= 10 && p == 1)
                                        {
                                             fuera_area++;
                                             f = true;
                                        }

                              pob.acomodo.at<uchar>(k, l) =
                                 m + p;

                         }
                    if(t)
                         tr +=1;

                    if(f)
                         fa +=1;
               }
          }
          array[0] = traslape;    //n mero de pixeles
             traslapados
          array[1] = fuera_area; //n mero de pixeles
             fuera del  rea  de acomodo
          array[2] = tr;          //n mero de patrones
             traslapados
          array[3] = fa;          //n mero de patrones
             fuera del  rea  de acomodo
          array[4] = Area_patrones;  //n mero de pixeles
              que est n bien acomodados dentro del  rea
              de acomodo.
          array[5] = cant_pat;

        return;
     }
```

Bibliografía

[1] F. Cuevas, O. Gonzalez, Y. Suzuki, D. Hernández, M. Roche, and N. Alcalá, "Genetic algorithms applied to optics and engineering," vol. 6046, 2006. [Online]. Available: https://doi.org/10.1117/12.674556

[2] R. E. González, Rafael C. y Woods, *Tratamiento Digital de Imágenes*, 1st ed. Wilmington, Delaware, E.U.A.: Addison Wesley Iberoamericana S.A. y Ediciones Díaz de Santos S.A., 1996, versión en Español.

[3] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 3rd ed. Upper Saddle River, New Jersey 07458: Pearson Education, Inc., 2008.

[4] R. Martí, *Multi-Start Methods*. Boston, MA: Springer US, 2003, pp. 355–368. [Online]. Available: https://doi.org/10.1007/0-306-48056-5_12

[5] C. R. Reeves, "Modern heuristic techniques for combinatorial problems," C. R. Reeves, Ed. New York, NY, USA: John Wiley & Sons, Inc., 1993, ch. Genetic Algorithms, pp. 151–196. [Online]. Available: http://dl.acm.org/citation.cfm?id=166648.166657

[6] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing." *Science*, vol. 220 4598, pp. 671–80, 1983.

[7] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization," *Swarm Intelligence*, vol. 1, no. 1, pp. 33–57, Jun 2007. [Online]. Available: https://doi.org/10.1007/s11721-007-0002-0

[8] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, Oct 1995, pp. 39–43.

[9] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, 1st ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989.

[10] R. E. Leakey., *Charles Darwing, El origen de las especies. Versión Abreviada*, 1st ed. DF, México.: Martín Casillas Editores, S. A., 1980.

[11] J. H. Holland, *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence.* Oxford, England: University Michigan Press, 1975.

[12] E. Hopper and B. C. H. Turton, "A review of the application of meta-heuristic algorithms to 2d strip packing problems," *Artificial Intelligence Review*, vol. 16, no. 4, pp. 257–300, Dec 2001. [Online]. Available: https://doi.org/10.1023/A:1012590107280

[13] C. Gil, J. Pablo Orejuela Cabrera, and D. Peña, "El problema de patrones de corte, clasificación y enfoques/cutting stock problem, classification and approaches," *Prospectiva*, vol. 15, p. 112, 02 2017.

[14] R. C. E. Gilmore and R. Gomory, "A linear programming approach to the cutting stock problem i," *Oper Res*, vol. 9, 01 1961.

[15] R. W. Haessler and P. E. Sweeney, "Cutting stock problems and solution procedures," *European Journal of Operational Research*, vol. 54, no. 2, pp. 141 – 150, 1991. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0377221791902935

[16] L. V. Kantorovich, "Mathematical methods of organizing and planning production," *Manage. Sci.*, vol. 6, no. 4, pp. 366–422, Jul. 1960. [Online]. Available: http://dx.doi.org/10.1287/mnsc.6.4.366

[17] Y. M. S. Marin, "Simulación del acomodo de patrones de corte en materiales," Ph.D. dissertation, Instituto Tecnológico de León, León, Guanajuato, México, 2003.

[18] G. C. Onwubolu and M. Mutingi, "A genetic algorithm approach for the cutting stock problem," *Journal of Intelligent Manufacturing*, vol. 14, no. 2, pp. 209–218, Apr 2003. [Online]. Available: https://doi.org/10.1023/A:1022955531018

[19] P. Poshyanonda and C. H. Dagli, "Genetic neuro-nester," *Journal of Intelligent Manufacturing*, vol. 15, no. 2, pp. 201–218, Apr 2004. [Online]. Available: https://doi.org/10.1023/B:JIMS.0000018033.05556.65

[20] M. Hma Jahromi, R. Tavakkoli-Moghaddam, A. Makui, and A. Shamsi, "Solving an one-dimensional cutting stock problem by simulated annealing and tabu search," *Journal of Industrial Engineering International*, vol. 8, 03 2012.

[21] E. López-Camacho, H. Terashima-Marin, P. Ross, and G. Ochoa, "A unified hyper-heuristic framework for solving bin packing problems," *Expert Syst. Appl.*, vol. 41, no. 15, pp. 6876–6889, Nov. 2014. [Online]. Available: http://dx.doi.org/10.1016/j.eswa.2014.04.043

[22] H. Terashima-Marín, P. Ross, C. J. Farías-Zárate, E. López-Camacho, and M. Valenzuela-Rendón, "Generalized hyper-heuristics for solving 2d regular and irregular packing problems," *Annals of Operations Research*, vol. 179, no. 1, pp. 369–392, Sep 2010. [Online]. Available: https://doi.org/10.1007/s10479-008-0475-2

[23] A. M. Del Valle, T. A. De Queiroz, F. K. Miyazawa, and E. C. Xavier, "Heuristics for two-dimensional knapsack and cutting stock problems with items of irregular shape," *Expert Syst. Appl.*, vol. 39, no. 16, pp. 12 589–12 598, Nov. 2012. [Online]. Available: http://dx.doi.org/10.1016/j.eswa.2012.05.025

[24] J. F. Gonçalves and M. C. Resende, "A biased random key genetic algorithm for 2d and 3d bin packing problems," 2013.

[25] Y.-P. Cui, Y. Cui, and T. Tang, "Sequential heuristic for the two-dimensional bin-packing problem," *European Journal of Operational Research*, vol. 240, no. 1, pp. 43–53, 2015. [Online]. Available: https://ideas.repec.org/a/eee/ejores/v240y2015i1p43-53.html

[26] F. Brandão and J. P. Pedroso, "Bin packing and related problems: General arc-flow formulation with graph compression," *Computers OR*, vol. 69, pp. 56–67, 2016.

[27] D. Zhang, L. Shi, S. Leung, and T. Wu, "A priority heuristic for the guillotine rectangular packing problem," *Information Processing Letters*, vol. 116, pp. 15–21, 01 2016.

[28] J. Martinovic, G. Scheithauer, and J. Valério de Carvalho, "A comparative study of the arcflow model and the one-cut model for one-dimensional cutting stock problems," *European Journal of Operational Research*, vol. 266, no. 2, pp. 458–471, 2018. [Online]. Available: https://ideas.repec.org/a/eee/ejores/v266y2018i2p458-471.html

[29] G. Martinsanz, J. de la Cruz García, and J. de la Cruz García, *Visión por computador: imágenes digitales y aplicaciones*, ser. Alfaomega Grupo Editor. Alfaomega Grupo Editor, 2002. [Online]. Available: https://books.google.com.mx/books?id=drDJAAAACAAJ

[30] C. A. Coello Coello, D. A. Van Veldhuizen, and G. Lamont, *Genetic Algorithms and Evolutionary Computation*, 01 2002, vol. 5.

[31] C. R. Reeves, Ed., *Modern Heuristic Techniques for Combinatorial Problems*. New York, NY, USA: John Wiley & Sons, Inc., 1993.

[32] F. Glover and M. Laguna, *Tabu Search*. Norwell, MA, USA: Kluwer Academic Publishers, 1997.

[33] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983. [Online]. Available: https://science.sciencemag.org/content/220/4598/671

[34] J. Kennedy, *Particle Swarm Optimization*. Boston, MA: Springer US, 2010, pp. 760–766. [Online]. Available: https://doi.org/10.1007/978-0-387-30164-8_630

[35] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360)*, May 1998, pp. 69–73.

[36] J. Holland, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. University of Michigan Press, 1975. [Online]. Available: https://books.google.com.mx/books?id=JE5RAAAAMAAJ

[37] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, 1st ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989.

[38] K. A. D. Jong, "An Analysis of the Behavior of a Class of Genetic Adaptive Systems," PhD thesis, University of Michigan College of Literature, Science, and the Arts Computer and Communication Sciences Department. Ann Arbor, MI, USA, August 1975.

[39] J. E. Baker, "Reducing bias and inefficiency in the selection algorithm," in *Proceedings of the Second International Conference on Genetic Algorithms on Genetic Algorithms and Their Application*. Hillsdale, NJ, USA: L. Erlbaum

Associates Inc., 1987, pp. 14–21. [Online]. Available: http://dl.acm.org/citation.cfm?id=42512.42515

[40] D. E. Goldberg and K. Deb, "A comparative analysis of selection schemes used in genetic algorithms," in *FOGA*, 1990.

[41] D. E. Goldberg, "A note on boltzmann tournament selection for genetic algorithms and population-oriented simulated annealing," *Complex Systems*, vol. 4, 1990.

[42] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, Jan 1979.

[43] M. A. S. Elmohamed, P. Coddington, and G. Fox, "A comparison of annealing techniques for academic course scheduling," in *Practice and Theory of Automated Timetabling II*, E. Burke and M. Carter, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 92–112.