Universidad Veracruzana

Instituto de Investigaciones en Inteligencia Aritifial

# Un estudio experimental de operadores de mutación para el algoritmo genético de agrupación para la segmentación de imágenes en RGB

T E S I S

que para obtener el título de:

Maestro en Inteligencia Artificial

presenta:

Oscar Fernández Solano

Directora:

Dra. Marcela Quiroz Castellanos

Codirector:

Dr. Nicandro Cruz Ramírez

Xalapa, Veracruz, México 2023

*A mi linda Chío, por su amor tan bonito y especial, por ser mi inspiración y motivarme siempre a seguir adelante.*

# Agradecimientos

Quiero agradecer a todas las personas que han aparecido en mi vida y han influido en lo que ahora soy, en lo que he logrado hasta ahora.

Primero que nada, quiero agradecer a CONAHCYT por brindarme el apoyo económico durante mis estudios de maestría. Así como a la Universidad Veracruzana y especialmente al Instituto de Investigaciones en Inteligencia Artificial por brindar acceso a posgrados de calidad y apoyarme durante mis estudios.

Agradezco a la Dra. Marcela Quiroz Castellanos por su apoyo como directora de mi trabajo de tesis y por darme la oportunidad de conocer personas importantes de mi área y de asistir a increíbles eventos. Así como a mi codirector de tesis, el Dr. Nicandro Cruz Ramírez por habernos recibido en la maestría con sus cursos tan interesantes y siempre preocupado por nuestro bienestar. También agradezco a las personas que he tenido la fortuna de conocer, tanto de la UV como de otras universidades, por la dedicación que tienen, por compartir sus conocimientos con los estudiantes y por brindarme su amistad.

Quiero agradecer también al equipo de trabajo de Gatitos G. He aprendido mucho de ustedes y agradezco mucho su amistad tan especial que me han brindado.

A mis compañeros de generación les agradezco mucho por su amistad, siempre fuimos un grupo genial y unido.

Agradezco a mis amigos y a mi familia por su apoyo y por su paciencia, ya que me fue más difícil estar con ellos, pero siempre están en mi mente.

Agradezco a mi novia, mi dulce Chío, quien ha sido mi inspiración y mi motivación. Así como los algoritmos genéticos se inspiraron de la idea de la evolución de las especies, yo me inspiré en mi bióloga favorita para seguir adelante.

# Resumen

La optimización combinatoria tiene una gran relevancia debido a su aplicabilidad y a que puede ayudar a resolver problemas complejos de la vida real. Dentro de estos problemas, existen un tipo de problemas relacionados con agrupación de objetos, estos son llamados problemas de agrupación. Uno de los problemas de agrupación, que se estudia en este trabajo es la segmentación de imágenes. Esta técnica es un paso fundamental que permite simplificar la información contenida en una imagen para poder realizar tareas subsecuentes de análisis de imágenes de forma más eficiente y con mejor precisión.

Debido a que en la segmentación se trata de agrupar los elementos que conforman una imagen, esta puede verse como un problema de agrupación en la que la similitud está basada en el color de los objetos presentes en la imagen. Además, la complejidad de este problema es tal que es necesario recurrir a técnicas aproximadas para su resolución. Así, este problema es abordado con el uso de algoritmos genéticos de agrupación, un tipo de algoritmo genético diseñado para problemas de agrupación y que presenta un buen desempeño resolviendo este tipo de problemas. Dentro de la literatura, existe una propuesta de este tipo de algoritmos que presenta un muy buen desempeño para otro problema específico de esta clase, por lo que se parte de la adaptación de dicho algoritmo para el problema de la segmentación de imágenes, el cual no ha sido abordado en la literatura con este tipo de algoritmos.

Sin embargo, los componentes utilizados, así como la complejidad y características del problema influyen en el desempeño de un algoritmo genético, de forma que es posible mejorar el algoritmo modificando sus componentes e incorporando conocimiento del dominio del problema. En ese sentido, este trabajo se centró en mejorar el desempeño del algoritmo a través de un estudio experimental de uno de sus elementos, el operador de mutación, el

cual influye en gran medida dicho desempeño. Para ello, se sigue un diseño experimental en tres fases para probar distintos operadores de mutación y sus parámetros a configurar segmentando un conjunto de imágenes de control. A partir de estos experimentos, surgió una nueva versión que incluye un operador de mutación que proporciona mejores resultados. Estos fueron validados segmentando más imágenes y realizando pruebas estadísticas, incluyendo una comparación con K-Means.

Los resultados experimentales muestran una mejora en el desempeño del algoritmo con el nuevo operador de mutación en cuanto a la calidad de las soluciones. Dicha mejora es más notable con las imágenes más difíciles de segmentar, donde la nueva versión del algoritmo tuvo un desempeño notablemente mejor. Con este trabajo se obtuvieron dos algoritmos para la segmentación de imágenes a color y se demuestra la importancia de hacer un estudio experimental para diseñar un nuevo operador de mutación en un problema de agrupación nuevo en la literatura de algoritmos genéticos de agrupación.

Este estudio representa una primera aproximación al problema de segmentación de imágenes utilizando este tipo de algoritmos genéticos y, considerando que los resultados son muy cercanos a los que se obtienen con K-Means y que el enfoque fue únicamente el operador de mutación, existe un margen muy amplio de oportunidad de mejora de este algoritmo.

# Aportaciones del trabajo de tesis

A continuación se mencionan las aportaciones obtenidas durante la realización del presente trabajo de investigación:

1. Artículo “Un estudio experimental de operadores de mutación para el algoritmo genético de agrupación para la segmentación de imágenes en RGB” aceptado para ser publicado en las memorias del Second Ibero-American Symposium of Master and Doctorate in Artificial Intelligence 2023.

2. Participación en el International Seminar of Computational Intelligence (ISCI, 2022), Tijuana, con el trabajo “A Grouping Genetic Algorithm for RGB Image Segmentation”.

3. Participación en el 10th International Workshop on Numerical and Evolutionary Optimization (NEOX2022), México, con el trabajo “A Grouping Genetic Algorithm for RGB Image Segmentation”.

4. Participación en el primer congreso virtual Expo Virtual Meet-Conf 2022, impartiendo el taller: “Algoritmos Genéticos para Problemas de Agrupación”.

5. Participación en el 8vo. Seminario de Socialización del Aprendizaje Computacional (SAC), Xalapa, con el trabajo “IA con imágenes: más allá de lo que ves".

# Índice general

# Índice de figuras

# Índice de tablas

# Índice de algoritmos

# Capítulo 1

# Introducción

Los problemas de optimización se dividen en continuos y discretos. Como su nombre lo indica, en los problemas continuos las variables toman valores de un conjunto infinito. En los problemas discretos, estas variables pueden tomar valores de un conjunto finito o infinito, pero estos son discretos, como los números enteros o binarios. Dentro de los problemas discretos están los problemas de optimización combinatoria, en los cuales se busca un conjunto, permutación o combinación de objetos, un grafo o un valor entero dentro de un conjunto finito de elementos (Papadimitriou and Steiglitz, 1998). Muchos de los problemas de optimización pueden ser resueltos utilizando un algoritmo con complejidad polinomial. Sin embargo existen problemas muy difíciles de resolver que un algoritmo con esta característica no garantiza que una solución óptima pueda ser encontrada en todos los casos, estos problemas son conocidos como problemas NP-duros (Yu and Gen, 2010). Ante esta situación, lo deseable es encontrar soluciones aceptables para cualquier instancia de un problema, razón por la que se recurre a técnicas aproximadas y metaheurísticas (Eiben et al., 2003), como los algoritmos de la Computación Evolutiva (EC, por sus siglas en inglés), uno de los paradigmas de la Computación Inteligente (CI, por sus siglas en inglés).

Los algoritmos de la EC están inspirados en la metáfora de la evolución natural, donde el concepto principal es la supervivencia del más apto. Estos algoritmos tratan de imitar la evolución natural, creando descendientes a partir de un conjunto de padres que transmiten sus características. La aleatoriedad juega un papel importante, pero también es una cuestión de prueba y error donde los individuos menos aptos no sobreviven. En la EC, se

usa el término de población, conformada por un conjunto de individuos. Estos individuos son llamados cromosomas, que a su vez están formados por un conjunto de características o genes, cuyos valores reciben el nombre de alelos. El proceso evolutivo se lleva a cabo por generaciones. En cada generación, los individuos compiten para trasmitir sus genes, donde los mejores tienen una mayor probabilidad de ser seleccionados para generar nuevos individuos a través la cruza o recombinación, también interfieren algunos procesos como la mutación, que altera algunos genes de un cromosoma. Al final de cada generación los nuevos individuos son introducidos a la población reemplazando a individuos de dicha generación. La forma en que se evalúa el nivel de aptitud de un individuo es dependiente del problema. Dada la gran diversidad de problemas que se pueden encontrar, diferentes algoritmos evolutivos (EAs, por sus siglas en inglés) han sido propuestos: algoritmos genéticos (GAs, por sus siglas en inglés), programación genética, programación evolutiva, estrategias evolutivas, evolución diferencial, evolución cultural y coevolución (Engelbrecht, 2007). Dentro de estos, destacan los GAs en la resolución de problemas complejos, especialmente los algoritmos genéticos de agrupación (GGA, por sus siglas en inglés) que han mostrado buenos resultados para resolver problemas de agrupación (Agustı et al., 2012), (Ramos-Figueroa et al., 2020).

## 1.1. Justificación

Muchas aplicaciones del área de visión por computadora emplean la segmentación de imágenes como un pre-procesamiento que permita entender de manera más sencilla las partes que conforman la imagen, por lo que es un pilar importante para esta área y para la inteligencia artificial. La importancia de la segmentación de imágenes radica en que esta técnica es considerada el cuello de botella para el desarrollo de tecnologías de procesamiento de imágenes, debido a que las tareas de más alto nivel dependen en gran medida de ella. También es de las tareas de análisis de imágenes de bajo nivel más difíciles (Tao et al., 2003), lo que se puede atribuir a la múltiple cantidad de objetos que puede haber en una imagen y la gran variación que existe entre ellos (Halder et al., 2011).

Algunas aplicaciones que tiene la segmentación de imágenes o para las que se requiere

esta técnica son el reconocimiento de patrones, detección de objetos, detección de bordes, extracción de características, reconocimiento de objetos (Awad et al., 2007), aplicaciones médicas como detección y extracción de tumores en imágenes de resonancia magnética (Halder et al., 2016), aplicaciones de sensado remoto, como las imágenes satelitales (Maulik and Bandyopadhyay, 2003). Como se puede observar, el campo de aplicación es muy amplio y relevante en todas estas tareas, por lo que proponer técnicas de segmentación que brinden buenas soluciones resulta oportuno y de gran importancia para el correcto funcionamiento de estas técnicas.

Los métodos propuestos usan varios criterios como el nivel de gris, texturas o color (Awad et al., 2009). El color es uno de los aspectos más importantes en la visión y es muy útil para discriminar y reconocer información (Hassanien and Oliva, 2017), por lo que considerar los colores para la tarea de segmentación puede resultar interesante.

Los problemas de optimización combinatoria han sido de gran interés por su aplicación en problemas complejos de la vida real. Entre estos están los problemas de clustering, los cuales, desde el punto de vista del aprendizaje automático, se consideran desafiantes, particularmente por su naturaleza no supervisada. Además, de acuerdo con Welch (1982), el clustering también es considerado un problema NP-duro, lo que justifica el uso de técnicas aproximadas adecuadas, especialmente el uso de EAs y de forma particular, de GGAs, pues se trata de un problema de agrupación.

Al tratarse de un problema NP-duro, no existe una propuesta que se pueda considerar como la mejor, el área de oportunidad siempre está abierta y el uso de GGAs para la segmentación de imágenes no ha sido explorado, por lo que la aportación al área de visión por computadora es importante y novedosa.

Uno de los métodos empleados para la segmentación de imágenes es el clustering, el cual tiene un enfoque no supervisado. Esto presenta algunas ventajas sobre métodos supervisados, como el no requerir de un especialista para definir las clases del conjunto de datos, pues el algoritmo de clustering se encarga de encontrarlas. También es posible que algunas características de los objetos no sean conocidas en ese momento, las cuales podrían ser marcadas más adelante por el algoritmo (Omran et al., 2005).

Lo que se ha observado en la literatura es que las propuestas para segmentación de

imágenes se basan en la umbralización, el cual es un enfoque en el que se busca dividir el histograma de intensidades de los píxeles (Nakane et al., 2020), donde los GAs suelen ser empleados para encontrar los rangos adecuados para realizar esta umbralización. Por otro lado, también existen propuestas de GAs para realizar el clustering de forma particional; es decir, que el propio algoritmo y sus operadores lidian con los píxeles como objetos y busca asignarlos en las particiones (o clusters) adecuadas. Sin embargo, Falkenauer (1996) identificó que los GAs presentan algunos inconvenientes para resolver problemas de agrupación debido a la codificación empleada para representar las soluciones del problema. Estos inconvenientes generan soluciones redundantes y hace que los operadores de cruza, los cuales se encargan de explotar las buenas soluciones, pierdan información importante en el proceso evolutivo y que los operadores de mutación, que se encargan de explorar el espacio de búsqueda, destruyan soluciones muy buenas, es decir, deshace el trabajo del operador de cruza cuando este último ha empezado a generar buenas soluciones. Debido a esto, emplear un GA para resolver el problema de segmentación de imágenes, visto como un problema de clustering, no es la opción más adecuada, pues el poder de búsqueda del GA se ve ampliamente afectado por estos inconvenientes en la representación de las soluciones.

Debido a lo anterior, para este tipo de problemas lo más adecuado es usar una representación basada en grupos; es decir, el uso de GGAs. Lo que implica emplear operadores de variación diseñados para este tipo de representación. También, el visualizar el problema de esta forma resulta más natural y fácil de trabajar.

## 1.2. Planteamiento del problema

El Algoritmo Genético de Agrupación con Transmisión de Genes Controlada (GGA-CGT, por sus siglas en inglés) propuesto por Quiroz-Castellanos et al. (2015) es uno de los mejores algoritmos del estado del arte para resolver el problema de empacado de objetos, su metodología de diseño se basó en la experimentación con distintas estrategias e incorporando conocimiento del dominio de dicho problema, por lo que resulta oportuno adaptarlo al problema de segmentación de imágenes y, a partir de este, mejorar su desem-

peño siguiendo una metodología similar bajo el contexto del problema de la segmentación de imágenes.

La elección de los operadores de variación tiene un impacto importante en el desempeño de los GGAs (Ramos-Figueroa et al., 2023). Dicho desempeño varía de acuerdo con las características del problema, por lo que un algoritmo que ofrezca buenos resultados para un determinado problema, podría no tener la misma eficiencia para un problema de agrupación distinto. De forma que es importante considerar las características del problema para el diseño de los operadores de variación. Para fines de este trabajo, el interés radica en el operador de mutación que utiliza la versión del GGA-CGT adaptada al problema de la segmentación de imágenes, que lleva por nombre GGA-CGT-RGB-IS, derivado de la segmentación de imágenes en RGB (RGB-IS, por sus siglas en inglés).

Una forma de observar el efecto de los operadores de mutación en el desempeño del algoritmo puede ser mediante un análisis experimental; es decir, implementar los operadores disponibles y observar el comportamiento del algoritmo para cada uno y así elegir el más adecuado. Las características del mejor operador para este problema deberán ser analizadas cuidadosamente para diseñar un nuevo operador de mutación que favorezca la búsqueda del GGA-CGT-RGB-IS y que no converja en óptimos locales.

La literatura muestra ejemplos de como la aplicación de una metodología para el diseño de operadores de variación ha mejorado exitosamente los GGAs propuestos como en los trabajos de Quiroz-Castellanos et al. (2015), Amador-Larrea et al. (2022) y Ramos-Figueroa et al. (2023), por lo que resulta oportuno seguir una metodología similar para un problema distinto y con un enfoque en el operador de mutación.

## 1.3. Objetivos

De acuerdo con lo observado en la revisión de la literatura y debido a la naturaleza de los problemas de clustering, los cuales pueden verse como problemas de agrupación, la implementación de GGAs resulta una de las estrategias más adecuadas para resolver este tipo de problemas, en lugar de emplear un GA tradicional. La literatura especializada cuenta con GGAs que usan distintos operadores de mutación y tienen resultados muy

buenos. Sin embargo, cuando se diseña un algoritmo, la implementación del operador de mutación se debe adaptar a las características y propiedades del problema, por lo que es importante estudiar el comportamiento del algoritmo con distintos operadores de mutación para el problema de segmentación de imágenes en RGB y elegir el más adecuado.

El GGA-CGT propuesto por Quiroz-Castellanos et al. (2015) es uno de los algoritmos más competitivos en el estado del arte para resolver problemas relacionados con la agrupación de elementos. Ante esto, se propone adaptar el GGA-CGT al problema de segmentación de imágenes con sus operadores originalmente utilizados y realizar un estudio del comportamiento de dicho algoritmo con distintos operadores de mutación disponibles en la literatura, así como el diseño de un operador de mutación basado en las estrategias que brinden mejores resultados para este problema, poniendo énfasis en el efecto que el operador de mutación propuesto tiene en el desempeño del algoritmo al segmentar las imágenes de los bancos de pruebas.

A continuación se presentan los objetivos que fueron planteados para este trabajo de investigación.

### 1.3.1. Objetivo general

Diseñar un operador de mutación para el GGA-CGT adaptado para el problema de segmentación de imágenes en RGB, para mejorar el desempeño con respecto a la función de aptitud y el número de generaciones del algoritmo.

### 1.3.2. Objetivos específicos

Para conseguir el objetivo general planteado para este trabajo, se propusieron los siguientes objetivos específicos:

1. Adaptar el GGA-CGT para el problema de segmentación de imágenes en RGB.
2. Analizar el comportamiento del GGA-CGT para segmentación de imágenes con sus operadores y parámetros utilizados por defecto.

3. Implementar los operadores de mutación del estado del arte: Swap, Insertion, Item elimination, Elimination, Creation y Merge and split.

4. Comparar el desempeño del algoritmo con los distintos operadores implementados.

5. Identificar las características de cada operador implementado que permitan obtener mejores resultados.

6. Diseñar un operador de mutación con base en el conocimiento adquirido durante la experimentación.

7. Validar la mejora del algoritmo mediante una prueba estadística no paramétrica con los resultados de la segmentación de imágenes con ambas versiones del algoritmo.

## 1.4. Hipótesis

La hipótesis planteada para este trabajo es la siguiente:

**H1:** Es posible mejorar el desempeño del GGA-CGT adaptado al problema de segmentación de imágenes en RGB con respecto a la función de aptitud y al número de generaciones, mediante el diseño de un operador de mutación adecuado para el problema planteado.

## 1.5. Alcance y limitaciones

El alcance de este trabajo está enfocado en el diseño de un operador de mutación para el GGA-CGT propuesto por Quiroz-Castellanos et al. (2015) adaptado al problema de la segmentación de imágenes en RGB. Los demás elementos que conforman el algoritmo, así como la estrategia de cruza, permanecerán tal y como están en su diseño original, conservando sus parámetros originales, excepto por los porcentajes de cruza y mutación, los cuales serán calibrados.

## 1.6. Organización del documento

El presente documento se encuentra organizado de la siguiente manera: en el Capítulo 2 se describen algunos conceptos sobre optimización combinatoria y los elementos esenciales de un problema de optimización. Se introduce el espacio de color RGB, así como la descripción del problema de segmentación de imágenes y las principales funciones de evaluación que se pueden emplear para evaluar la calidad de las soluciones en un algoritmo genético. También se introduce la definición del problema con el que se trabaja en este proyecto. En el Capítulo 3 se habla del estado del arte en segmentación de imágenes con un enfoque en los algoritmos bioinspirados y se mencionan las funciones de aptitud que emplean. También se introducen los conjuntos de imágenes que se utilizaron para evaluar el desempeño del algoritmo y se describe el GGA-CGT, que es el algoritmo de referencia del cual parte el estudio. En el Capítulo 4 se detalla la adaptación del GGA-CGT al problema de la segmentación de imágenes en RGB y algunos experimentos con esta primera versión segmentando las imágenes de prueba de BSDS500. En el Capítulo 5 se describen los operadores de mutación del estado del arte para la representación basada en grupos y en el Capítulo 6 se aplica una metodología para evaluar el desempeño de estos operadores de mutación de acuerdo con los resultados de la segmentación del banco de imágenes de control. Con base en los resultados se diseña un nuevo operador para usarlo en el GGA-CGT-RGB-IS, generando así una nueva versión del algoritmo la cual es presentada en el Capítulo 7, donde se realiza una configuración de algunos parámetros y se segmentan los dos bancos de imágenes mencionados anteriormente para comparar su desempeño con la versión inicial. En ese mismo capítulo se analiza el comportamiento algorítmico de ambas versiones. Finalmente, las conclusiones y trabajo futuro son presentadas en el Capítulo 8.

# Capítulo 2

# Marco teórico

En este capítulo se abordan los problemas de optimización combinatoria, así como sus componentes principales. También se muestra la teoría acerca del espacio de color RGB, el cual es utilizado en este proyecto. Asimismo, se aborda el problema de la segmentación de imágenes en dicho espacio de color con un enfoque de agrupación. También se describen algunas de las funciones de evaluación en la literatura que pueden ser utilizadas como objetivos para resolver este problema y después se define el problema de la segmentación de imágenes como un problema de optimización donde se busca minimizar una de estas funciones de evaluación.

La optimización combinatoria se deriva de las matemáticas discretas. Dentro de esta rama, en un problema discreto de optimización se busca maximizar (o minimizar) una función objetivo en los números reales en un conjunto finito de soluciones factibles $S$ (Lee, 2004). Es común encontrar los términos enumeración, combinación y permutación. Así, en un problema de optimización combinatoria, la solución consiste en una combinación de componentes únicos seleccionados de un conjunto generalmente finito. El objetivo es encontrar la combinación óptima de componentes (Luke, 2009).

Dentro de los problemas de optimización se encuentran los problemas de agrupación, en los cuales se trata de particionar un conjunto de elementos en un número específico de grupos, donde el objetivo es obtener la mejor distribución posible de estos elementos en los grupos. En los problemas de agrupación, cada grupo no debe compartir elementos en común con el resto de los grupos y generalmente se tienen otras restricciones dependientes

del problema. De esta forma, la solución a este tipo de problemas consiste en encontrar una partición eficiente de un conjunto $V$ con $n$ elementos, en una colección de $D$ grupos disjuntos $G_i$, de forma que $V = \bigcup_{i=1}^{D} G_i$ y $G_i \bigcap G_j = \emptyset$, con $i \neq j$ (Ramos-Figueroa, 2022).

Tal y como señalan los autores en Mutingi and Mbohwa (2017), la optimización de problemas de agrupación se definen por la combinación de los elementos en los grupos; estos problemas pueden ser resueltos efectivamente con algoritmos computacionales dada su estructura de grupos; su naturaleza es altamente combinatoria, son NP-duros y son computacionalmente costosos, además, tienen muchas restricciones, lo cual añade complejidad.

## 2.1. Problemas de optimización combinatoria

De acuerdo con (Blum and Roli, 2003), un problema de optimización combinatoria $P = (S, f)$ puede ser definido por:

- Un conjunto de variables $X = \{x_1, \ldots, x_n\}$.
- Variables de dominio $D_1, \ldots, D_n$.
- Restricciones entre las variables.
- Una función objetivo $f$ a minimizar, donde $f : D_1 \times \cdots \times D_n \to \mathbb{R}^+$.

El conjunto de posibles asignaciones factibles es:

$$S = \{s = \{(x_1, v_1), \ldots, (x_n, v_n)\} \mid v_i \in D_i,\ s \text{ satisface todas las restricciones}\}$$

En todo problema de optimización, siempre se deben especificar estos tres componentes esenciales: la representación, el objetivo y la función de evaluación.

### 2.1.1. Representación

Para un problema de optimización combinatoria, pueden existir diversas formas de representar una solución. La representación juega un papel importante en la definición del vecindario de búsqueda, pues este último dependerá del esquema utilizado para la codificación de las soluciones y no del problema en sí. Es por eso que es importante elegir una representación adecuada para el problema de optimización.

De acuerdo con Blum and Roli (2003), la estructura del vecindario es una función $N : S \to 2^S$ que asigna un conjunto de soluciones vecinas $N(s) \subseteq S$ a cada $s \in S$.

En problemas continuos, un vecindario puede ser explorado haciendo cambios en el valor continuo, entonces existe una noción de dirección, pero en problemas combinatorios, esto no existe y el tamaño del vecindario depende del tipo de representación que se utilice, lo que dificulta la exploración del paisaje de soluciones (Yu and Gen, 2010).

Esto último está relacionado con la complejidad de estos problemas. En una primera instancia, resultaría sencillo intentar enumerar todas las posibles soluciones al problema y elegir la mejor; sin embargo, para problemas más grandes, la complejidad computacional crece demasiado y se vuelve imposible enumerar todas las posibles soluciones al problema.

En optimización combinatoria, la noción de cercanía en un vecindario se puede definir principalmente de dos formas (Michalewicz and Fogel, 2013):

- Estableciendo una función de distancia que define las soluciones como parte del vecindario si están dentro de un límite de distancia.
- Definiendo una función de mapeo en el espacio de búsqueda. En este caso, una solución puede ser modificada con alguna función de mapeo, por lo que un posible cambio en esta solución mediante el uso de dicha función, genera un vecino de esta.

### 2.1.2. Objetivo

El objetivo es una expresión matemática de la tarea que se busca alcanzar. De acuerdo con Michalewicz and Fogel (2013) no se trata de una función, sino de una expresión y permite identificar qué es lo que se está buscando en el problema.

### 2.1.3. Función de evaluación

La función de evaluación se trata de una función de mapeo del espacio de las posibles soluciones candidatas que se encuentran definidas bajo la representación utilizada, a un conjunto de números (por ejemplo, reales). De esta forma, cada solución del espacio de posibles soluciones tiene asignado un valor numérico que permite diferenciar la calidad entre las soluciones disponibles. Esta función permite discriminar soluciones y decidir cuál es mejor, en términos de dicha función.

De esta forma, haciendo uso de estos elementos, se puede definir un problema de búsqueda. Dado un espacio de búsqueda $S$, junto con su parte factible $F \subseteq S$, hallar $x \in F$, tal que $eval(x) \leq eval(y)$ para todo $y \in F$, donde $eval()$ hace referencia a la función de evaluación empleada. Esto considerando que se trata de un problema de minimización. Como se puede observar, todo lo que se toma en cuenta para este problema está relacionado con la función de evaluación elegida, más no del objetivo, es aquí donde radica la importancia de la elección de una buena función de evaluación que refleje el objetivo del problema (Michalewicz and Fogel, 2013).

## 2.2. Espacio de color RGB

De acuerdo con Gonzalez (2009), los espacios de color especifican un sistema coordenado, de forma que dentro de este modelo un color puede ser representado por un solo punto. Así, su objetivo es facilitar la especificación de los colores de una forma estándar. El mismo autor menciona que la mayoría de modelos de color están orientados a su uso en hardware, como monitores e impresoras, así como aplicaciones que tienen que ver con la manipulación de colores. El espacio de color RGB suele ser el más utilizado para tareas de procesamiento digital de imágenes, así como para dispositivos de hardware como monitores y cámaras de video a color.

El espacio de color RGB hace referencia a los tres colores primarios: rojo (R), verde (G) y azul (B). De ahí el nombre RGB, por sus siglas en inglés. Al mezclar estos colores primarios, se pueden obtener una gran variedad de diferentes colores. Desde el punto de vista del espacio de color, esta combinación se da con las intensidades de cada uno de

estos tres colores primarios en el sistema coordenado. Dicho sistema está formado por tres ejes que corresponden a cada color primario (Stone, 2016). Usualmente a cada eje se le llama canal. En la Figura 2.1 se muestra el espacio de color RGB con coordenadas entre 0 y 1 para cada eje. Aquí se pueden observar los colores rojo, verde y azul en algunas de las esquinas. Los colores secundarios cían, magenta y amarillo aparecen en las demás esquinas. En el origen del plano coordenado se encuentra el color negro, mientras que en las coordenadas $[1, 1, 1]$ se encuentra el color blanco. En este espacio están contenidos también los colores en escala de grises, los cuales se consiguen con una intensidad igual en los tres colores y van desde el negro hasta el blanco.

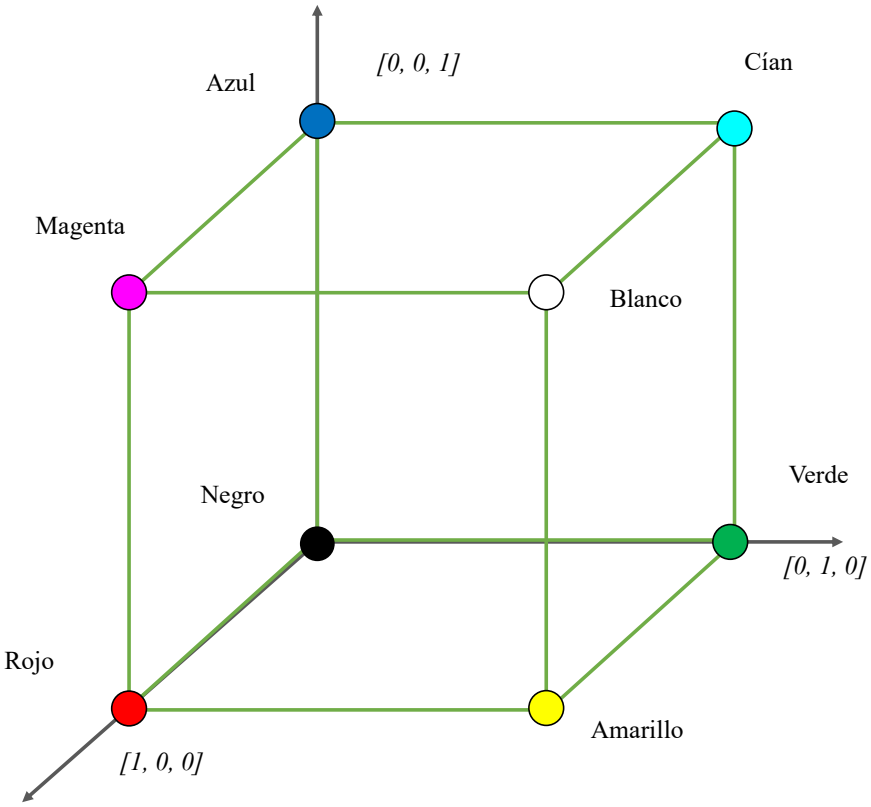



Figura 2.1: Espacio de color RGB.

En el caso de imágenes digitales, cada canal puede dividir su rango en $[0, L-1]$, donde $L = 2^b$ y $b$ corresponde al número de bits. De forma que una imagen con color completo tiene 8 bits, por lo que su rango está entre $[0, 2^8 - 1] = [0, 255]$ para cada canal de color. De esta forma se pueden utilizar valores de intensidad en este rango para cada canal, de

forma que el color blanco estaría ubicado en la coordenada $[255, 255, 255]$. Con este rango, una imagen RGB de 24 bits (8 por canal) es posible representar $(2^8)^3 = 16,777,216$ colores diferentes (Gonzalez, 2009).

## 2.3. Segmentación de imágenes en RGB

La segmentación de imágenes puede verse como una técnica de procesamiento de imágenes empleada para dividir una imagen en múltiples regiones. También puede verse como un proceso para delimitar objetos semánticamente en una imagen (Ghosh et al., 2019). Desde un punto de vista más técnico, la segmentación de imágenes puede formularse como un problema de clasificación de píxeles, de forma que los píxeles que comparten la misma etiqueta están conectados de alguna forma, ya sea con base en algún aspecto visual de bajo nivel (como el color) o de más alto nivel (semántico) (Minaee et al., 2021).

La segmentación semántica se refiere a determinar la naturaleza de las diferentes partes de una imagen. Busca asociar cada píxel de la imagen con una etiqueta de la clase a la que pertenece (Sultana et al., 2020). Esta etiqueta tiene un sentido semántico que va más allá de las características del mismo píxel. Es decir, las regiones en las que se segmenta esta imagen corresponden a objetos, de forma que en una imagen, se pueden delimitar y agrupar los píxeles que representen un avión, el fondo de la imagen, etc. Para este tipo de segmentación se suelen emplear técnicas más avanzadas como las redes neuronales convolucionales.

Un problema relacionado, pero diferente es la segmentación de imágenes de bajo nivel, el cual es el que se abordar en este trabajo. Dicha segmentación consiste en una partición no supervisada de la imagen en regiones coherentes con respecto a señales de bajo nivel, como el color, la textura o la profundidad (Csurka et al., 2021).

El objetivo de la segmentación es simplificar la representación de una imagen en algo que sea más significativo y fácil de analizar (Halder et al., 2011). La segmentación de imágenes consiste en particionar una imagen en múltiples segmentos de acuerdo con la información extraída de los píxeles. Cabe destacar que, a medida que el número de segmentos aumenta, la dificultad para realizar la tarea también aumenta. De acuerdo con Nakane

et al. (2020), los principales métodos empleados para la segmentación de imágenes son el de umbralización y el de clustering. El método de umbralización divide el histograma de las intensidades de los píxeles usando funciones de membresía, que representan el grado de pertenencia, o el umbral más adecuado para la imagen, en el caso del algoritmo de Otsu. La búsqueda de estos parámetros hace atractivo el uso de GAs. Por otro lado, en segmentación de imágenes, el clustering se basa en centroides de grupos a los que pertenecen los píxeles, donde se busca minimizar la distancia entre los píxeles pertenecientes a un determinado grupo y su centroide (Nakane et al., 2020). El clustering es una tarea que tiene el objetivo de hallar un conjunto finito de categorías (clusters). Se centra principalmente en conseguir que los elementos dentro de cada grupo sean lo más parecidos posible y, al mismo tiempo, que los grupos sean muy diferentes entre sí (Hruschka et al., 2009). El clustering se divide en jerárquico y particional. El jerárquico tiene una estructura de árbol con una secuencia de clustering en la que cada cluster es una partición del conjunto de datos. Mientras que el particional divide los grupos en un número específico de clusters, minimizando algún criterio. Este último puede verse como un problema de optimización.

Un ejemplo muy popular de clustering particional es el algoritmo de K-Means. Este algoritmo empieza con $K$ centroides, elegidos de forma aleatoria (o utilizando alguna información a priori). Una vez que se tienen los centroides, el resto de los píxeles se agrupan tomando como referencia el centroide al que estén más cerca y el centroide se vuelve a calcular cuando los elementos han sido asignados. Este proceso se repite hasta que exista una convergencia del algoritmo (Omran et al., 2005). En la Figura 2.2 se ilustran los dos métodos principales de segmentación de imágenes.

Se han propuesto algoritmos para segmentar imágenes en escala de grises o con diferentes componentes, como en el caso de las imágenes satelitales. También es posible emplear imágenes en el espacio de color RGB (rojo, verde y azul). Con respecto a la segmentación en RGB, para una imagen $I$ de $n$ píxeles, a esta se le asocia un conjunto $V = \{1, \ldots, n\}$ de $n$ tuplas de la forma $(r_i, g_i, b_i)$, donde cada tupla corresponde a los valores de intensidad en el espacio RGB para el píxel $i$ de la imagen.

El problema de segmentación de imágenes se puede definir de la siguiente forma: Sea $k$ un número entero positivo mayor que 1, se busca particionar el conjunto $N$ de $n$ píxeles de

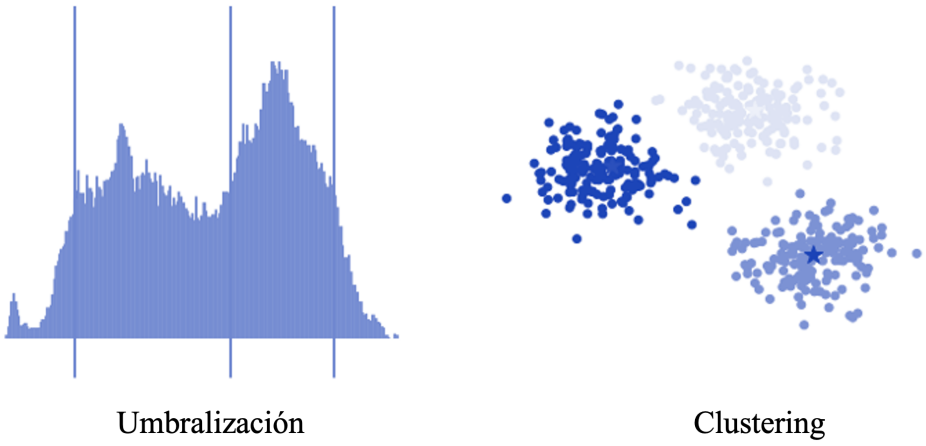



Figura 2.2: Principales métodos de segmentación de imágenes.

la imagen $I$ en $k$ segmentos de tal manera que los píxeles que estén en un mismo segmento tengan la mayor similitud posible.

Tomando como ejemplo la imagen de la Figura 2.3, la cual tiene $n = 9$ píxeles, por lo que se tiene un conjunto $V = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$. Dadas las características de la imagen, el píxel 5, que es el del centro, tiene valores de intensidad distintos a los de los demás. Suponiendo que el número de segmentos es $k = 2$, se busca asignar los $n$ píxeles de la imagen en 2 segmentos, donde los píxeles que estén dentro de un segmento deberían de tener una mayor similitud que la que comparten con los píxeles asignados en el otro segmento. Con este ejemplo, es fácil observar que lo que se esperaría es que el píxel 5 quede en un segmento, mientras que el resto de los píxeles deberían quedar en otro segmento; por lo tanto, la segmentación esperada podría representarse de la siguiente forma: $S = \{[5], [1, 2, 3, 4, 6, 7, 8, 9]\}$. Donde $S$ representa la segmentación de los 9 píxeles en dos grupos. Se tiene un conjunto de tamaño $k = 2$, cuyos elementos son listas que representan los segmentos o grupos. Estas listas contienen los índices de los píxeles asignados en cada grupo.

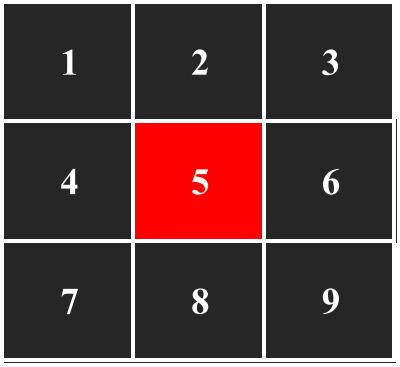



Figura 2.3: Ejemplo de imagen RGB.

### 2.3.1. Funciones de evaluación

Como se mencionó anteriormente, uno de los componentes esenciales de un problema de optimización combinatoria es la función de evaluación. Debido a que la segmentación de imágenes puede verse como un problema de clustering, las mismas funciones de evaluación que existen para el clustering pueden ser empleadas para resovler este problema. En ese sentido, se describen brevemente tres de estas funciones que pueden ser aplicadas para resolver la segmentación de imágenes como un problema de optimización combinatoria.

#### Distancia intra-cluster

Una forma muy sencilla de evaluar el clustering realizado por un algoritmo es mediante la distancia intra-cluster. Esta es la suma de las distancias euclidianas entre cada punto y su centroide, calculada como se muestra en la Ecuación 2.1 (Turi, 2001):

$$intra(S) = \sum_{G_k \in S} \sum_{x \in G_k} d(x, \mu_k). \tag{2.1}$$

Donde $d(x, \mu_k)$ hace referencia a la distancia euclidiana entre cada elemento $x$ del cluster $G_k$ y $\mu_k$, que corresponde al centroide del mismo cluster. En términos de la segmentación de imágenes, cada elemento corresponde a un píxel y los clusters son los segmentos en los que se particiona el conjunto de píxeles de la imagen. De esta forma, $S$ es una solución conformada por cada uno de los $G_k$ grupos.

La distancia intra-cluster representa qué tan compacto se encuentra un grupo o seg-

mento, de forma que entre mayor sea la distancia intra-cluster, menor similitud habrá entre los píxeles en dicho segmento. Por el contrario, si esta distancia es menor, es un indicador de que los píxeles asignados al segmento en cuestión comparten un mayor grado de similitud. Por lo tanto, lo deseable es que la distancia intra-cluster tenga el menor valor posible (idealmente de cero).

**Distancia inter-cluster**

Otra forma de evaluar la calidad de la segmentación de una imagen es calculando la distancia inter-cluster. Esta es una medida complementaria a la de la distancia intra-cluster donde lo que se busca es identificar qué tan separados entre sí están los segmentos (Turi, 2001). Existen diferentes formas de medir esta distancia en la literatura, algunos criterios, como mencionan Aljarah et al. (2021), son con enlace único, completo, promedio o por centroide. En el primer criterio, la distancia inter-cluster se obtiene calculando la distancia mínima entre pares de elementos contenidos en dos distintos grupos, y así para cada par de grupos que se puedan formar como se muestra en la Ecuación 2.2.

$$d(G_i, G_j) = Min(|x_k - x_l|), \quad x_k \in G_i, x_l \in G_j. \tag{2.2}$$

Donde $G_i$ y $G_j$ corresponden a un par de segmentos de la solución, con $i \neq j$. $x_k$ y $x_l$ corresponden a los píxeles que pertenecen a los segmentos $G_i$ y $G_j$, respectivamente.

Contrario al primer criterio, la distancia inter-cluster de enlace completo se obtiene tomando la máxima distancia entre un par de elementos de dos grupos distintos. El procedimiento es, por lo tanto, similar al del primer criterio, solo que en este caso se toma la distancia máxima, como se muestra en la Ecuación 2.3.

$$d(G_i, G_j) = Max(|x_k - x_l|), \quad x_k \in G_i, x_l \in G_j. \tag{2.3}$$

El criterio del promedio calcula todas las distancias posibles entre pares de elementos de dos grupos diferentes y dividiéndola entre el número de pares posibles que se pueden formar. Esto se hace como en la Ecuación 2.4.

$$d(G_i, G_j) = \frac{1}{n_i n_j} \sum_{x_k \in G_i, x_l \in G_j} |x_k - x_l|. \tag{2.4}$$

Donde $n_i$ y $n_j$ corresponden a la cantidad de píxeles que hay en los segmentos $G_i$ y $G_j$, respectivamente.

Con el último enfoque se calcula la distancia inter-cluster mediante los centroides de cada grupo. De forma que solo se calculan las distancias entre centroides para cada par de grupos, como en la Ecuación 2.5.

$$inter(S) = \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} (\mu_i - \mu_j)^2. \tag{2.5}$$

Donde $\mu_i$ y $\mu_j$ corresponden al centroide $i$ y $j$ en la solución $S$.

Sea cual sea el criterio que se utilice, la distancia inter-cluster nos indica qué tan delimitados están los segmentos de la solución. Una distancia muy alta es deseable, pues es una señal de que los segmentos se encuentran bien separados entre sí. Por el contrario, si la distancia es baja, es una señal de que la solución tiene una mala calidad.

### Promedio de Silhouette

Pese a que las distancias intra e inter-cluster son buenas medidas de la calidad del clustering, a veces se requiere un enfoque diferente cuando no se tienen las etiquetas verdaderas de las clases a las que debería pertenecer cada elemento. En este caso, cuando no sabemos a qué segmento debería pertenecer cada píxel. Dada esta situación, el Coeficiente de Silhouette fue propuesto por Rousseeuw (1987). Esta es una forma de medir qué tan bien se realizó la agrupación mediante clustering. El Coeficiente de Silhouette para una muestra $x$ se define como se muestra en la siguiente ecuación:

$$s(x) = \frac{b(x) - a(x)}{max\{a(x), b(x)\}}. \tag{2.6}$$

Sujeto a:

$$-1 \leq s(x) \leq 1. \tag{2.7}$$

Donde $a(x)$ es la disimilitud promedio de $x$ con todos los demás objetos del mismo cluster. Esto es, la distancia promedio que hay de $x$ a los demás puntos que pertenecen al mismo grupo. De manera similar, para definir $b(x)$ se necesita la disimilitud promedio de $x$ con todos los objetos de los demás clusters y al final se toma la distancia promedio mínima de todas. Para esta métrica, un valor de 1 indica una solución con una calidad alta. Mientras que un valor de -1 indica que la muestra $x$ fue asignada al cluster incorrecto. En términos de la segmentación de imágenes, la muestra $x$ correspondería a un píxel de la imagen y los clusters son los segmentos.

Esto se hace para cada píxel de la imagen, por lo que se puede obtener el promedio de Silhouette a partir de estos coeficientes como se calculan con la Ecuación 2.6. Por lo que el promedio de Silhouette de una solución $S$ con $n$ píxeles es el siguiente:

$$\bar{s}(S) = \frac{1}{n}\sum_{x=1}^{n} s(x). \tag{2.8}$$

### 2.3.2. Definición del problema

Desde el punto de vista de la optimización, la segmentación de imágenes se puede definir como un problema de minimización tomando como función de evaluación $F(S)$ la medida su distancia intra-cluster, definida en la Ecuación 2.1, donde el objetivo consiste en minimizar $F(S)$. Donde $S$ corresponde a una solución o la segmentación de una imagen con un conjunto $N$ de $n$ píxeles en $k$ grupos. Así, el problema de segmentación de imágenes en RGB se puede formular como:

$$\textit{minimizar } F(S). \tag{2.9}$$

Sujeto a:

$$\sum_{i=1}^{k} x_{ij} = 1 \quad \forall j \in N. \tag{2.10}$$

$$\sum_{i=1}^{k}\sum_{j=1}^{|G_i|} x_{ij} = n. \tag{2.11}$$

Donde $x_{ij}$ tomará el valor de 1 si el píxel $j$ se encuentra en el grupo $i$, en otro caso tomará el valor de 0, y $|G_i|$ corresponde a la cantidad de píxeles que están en el grupo $G_i \in S$. De esta forma, para la segmentación de imágenes se busca minimizar la distancia intra-cluster con un número de segmentos $k$ fijo, con la restricción de que un píxel puede pertenecer solo a un grupo y que no puede haber píxeles sin asignar.

## 2.4. Conclusiones

Como conclusiones de este capítulo se puede mencionar que los problemas de agrupación son problemas combinatorios difíciles de resolver, cuya complejidad aumenta con las restricciones que estos tienen. Además, el vecindario depende de la representación que se utilice, lo cual puede hacer más o menos compleja la búsqueda. Y se observa que todo problema de optimización comparte elementos básicos que permiten definirlos y entender qué es lo que se trata de minimizar (o maximizar).

El problema de la segmentación de imágenes puede ser tratado como un problema de clustering con un número de segmentos fijos para el cual se busca minimizar alguna de las funciones de evaluación presentadas en este capítulo y tomando como criterio la intensidad de color en el espacio RGB.

# Capítulo 3

# Marco referencial

En este capítulo se muestra un panorama del estado del arte con respecto a la segmentación de imágenes clásica empleando algoritmos bioinspirados, haciendo énfasis en los algoritmos genéticos. Además, se introducen los conjuntos de imágenes utilizados para probar el algoritmo propuesto en este trabajo. También se describe el GGA-CGT, el cual es el algoritmo que se tiene como antecedente en este proyecto y es utilizado para realizar la adaptación al problema definido en el capítulo anterior.

Como se pudo observar en la sección de segmentación de imágenes del capítulo anterior, la segmentación de imágenes comprende tanto la segmentación semántica como la clásica, donde la diferencia radica en las características que se toman como referencia para realizar dicha tarea. Se habló de la segmentación semántica, que se considera un problema de segmentación de imágenes de alto nivel, mientras que el problema abordado en este trabajo es el de la segmentación de bajo nivel. Como mencionan Csurka et al. (2021), la segmentación clásica basada en características de bajo nivel suele ser abordada como una tarea no supervisada. Particularmente en este trabajo, el problema de la segmentación de imágenes puede ser visto como un problema de clustering, para el cual resulta interesante el uso de los GGAs, ya que este es también un problema de agrupación. Por esta razón, el uso de técnicas de aprendizaje profundo no se consideró en este trabajo. Sin embargo, debido a que se trata de una área muy activa con mucho éxito en la segmentación semántica, así como otras tareas del área de visión por computadora, vale la pena mencionar lo que se ha logrado en el campo de la segmentación de imágenes.

El desarrollo de algoritmos de aprendizaje profundo no solo tuvo un impacto en las típicas tareas de clasificación de objetos, sino que también tuvo un aporte significativo en otras tareas de visión por computadora relacionadas como la detección de objetos, localización, seguimiento y la segmentación semántica de imágenes. Una de las técnicas más utilizadas en la visión por computadora fue introducida con el nombre de redes neuronales convolucionales (CNN). A partir de aquí, varias técnicas tomaron inspiración de redes populares como AlexNet, autoencoders convolucionales, redes neuronales recurrentes (RNNs), redes residuales, entre otras (Ghosh et al., 2019).

En una revisión de literatura, Minaee et al. (2021) destacan que, entre las arquitecturas de aprendizaje profundo más prominentes utilizadas por la comunidad de visión por computadora, se encuentran las CNNs, RNNs, memoria de largo y corto término (LSTM), codificadores-decodificadores, redes adversarias generativas (GANs), entre otras. En un trabajo del mismo año, Csurka et al. (2021) revisaron algunas técnicas de segmentación semántica de imágenes adaptadas específicamente al dominio del problema que buscaban resolver. Entre sus aportes, realizaron una clasificación de estas técnicas y mencionaron la arquitectura base que emplean estas técnicas. Entre las arquitecturas en las que están basadas las técnicas de segmentación revisadas por los autores resulta interesante observar que la mayoría tienen como base las FCNs. También se emplearon arquitecturas como DLab, DeepLab, U-Net, entre otras.

Uno de los problemas más importantes que se tienen con el uso de técnicas de aprendizaje profundo tiene que ver con el tiempo de entrenamiento y la cantidad de datos que se requieren para dicho entrenamiento. Sin embargo, en los últimos años, gracias a la transferencia de aprendizaje, se ha facilitado mucho el entrenamiento de estos modelos, por lo que ya no es necesario emplear mucho tiempo ni recursos computacionales para esta tarea. En la transferencia de aprendizaje, un modelo primero es pre-entrenado con un conjunto de datos. Este modelo puede ser utilizado para algún problema en específico haciendo un re-entrenamiento con el conjunto de datos de dicho problema, pero ya no es necesario entrenarlo desde cero. En el caso de la segmentación semántica de imágenes, muchos de los modelos propuestos no son entrenados desde cero, sino que parten de modelos pre-entrenados con conjuntos de imágenes, sobre todo con el conjunto llamado ImageNet

(Minaee et al., 2021).

Como se mencionó anteriormente, la segmentación de bajo nivel es un problema diferente al de la segmentación semántica y suele verse como un problema de naturaleza no supervisada. Mientras que para la segmentación semántica suelen emplearse técnicas de aprendizaje profundo, para la segmentación de bajo nivel se utilizan técnicas más clásicas. Es aquí donde los algoritmos evolutivos y otros algoritmos bioinspirados suelen ser utilizados.

De acuerdo con Nakane et al. (2020), la mayor parte de trabajos que emplean algoritmos bioinspirados como GAs, optimización con enjambre de partículas (PSO, por sus siglas en inglés) y algoritmos de colonia de hormigas (ACOs, por sus siglas en inglés) para la segmentación de imágenes, suelen enfocarse en la segmentación clásica empleando los principales métodos de segmentación (ya sea mediante umbralización o clustering).

## 3.1. Algoritmos bioinspirados del estado del arte para segmentación de imágenes

Para la búsqueda de literatura, se partió del survey de Nakane et al. (2020), donde se mencionan las aplicaciones de GAs para tareas de segmentación de imágenes. Para buscar aplicaciones de GGAs se decidió buscar en Google Scholar usando como palabras clave *grouping genetic algorithm, segmentation* y *clustering*. También se realizaron búsquedas en bases de datos especializadas como las del IEEE y el ACM con las mismas palabras clave. Al no encontrar suficientes resultados, la búsqueda se dividió en dos partes: en la primera, el interés está en el uso de GAs para la segmentación de imágenes, mientras que en la segunda parte, la búsqueda tenía como prioridad encontrar aplicaciones de GGAs para clustering. Los hallazgos se exponen a continuación.

Tao et al. (2003) proponen un algoritmo para la segmentación de imágenes en escala de grises basado en probabilidades, partición difusa y entropía. Aquí se usa un GA para determinar los parámetros más adecuados para las funciones de membresía, maximizando la entropía. De manera similar, Maulik and Bandyopadhyay (2003) proponen un algoritmo

difuso que emplea un GA para obtener los parámetros óptimos para las funciones de membresía, pero que como función de evaluación toma el índice Xie-Beni, el cual se debe minimizar.

También hay propuestas de otros paradigmas, como la propuesta de un algoritmo de PSO de Omran et al. (2005) que forma parte del paradigma de la Inteligencia Colectiva. Dicho algoritmo encuentra los centroides para un número de clusters determinado, donde se busca minimizar la distancia intra-cluster, maximizar la distancia inter-cluster y minimizar el error de cuantización en una sola función de evaluación.

Algunas propuestas también combinan distintos métodos como proponen Awad et al. (2009), así como el uso de redes neuronales artificiales propuesto por Awad et al. (2007), donde se busca minimizar la distancia intra-clúster. Así como algoritmos que toman más información acerca del problema, como el propuesto por Halder et al. (2011), que también busca minimizar la distancia intra-clúster. Este algoritmo posteriormente fue aplicado con éxito para segmentación de imágenes médicas por Halder et al. (2016).

Si bien estos algoritmos hacen uso de GAs, lo que hacen en la mayoría de los casos es encontrar los parámetros adecuados para realizar la segmentación mediante umbralización. El GA en sí no está tomando los píxeles como objetos para posteriormente realizar la agrupación mediante un clustering particional, aunque también existen propuestas para resolver problemas de clustering con un número fijo de clústers como mencionan Hruschka et al. (2009). Sin embargo, intentar este método de clustering directamente con los GAs resultaría muy costoso. Falkenauer Falkenauer (1996) identificó que el inconveniente con los GAs para resolver problemas de agrupación tiene que ver con que están orientados a los objetos, en lugar de orientarse a los grupos. La representación usada por los GAs es muy redundante y hace que se pierda el objetivo de resolver un problema de agrupación. Es por eso que propuso los GGAs.

Dado que el diseño de los GGAs se enfoca en una representación basada en grupos, los operadores de variación trabajan con la parte del grupo, que es considerado el elemento más pequeño que puede haber. Con este enfoque se resuelve el inconveniente para trabajar con problemas de agrupación y se puede seguir con la idea que plantea el paradigma del GA, que es realizar una exploración en el espacio de búsqueda, identificando regiones

promisorias, así como una explotación de estas regiones.

Continuando con la primera propuesta de GGA de Falkenauer, la cruza busca promover la herencia de los grupos más promisorios. En el caso de la mutación, destaca el uso de tres estrategias generales que también trabajan a nivel de grupo: creación de nuevos grupos, eliminación de grupos existentes o la mezcla de una pequeña cantidad de elementos entre los grupos (Falkenauer, 1996). Mientras que el operador de cruza se considera como primario, la mutación se considera como un operador secundario. Aunque, junto con la selección y la codificación, Lim et al. (2017) coinciden en que su importancia es la misma tanto en GAs como en GGAs.

Gracias a este enfoque de grupos en los operadores del GGA y a su representación de soluciones tan afín a los problemas de agrupación, el uso de este algoritmo para resolver un problema como el de clustering parece ser una opción viable. En ese sentido, Hong et al. (2012) proponen un GGA para realizar la tarea de selección de características basado en clustering. Señalan nuevamente la dificultad de un GA para realizar esta tarea, aunque también indican que hay propuestas que usan GAs. En este algoritmo, el operador de cruza toma un cromosoma de base y después inserta grupos de otro cromosoma. Posteriormente, elimina los grupos que aparecen duplicados. La mutación trabaja a nivel de objetos, donde se reasignan objetos de manera aleatoria a otros grupos. También usa el operador de inversión, el cual tiene la función de ayudar al operador de cruza a seleccionar una combinación diferente de grupos en dicho proceso. En esta propuesta, los resultados del GGA fueron mejores que los del GA. Yusoh and Tang (2012) proponen el uso de un GGA para clustering, donde el problema a tratar intenta garantizar un uso óptimo de los recursos de cómputo en la nube para el esquema de *Software as a Service*. El operador de cruza que utiliza trabaja a nivel de grupos y lo que hace es tomar un punto de corte entre estos para combinarlos. La mutación trabaja a nivel de objetos, intercambiando estos objetos (máquinas virtuales, para este problema) entre los grupos disponibles. El algoritmo siempre produce soluciones factibles para cada instancia de prueba, aunque con un tiempo computacional alto. Un GGA para problemas de clustering es propuesto por Agustı et al. (2012), este algoritmo usa una versión modificada del operador de cruza propuesto en el GGA canónico de Falkenauer, para adaptarlo al problema de clustering. La mutación se

realiza en dos pasos: primero se separa un grupo, formando dos grupos distintos; después se juntan un par de grupos de forma aleatoria. El algoritmo emplea además, una búsqueda local y un método de paralelización para mejorar su desempeño. Los resultados que obtuvieron fueron mejores que algunos algoritmos diseñados para clustering, como K-Means, en cuanto a la similitud con respecto a los óptimos conocidos.

De acuerdo con la literatura, el uso de GGAs para realizar la tarea de clustering particional proporciona buenos resultados, además de que la tarea justifica el uso de un EA, pues se trata de un problema NP-duro. En ese sentido, el uso de un GGA para resolver el problema de segmentación de imágenes parece ser adecuado.

## 3.2. Conjuntos de imágenes

Un conjunto de imágenes a color es necesario para poder probar los algoritmos de segmentación de imágenes. Aunque hoy en día es muy común poder obtener imágenes en RGB, en la literatura existen diversas propuestas, desde imágenes individuales, hasta conjuntos de imágenes de pruebas. En el presente proyecto se trabajó con un conjunto de imágenes de la literatura y otro conjunto de datos sintéticos generados con Python. Estos conjuntos de imágenes fueron utilizados para evaluar el desempeño del algoritmo realizando la tarea de la segmentación.

### 3.2.1. Imágenes de prueba de BSDS500

La base de datos BSDS500, propuesta por Arbelaez et al. (2010), es una extensión de la base de datos BSDS300. Esta última fue propuesta por Martin et al. (2001) y contiene 300 imágenes a color con dimensiones de $481 \times 321$ píxeles. La finalidad de esta base de datos es proveer recursos para la evaluación del desempeño de algoritmos de segmentación y detección de contornos. Para dicho fin, las imágenes fueron segmentadas manualmente por 30 personas. BSDS500 extiende la colección de imágenes agregando 200 imágenes más. De esta forma, las 300 imágenes de la anterior base de datos se usan para entrenamiento y las 200 imágenes nuevas se emplean para probar los algoritmos de segmentación o detección de contornos. En BSDS500 las imágenes fueron segmentadoas por 5 personas en promedio. De

esta forma, la base de datos incluye la imagen en formato *jpg* y un archivo *mat* que contiene las etiquetas de los píxeles de cada imagen, de acuerdo con la segmentación realizada por cada persona.

Figura 3.1: Ejemplos de imágenes de la base de datos BSDS500.

### 3.2.2. Imágenes de control

En este trabajo, se propone un banco de imágenes de control para realizar pruebas con algoritmos de segmentación que utilicen como función de aptitud la distancia intra-cluster. Dicha base de datos consiste en 100 imágenes en RGB de $50 \times 50$ píxeles. Estas imágenes están compuestas de cuadros con un color en específico que puede repetirse en cualquier otra parte de la imagen. Para generar estas imágenes, lo primero es elegir una cantidad $k$ de colores distintos en el espacio de color RGB generando tres valores (uno para cada canal) aleatorios uniformemente en el rango de 0 a 255, que corresponde con el rango de intensidades para este espacio de color. Después, se genera una matriz con estos colores, asignándolos de forma aleatoria y es así como se crea una imagen con un número $k$ de colores distintos. El rango de grupos para estas imágenes es de 5 a 54, con dos imágenes por cada valor de $k$, dando como resultado 100 imágenes. Lo que se espera es que, al aplicar la segmentación a estas imágenes, el algoritmo sea capaz de agrupar todos los píxeles de la imagen en $k$ grupos: uno para cada color. Al conseguir este resultado, la suma de las

distancias intra-cluster de cada grupo en la segmentación debería ser de 0. En algoritmos genéticos, estas imágenes pueden servir para observar qué tan rápido, si es que se consigue, el algoritmo converge en la mejor agrupación posible.

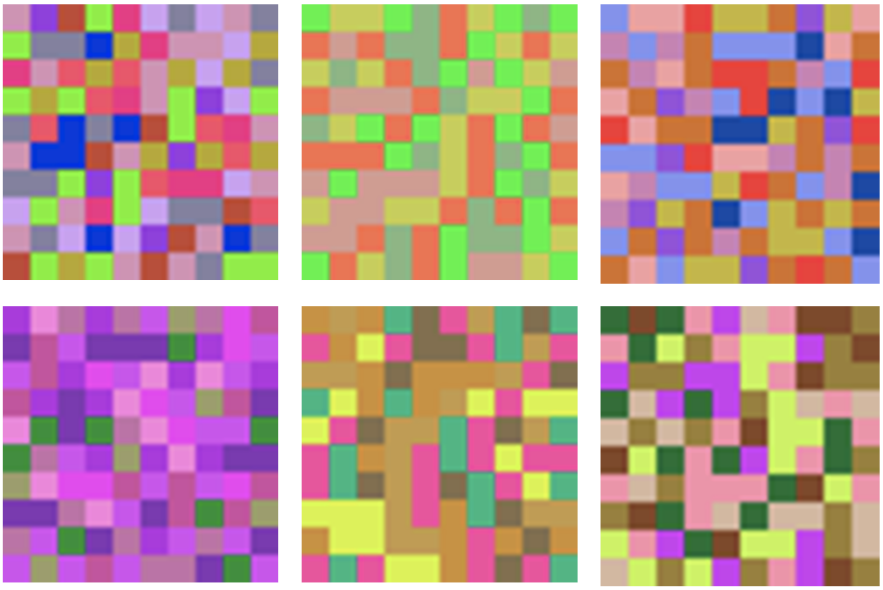


Figura 3.2: Ejemplos de imágenes de control.

## 3.3. GGA-CGT

Los GAs fueron propuestos por Holland en la década de los sesenta (Holland, 1992), sus elementos son cinco: representación de individuos, selección de padres, recombinación o cruza, mutación y reemplazo. Donde la mutación, en un principio no era tan importante, pero después fue tomando relevancia para explorar más soluciones. Desde esa primera versión, se han propuesto distintas variantes de la misma que usan otros componentes e introducen algunos conceptos nuevos (Engelbrecht, 2007). Sin embargo, Falkenauer (1996) identificó que los GAs tienen inconvenientes para resolver problemas combinatorios de agrupación. Por lo cual propuso un nuevo algoritmo: el GGA. El cual se enfoca en trabajar con grupos, en lugar de los elementos de forma individual, como ocurre con el GA. En la Figura 3.3 se puede observar la principal diferencia entre el GGA y el GA. Suponiendo que se tienen 6 objetos los cuales son agrupados en 4 grupos distintos (identificados por el color), una forma de representar una solución con un enfoque en elementos es la que se

podría usar en un GA, en la que a cada elemento se le asigna una etiqueta con el número de grupo al que pertenecen. Por otro lado, la representación enfocada en grupos trabaja directamente con los 4 grupos, los cuales contienen los mismos elementos asignados al grupo correspondiente.

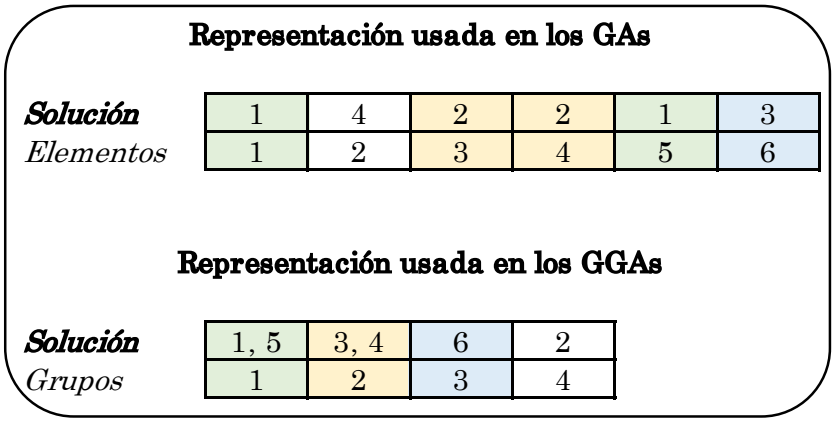



Figura 3.3: Diferencia entre representación usada por GAs y GGAs.

Numerosos problemas combinatorios del mundo real y con una complejidad alta se han resuelto exitosamente empleando los GGAs (Mutingi and Mbohwa, 2017). Al usar este tipo de algoritmos, resulta más sencillo plantear problemas de optimización combinatoria que involucran particiones de conjuntos, por ejemplo los problemas de clustering.

El GGA-CGT es uno de los mejores del estado del arte para el problema de empacado de objetos en contenedores en una dimensión (1D-BPP, por sus siglas en inglés). Incluye elementos que permiten tener una mayor diversidad en la población de soluciones.

El 1D-BPP busca asignar un conjunto de objetos en contenedores, haciendo uso de la menor cantidad de contenedores posible. Se trata de un problema NP-duro. A continuación se explican brevemente los componentes principales del GGA-CGT como aparecen en el trabajo de Quiroz-Castellanos et al. (2015).

En el Algoritmo 1 se muestran los componentes y el procedimiento que sigue el GGA-CGT para encontrar la mejor solución. Primero, se genera una población utilizando la heurística FF-ñ. Después, mientras no se encuentre el óptimo teórico conocido ($L_2$) ni se alcance el número máximo de generaciones, se ejecuta el ciclo en el que se realiza la cruza y mutación de los individuos en dos fases. En la priemra fase, se lleva a cabo la selección

controlada de individuos para luego aplicar la cruza a los $n_c$ individuos seleccionados empleando el operador GLX, junto con su respectivo método de reparación de soluciones FFD. Los descendientes se introducen a la población con el reemplazo controlado para la cruza. En la segunda fase, se lleva a cabo la mutación. Previo a este procedimiento, se clonan las soluciones del grupo élite cuya edad sea menor a la establecida (*life_span*) y posteriormente se aplica la mutación a los $n_m$ mejores individuos de la población. Para la reparación en la mutación se emplea el método de reacomodo por pares (RP). Por último, los clones se introducen a la población de forma controlada y se actualiza la mejor solución global. La calidad de las soluciones se mide con respecto a la función de evaluación de la Ecuación 3.1.

**Algoritmo 1:** GGA-CGT.

**Entrada:** $P$ tamaño de la población, $N$ número máximo de generaciones, $n_c$ número de individuos a cruzar, $n_m$ número de individuos a mutar.
**Salida:** $S$, la mejor solución encontrada.

```
1  Generar población inicial con FF-ñ.
2  mientras generacion < N y Tamaño(mejor_solucion) > L2 hacer
3      Seleccionar n_c individuos por medio de la selección controlada.
4      Aplicar cruza con GLX+FFD a los n_c individuos seleccionados.
5      Introducir hijos mediante reemplazo controlado para cruza.
6      Seleccionar n_m individuos por medio de la selección controlada.
7      Clonar soluciones élite con edad menor que life_span.
8      Aplicar mutación adaptativa con RP a los mejores n_m individuos.
9      Introducir los clones con reemplazo controlado para mutación.
10     Actualizar la mejor solución global S.
11 fin
```

A continuación, se describen con mayor detalle los componentes del GGA-CGT.

### 3.3.1. Función de evaluación

La función de evaluación utilizada en el GGA-CGT se basa en el principio de que es mejor tener contenedores que estén casi llenos y también tener contenedores casi vacíos, de esta forma resulta más fácil eliminar los contenedores que estén casi vacíos para después acomodar estos objetos sobrantes en los contenedores que están más llenos. Así, esta función evalúa la proporción de llenado de los $m$ contenedores con capacidad $c$ utilizados

en una solución con la siguiente ecuación:

$$F_{BPP} = \frac{\sum_{i=1}^{m} \left(\frac{S_i}{c}\right)^2}{m}. \tag{3.1}$$

El objetivo es maximizar esta función. Donde $S_i$ es la suma de pesos de los objetos contenidos en el contenedor $i$.

### 3.3.2. Inicialización de la población

La población se inicializa utilizando una heurística llamada FF-ñ. Con esta heurística primero se seleccionan los objetos que sobrepasen el 50 % de la capacidad del contenedor. Cada uno de estos objetos es asignado a un solo contenedor y cuando ya no hay más objetos que cumplan con esta característica, el resto de los objetos son permutados aleatoriamente y se asignan utilizando la heurística First-Fit (FF), es decir, cada objeto se asigna al primer contenedor en el que exista suficiente capacidad para almacenarlo.

### 3.3.3. Cruza

El operador de cruza utilizado lleva por nombre Gene Level Crossover (GLX). Dicho operador está basado en grupos, por lo que busca heredar los mejores grupos de las soluciones padre a las soluciones hijo. Con este operador, se tienen dos padres que generan un par de hijos. Primero los genes de ambos padres son ordenados de forma descendente de acuerdo con el llenado de los contenedores. Después se recorren los genes de ambos padres y se comparan uno a uno: si el del primer padre es mejor, se transmite primero, seguido del gen del segundo padre; de lo contrario, el segundo padre transmite su gen primero. En caso de que los grupos tengan la misma calidad, se le da prioridad al primer padre. Para generar el segundo hijo, se alternan las posiciones de los padres y se realiza el mismo procedimiento mencionado anteriormente. En este caso, los genes se comparan dándole prioridad al segundo padre en caso de que la calidad de los genes sea la misma. Cabe destacar que, cuando el número de grupos es variable, el exceso de grupos es heredado directamente a los hijos sin realizar una comparación. Adicionalmente, cuando se heredan

grupos que contienen objetos repetidos, estos son eliminados de la solución hijo.

### 3.3.4. Mutación

Para la mutación se utiliza una versión mejorada del operador Elimination, llamada Adaptive Mutation Operator. Este operador también trabaja a nivel de grupos. Su parte adaptativa está en considerar el número de grupos a eliminar de acuerdo con la cantidad y el llenado de los contenedores que se tienen en la solución, en lugar de eliminar un porcentaje fijo de estos. De esta forma, cuando la solución tiene una gran cantidad de contenedores, la proporción de grupos a eliminar no es la misma que si se tuvieran pocos contenedores, sino que se busca eliminar una menor cantidad de estos con el objetivo de reducir el costo computacional.

### 3.3.5. Método de reparación de soluciones

Al tener restricciones en el problema, es más probable que las soluciones generadas en la cruza y mutación violen dichas restricciones. Al aplicar los operadores de variación para este problema, muchas de las soluciones que se producen son infactibles, por lo que es necesario repararlas. Para esto, se aplican algunas estrategias que permiten reasignar objetos perdidos, o eliminar objetos repetidos, etc.

Debido a que durante el proceso de cruza se eliminan los contenedores que tienen objetos repetidos, la reparación de soluciones es más sencilla y solo se aplica la heurística Firs-Fit Decreasing (FFD). Esta heurística ordena los objetos perdidos de forma descendente de acuerdo con el peso de los objetos y estos son asignados a aquel contenedor que tenga suficiente capacidad para almacenarlo.

En el caso de la mutación, las soluciones infactibles son reparadas mediante la heurística de reacomodo por pares, la cual hace intercambios entre pares de objetos libres y pares de objetos en contenedores con el fin de intercambiar objetos más pesados por objetos menos pesados y al final intentar insertarlos usando la heurística FF.

### 3.3.6. Esquemas de selección

La selección de padres tiene un efecto importante en la diversidad y calidad de la población. Por un lado, elegir únicamente buenos padres podría afectar la diversidad de las soluciones obtenidas, mientras que elegir padres de forma totalmente aleatoria, podría resultar inconveniente para que el algoritmo logre converger y hallar una buena solución. Los esquemas de selección empleados en el GGA-CGT permiten mantener un buen balance con respecto a la presión de selección.

Para la cruza, la selección de padres contempla una selección de la primera mitad de padres con un valor de aptitud alto, y la otra mitad provenientes de una selección aleatoria, excluyendo al grupo élite de la solución. El grupo élite tiene un tamaño definido (generalmente del 10 %) y este es excluido de la selección aleatoria, pero sí se toma en cuenta para la selección de la primera mitad de padres con mejor aptitud.

La mutación se aplica a los mejores individuos de la población, pero siempre se clonan los individuos del grupo élite cuya edad es menor a algún valor establecido llamado *life_span*, el cual se mide con respecto al número de generaciones que una solución lleva en la población. Esto se hace con la finalidad de no perder las mejores soluciones.

### 3.3.7. Estrategias de reemplazo

Estos mecanismos permiten mantener una diversidad de la población y evitar que el algoritmo converja rápidamente o que se quede atorado en óptimos locales. Para el GGA-CGT se tienen dos estrategias de reemplazo, una para la cruza y otra para la mutación, las cuales son explicadas brevemente a continuación.

Después de aplicar el operador de cruza, los hijos son introducidos a la población de forma controlada. Es decir, primero se reemplaza a los padres seleccionados de forma aleatoria, y el resto de hijos reemplazan primero a las soluciones con aptitud duplicada y después a las soluciones con peor valor de aptitud.

Como se menciona en el esquema de selección para la mutación, se tiene un conjunto de copias de las mejores soluciones del grupo élite. Después de la mutación, estas soluciones son introducidas a la población reemplazando a las soluciones con aptitud duplicada y

después a las de peor aptitud.

## 3.4. Conclusiones

De acuerdo con lo observado ne la literatura, los algoritmos bioinspirados suelen tener una mayor aplicación en la segmentación de bajo nivel, donde se emplean los métodos más utilizados como umbralización y clustering. Además, la función que más suele ser utilizada para evaluar la calidad de las soluciones es la distancia intra-cluster.

Dentro de los algoritmos bioinspirados que se emplean para la segmentación de imágenes, destaca el uso de GAs para encontrar los parámetros óptimos en el caso del método de umbralización. En el caso del método de clustering, también existen propuestas que emplean GAs. Sin embargo, dado que la segmentación se trata como un problema de agrupación, los GAs no suelen ser la mejor opción.

Finalmente, se puede ver que los GGAs son una buena e interesante opción para realizar la segmentación de imágenes. Ante esto, se propone adaptar el GGA-CGT que, como se mencionó, fue diseñado para resolver el 1D-BPP y tiene elementos que permiten hacer una búsqueda más inteligente y controlada. Debido a su destacado desempeño para resolver este problema, resulta de interés adaptarlo al problema de la segmentación de imágenes y observar su desempeño. La adaptación del GGA-CGT para el problema anteriormente mencionado es el tema a tratar en el siguiente capítulo.

# Capítulo 4

# Adaptación del GGA-CGT a la segmentación

Debido a las características del GGA-CGT de Quiroz-Castellanos et al. (2015), así como el buen desempeño mostrado en sus resultados, se decidió trabajar con este algoritmo para adaptarlo al problema de la segmentación de imágenes en RGB. Dado que este es un problema de agrupación, resultó sencillo realizar su adaptación para dicho problema, salvo algunos aspectos que no pudieron considerarse por las características del problema de la segmentación de imágenes. En este capítulo se describe la adaptación del GGA-CGT para este problema, así como las consideraciones y diferencias que existen. Este algoritmo recibe el nombre de GGA-CGT-RGB-IS que, como se mencionó anteriormente, recibe su nombre por los elementos de la transmisión controlada de genes y la segmentación de imágenes en RGB. También se realizaron experimentos para observar su desempeño comparado con K-Means con las imágenes de prueba de BSDS500 utilizando los mismos parámetros del GGA-CGT para el 1D-BPP.

El GGA-CGT-RGB-IS utiliza una representación basada en grupos, donde un cromosoma representa una solución potencial; es decir, el esquema codificado de segmentación de la imagen. Este cromosoma está formado por genes, los cuales representan los grupos en los que se busca segmentar la imagen. A su vez, estos grupos contienen los índices de los píxeles que han sido asignados a dicho grupo, siguiendo las restricciones planteadas en la definición del problema: los píxeles solo pueden pertenecer a un solo grupo y no debe

haber píxeles sin asignar. Como información adicional, se guarda la cantidad de píxeles contenidos en cada grupo, así como la distancia intra-cluster de cada grupo.

## 4.1. Diseño del algoritmo

El algoritmo sigue el esquema clásico de un GA. En el Algoritmo 2 se muestra el pseudocódigo del GGA-CGT-RGB-IS, el cual es muy similar al del GGA-CGT original. En dicho algoritmo se muestra la forma en la que trabajan las estrategias del GGA-CGT adaptadas al problema de la segmentación de imágenes. Los parámetros de entrada son el tamaño de la población, el número máximo de generaciones del algoritmo, número de individuos a cruzar, número de individuos a mutar, la imagen a color que se va a segmentar, así como el número de segmentos en los que se van a agrupar los píxeles de la imagen. Como salida se tiene la mejor solución global que se haya obtenido en cualquiera de las generaciones en las que se ejecutó el algoritmo. A diferencia del GGA-CGT, este algoritmo emplea la distancia intra-cluster (Ecuación 4.1) como función de evaluación. En este algoritmo se manejan dos condiciones de paro. La primera es el número máximo de generaciones y la segunda es cuando el algoritmo ha encontrado un óptimo global, es decir, que la mejor solución tiene una distancia intra-cluster de cero.

**Algoritmo 2:** GGA-CGT-RGB-IS.

**Entrada:** $P$ tamaño de la población, $N$ número máximo de generaciones, $n_c$ número de individuos a cruzar, $n_m$ número de individuos a mutar, *img* imagen y $k$ número de segmentos.

**Salida:** $S$, la mejor solución encontrada.

```
1  Generar población inicial de tamaño P de forma aleatoria.
2  mientras generacion < N e intra(S) > 0 hacer
3      Seleccionar n_c individuos por medio de la selección controlada.
4      Aplicar cruza con GLX a los n_c individuos seleccionados.
5      Introducir hijos mediante reemplazo controlado para cruza.
6      Seleccionar n_m individuos por medio de la selección controlada.
7      Clonar soluciones élite con edad menor que life_span.
8      Aplicar mutación con Elimination a los mejores n_m individuos.
9      Introducir los clones con reemplazo controlado para mutación.
10     Actualizar la mejor solución global S.
11 fin
```

El proceso inicia con la generación de una población inicial $P$ de forma aleatoria. Después, mientras no se encuentre un óptimo global, se realiza el proceso evolutivo por un número máximo de generaciones $N$. Los individuos seleccionados son cruzados y mutados en dos fases. Primero se seleccionan $n_c$ individuos mediante la selección controlada y se aplica el operador GLX a estos individuos. Los hijos se introducen mediante el reemplazo controlado para la cruza. En la segunda fase, $n_m$ individuos son seleccionados con la selección controlada. Antes de aplicar la mutación, se clonan las soluciones del grupo élite con edad menor que *life_span*. La mutación se realiza con el operador de Elimination. Posteriormente, los clones se introducen con el reemplazo controlado para la mutación. Finalmente, la mejor solución global es actualizada.

A continuación se describen con mayor detalle los elementos del GGA-CGT-RGB-IS.

### 4.1.1. Esquema de representación

La representación utilizada por el GGA-CGT-RGB-IS es la misma que la del GGA-CGT: la representación basada en grupos. Para ilustrar la idea acerca de la representación de las soluciones, se revisa la estructura del padre 1 de la Figura 4.3. Dicho cromosoma está formado por 2 genes o grupos (columnas marcadas en gris), los cuales contienen los índices de los píxeles que pertenecen a determinado grupo, en la fila que está debajo de la representación de grupo se muestra la distancia intra-cluster de dicho grupo, obtenida a partir de la distancia euclidiana de cada píxel al centroide del grupo en cuestión. Así, el grupo $G_1$ del padre 1 está formado por los píxeles del 1 al 4 y la distancia intra-cluster de este grupo es de 0. Mientras que el grupo $G_2$ contiene los píxeles del 5 al 9 y su distancia intra-cluster es de 81.6.

### 4.1.2. Función de evaluación

El primer elemento en el que existe una diferencia es en la función de evaluación. La función de evaluación con la que se trabajó es la distancia intra-cluster, definida en la Ecuación 4.1, misma que ha sido utilizada por Awad et al. (2007) Awad et al. (2009) y Halder et al. (2011) Halder et al. (2016). Esta función debe ser minimizada, pues una dis-

tancia intra-cluster menor es deseable, ya que nos indica que los píxeles han sido asignados de una mejor manera en los segmentos.

$$intra(S) = \sum_{G_k \in S} \sum_{x \in G_k} d(x, \mu_k). \tag{4.1}$$

La elección de esta función se basó en una serie de pruebas con algunas imágenes del banco de imágenes de control usando las siguientes funciones de evaluación: distancia intra-cluster, distancia inter-cluster, razón distancia intra/inter-cluster. Sin embargo, la función de evaluación que consiguió mejores resultados fue intra-cluster. Con el resto de funciones el algoritmo fue incapaz de converger al óptimo global. Aunque la razón por la que estas funciones de evaluación no proporcionaron buenos requeriría un análisis más detallado, esto podría deberse a las características de las instancias de prueba, ya que estas imágenes están diseñadas para conseguir una distancia intra-cluster de 0 en el mejor de los casos, mientras que no se conoce el óptimo global para la distancia inter-cluster.

### 4.1.3. Inicialización de la población

La inicialización de una solución de la población se realiza de forma completamente aleatoria. De forma que se crean los grupos conteniendo píxeles que muy probablemente no tienen algún grado de similitud. La forma en que se inicializa la población garantiza que se creen grupos de tamaño no proporcional y que contengan píxeles seleccionados de forma aleatoria, sin violar las restricciones del problema.

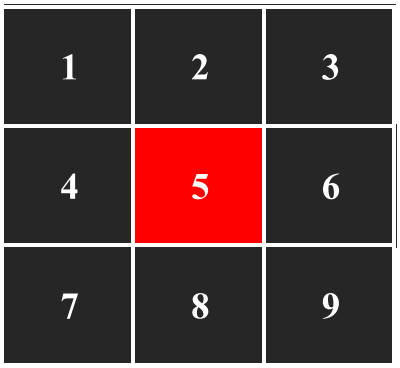



Figura 4.1: Imagen RGB para ejemplo de inicialización de la población.

En la Figura 4.2 se muestra un ejemplo de la inicialización de la población para la

imagen de la Figura 4.1. Primero se parte de una lista con los índices de los píxeles de la imagen que tienen asociados sus valores de intensidad RGB. Posteriormente se realiza una permutación de esta lista y a partir de esta permutación se eligen $k-1$ puntos de corte, los cuales se asignan de forma completamente aleatoria, evitando que estos queden en la primera o última posición de la lista, pues esto generaría grupos vacíos. De esta forma, los píxeles contenidos en cada división forman un segmento. En el ejemplo, se buscan tres segmentos, por lo que se seleccionan dos puntos de corte de forma aleatoria a lo largo de la lista de 9 píxeles, formando así tres grupos. El primer segmento contiene los píxeles 7 y 8, el segundo contiene los píxeles 5, 4, 1 y 3, mientras que los píxeles 9, 2 y 6 pertenecen al tercer segmento.

**Lista de los píxeles de la imagen de ejemplo:**

| ***Píxel*** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| ***Intensidad (RGB)*** | 0, 0, 0 | 0, 0, 0 | 0, 0, 0 | 0, 0, 0 | 255, 0, 0 | 0, 0, 0 | 0, 0, 0 | 0, 0, 0 | 0, 0, 0 |

**Lista permutada de los índices de los píxeles con *k-1* puntos de corte:**

| ***Píxel*** | 7 | 8 | 5 | 4 | 1 | 3 | 9 | 2 | 6 |
|---|---|---|---|---|---|---|---|---|---|
| ***Intensidad (RGB)*** | 0, 0, 0 | 0, 0, 0 | 255, 0, 0 | 0, 0, 0 | 0, 0, 0 | 0, 0, 0 | 0, 0, 0 | 0, 0, 0 | 0, 0, 0 |

Figura 4.2: Ejemplo de la inicialización de una solución de la población para una imagen cuando se busca segmentar en 3 segmentos, $k = 3$.

### 4.1.4. Cruza

Para la cruza, se utilizó el operador GLX original del GGA-CGT. El operador sigue funcionando de la misma forma, generando dos hijos por cada par de padres. En este caso, el criterio de ordenamiento de los genes es la distancia intra-cluster por segmento. De forma que los mejores genes serán los segmentos que tengan la menor distancia intra-cluster. Cuando los genes tienen la misma calidad, se le da prioridad al primer padre, como ocurre en la versión inicial del algoritmo.

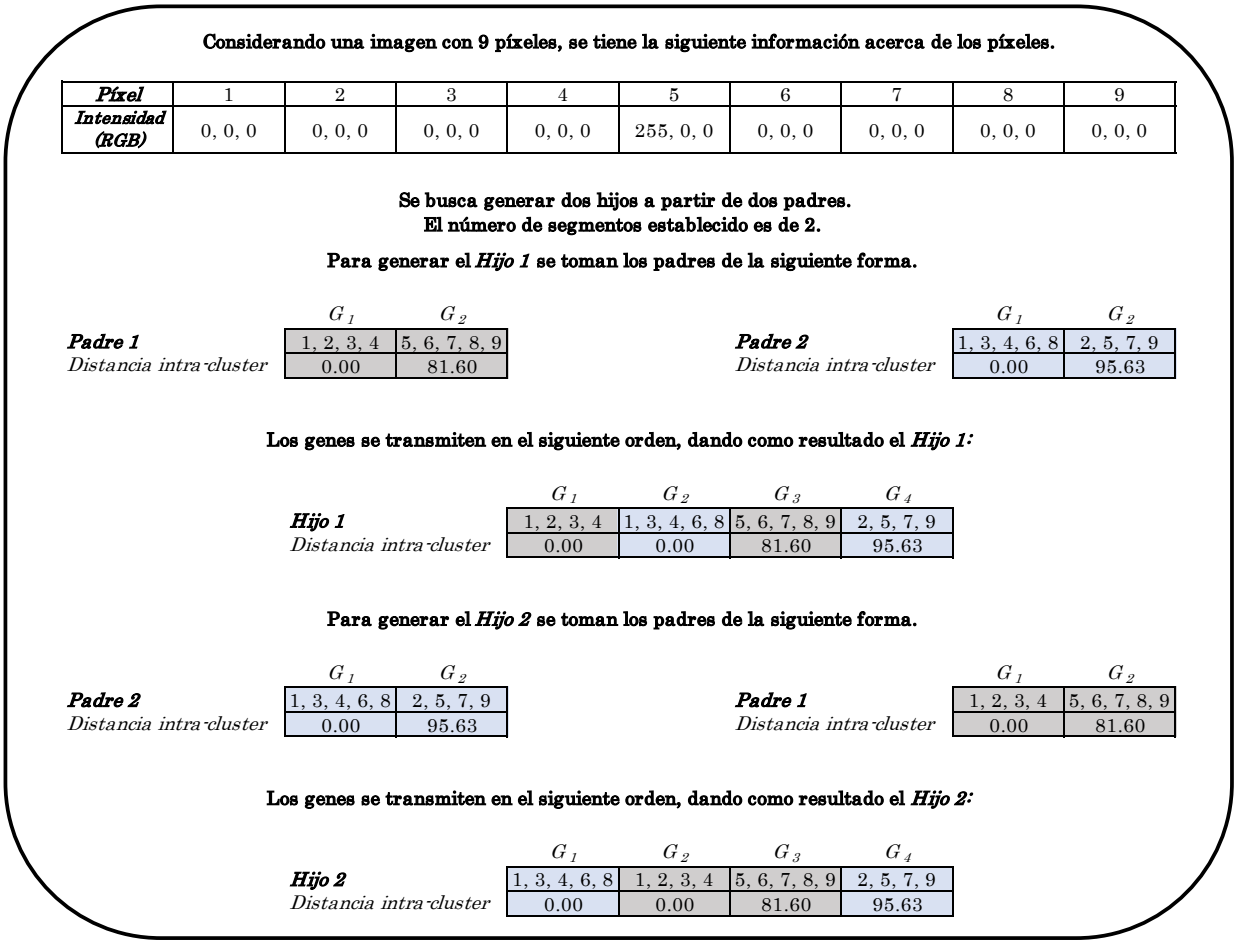



Figura 4.3: Operador de cruza GLX empleado para el problema de segmentación de imágenes en RGB con el GGA-CGT-RGB-IS.

Este operador toma un par de padres y ordena sus genes de acuerdo a su valor de calidad, donde los primeros genes serán los grupos con menor distancia intra-cluster. Después, se recorren estos genes, uno a uno y se compara su valor de aptitud: si la distancia es la misma, se le da prioridad al primer padre, el cual heredará su primer gen al hijo, seguido del gen del segundo padre. Si la distancia es diferente, el gen con la menor distancia va primero, seguido del otro gen. En la Figura 4.3 se puede observar el funcionamiento del operador de cruza. En este ejemplo, los genes se encuentran ordenados de mejor a peor (recordando que una distancia intra-cluster baja es mejor). Siguiendo con el procedimiento descrito anteriormente, se puede observar que la calidad de estos genes es la misma, por lo que se le da prioridad al padre 1, de forma que el hijo tiene el primer gen del padre 1 seguido del gen del segundo padre en esta primera iteración. Para el segundo y último

gen, se comparan las distancias intra-cluster, donde se observa que la del padre 1 es menor, razón por la cual este gen va primero, seguido del gen del segundo padre. Así, el hijo generado tiene un total de 4 grupos y 18 píxeles en total. Para el generar el segundo hijo, los padres se toman en un orden distinto. En esta ocasión, el padre 2 va primero, por lo que la prioridad se le da a este. Es así como se genera el segundo hijo, donde los primeros genes de cada padre tienen la misma calidad, por lo que se elige primero el gen del segundo padre, seguido del del padre 1. Para el segundo gen, el orden sigue siendo el mismo que para el primer hijo, pues el mejor gen sigue siendo el del padre 2. Como se puede observar, las soluciones generadas violan las restricciones del problema: se tienen píxeles repetidos y además el número de segmentos es mayor al establecido. Ante esto, es necesario aplicar un método de reparación de las soluciones, el cual se describe más adelante.

### 4.1.5. Mutación

El operador de mutación utilizado es el de Elimination. Debido a las características del problema de la segmentación de imágenes, la estrategia adaptativa que determina el número de grupos a eliminar no se puede aplicar para este problema, pues el número de segmentos que se pueden tener no es tan alto como en el caso del problema de empacado de objetos. El conjunto de imágenes de prueba de BSDS500 tiene un rango de segmentos que va desde 2 hasta 54 segmentos. Con base en esto, el operador trabaja eliminando el 20 % de los segmentos de la solución a mutar. Sin embargo, cuando el número de segmentos es menor que 5, el 20 % de los grupos es menor que 1. Esto deja la posibilidad de que no se utilice el operador de mutación bajo estos casos. Para mitigar este posible problema, se decidió emplear la función techo para asegurarse de que al menos un segmento sea eliminado en todo momento.

Para ilustrar el funcionamiento de este operador bajo el contexto del problema, en la Figura 4.4 se muestra un hijo con 2 grupos. Esta solución tiene sus genes ordenados de mejor a peor, por lo que el siguiente paso es eliminar un grupo de dicha solución. En este caso, el peor grupo de la solución es $G_2$, pues su distancia intra-cluster es más alta que la del grupo $G_1$. Una vez eliminado dicho grupo, el hijo mutado ahora tiene un solo grupo que contiene los píxeles 1, 3, 4, 6. Esta solución no es factible, por lo que es necesario repararla

con la misma heurística de reparación que se usa en la cruza, la cual al ser aplicada produce la solución llamada "reparación" que nuevamente es una solución factible.

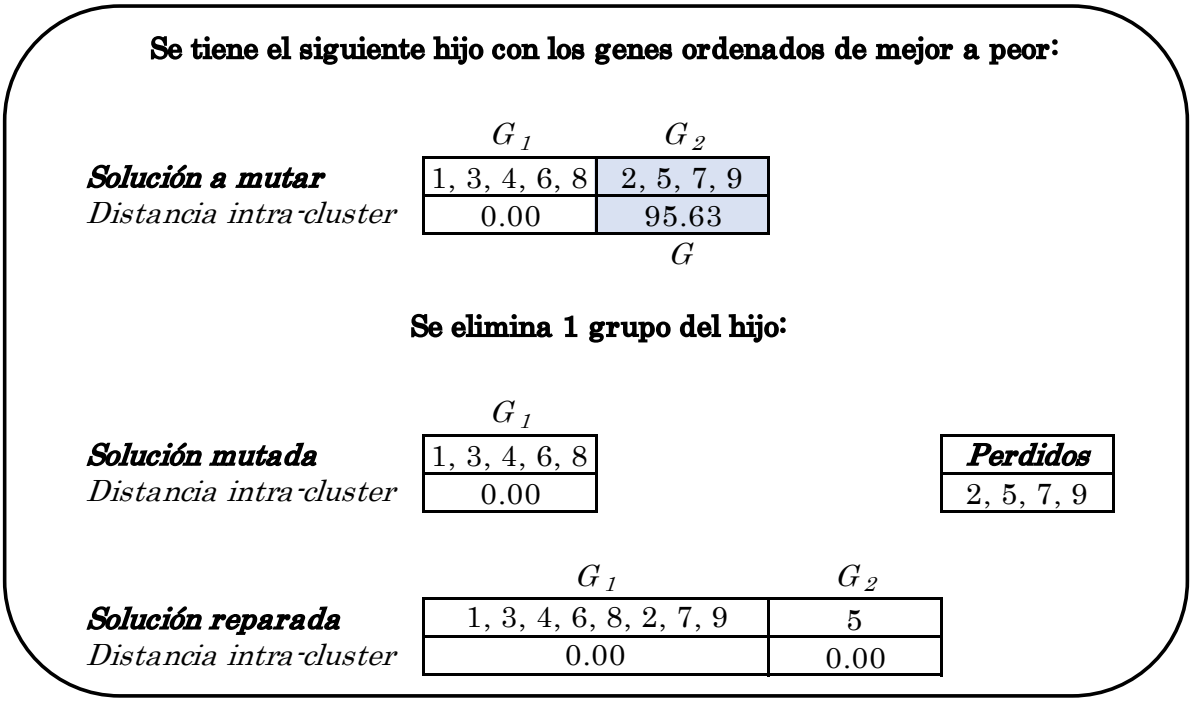



Figura 4.4: Operador de mutación Elimination empleado para el problema de segmentación de imágenes en RGB con el GGA-CGT-RGB-IS.

### 4.1.6. Método de reparación de soluciones

De las estrategias utilizadas para reparar las soluciones en el GGA-CGT, el FFD no se puede utilizar para reparar las soluciones producidas mediante la cruza, pues no existen límites de capacidad en los grupos ni se puede considerar algún atributo parecido a la capacidad de los contenedores. De manera similar, con las soluciones infactibles producidas por la mutación no es posible utilizar el reacomodo por pares (RP), pues en el problema de la segmentación de imágenes, los píxeles no tienen el atributo de peso, ni existe un nivel de llenado de los grupos. De forma que para reparar las soluciones se propuso una nueva estrategia que toma en cuenta dos casos que pueden presentarse de manera no excluyente:

- El número de grupos es diferente al establecido ($k$).

- El número de píxeles es diferente al número total de píxeles de la imagen ($n$). Esto implica que puede haber píxeles perdidos y/o repetidos.

Este método de reparación se encarga eliminar o crear grupos para que se siga teniendo la misma cantidad. Cuando es necesario eliminar grupos, se eliminan los peores, mientras que cuando faltan grupos, se asigna uno de los píxeles perdidos de forma aleatoria a un grupo nuevo. Un caso especial puede ocurrir cuando no hay suficientes píxeles para crear los grupos perdidos. En ese caso, se toman píxeles aleatoriamente de grupos cuya cardinalidad es mayor que 1. La inserción de los píxeles se basa en la heurística Best-Fit (BF) y los asigna de la siguiente forma: elige un píxel de forma aleatoria y se evalúa el efecto de agregar dicho píxel a cada uno de los grupos, de forma que se asigna al grupo donde tiene una mayor afinidad; es decir, se asigna al grupo cuyo centroide sea el más cercano al del píxel en cuestión. El efecto deseable de poner ese píxel en el grupo es que mejore la distancia intra-cluster de dicho grupo.

### 4.1.7. Esquemas de selección

Se adaptaron los esquemas de selección del GGA-CGT para el problema de la segmentación de imágenes. Debido a que estos esquemas son genéricos, es posible aplicarlos tomando como valor de aptitud la suma de las distancias intra-cluster. El tamaño del grupo élite sigue siendo el mismo que en el algoritmo original, cuyo porcentaje es del 10 % del tamaño de la población. Aunque el análisis del efecto de este parámetro está fuera del alcance de este proyecto, Quiroz-Castellanos et al. (2015) mencionan que el tamaño del grupo élite permite ajustar el balance entre la presión selectiva y la diversidad de la población del algoritmo.

### 4.1.8. Estrategias de reemplazo

La adaptación de las estrategias de reemplazo del GGA-CGT se aplicaron para esta versión del algoritmo. Tal y como se describen en el GGA-CGT, estas estrategias se basan en el reemplazo de soluciones con valor de aptitud repetido y los peores, así como el

reemplazo de los padres aleatorios, en el caso del reemplazo para la cruza. También se conservan las mejores soluciones del grupo élite con una edad menor que 10 generaciones.

## 4.2. Experimentos con imágenes

Con el objetivo de evaluar el desempeño de la nueva versión del algoritmo adaptada a la segmentación de imágenes, se realizó la segmentación de las imágenes de prueba disponibles en la base de BSDS500, que consiste en conjunto de 200 imágenes a color. Estas imágenes fueron reescaladas de su resolución original de $431 \times 321$ a $80 \times 53$ píxeles, esto con el fin de reducir el costo computacional. Con respecto al número de segmentos proporcionados por la base de datos, se tenían cinco archivos en promedio por cada imagen, estos archivos contenían las etiquetas de los píxeles de acuerdo con el segmento al que pertenecían. Para fines de los experimentos, se decidió trabajar con el mínimo número de segmentos de cada imagen y se realizaron 30 ejecuciones por imagen utilizando semillas diferentes.

Para estos experimentos se utilizaron los parámetros originales a excepción del número de generaciones. En el algoritmo original se utilizaban 500 generaciones. Sin embargo, en los experimentos realizados con algunas de las imágenes de prueba de BSDS500 se observó que más allá de la generación 100, la distancia intra-cluster de la mejor solución encontrada con el algoritmo ya no tiene un cambio significativo, por esta razón, se limitó a 100 generaciones. A continuación se muestran los parámetros utilizados en el algoritmo:

- Tamaño de población, $P$: 100 individuos.
- Número máximo de generaciones, $N$: 100.
- Número de individuos a cruzar, $n_c$: 20 %
- Número de individuos a mutar, $n_m$: 83 %
- Número de individuos en el grupo élite: 10.
- Número máximo de generaciones de un individuo para ser clonado, *life_ span*: 10.

Adicionalmente, se utilizó K-Means como referencia. Se utilizó una versión de Python de la librería Scikit-learn (Pedregosa et al., 2011). Siguiendo la documentación sobre dicha implementación, se configuraron los siguientes parámetros del algoritmo:

- Método de inicialización: aleatorio.
- Número de ejecuciones simultáneas: 1.
- Máximo número de iteraciones: 100.

En la Tabla 4.1 se observan los mejores resultados obtenidos por el GGA-CGT-RGB-IS agrupados por rangos de número de segmentos. Se tienen cuatro grupos de acuerdo con el rango de segmentos de las imágenes de prueba de BSDS500. La primera columna corresponde al rango de número de segmentos por el que fueron agrupados los resultados de segmentación y las siguientes columnas muestran tres estadísticas para los valores de *RPD* de las imágenes segmentadas que pertenecen al grupo en cuestión. Para las estadísticas de mejor, media y peor, se usan los promedios de los valores contenidos en cada grupo.

Estos resultados están reportados en términos de la diferencia de porcentaje relativo *RPD*, en lugar de la distancia intra-cluster. Esto con el objetivo de comparar fácilmente el algoritmo propuesto con respecto a K-Means. La forma de en que se calculó el *RPD* fue utilizando la Ecuación 4.2. Donde $intra(i)$ es el valor de la distancia intra-cluster de la mejor solución global encontrada por el GGA-CGT-RGB-IS para la imagen $i$, mientras que $intra*(i)$ corresponde a la distancia intra-cluster de la solución encontrada por K-Means.

$$RPD = \frac{intra(i) - intra*(i)}{intra*(i)}. \tag{4.2}$$

Adicionalmente, los resultados de la segmentación fueron agrupados por el número de segmentos asignado a cada imagen. De esta forma, en la Figura 4.5 se muestran los resultados del *RPD* promedio por número de segmentos con intervalos de confianza del 95 %. De acuerdo con esta medida de *RPD* promedio, lo ideal es que este valor esté lo más cercano posible al 0, incluso menor que 0, pues sería una señal de que la calidad de la solución obtenida por el algoritmo fue mejor que la de referencia (K-Means, en este caso).

Tabla 4.1: *RPD* promedio de la segmentación de las imágenes de prueba de BSDS500 agrupadas por número de segmentos con el GGA-CGT-RGB-IS.

| **No. Segmentos** | **Mejor** | **Media** | **Peor** |
|---|---|---|---|
| *(1, 14]* | 0.024 (0.0087) | 0.0099 (0.0211) | -0.0262 (0.0596) |
| *(14, 24]* | 0.0264 (0.0047) | 0.0093 (0.0132) | -0.0261 (0.0442) |
| *(24, 55]* | 0.0345 (0.0107) | 0.0161 (0.0176) | -0.0382 (0.0607) |


Figura 4.5: *RPD* promedio comparado con el número de segmentos definido para las 200 imágenes de prueba de BSDS500.

De acuerdo con los valores de *RPD* calculados, que se muestran en la Figura 4.5, se puede observar que la variación más alta es menor al 5 % con respecto a K-Means, donde los valores más altos coinciden con las imágenes que tienen un mayor número de segmentos. También se observa que algunos valores de RPD son menores a 0 para algunas imágenes que se encuentran entre el rango de 12 a 32 segmentos.

Es importante señalar que, dado que el algoritmo está utilizando los parámetros originales que fueron elegidos para resolver el 1D-BPP, es necesario realizar una calibración del algoritmo para obtener mejores resultados, así como explorar los diferentes operadores de mutación para GGAs que se han propuesto en el estado del arte para usar el más adecuado

para este problema. De acuerdo con Quiroz-Castellanos et al. (2015) y Ramos-Figueroa et al. (2023), el desempeño del GGA-CGT está altamente relacionado con el operador de mutación. Generalmente este operador es útil para promover la exploración en el espacio de búsqueda, especialmente en problemas de agrupación con muchas restricciones, donde la posibilidad de quedar atrapado en un óptimo local es muy alta. Considerando lo anterior, la selección de un operador de mutación para el problema de segmentación de imágenes es relevante. Por lo tanto, el desempeño del GGA-CGT-RGB-IS puede ser mejorado con la selección adecuada de un operador de mutación, es por eso que estos operadores se implementarán y se pondrán a prueba con una serie de experimentos observando el desempeño del algoritmo bajo el uso de dichos operadores. En el siguiente capítulo se abordan los operadores de mutación de agrupación del estado del arte.

# Capítulo 5

# Operadores de mutación del estado del arte para GGAs

En este capítulo se describen detalladamente los operadores de mutación del estado del arte que se adaptaron para resolver el problema de la segmentación de imágenes en RGB con el GGA-CGT-RGB-IS. Para cada operador se muestra su algoritmo y se explica cómo funciona de forma general y qué consideraciones se toman específicamente para el problema de la segmentación de imágenes.

Debido a que la selección de operadores de variación está en función del problema a resolver, es importante seguir una metodología adecuada para la selección de estos. En el GGA-CGT para el 1D-BPP, propuesto por Quiroz-Castellanos et al. (2015), se puso especial cuidado en el diseño del algoritmo para que este tuviera un mejor desempeño, incorporando conocimiento del dominio del problema. De acuerdo con los resultados obtenidos, este es uno de los mejores algoritmos del estado del arte para resolver el 1D-BPP y ha sido adaptado para resolver otros problemas de agrupación. Siguiendo esta idea, Amador-Larrea (2022) aplicó con éxito una metodología para el diseño de un operador de cruza, mejorando el GGA-CGT. En el trabajo mencionado anteriormente se puso énfasis en la experimentación con distintos operadores de cruza, utilizando diferentes estrategias de ordenamiento y transmisión de genes, así como el criterio de generar uno o dos descendientes por cada par de padres durante la cruza. De esta forma, el conocimiento obtenido en el estudio experimental permitió diseñar el nuevo operador de cruza mencionado, mejorando

el desempeño del GGA-CGT. También, Ramos-Figueroa (2022) diseñó el primer GGA para el problema de programación de máquinas no paralelas con máquinas no relacionadas y minimización del tiempo de producción ($R||C_{max}$) donde se siguió una metodología similar para evaluar las estrategias y operadores, permitiendo seleccionar el más adecuado para el problema en cuestión. En el trabajo citado anteriormente se puso énfasis en conocer cada aspecto del problema para poder identificar las estrategias que favorecieran la obtención de mejores soluciones. La correcta selección de los operadores de variación puede garantizar un buen desempeño del algoritmo.

En el caso de los operadores de mutación, Mutingi and Mbohwa (2017) mencionan que los detalles de la implementación de este operador para los GGAs dependerá del problema específico en cuestión, por lo que es crucial el desarrollo de heurísticas para guiar de forma inteligente el procedimiento de mutación. En ese sentido, la mutación debe evitar que la búsqueda se quede atrapada en un óptimo local y debe incrementar la probabilidad de alcanzar un óptimo global. De acuerdo con Hong et al. (2000), esto se consigue con una serie de cambios en pequeña escala y de forma aleatoria. La mutación se puede considerar como un simple operador de búsqueda. En contraste con el operador de cruza, el cual busca explotar una solución para encontrar mejores soluciones, lo que se busca con la mutación es ayudar en la exploración del espacio de búsqueda, como indican Abdoun et al. (2012). Gracias al operador de mutación se mantiene una diversidad genética en la población.

De acuerdo con la revisión de operadores de variación para GGAs presentada por Ramos-Figueroa et al. (2021), se han diseñado siete operadores de mutación diferentes, dentro de los cuales, seis de ellos pueden ser aplicados al problema de segmentación de imágenes. A su vez, estos operadores se clasifican de acuerdo con su enfoque, ya sea en elementos o en grupos. Dentro de los de los operadores orientados a elementos, se encuentran los siguientes: Swap, Insertion, Item elimination. Mientras que para los operadores orientados a grupos existen los siguientes: Elimination, Creation y Merge and split. Dentro de estos, destaca la popularidad el operador de Elimination, el cual ha sido ampliamente utilizado para resolver distintos problemas, donde la única diferencia en la implementación de este operador radica en la heurística utilizada para elegir los grupos que se eliminarán. A continuación, se describen los seis operadores de mutación mencionados anteriormente

para representación basada en grupos. Se considera un ejemplo en el que se tienen 16 objetos que se acomodan en 4 grupos. Estos ejemplos son ilustrados en las Figuras 5.1 y 5.2.

## 5.1. Operadores con enfoque en los elementos

Los operadores de mutación con este enfoque trabajan a nivel de elementos, es decir, toman elementos de un grupo para producir cambios. Los operadores de mutación que se encuentran en esta categoría y que pueden ser aplicados al problema de la segmentación de imágenes son los siguientes: Swap, Insertion e Item elimination. Estos se describen a continuación.

### 5.1.1. Swap

El primer operador de mutación enfocado en los elementos es Swap. Este operador realiza intercambios de elementos entre dos grupos. La forma en la que modifica una solución es la siguiente: se toman dos grupos $G_A$ y $G_B$ de forma completamente aleatoria o bajo algún criterio dependiente del problema a resolver. Una vez que se tienen los dos grupos seleccionados, se realiza un intercambio de elementos entre estos dos grupos. La cantidad de elementos que se van a intercambiar también depende del problema o puede ser establecido algún esquema clásico, como un intercambio 1-1, 2-2, etc. En el ejemplo de la Figura 5.1, se parte de la solución $S$, en la cual se seleccionan los grupos 1 y 3 (con color de fondo en azul) para realizar el intercambio. Para el primer grupo se seleccionan los elementos 1 y 5, mientras que para el segundo grupo se seleccionan los elementos 10, 16 y 9 (marcados en rojo). Estos son los elementos que se van a intercambiar entre estos dos grupos, es decir, los elementos 1 y 5 ahora pasaran a ser parte del grupo 3, mientras que los elementos 10, 16 y 9 pertenecerán al grupo 1. En la solución que lleva el nombre de $S_m$, se puede observar el efecto de realizar dicho intercambio.

Para el problema abordado en esta tesis, realizar este intercambio no produce soluciones infactibles, no hay elementos que falten en la solución ni grupos faltantes. Sin embargo, para problemas donde exista una capacidad de llenado en los grupos, siempre se debe verificar

que el intercambio sea factible, es decir, que no viole las restricciones de capacidad de los grupos involucrados en el intercambio. En el caso del problema de segmentación de imágenes en RGB, este intercambio puede ser realizado sin ningún inconveniente y no es necesario verificar que dicho intercambio sea factible o no, pues los grupos no tienen ninguna restricción de capacidad. En el Algoritmo 3 se muestra el pseudocódigo para este operador adaptado al problema de la segmentación de imágenes en RGB.

**Algoritmo 3:** Operador de mutación Swap.

**Entrada:** $S$ una solución, $percentage\_value$ el porcentaje de píxeles a seleccionar de cada grupo.

**Salida:** $S_m$ una solución mutada.

1 Crear una copia de $S$ llamada $S_m$.
2 Seleccionar un grupo $G_a$ de forma aleatoria de $S_m$.
3 Seleccionar otro grupo $G_b$ de forma aleatoria de $S_m$.
4 $n\_a = |S_m[G_a]| \times percentage\_value$.
5 $n\_b = |S_m[G_b]| \times percentage\_value$.
6 Seleccionar aleatoriamente $n\_a$ píxeles del grupo $G_a$.
7 Seleccionar aleatoriamente $n\_b$ píxeles del grupo $G_b$.
8 Intercambiar píxeles seleccionados entre los grupos $G_a$ y $G_b$.

### 5.1.2. Insertion

El segundo operador que trabaja a nivel de elementos es el de Insertion. Este operador elimina elementos de un grupo para reinsertarlos mediante alguna heurística. Su funcionamiento es el siguiente: primero se selecciona un grupo $G$, de este grupo se van a elegir $n$ elementos que posteriormente serán eliminados de dicho grupo. La selección del grupo, así como de los elementos a eliminar puede ser completamente aleatoria, pero también es válido realizar ajustes de acuerdo con las características del problema para saber cuántos elementos se van a eliminar y bajo qué criterio se seleccionarán. Retomando la Figura 5.1, en el ejemplo de mutación con el operador de Insertion, se parte de la solución $S$, de esa solución se elige el grupo 3 (con color de fondo en azul) para eliminar 3 elementos de este, los cuales son los elementos 6, 10 y 9 (marcados en rojo). Como se puede observar, este procedimiento hace que se tengan elementos faltantes en las soluciones, los cuales pueden observarse en la lista llamada "Perdidos". Estos elementos deben ser reinsertados en la

solución utilizando alguna heurística de acomodo, como BF. Una vez que los elementos perdidos han sido insertados en la solución nuevamente, la solución mutada que lleva el nombre de $S_m$ queda con los 3 elementos perteneciendo a los grupos 1 y 2, en lugar del grupo 3, que era al que originalmente pertenecían.

Este operador de mutación requiere alguna heurística de reparación, pues las soluciones que produce al remover elementos de un grupo son infactibles. Para el caso del problema de segmentación de imágenes, este operador puede ser utilizado en conjunto con la heurística de reparación BF. El Algoritmo 4 muestra este operador de mutación adaptado al problema de la segmentación de imágenes en RGB.

**Algoritmo 4:** Operador de mutación Insertion.

**Entrada:** $S$ una solución, $percentage\_value$ el porcentaje de píxeles a seleccionar del grupo.
**Salida:** $S_m$ una solución mutada.

1 Crear una copia de $S$ llamada $S_m$.
2 Seleccionar un grupo $G$ de forma aleatoria de $S_m$.
3 $n = |S_m[G_a]| \times percentage\_value$.
4 Quitar $n$ píxeles del grupo $G$ de forma aleatoria.
5 Reparar $S_m$.

### 5.1.3. Item elimination

El tercer operador que trabaja con un enfoque en elementos, es el de Item elimination. Este operador asigna una probabilidad a cada elemento para decidir si se elimina o no del grupo al que pertenece. Su funcionamiento es el siguiente: se genera un vector de tamaño del número de elementos en la solución, para cada posición se asigna un valor aleatorio con distribución uniforme entre 0 y 1. Después se establece un umbral $t$ que permite definir el límite para decidir su un elemento es eliminado de la solución o no. Aquellos elementos con una probabilidad menor que el umbral son removidos de la solución. En el ejemplo de la Figura 5.1 para este operador, se elige un umbral $t = 0.5$. Se tiene una lista de probabilidades de tamaño 16, pues el ejemplo consiste de 16 elementos, cada uno con su valor entre 0 y 1 elegido aleatoriamente. Los elementos con probabilidad menor a la del umbral son marcados con color de fondo naranja, lo que indica que estos elementos serán

eliminados de la solución. En la solución $S$ estos elementos son marcados en rojo, los cuales son 4, 5, 12, 9 y 13. Una vez que son removidos de la solución, pasan a pertenecer a la lista de elementos perdidos, por lo que es necesario reinsertarlos utilizando alguna heurística.

Para este operador, la configuración del parámetro $t$ es de suma importancia, pues una probabilidad muy baja, tendría un efecto muy pequeño en la solución modificada, mientras que un valor muy alto, sería más disruptivo, pues la probabilidad de eliminar un objeto de la solución es mayor. Este parámetro puede ser usado para beneficiar la mutación y podría contribuir al problema que se busca resolver si se elige un valor adecuado. En el caso del problema de segmentación de imágenes en RGB, este operador puede ser implementado y requiere que las soluciones sean reparadas con la heurística de reinserción de píxeles perdidos. El pseudocódigo para este operador adaptado al presente problema se muestra en el Algoritmo 5.

**Algoritmo 5:** Operador de mutación Item elimination.

**Entrada:** $S$ una solución, $t$ umbral de eliminación.
**Salida:** $S_m$ una solución mutada.

```
1 Crear una copia de S llamada S_m.
2 para píxel ∈ S_m hacer
3     si random_uniform() < t entonces
4         Eliminar píxel de la solución S_m.
5     fin
6 fin
7 Reparar la solución S_m.
```

## 5.2. Operadores con enfoque en grupos

Dentro de esta categoría se encuentras los siguientes operadores de mutación: Elimination, Creation y Merge and split. En estos operadores el grupo entero es eliminado o perturbado, por lo que cualquier modificación implica mover todos los elementos dentro del mismo grupo. A continuación se describen estos operadores que pueden ser aplicados al problema de la segmentación de imágenes.

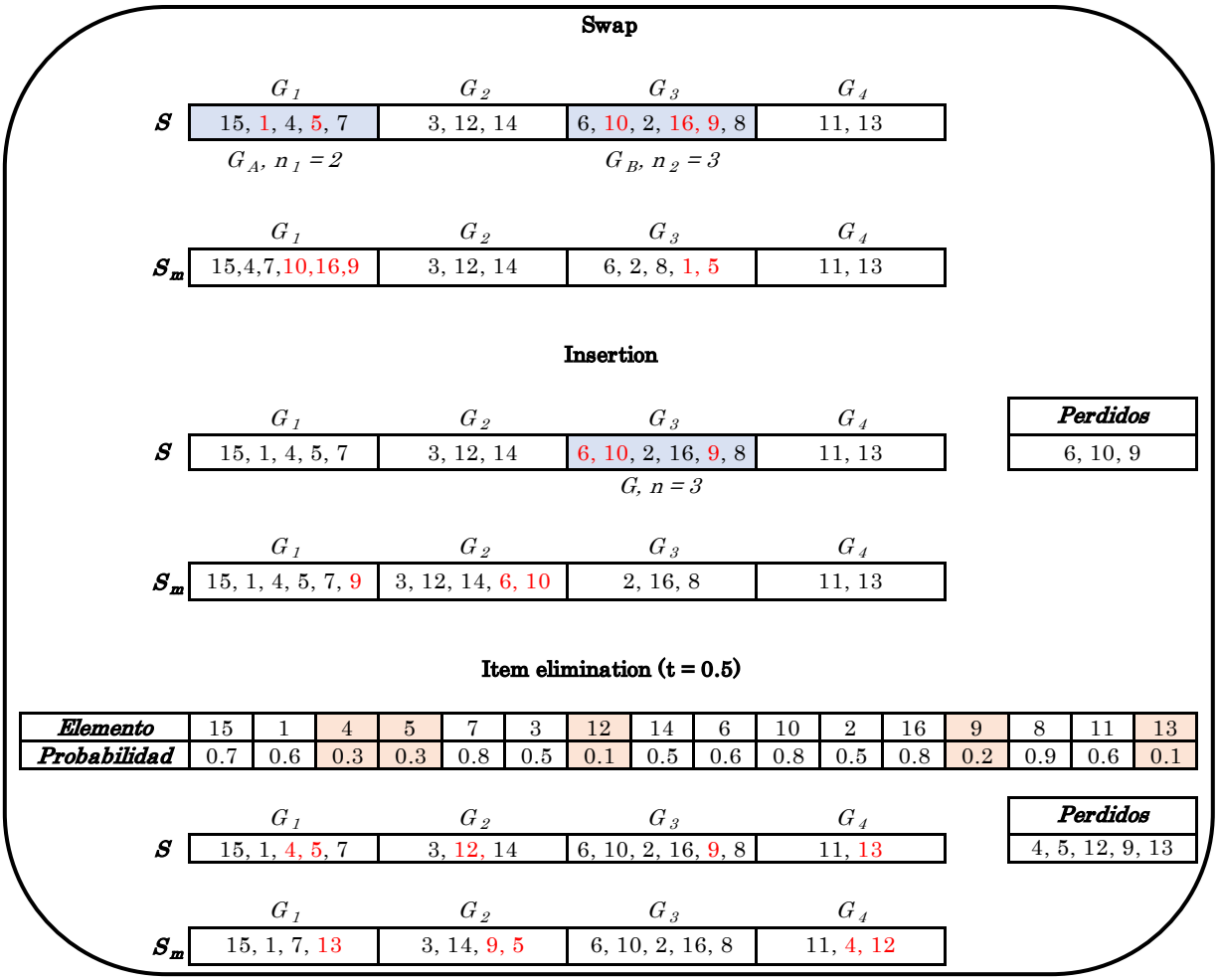



Figura 5.1: Operadores de mutación con enfoque en elementos.

### 5.2.1. Elimination

Los operadores de mutación con enfoque de grupos, se ilustran en la Figura 5.2. El primero de ellos es el de Elimination. Este operador elimina una cantidad de grupos de la solución. Su funcionamiento es el siguiente: se determina una cantidad de grupos que serán eliminados. Después se elige cada grupo $G$ y este es eliminado de la solución. En el ejemplo de la Figura 5.2 para este operador, la cantidad de grupos a eliminar es de 1, de forma que se elige el último grupo y sus elementos (marcados en rojo) son eliminados de la solución $S$. Los elementos faltantes que se encuentran en la lista "Perdidos" deben ser reinsertados bajo alguna heurística. Pero no solo se tienen elementos perdidos, sino que ahora el número de grupos que tiene la solución es menor al establecido, es decir, también

hay grupos faltantes y en este caso, falta un grupo en la solución para que se considere factible. Para explicar la reparación de esta solución, se presenta el siguiente operador de mutación.

La selección del número de grupos a eliminar y cuales se eliminarán puede hacerse de forma aleatoria o también puede usarse conocimiento del problema para usar algún criterio de eliminación. Para el problema de segmentación de imágenes, el uso del operador de Elimination, como se mencionó en el capítulo anterior, realiza la eliminación con base en un porcentaje de grupos y selecciona los grupos con peor valor de calidad. En el Algoritmo 6 se muestra el pseudocódigo de este operador adaptado al problema de la segmentación de imágenes.

**Algoritmo 6:** Operador de mutación Elimination.

**Entrada:** $S$ una solución, $k$ número de segmentos, $percentage\_value$ porcentaje de grupos a eliminar.
**Salida:** $S_m$ una solución mutada.

1 Crear una copia de $S$ llamada $S_m$.
2 $k\_elim = \lceil k \times percentage\_value \rceil$.
3 Eliminar $k\_elim$ grupos de la solución $S_m$.
4 Reparar la solución $S_m$.

### 5.2.2. Creation

El operador de Creation también trabaja con un enfoque de grupos y, como su nombre sugiere, se encarga de crear grupos. Su funcionamiento es el siguiente: dada una solución con una cantidad de grupos, el operador crea un nuevo grupo y le asigna elementos de forma completamente aleatoria o siguiendo alguna heurística del problema. Volviendo al ejemplo de la Figura 5.2 para este operador, se toma la solución mutada del ejemplo anterior, en el cual se aplicó el operador de mutación Elimination para eliminar un grupo de la solución, dejando a esta con tres grupos en lugar de los cuatro establecidos y los elementos 11 y 13 aparecen como elementos perdidos. A partir de esta solución, se aplica el operador de mutación Creation para crear un nuevo grupo, de forma que la solución ahora tendrá cuatro grupos, justo como se había establecido en el problema de ejemplo.

Posteriormente, a este nuevo grupo se le asignan los elementos perdidos utilizando alguna heurística de reinserción, produciendo la solución llamada $S_m$. Esta solución es factible, pues ahora tiene cuatro grupos y no hay elementos perdidos.

Como se puede observar, resulta conveniente usar los operadores de Elimination y Creation en conjunto para eliminar grupos y posteriormente reparar la solución cuando es necesario mantener un número fijo de grupos. En el caso del problema de segmentación de imágenes, esta combinación de operadores resulta muy útil, pues una vez que se aplica el operador de eliminación, además de tener píxeles perdidos, también es necesario reparar las soluciones creando los grupos faltantes. El pseudocódigo para este operador se muestra en el Algoritmo 7.

**Algoritmo 7:** Operador de mutación Creation.

**Entrada:** $S$ una solución, $k$ número de segmentos, $ms\_pxs$ píxeles perdidos.
**Salida:** $S_m$ una solución mutada.

```
1 Crear una copia de S llamada S_m.
2 si |S_m| < k entonces
3     Crear un nuevo grupo en la solución con un píxel perdido de ms_pxs tomado
        de forma aleatoria.
4 fin
5 Reparar la solución S_m.
```

### 5.2.3. Merge and split

El último operador de este tipo es el de Merge and split. Este operador se encarga de fusionar dos grupos, y dividir un grupo en dos. Su funcionamiento está basado en dos fases. En la primera fase, se toman un par de grupos $G_A$ y $G_B$ y estos son unidos, formando un nuevo grupo que contiene los elementos de ambos grupos. En la segunda fase, se selecciona un grupo de la solución y se divide en dos grupos nuevos. La elección de dichos grupos puede ser aleatoria o también puede depender del problema. Continuando con el ejemplo de la Figura 5.2, se parte de la solución $S$, de la cual los grupos 1 y 2 (con color de fondo azul) fueron elegidos de forma aleatoria. Una vez que se tienen los dos grupos, estos son unidos, formando un nuevo grupo 1. En esta primera fase, la solución ahora consta de tres grupos, en lugar de cuatro. En la siguiente fase, el grupo 1 es elegido de forma aleatoria y

se realiza la división del mismo, formando dos nuevos grupos, por lo que la solución final tendrá cuatro grupos, pero se puede observar que el conjunto de elementos que pertenecen a los primeros dos grupos son distintos en comparación con la solución inicial. A pesar de que la elección del grupo se realiza de forma aleatoria, existe una consideración con respecto al número de píxeles que tiene un grupo para ser considerado como una opción para dividir. De forma que los grupos que tienen un píxel no son considerados, pues no es posible dividir un grupo con un solo elemento. Adicionalmente, el punto de corte para dividir el grupo, se elige de forma aleatoria, evitando que al dividir el grupo se obtengan grupos vacíos. Cuando el grupo contiene dos píxeles, el único punto de corte es entre estos dos píxeles, por lo que se formarían dos grupos con un píxel cada uno.

A pesar de que en la primera fase de este operador se pierde un grupo, con la segunda fase ese grupo faltante se restablece, de forma que este operador de mutación mantiene el número de grupos constante y tampoco es necesario reparar las soluciones, pues no es posible tener elementos perdidos. Para el problema de la segmentación de imágenes, este operador resulta interesante pues mantiene el número de segmentos constante y no es necesario realizar una reparación que consume demasiados recursos. El Algoritmo 8 muestra el pseudocódigo para este operador adaptado a la segmentación de imágenes.

**Algoritmo 8:** Operador de mutación Merge and split.

**Entrada:** $S$ una solución.
**Salida:** $S_m$ una solución mutada.

1 Crear una copia de $S$ llamada $S_m$.
2 Seleccionar un grupo $G_a$ de forma aleatoria de $S_m$.
3 Seleccionar otro grupo $G_b$ de forma aleatoria de $S_m$.
4 Unir los grupos $G_a$ y $G_b$.
5 Seleccionar aleatoriamente un grupo $G$ de $S_m$ con más de un píxel.
6 Dividir el grupo $G$ en dos.

La mutación en un algoritmo genético y en un GGA, juega un papel importante. Para los GGAs, los operadores de mutación pueden producir cambios a nivel de los elementos que contienen las soluciones, o a nivel de gen, es decir, de los grupos que contienen. La disrupción de estos cambios depende de los parámetros propios de cada operador, como el número de grupos involucrados o el número de elementos. El control de los paráme-

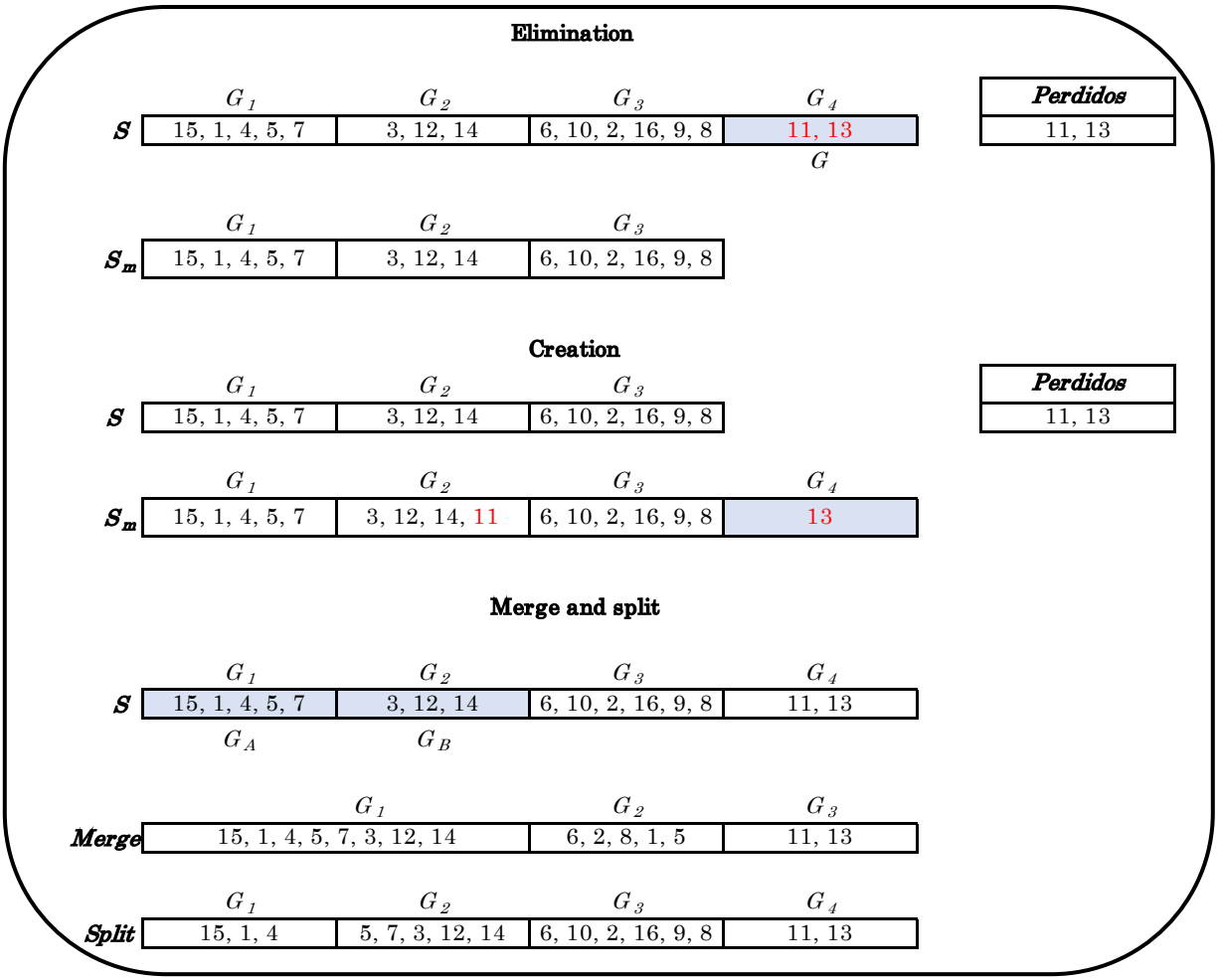



Figura 5.2: Operadores de mutación para representación basada en grupos con enfoque en grupos.

tros puede jugar un papel importante en favor o en contra del desempeño del algoritmo dependiendo del problema a resolver, por lo que es esencial elegir estos parámetros adecuadamente. En el siguiente capítulo se utiliza una metodología para evaluar el desempeño de los operadores de mutación vistos en las secciones anteriores y elegir los parámetros más adecuados para estos.

# Capítulo 6

# Diseño experimental del operador de mutación

En este capítulo se aborda el diseño experimental para probar los distintos operadores de mutación descritos en el capítulo anterior. Los experimentos fueron divididos en 3 fases y se utilizaron únicamente las 100 imágenes de control con una semilla por imagen para reducir el costo computacional de estos experimentos. La razón por la que se utilizaron las imágenes de control es porque se conoce el óptimo global de estas. Este óptimo global se conoce porque el número de colores diferentes que contiene cada imagen es conocido y este equivale al número de segmentos, por lo que es posible agrupar cada uno de los píxeles de acuerdo con su color, obteniendo así una distancia intra-cluster de 0. Para hacer una comparación justa, la semilla utilizada fue la misma durante todos los experimentos. Esto significa que en todas las ejecuciones del algoritmo se partió siempre de la misma población inicial. La primera fase consintió en utilizar distintos porcentajes de selección de grupos o elementos (según el tipo de operador) y seleccionar el mejor operador con enfoque en elementos y el mejor con enfoque en grupos. En la segunda fase se probaron diferentes estrategias de selección con los dos mejores operadores, esto con la finalidad de elegir la mejor estrategia para cada operador. En las primeras dos fases únicamente se utilizó el operador de mutación con el fin de observar si el operador es capaz de encontrar los óptimos globales de forma aislada, sin ayuda de los demás elementos del GGA. Esto significa que toda la población era mutada en cada generación, por lo que no se utilizó el

operador de cruza, ni los esquemas de selección, ni las estrategias de reemplazo del GGA-CGT-RGB-IS. Finalmente, en la Fase 3 de los experimentos se probó el mejor operador por enfoque con el algoritmo completo para observar si los elementos que ayudaban a obtener mejores resultados se conservaban. A partir de estos resultados, se eligió el operador que mejores resultados proporcionaba en términos de la distancia intra-cluster y del número de generaciones que le tomó alcanzar el óptimo global. A continuación se describen con mayor detalle las fases de los experimentos.

## 6.1. Fase 1: porcentajes de selección

Como se recordará, los operadores se pueden clasificar como operadores basados en elementos o en grupos. Los que se basan en elementos, seleccionan un grupo y manipulan una cantidad de elementos dentro de este grupo, por lo que un porcentaje de selección nos indica qué proporción de elementos de un grupo se van a seleccionar para aplicar la mutación. Mientras que los operadores que se enfocan en grupos, seleccionan un grupo o una cantidad de grupos, pero si se realizan modificaciones, estas son para todo el grupo y no solo una cantidad de elementos dentro del grupo. En este caso, un porcentaje de selección nos indica la proporción de grupos que se van a seleccionar de la solución.

La mayoría de los operadores de mutación revisados tienen este parámetro en común. Por lo que en la primera fase se analiza el comportamiento de los 5 operadores de mutación con diferentes porcentajes de selección. La estrategia consiste en segmentar el banco de imágenes de control usando los siguientes porcentajes de selección en los operadores: 20 %, 40 %, 60 % y 80 %. Cabe señalar que en el caso del operador de Item elimination, este porcentaje es interpretado como el umbral $t$. Y en el caso del operador Merge and split, no existe algún parámetro similar que se pueda controlar. Para esta fase, la selección de los elementos o grupos se realizó de forma completamente aleatoria.

En la Tabla 6.1 se muestran los resultados obtenidos en esta fase, marcando en negritas los mejores resultados (mayor porcentaje de casos exitosos). En la primera columna se muestran las dos categorías de operadores: los que se enfocan en elementos y los de grupos. En la segunda columna se muestran las configuraciones de los distintos operadores de

mutación probados. Para cada operador, excepto Merge and split, se tienen cuatro valores de porcentaje de selección de elementos o grupos. En el caso de Item elimination, estos porcentajes corresponden al valor del umbral. La tercera columna muestra el porcentaje de casos exitosos, donde un caso exitoso indica que el operador encontró la solución óptima, con una distancia intra-cluster de 0. Mientras que un caso no exitoso es aquel en el que el algoritmo no encontró el óptimo global en las 100 generaciones. Para estos experimentos, el porcentaje de casos exitosos resulta de interés para evaluar qué tan bueno es el desempeño del operador de mutación.

A partir de estos resultados se puede observar que el operador de Swap no consiguió alcanzar el óptimo global con ninguna configuración. Esto puede deberse a que los cambios que se realizan con el operador no son tan disruptivos, por lo que la solución es apenas modificada. Y como se recordará, la solución es inicializada completamente aleatoria, de forma que las soluciones mutadas quedan muy parecidas a las de la población inicial. Con este operador, las soluciones son factibles, por lo que no es necesario repararlas. Esto representa tanto una ventaja en costo computacional, como una desventaja en la calidad de las soluciones. Ya que al no aplicar la función de reparación, que incluye la heurística BF, las soluciones generadas con el operador de mutación son completamente aleatorias.

Algo similar ocurre con el operador Merge and split, con el cual no se obtuvo ningún caso exitoso. Pese a que el costo computacional de dicho operador es mucho más bajo, al no requerir reparar las soluciones, esto tampoco ayuda mucho a que se encuentren mejores soluciones, pues el proceso es más aleatorio.

Con el operador de Insertion, se empezaron a tener casos exitosos a partir del porcentaje de selección de píxeles de 40 %. Este operador sigue siendo poco disruptivo, pues solo se selecciona un grupo de la solución para quitar el porcentaje de elementos indicado. Con este operador, las soluciones mutadas no son factibles, por lo que sí se hace uso de la heurística de reparación, donde únicamente se presenta el caso en el que faltan píxeles por asignar a algún segmento. Dada esta condición, la función de reparación emplea la heurística BF para asignar los píxeles al segmento más cercano. Es por eso que al aumentar el porcentaje de selección de elementos, el número de casos exitosos aumenta, pues del grupo seleccionado, se están seleccionando más píxeles, lo que presenta una oportunidad para acomodarlos

mejor.

En esta fase, los operadores que mejores resultados obtuvieron fueron Item elimination y Elimination. Estos dos operadores son los más disruptivos. En el caso de Elimination, contrario a las tendencias observadas para los demás operadores, parece que eliminar menos grupos es más conveniente. Esto también presenta una ventaja computacional, pues entre menos grupos, la función de reparación se utiliza menos veces. El operador que más llama la atención es Item elimination, con el cual se obtienen la mayoría de casos exitosos con tan solo seleccionar un umbral del 40 %, aunque desde el 20 % se observan buenos resultados con casi el 100 % de casos exitosos.

Tabla 6.1: Resultados de la Fase 1 para la segmentación del banco de imágenes de control con una semilla por imagen.

| **Enfoque** | **Operador** | **% Exitosos** |
|---|---|---|
| Elementos | Swap 20 % | 0 |
| | Swap 40 % | 0 |
| | Swap 60 % | 0 |
| | Swap 80 % | 0 |
| | Insertion 20 % | 0 |
| | Insertion 40 % | 1 |
| | Insertion 60 % | 6 |
| | Insertion 80 % | 12 |
| | **Item elimination 20 %** | 98 |
| | **Item elimination 40 %** | **100** |
| | **Item elimination 60 %** | **100** |
| | **Item elimination 80 %** | **100** |
| Grupos | **Elimination 20 %** | **19** |
| | Elimination 40 % | 16 |
| | Elimination 60 % | 16 |
| | Elimination 80 % | 17 |
| | Merge and split | 0 |

En la Tabla 6.2 se observan las estadísticas de los resultados obtenidos para el operador de Item elimination. Estas estadísticas se muestran en términos del número de generaciones que el algoritmo necesitó para alcanzar el óptimo global. En la primera columna se tienen las 5 estadísticas que se analizan. En las siguientes columnas se tienen las configuraciones del operador. Estos porcentajes corresponden al valor del umbral de eliminación y para cada columna se muestran las estadísticas mencionadas. Los valores se presentan

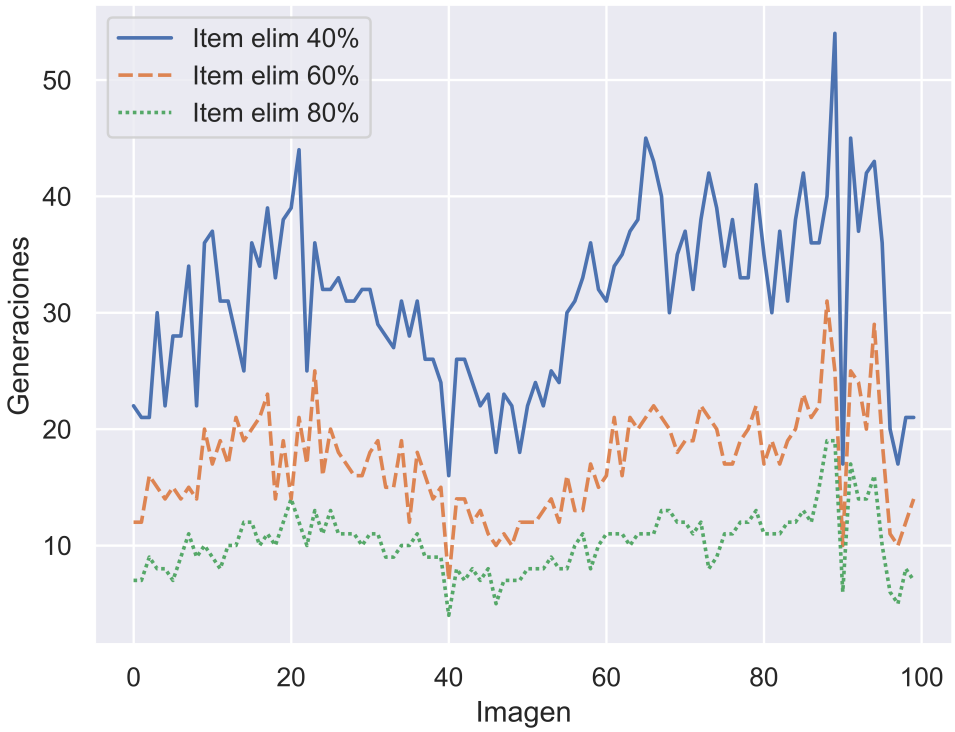



Figura 6.1: Comportamiento del operador Item elimination con distintos valores de umbral.

en términos del número de generaciones tomadas para alcanzar el óptimo global.

A partir de esta tabla, se puede observar que lo que más conviene para que el algoritmo converja rápidamente, de acuerdo con el número de generaciones, es aumentar el umbral al 80 %, pues el algoritmo alcanza el óptimo en 10.22 generaciones en promedio. Esto también lo hace más consistente, pues la desviación estándar disminuye.

En la Figura 6.1 se muestra una gráfica de la cantidad de generaciones por imagen tomadas para alcanzar el óptimo global para los distintos valores de umbral establecidos. En el eje $x$ se muestran las imágenes del banco de imágenes de control y en el eje $y$ se muestra el número de generaciones que le tomó a cada operador para alcanzar el óptimo global. En dicha figura lo que se observa es que un umbral del 80 % garantiza que el algoritmo converja rápidamente, pues la línea es la que más cercana al cero se encuentra. Por lo que el operador que mejores resultados proporciona en la categoría de enfoque de elementos es el de Item elimination con un umbral $t = 0.80$. Este comportamiento resulta interesante, pues los píxeles que se eliminan de la solución pueden pertenecer a cualquier segmento. De forma que los segmentos están en constante reacomodo y cada vez los píxeles

se empiezan a asignar de una mejor forma hasta que se encuentra una buena distribución en los segmentos gracias a la heurística BF de la función de reparación, que garantiza que cada vez que hay un píxel sin asignar, este sea asignado al mejor grupo posible.

Tabla 6.2: Número de generaciones de los mejores resultados del operador Item elimination con la segmentación del banco de imágenes de control con una semilla.

| | **Item elimination 40 %** | **Item elimination 60 %** | **Item elimination 80 %** |
|---|---|---|---|
| *Media* | 31.23 | 17.09 | 10.22 |
| *Desv. Est.* | 7.48 | 4.32 | 2.67 |
| *Mejor* | 16 | 7 | 4 |
| *Mediana* | 31.50 | 17.00 | 10.00 |
| *Peor* | 54 | 31 | 19 |

Tabla 6.3: Número de generaciones de los mejores resultados del operador Elimination con la segmentación del banco de imágenes de control con una semilla.

| | **Elimination 20 %** | **Elimination 40 %** | **Elimination 60 %** | **Elimination 80 %** |
|---|---|---|---|---|
| *Media* | 27.84 | 11.38 | 12.5 | 16.47 |
| *Desv. Est.* | 30.73 | 13.77 | 17.9 | 24.89 |
| *Mejor* | 4 | 3 | 2 | 2 |
| *Mediana* | 4 | 6 | 6 | 5 |
| *Peor* | 96 | 54 | 70 | 92 |

En la Tabla 6.3 se observan las estadísticas de los mejores resultados obtenidos con el operador de Elimination. En la primera columna se tienen las 5 estadísticas que se analizan. En las siguientes columnas se tienen las configuraciones del operador. Estos porcentajes corresponden al valor del umbral de eliminación y para cada columna se muestran los valores para cada una de las estadísticas mencionadas. Nuevamente, estos resultados se muestran en términos del número de generaciones tomadas para alcanzar el óptimo global.

En la Figura 6.2 se observa el comportamiento del operador con los porcentajes de eliminación de grupos con los que se probó. En el eje $x$ se muestran las imágenes del banco de imágenes de control y en el eje $y$ se muestra el número de generaciones que le tomó a cada operador para alcanzar el óptimo global. En esta figura se grafica el número de generaciones que le toma al operador de Elimination en alcanzar el óptimo global por imagen para cada configuración del operador. Se observa que el comportamiento no es muy consistente a lo largo del banco de prueba. A partir de esta figura y de las estadísticas de la tabla anterior, se puede observar que el porcentaje de eliminación de grupos del

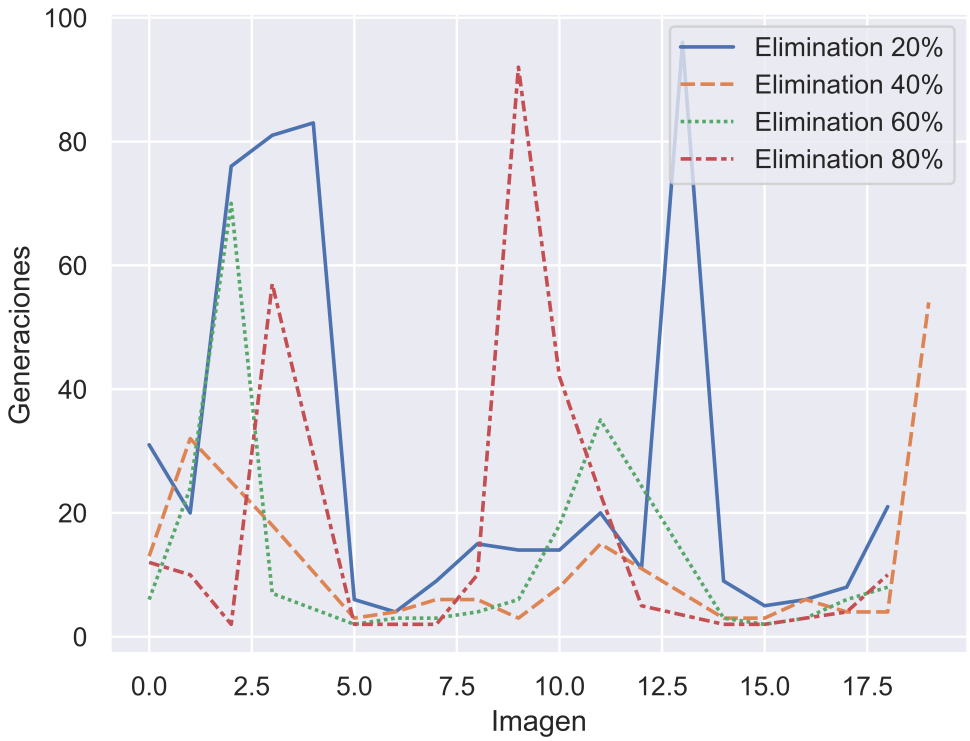



Figura 6.2: Comportamiento del operador Elimination con distintos porcentajes de eliminación de grupos.

40 % parece hacer que el operador sea más consistente, sin embargo, un porcentaje del 20 % proporciona más casos exitosos en esta fase. Por esta razón, se decide seleccionar el operador de Elimination con 20 % de selección de grupos como mejor operador con enfoque en grupos.

## 6.2. Fase 2: estrategias de selección

Una vez que se seleccionó el mejor operador por categoría, se realizaron pruebas de estrategias de selección para estos dos operadores. La idea es elegir la estrategia que permita obtener mejores resultados. Siguiendo la metodología empleada por Ramos-Figueroa et al. (2023), se probaron las siguientes estrategias de selección de grupos: *aleatorio, peor, peor-mejor* y *peor-aleatorio*. Para cada una de estas estrategias, se volvieron a utilizar los porcentajes de selección de grupos que se utilizaron en la Fase 1. De forma que cada estrategia se seleccionaron el 20 %, 40 %, 60 % y 80 %. Cabe destacar que estas estrategias

aplican únicamente para el operador de Elimination, pues el operador de Item elimination trabaja con todo el conjunto de píxeles, asignándoles probabilidades de ser eliminados. De forma que los resultados de Item elimination son los mismos que los de la Fase 1, con un umbral $t = 0.8$.

Como su nombre lo indica, la estrategias de selección de grupos se encargan de elegir el porcentaje de los grupos según su criterio. En este caso, con el operador de eliminación, cuando el porcentaje de selección de grupos es del 20 %, la primera estrategia elimina el 20 % de los grupos seleccionados de forma aleatoria. Mientras que la segunda elimina el 20 % de los peores grupos. En el caso de las siguientes, una parte corresponde a peores grupos, mientras que el resto pueden ser grupos con la mejor calidad, o pueden ser grupos elegidos de forma aleatoria. En la medida de lo posible, la distribución de los grupos a seleccionar es equitativa. Por ejemplo, si se usa la estrategia de selección *peor-aleatorio* para una imagen con 20 segmentos con selección del 20 %, dicho porcentaje de segmentos a eliminar correspondería a 4 segmentos. De estos 4 segmentos, la mitad serían los segmentos con mayor distancia intra-cluster y la otra mitad serían segmentos elegidos de forma aleatoria. Desde luego que los segmentos elegidos no se pueden repetir. Uno de los inconvenientes con las últimas dos estrategias cuando el porcentaje de selección de grupos es del 20 % es que cuando el número de segmentos es menor que 5, esta estrategia se comporta simplemente como la que selecciona los peores grupos, pues el 20 % de segmentos corresponde solo a 1 segmento.

En la Tabla 6.4 se muestran los resultados de la Fase 2. En la primera columna se tienen las estrategias de selección de grupos con los cuatro diferentes porcentajes de selección de la Fase 1. En la segunda columna se muestra el porcentaje de imágenes en las que la segmentación alcanzó el óptimo global, indicando en negritas el mejor resultado encontrado. De acuerdo con esta tabla, la estrategia de selección que mejores resultados proporciona es la de *peor-aleatorio*, lo cual coincide con lo encontrado por Ramos-Figueroa et al. (2023) en el diseño de un operador de mutación para el problema de $R||C_{max}$. Como se había observado en la fase anterior, la selección completamente aleatoria no proporciona buenos resultados. Y las siguientes tres estrategias tienen resultados similares, donde se puede observar que entre menor porcentaje de selección, se obtienen más casos exitosos.

Tabla 6.4: Resultados de la Fase 2 con las estrategias de selección de grupos para el operador Elimination con el banco de imágenes de control utilizando una semilla por imagen.

| **Estrategia de selección** | **% Exitosos** |
|---|---|
| *Aleatorio 20 %* | 19 |
| *Aleatorio 40 %* | 16 |
| *Aleatorio 60 %* | 16 |
| *Aleatorio 80 %* | 17 |
| *Peor 20 %* | 40 |
| *Peor 40 %* | 25 |
| *Peor 60 %* | 23 |
| *Peor 80 %* | 16 |
| *Peor-mejor 20 %* | 41 |
| *Peor-mejor 40 %* | 27 |
| *Peor-mejor 60 %* | 21 |
| *Peor-mejor 80 %* | 14 |
| ***Peor-aleatorio 20 %*** | **45** |
| *Peor-aleatorio 40 %* | 25 |
| *Peor-aleatorio 60 %* | 23 |
| *Peor-aleatorio 80 %* | 16 |

Tabla 6.5: Número de generaciones de las mejores estrategias de selección de grupos para el operador de Elimination con 20 % de eliminación de grupos.

| | ***Aleatorio*** | ***Peor*** | ***Peor-mejor*** | ***Peor-aleatorio*** |
|---|---|---|---|---|
| *Media* | 27.84 | 19.20 | 25.68 | 26.47 |
| *Desv. Est.* | 30.73 | 16.62 | 22.45 | 26.61 |
| *Mejor* | 4 | 4 | 5 | 4 |
| *Mediana* | 14 | 14 | 17 | 14 |
| *Peor* | 96 | 72 | 100 | 97 |

En general, cualquiera de estas tres últimas estrategias proporciona resultados muy similares. En la Tabla 6.5 se pueden observar más detalles sobre estas estrategias con un porcentaje de selección de grupos del 40 %. En la primera columna se muestran las mismas estadísticas que se han analizado. En las siguientes columnas se muestran dichas estadísticas para cada estrategia de selección de grupos. Los valores se presentan en términos del número de generaciones tomadas para alcanzar el óptimo global. A partir de esta tabla se puede observar que la estrategia de *peor* parece ser la más consistente, pues tiene la menor desviación estándar de las cuatro estrategias probadas. Sin embargo, la estrategia de *peor-aleatorio* obtiene la mayor cantidad de casos exitosos, por lo que es seleccionada como la

estrategia más adecuada para el operador de mutación en el problema de la segmentación de imágenes.

## 6.3. Fase 3: comparación del desempeño de los operadores en el GGA-CGT-RGB-IS

Finalmente, en la Fase 3 se utilizan los dos mejores operadores para este problema para ser añadidos al GGA-CGT-RGB-IS y observar su desempeño en conjunto con los demás elementos que lo conforman. Adicionalmente, se comparan los resultados con la versión inicial del GGA-CGT-RGB-IS, que utiliza el operador de mutación Elimination eliminando el 20 % de los peores grupos.

Tabla 6.6: Resultados de la Fase 3 comparando las nuevas versiones que incorporan los operadores de mutación propuestos con la versión inicial del GGA-CGT-RGB-IS.

| | **GGA-CGT-RGB-IS** | **GGA-CGT-RGB-IS+E-WR** | **GGA-CGT-RGB-IS+IE** |
|---|---|---|---|
| *Media* | 15.64 | 15.30 | **6.46** |
| *Desv. est.* | 15.50 | 12.52 | **2.01** |
| *Mejor* | **1** | **1** | 2 |
| *Mediana* | 13 | 13 | **7** |
| *Peor* | 97 | 77 | **11** |

En la Tabla 6.6 se muestran los resultados de la segmentación del banco de imágenes de control con las tres versiones del algoritmo con respecto al número de generaciones que le tomó a cada versión alcanzar el óptimo global. La primera columna presenta las estadísticas que se analizan. En las siguientes columnas se muestran tres versiones del GGA-CGT-RGB-IS: una que utiliza el operador de Item elimination (GGA-CGT-RGB-IS+IE), otra con el de Elimination con la estrategia de selección de grupos *peor - aleatorio* (GGA-CGT-RGB-IS+E-WR) y la versión inicial del algoritmo que utiliza el operador de Elimination con la estrategia de selección de grupos de *peor*. Los valores se presentan en términos del número de generaciones tomadas para alcanzar el óptimo global. Cabe destacar que con los tres algoritmos se consiguieron los 100 casos exitosos.

Con base en los resultados presentados en dicha tabla, se puede observar que la versión del GGA que utiliza el operador de Item elimination con umbral $t = 0.8$ es más consistente

y rápido que las otras dos versiones. Tanto con GGA-CGT-RGB-IS+E-WR como en la versión inicial, el comportamiento es muy similar, pues el operador es el mismo y solo cambia la estrategia de selección que, de acuerdo con los resultados de la Fase 2, también eran muy similares. La diferencia más significativa entre estas dos versiones está en el peor resultado obtenido, donde se puede ver que la versión que utiliza la estrategia de *peor-aleatoria* tiene un máximo número de generaciones de 77, lo cual representa 20 generaciones por debajo del peor de la versión inicial.

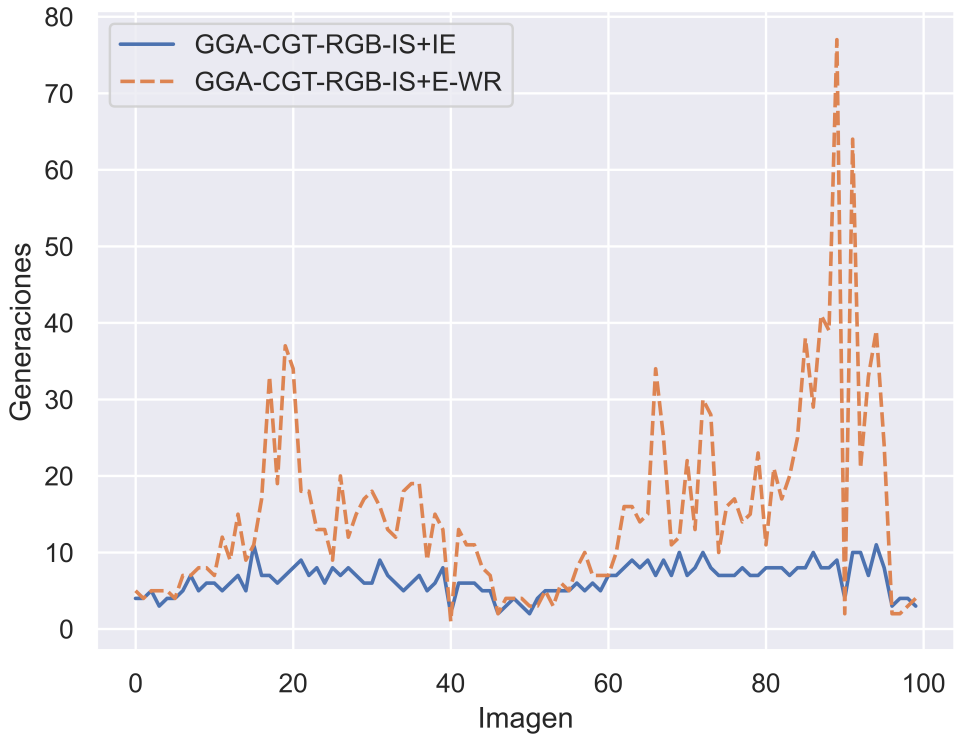



Figura 6.3: Desempeño de los operadores en el GGA-CGT-RGB-IS segmentando el banco de imágenes de control.

En la Figura 6.3 se observa el desempeño de las versiones del algoritmo que utilizan los dos operadores seleccionados en las fases anteriores. En el eje $x$ se muestran las 100 imágenes del banco de imágenes de control y en el eje $y$ se muestra el número de generaciones que le toma al algoritmo alcanzar el óptimo globla. Se omitieron los resultados de la versión inicial debido a que el comportamiento es muy similar al de la versión con Elimination. En esta figura se puede observar claramente que el operador de Item elimination permite al GGA-CGT-RGB-IS alcanzar un desempeño más robusto y eficiente, pues la línea se

mantiene cerca del eje $x$ en la mayoría de las imágenes, lo que indica que al algoritmo no le tomó muchas generaciones para alcanzar el óptimo global.

Con base en los resultados obtenidos en estos experimentos, se puede concluir que el operador que mejores resultados proporciona utilizando el banco de imágenes de control es el operador de Item elimination, pues es el más consistente y rápido para el banco de imágenes de control. El siguiente paso es evaluar la nueva versión del algoritmo con las 200 imágenes de prueba de BSDS500 y verificar que las características que hacen que este operador mejore el desempeño del algoritmo sean preservadas con las imágenes reales.

# Capítulo 7

# GGA-CGT-RGB-IS con Item elimination (GGA-CGT-RGB-IS+IE)

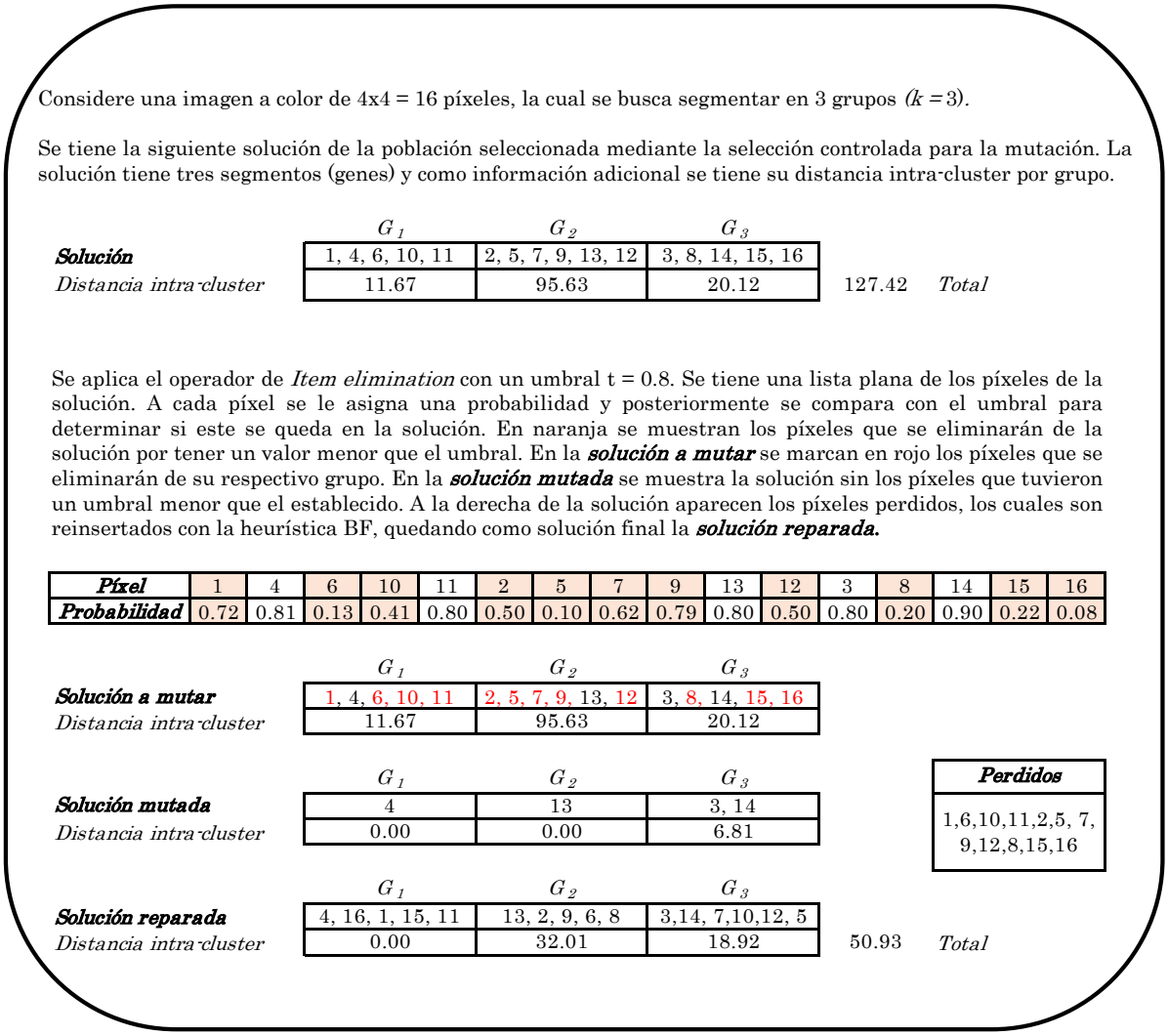
En este capítulo se muestran los resultados experimentales segmentando los dos conjuntos de imágenes disponibles: las 100 imágenes de control y las 200 imágenes de prueba de BSDS500 con 30 semillas por imagen. Se empleó una configuración de parámetros diferente a la del GGA-CGT y se realizaron pruebas estadísticas para validar las diferencias en el desempeño entre la versión inicial y la nueva. También se presenta un análisis del comportamiento algorítmico para comparar ambas versiones con base en sus gráficos de convergencia.

Con base en los resultados obtenidos en el capítulo anterior, se concluyó que el operador de mutación de Item elimination parece mejorar el desempeño del GGA-CGT-RGB-IS en términos de la distancia intra-cluster de las soluciones obtenidas. Pese a que también se diseñó un operador con enfoque de grupos basado en los resultados obtenidos con el banco de imágenes de control, este operador no proporcionó mejores resultados que los que se obtuvieron con Item elimination. Este último operador es más disruptivo, lo que permite reacomodar los píxeles durante el proceso de reparación de una mejor forma. Además, los píxeles que se eliminan de la solución provienen de diferentes grupos, de acuerdo con la probabilidad que se le asigna a cada elemento, lo que aumenta las posibilidades de mejorar la solución que cuando se enfoca en un porcentaje de grupos de la solución, como es el caso del operador de Elimination. Así, se propone una versión mejorada del algoritmo

llamada GGA-CGT-RGB-IS+IE, derivado del nombre de la versión inicial del algoritmo (GGA-CGT-RGB-IS) y el uso del operador de mutación Item elimination (IE).

El operador de mutación Item elimination fue usado por primera vez por Rossi et al. (2010) para el problema de programación de trabajos fijos. Este es un operador con enfoque en elementos que, en el contexto del problema de la segmentación de imágenes, asigna una probabilidad a cada píxel del conjunto de píxeles de la imagen y si este valor es menor que un umbral $t$ establecido, el píxel es eliminado del grupo al que pertenece. Debido a que se pierden píxeles, este operador produce soluciones infactibles, por lo que es necesario repararlas usando el método de reparación implementado para este problema. La reparación juega un papel importante en el éxito de este operador, pues cada vez que un píxel es eliminado de la solución, este es reinsertado de forma que la calidad de la solución mejora, pues el píxel se asigna a un segmento donde tiene una mayor similitud con los píxeles contenidos en dicho segmento.

Este operador no requiere una extensa configuración, pues el único parámetro a configurar es el umbral, que para este problema se estableció en $t = 0.8$. Comparado con la propuesta de Rossi et al. (2010), el umbral $t$ que utilizan los autores varía de acuerdo con las características de la instancia del problema y los trabajos asignados en la solución, pero también puede ser utilizado para generar soluciones más diversas elevando el umbral a un valor fijo de 0.5. En la Figura 7.1 se muestra un ejemplo ilustrativo del funcionamiento de este operador para la segmentación de imágenes en RGB.



Considere una imagen a color de 4x4 = 16 píxeles, la cual se busca segmentar en 3 grupos ($k = 3$).

Se tiene la siguiente solución de la población seleccionada mediante la selección controlada para la mutación. La solución tiene tres segmentos (genes) y como información adicional se tiene su distancia intra-cluster por grupo.

| | $G_1$ | $G_2$ | $G_3$ | | |
|---|---|---|---|---|---|
| ***Solución*** | 1, 4, 6, 10, 11 | 2, 5, 7, 9, 13, 12 | 3, 8, 14, 15, 16 | | |
| *Distancia intra-cluster* | 11.67 | 95.63 | 20.12 | 127.42 | *Total* |

Se aplica el operador de *Item elimination* con un umbral t = 0.8. Se tiene una lista plana de los píxeles de la solución. A cada píxel se le asigna una probabilidad y posteriormente se compara con el umbral para determinar si este se queda en la solución. En naranja se muestran los píxeles que se eliminarán de la solución por tener un valor menor que el umbral. En la ***solución a mutar*** se marcan en rojo los píxeles que se eliminarán de su respectivo grupo. En la ***solución mutada*** se muestra la solución sin los píxeles que tuvieron un umbral menor que el establecido. A la derecha de la solución aparecen los píxeles perdidos, los cuales son reinsertados con la heurística BF, quedando como solución final la ***solución reparada.***

| ***Píxel*** | 1 | 4 | 6 | 10 | 11 | 2 | 5 | 7 | 9 | 13 | 12 | 3 | 8 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ***Probabilidad*** | 0.72 | 0.81 | 0.13 | 0.41 | 0.80 | 0.50 | 0.10 | 0.62 | 0.79 | 0.80 | 0.50 | 0.80 | 0.20 | 0.90 | 0.22 | 0.08 |

| | $G_1$ | $G_2$ | $G_3$ |
|---|---|---|---|
| ***Solución a mutar*** | 1, 4, 6, 10, 11 | 2, 5, 7, 9, 13, 12 | 3, 8, 14, 15, 16 |
| *Distancia intra-cluster* | 11.67 | 95.63 | 20.12 |

| | $G_1$ | $G_2$ | $G_3$ |
|---|---|---|---|
| ***Solución mutada*** | 4 | 13 | 3, 14 |
| *Distancia intra-cluster* | 0.00 | 0.00 | 6.81 |

| ***Perdidos*** |
|---|
| 1,6,10,11,2,5, 7, 9,12,8,15,16 |

| | $G_1$ | $G_2$ | $G_3$ | | |
|---|---|---|---|---|---|
| ***Solución reparada*** | 4, 16, 1, 15, 11 | 13, 2, 9, 6, 8 | 3,14, 7,10,12, 5 | | |
| *Distancia intra-cluster* | 0.00 | 32.01 | 18.92 | 50.93 | *Total* |

Figura 7.1: Ejemplo ilustrativo del operador de mutación Item elimination para la segmentación de imágenes en RGB.

## 7.1. Configuración de parámetros

Antes de realizar los experimentos con el banco de imágenes de control y las imágenes de prueba de BSDS500 con más semillas, se decidió realizar una calibración del algoritmo adaptado en su versión inicial. Los valores de número de individuos a cruzar y mutar elegidos se usaron tanto para el algoritmo inicial como para el GGA-CGT-RGB-IS+IE.

Para la calibración, se seleccionó un subconjunto de 20 imágenes de las imágenes de prueba de BSDS500 de forma aleatoria y cuidando que se tuviera una representación

adecuada de las características de las imágenes con respecto al número de segmentos utilizando. Para la calibración se empleó una semilla por imagen. A partir de estos resultados, se encontró que los mejores parámetros para el GGA-CGT-RGB-IS son: 60 % de cruza y 30 % de mutación.

Los valores del resto de parámetros del algoritmo no fueron modificados, por lo que son los mismos que se utilizan en el GGA-CGT. De esta forma, los parámetros finales para ejecutar los experimentos se muestran a continuación:

- Tamaño de población, $P$: 100 individuos.
- Número máximo de generaciones, $N$: 100.
- Número de individuos a cruzar, $n_c$: 60.
- Número de individuos a mutar, $n_m$: 30.
- Número de individuos en el grupo élite: 10.
- Número máximo de generaciones de un individuo para ser clonado, *life_span*: 10.

De esta forma, la única diferencia entre las dos versiones del algoritmo está en el operador de mutación utilizado. Recordando que la versión inicial utiliza el operador de mutación de Elimination, eliminando el 20 % de los peores grupos de la solución y la versión GGA-CGT-RGB-IS+IE utiliza un umbral de 0.8

## 7.2. Segmentación del banco de imágenes de control

El banco de imágenes de control está formado por 100 imágenes con un número definido de colores diferentes. Existen 2 imágenes distintas por cada número de segmentos en un rango de 5 a 54 y se utilizaron 30 semillas por imagen. En estas imágenes se conoce el óptimo global, de forma que los algoritmos se comparan con respecto al número de casos exitosos obtenidos. Sin embargo, ambas versiones del algoritmo tuvieron el 100 % de casos exitosos, por lo que la comparación se realizó con respecto al número de generaciones tomadas para alcanzar el óptimo.

En la Tabla 7.1 se muestran los resultados de la segmentación del banco de imágenes de control. Debido a la cantidad de imágenes, los resultados se agruparon con respecto al número de segmentos de cada imagen, formando grupos de imágenes con un rango específico de número de segmentos de forma que cada grupo tuviera la misma cantidad de imágenes. En este caso, cada grupo contiene 20 imágenes que están dentro del correspondiente rango de segmentos de cada grupo. En la primera columna de esta tabla se tienen los grupos por rangos de número de segmentos. Se tienen cinco grupos diferentes de acuerdo con el número de segmentos de las imágenes del banco de imágenes de control. Las siguientes tres columnas muestran estadísticas de mejor, media y peor para las imágenes que pertenecen a cada grupo indicando en paréntesis la desviación estándar de estos valores. A su vez, cada columna se divide en dos columnas, donde se muestran los valores para cada versión del algoritmo. El GGA-CGT-RGB-IS+IE corresponde a la nueva versión que utiliza el operador de Item elimination, mientras que la otra versión es la inicial. Los resultados se muestran en términos del número de generaciones promedio tomadas para alcanzar el óptimo global.

Los resultados que se muestran en la tabla se obtuvieron de la siguiente forma. Primero se obtuvieron las estadísticas de mejor, media y peor con las 30 ejecuciones por imagen. Posteriormente, se agruparon y los valores que se muestran en la tabla corresponden al promedio de cada una de las estadísticas de cada imagen dentro de cada grupo, así como la desviación estándar de estos valores mostrada en paréntesis. Los resultados completos se pueden consultar en la Tabla A.1 del Apéndice A.

A partir de esta tabla se puede observar que la nueva versión del algoritmo con el operador de Item elimination es mejor que la versión inicial. También resulta interesante observar que el algoritmo sigue teniendo un buen desempeño aunque el número de segmentos aumente. Dicho comportamiento no se observa con la versión inicial, donde su desempeño tiende a disminuir a medida que el número de segmentos aumenta.

Tabla 7.1: Generaciones promedio de la segmentación del banco de imágenes de control agrupadas por número de segmentos.

| **No. Segmentos** | **Mejor** | | **Media** | | **Peor** | |
|---|---|---|---|---|---|---|
| | **GGA-CGT-RGB-IS+IE** | **GGA-CGT-RGB-IS** | **GGA-CGT-RGB-IS+IE** | **GGA-CGT-RGB-IS** | **GGA-CGT-RGB-IS+IE** | **GGA-CGT-RGB-IS** |
| *(4, 14]* | **2.0 (0.0)** | **2.0 (0.7947)** | **2.78 (0.5135)** | 3.49 (1.1655) | **3.7(0.5712)** | 5.7 (2.1788) |
| *(14, 24]* | **3.45 (0.8256)** | 5.35 (1.2258) | **5.11 (0.8795)** | 9.21 (2.5563) | **6.8 (1.3992)** | 14.65 (5.3339) |
| *(24, 34]* | **4.85 (0.7452)** | 9.4 (1.729) | **7.15 (0.7897)** | 20.34 (6.0741) | **9.65 (1.8432)** | 41.3 (18.8905) |
| *(34, 44]* | **5.1 (0.7182)** | 10.0 (2.5955) | **6.85 (0.8459)** | 18.86 (7.2622) | **8.6 (1.3534)** | 36.45 (19.2339) |
| *(44, 54]* | **5.4 (1.6351)** | 13.8 (8.1344) | **7.24 (1.9422)** | 28.23 (18.8924) | **9.05 (2.481)** | 51.95 (37.1618) |

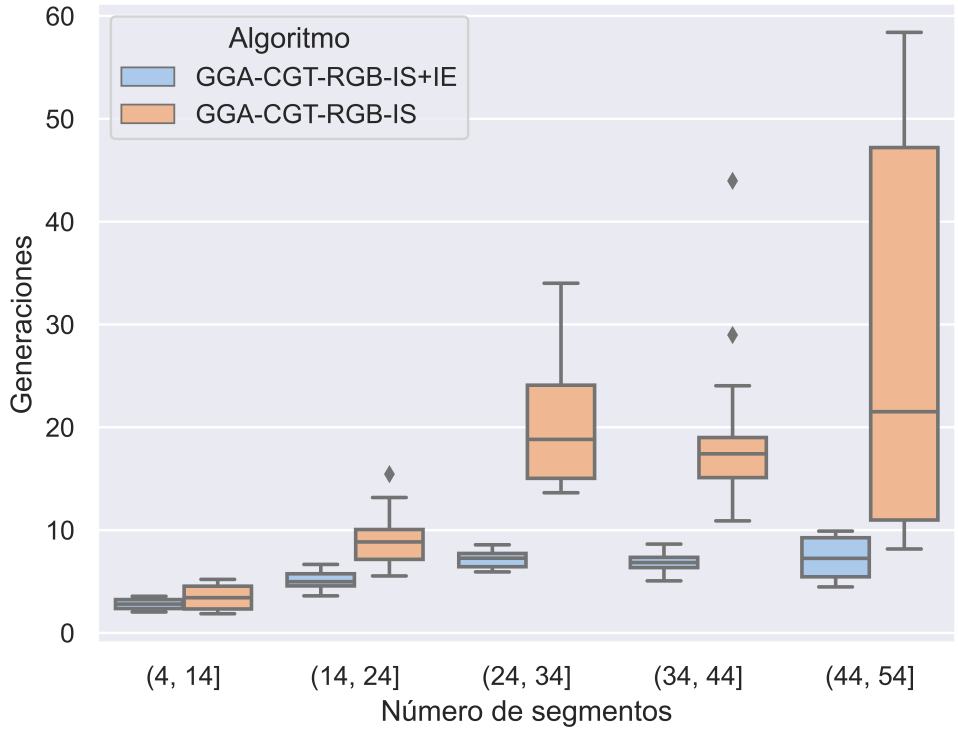



Figura 7.2: Generaciones promedio de la segmentación del banco de imágenes de control agrupados por número de segmentos.

En la Figura 7.2 se muestran los resultados de la tabla anterior gráficamente para la

estadística de la media. En el eje $x$ se tienen los grupos de imágenes por rango de segmentos, mientras que en el eje $y$ se muestra el número de generaciones. En esta figura se puede observar que el desempeño del GGA-CGT-RGB-IS+IE es mejor en todos los grupos de imágenes considerados con respecto al número de generaciones tomadas para encontrar el óptimo global. Especialmente, esta mejora se puede apreciar en los grupos donde el rango de segmentos es mayor.

Tabla 7.2: Resumen de la prueba de Wilcoxon Rank-Sum.

| **No. Segmentos** | **A favor** | **En contra** | **SdS** |
|---|---|---|---|
| *(4, 14]* | 12 | 0 | 8 |
| *(14, 24]* | 20 | 0 | 0 |
| *(24, 34]* | 20 | 0 | 0 |
| *(34, 44]* | 20 | 0 | 0 |
| *(44, 54]* | 20 | 0 | 0 |

Con los mismos datos de los experimentos se realizó la prueba de Wilcoxon Rank-Sum para validar si la diferencia en los resultados obtenidos es significativa o no. La prueba de Wilcoxon Rank-Sum es una prueba estadística no paramétrica en la que se busca comprobar la hipótesis nula de que un conjunto de datos $A$ y $B$ son muestras de distribuciones continuas cuyas medianas son idénticas (Dao, 2022). Con esta prueba se encuentra un valor conocido como *p-value*. De acuerdo con Saha et al. (2016), este valor es la probabilidad de observar que la hipótesis nula sea verdadera. Por lo tanto, un *p-value* pequeño implica que existe una probabilidad muy baja de que los datos observados hayan ocurrido por casualidad. Así, la hipótesis nula se rechaza cuando el *p-value* está por debajo del valor significativo, el cual generalmente es establecido en 5 % o 0.05 (Saha et al., 2016). En la Tabla 7.2 se muestra un resumen de los resultados de la prueba de Wilcoxon Rank-Sum agrupados también de acuerdo con el número de segmentos de las imágenes. En la primera columna se muestran los grupos y sus rangos de segmentos. Las siguientes columnas muestran los conteos de imágenes por grupo para los siguientes casos: cuando la nueva versión es mejor que la inicial y existe diferencia significativa (a favor), cuando la versión inicial es mejor que la nueva y existe diferencia significativa (en contra) y cuando no existe diferencia significativa (SdS); es decir, aquellas imágenes en las que el *p-value* es mayor que 0.05. En dicha tabla, se puede observar que para el grupo de

imágenes que tienen entre 5 y 14 segmentos, casi la mitad de imágenes que cumplen con esas características no presentan una diferencia significativa. Para el resto de los grupos, todas las imágenes presentan una diferencia significativa a favor de la nueva versión. Por lo que podemos decir que claramente existe una mejora en el desempeño del algoritmo con el operador de mutación propuesto para las instancias del banco de imágenes de control. Los resultados completos de la prueba de Wilcoxon Rank-Sum para este banco de imágenes se pueden consultar en la Tabla B.1 del Apéndice B.

## 7.3. Segmentación de las imágenes de prueba de BSDS500

El conjunto de imágenes de prueba está formado por 200 imágenes a color en formato *jpg*. Con el objetivo de reducir el costo computacional del algoritmo, las imágenes fueron reescaladas de 481x321 píxeles a 80x53 píxeles conservando su misma relación de aspecto. Para la segmentación de estas imágenes se utilizaron los mismos parámetros mostrados en la primera sección de este capítulo.

A diferencia de las imágenes de prueba, para las imágenes de BSDS500 no se tiene algún óptimo conocido, por lo que el algoritmo fue ejecutado hasta alcanzar el número máximo de generaciones, reportando la mejor solución encontrada de forma global durante las 100 generaciones. Debido a que no se conoce algún óptimo local, se utilizó K-Means como referencia para comparar los resultados entre ambas versiones. Para esta comparación se utilizó nuevamente el *RPD* como en la Ecuación 4.2. Donde $intra(i)$ es el valor de la distancia intra-cluster encontrado por cada versión del GGA para la imagen $i$, mientras que $intra*(i)$ corresponde al de K-Means. Lo deseable es que este valor sea lo más cercano posible al 0. Incluso un valor negativo es favorable, pues es una señal de que el algoritmo tuvo un mejor desempeño que K-Means en cuanto al objetivo a minimizar, el cual corresponde a la distancia intra-cluster.

$$RPD = \frac{intra(i) - intra*(i)}{intra*(i)}. \tag{7.1}$$

Para estos experimentos se utilizaron los mismos rangos que para las imágenes de control, con la diferencia de que los tres últimos grupos fueron unidos, ya que contenían

una cantidad de imágenes pequeña. La cantidad de imágenes por grupo se muestra en la Tabla 7.3. En la primera columna se tienen los rangos de los cuatro grupos de acuerdo con el número de segmentos de las imágenes. En la siguiente columna se muestra el conteo del número de imágenes que pertenecen a cada grupo. En esta tabla se puede observar que la mayor cantidad de imágenes se concentra en el primer grupo.

En la Tabla 7.4 se muestran los resultados de la segmentación de las imágenes de prueba. Para estos experimentos, cada versión del algoritmo se ejecutó 30 veces y se agruparon de acuerdo con los rangos de número de segmentos mencionados en el párrafo anterior. Asimismo, las estadísticas completas de los RPDs por imagen se muestran en la Tabla A.2 del Apéndice A.

Tabla 7.3: Distribución de las imágenes de prueba de BSDS500 en los grupos por número de segmentos.

| **No. Segmentos** | **No. Imágenes** |
|---|---|
| *(1, 14]* | 165 |
| *(14, 24]* | 24 |
| *(24, 54]* | 11 |

En la Tabla 7.4 se muestran los resultados de la segmentación del banco de imágenes de prueba de BSDS500. La primera columna corresponde al rango de número de segmentos de cada grupo. Las siguientes columnas muestran tres estadísticas para los valores de *RPD* de las imágenes segmentadas que pertenecen al grupo en cuestión. A su vez, cada una de estas columnas se divide en dos, donde cada columna corresponde a la versión del algoritmo, donde GGA-CGT-RGB-IS+IE corresponde a la nueva versión que utiliza el operador de mutación de Item elimination, mientras que la otra versión es la inicial. Para las estadísticas de mejor, media y peor, se usan los promedios de los resultados contenidos en cada grupo y en paréntesis se muestra la desviación estándar de estos valores.

A partir de los resultados de la Tabla 7.4 se puede observar la misma tendencia que con las imágenes de control para el GGA-CGT-RGB-IS+IE. En el primer grupo de imágenes con menos segmentos, se observa que ambas versiones tienen un comportamiento similar y que la versión inicial solo es en promedio mejor en la estadística de mejor para el primer grupo. En los siguientes grupos el GGA-CGT-RGB-IS+IE es mejor en las tres estadísticas

y esta diferencia es más notable conforme aumenta el número de segmentos.

En la Figura 7.3 se pueden observar los resultados con diagramas de caja utlizando los datos de la media de cada imagen por grupo. En el eje x se muestran los grupos de acuerdo con su rango de número de segmentos y en el eje y se muestra el RPD. En esta figura se observa claramente que el GGA-CGT-RGB-IS+IE es mejor cuando el número de segmentos es mayor, pero cuando se tienen menos (entre 2 y 14 segmentos), la diferencia no es tan notable. Es importante destacar que la mediana de cada grupo de la nueva versión es es menor que la de la versión inicial. También se puede observar que el desempeño de la nueva versión tiende a mejorar conforme aumenta la complejidad de las características de la imagen (el número de segmentos). En el último grupo, la nueva versión del algoritmo tiene un desempeño donde el valor de la mediana es incluso menor que K-Means, pues la línea del diagrama de caja indica que el *RPD* está por debajo del 0. Esto indica que por lo menos el 50 % de las imágenes de ese grupo que fueron segmentadas con la nueva versión tienen en promedio una calidad mejor que la encontrada por K-Means. Con respecto a la versión inicial del algoritmo, esta tiene un desempeño muy similar en todos los grupos y su mediana se mantiene por arriba de 0.01 en los tres grupos. De manera general, se puede ver que la nueva versión del algoritmo parece ser mejor en términos de la distancia intra-cluster para realizar la tarea de segmentación de imágenes, especialmente con imágenes con un mayor número de segmentos.

Tabla 7.4: *RPD* promedio de la segmentación de las 200 imágenes de prueba de BSDS500 agrupadas por número de segmentos.

| **No. Segmentos** | **Mejor** | | **Media** | | **Peor** | |
|---|---|---|---|---|---|---|
| | **GGA-CGT-RGB-IS+IE** | **GGA-CGT-RGB-IS** | **GGA-CGT-RGB-IS+IE** | **GGA-CGT-RGB-IS** | **GGA-CGT-RGB-IS+IE** | **GGA-CGT-RGB-IS** |
| *(1, 14]* | 0.0248 (0.0084) | **0.0241 (0.0087)** | **0.0096 (0.021)** | 0.0103 (0.021) | **-0.0285 (0.0594)** | -0.0256 (0.0596) |
| *(14, 24]* | **0.0232 (0.0037)** | 0.0261 (0.0048) | **0.0014 (0.0122)** | 0.01 (0.0141) | **-0.0405 (0.0424)** | -0.0265 (0.0445) |
| *(24, 55]* | **0.0186 (0.0045)** | 0.0356 (0.0128) | **-0.0081 (0.0122)** | 0.018 (0.0233) | **-0.0696 (0.0554)** | -0.0366 (0.0652) |

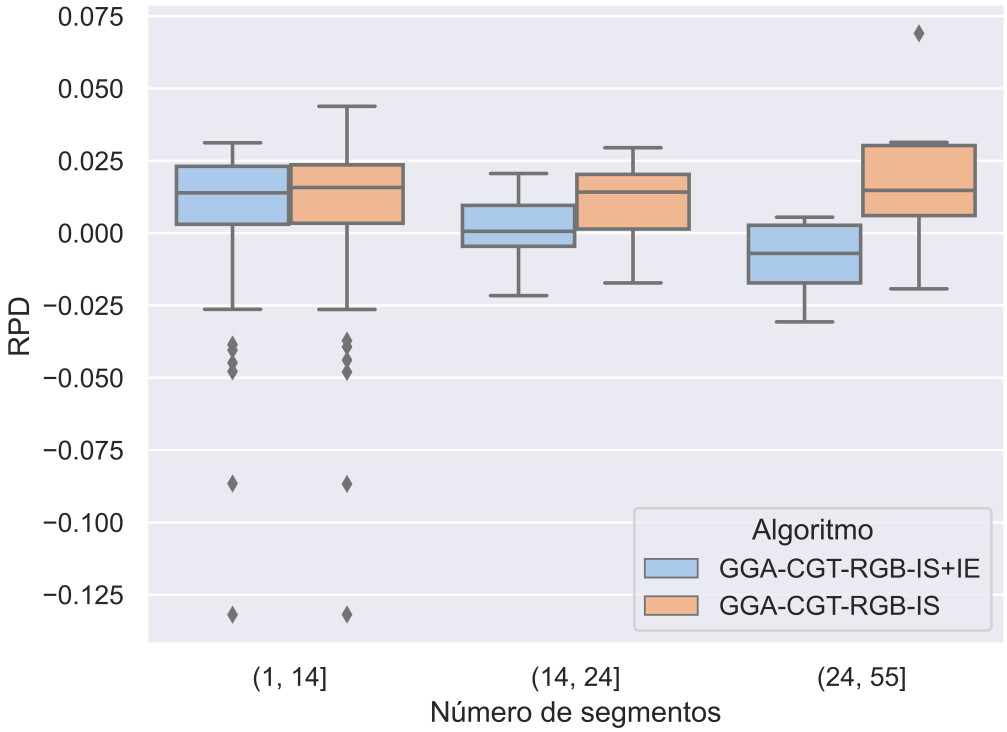



Figura 7.3: Resultados promedio de la segmentación las 200 imágenes de prueba de BSDS500 agrupados por número de segmentos.

Como se mencionó anteriormente, para la segmentación de estas imágenes se tienen los resultados de tres algoritmos: las dos versiones del GGA-CGT adaptado al problema y K-Means. Debido a que en los algoritmos genéticos se está minimizando la distancia intra-cluster, para K-Means los resultados también se evalúan con respecto a la distancia intra-cluster de la solución encontrada para cada instancia. Para hacer la prueba estadística, primero es necesario considerar las tres poblaciones o conjuntos de resultados de los algoritmos para determinar si existen diferencias entre estos algoritmos. Para ello, es necesario aplicar la prueba de Kruskal-Wallis. Esta prueba es básicamente una extensión de la de Wilcoxon Rank-Sum, pero para más de dos conjuntos de datos. La prueba de Kruskal-Wallis nos permite primero identificar si existen diferencias significativas entre los conjuntos de datos de más de dos poblaciones. Si el *p-value* es significativo; es decir, menor que 0.05, entonces se puede concluir que al menos dos poblaciones son diferentes. El siguiente paso es determinar cuáles son diferentes, por lo que ahora se utiliza Wilcoxon Rank-Sum por pares para encontrar diferencias significativas (Rumsey, 2007).

Una vez que realizada la prueba de Kruskal-Wallis, se encontró que 192 de las 200 imágenes de prueba de BSDS500 tuvieron un *p-value* menor de 0.05, por lo que con estas imágenes se procedió a aplicar la prueba de Wilcoxon Rank-Sum por pares de algoritmos. Así, el resumen de esta prueba se muestra en la Tabla 7.5. En la primera columna se muestran los grupos y sus rangos de segmentos. Las siguientes dos columnas muestran el conteo para cada par de algoritmos comparando la nueva versión con la inicial, así como con K-Means. Para cada comparación se tienen tres subcolumnas que muestran los conteos de imágenes por grupo para los siguientes casos: cuando la nueva versión es mejor que la inicial y existe diferencia significativa (a favor), cuando la versión inicial es mejor que la nueva y existe diferencia significativa (en contra) y cuando no existe diferencia significativa (SdS); es decir, aquellas imágenes en las que el *p-value* obtenido con la prueba de Wilcoxon Rank-Sum es mayor que 0.05.

A partir de los resultados que se muestran en la Tabla 7.5, se puede observar que la versión inicial es mejor que la nueva en la gran mayoría de las imágenes del primer grupo, aunque también existe una cantidad considerable de imágenes sin diferencia significativa. Mientras que para el resto de los grupos, la nueva versión es mejor en todas las imágenes cuyo *p-value* obtenido con la prueba de Wilcoxon Rank-Sum es significativo. Por lo que con estos resultados podemos afirmar que existe una mejora significativa en el desempeño del algoritmo con el operador de mutación Item elimination para imágenes con más de 14 segmentos y que las diferencias en el desempeño de las dos versiones no son significativas cuando se tiene un menor número de segmentos. En el caso de la comparación contra K-Means, este último algoritmo siempre tiene la mayoría de casos a su favor, pero también existe una gran proporción de imágenes sin diferencia significativa, ligeramente superior que en la comparación de las dos versiones del algoritmo. De forma que en estos casos no se puede afirmar cuál algoritmo es mejor. Los resultados completos de ambas pruebas estadísticas realizadas para las imágenes de BSDS500 se pueden consultar en la Tabla B.2 del Apéndice B.

Tabla 7.5: Resumen de la prueba de Wilcoxon Rank-Sum con las 192 imágenes de 200 cuyo *p-value* es significativo, de acuerdo con la prueba de Kruskal.

| **No. Segmentos** | **GGA-CGT-RGB-IS+IE vs GGA-CGT-RGB-IS** | | | **GGA-CGT-RGB-IS+IE vs K-Means** | | |
|---|---|---|---|---|---|---|
| | A favor | En contra | SdS | A favor | En contra | SdS |
| *(1, 14]* | 29 | 103 | 27 | 20 | 122 | 17 |
| *(14, 24]* | 21 | 0 | 1 | 3 | 12 | 7 |
| *(24, 55]* | 11 | 0 | 0 | 2 | 4 | 5 |

## 7.4. Comportamiento algorítmico: GGA-CGT-RGB-IS+IE vs GGA-CGT-RGB-IS

Como se pudo observar en los resultados obtenidos de la segmentación tanto del banco de imágenes de control como las imágenes de prueba de BSDS500, parece que existe una tendencia en la que, a medida que aumenta el número de segmentos, la diferencia entre las dos versiones del GGA-CGT-RGB-IS es más clara, donde la nueva versión tiene un desempeño más estable que la primera versión. Para profundizar más en el análisis de este comportamiento y verificar si existe una tendencia clara en la convergencia del algoritmo, se realizó la segmentación de 2 de las 200 imágenes de prueba de BSDS500. Una con pocos segmentos y otra con una cantidad muy grande de segmentos, que para este conjunto de imágenes corresponde a una imagen con 5 segmentos y otra con 54. Para cada imagen, se seleccionó la semilla que está en la mediana de las 30 ejecuciones con respecto a la distancia intra-cluster para cada algoritmo.

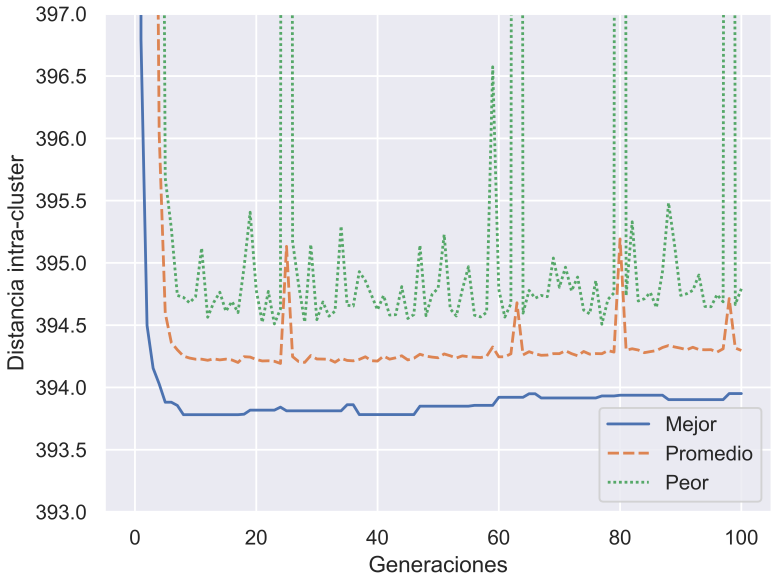



Figura 7.4: Gráfico de convergencia de GGA-CGT-RGB-IS+IE para la imagen 100075 de BSDS500 con 5 segmentos.

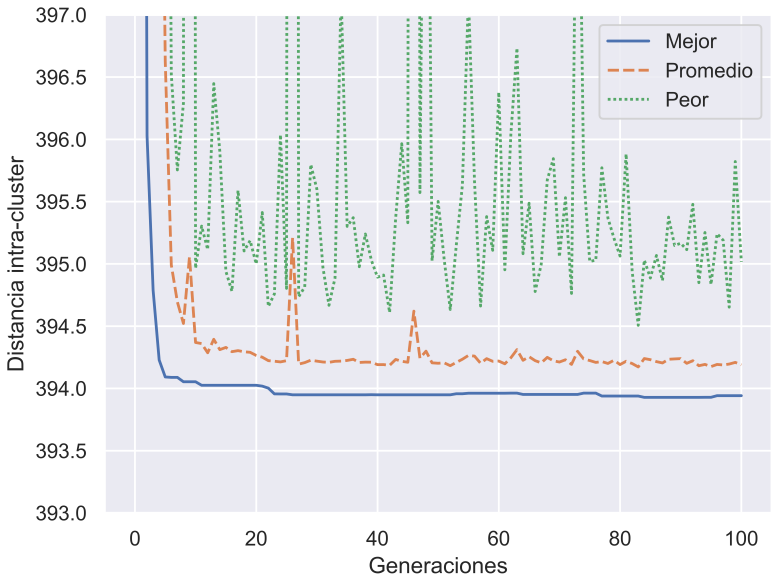



Figura 7.5: Gráfico de convergencia de GGA-CGT-RGB-IS para la imagen 100075 de BSDS500 con 5 segmentos.

En esta sección, se analiza el comportamiento algorítmico de ambas versiones segmentando estas dos imágenes que presentan características contrastantes. Para esto, se analizan los gráficos de convergencia para cada versión del algoritmo por imagen. Estos gráficos reportan la distancia intra-cluster de la mejor y peor solución, así como el promedio de la

población por generación. Analizar la convergencia permite observar el comportamiento del algoritmo durante la búsqueda. Esta convergencia determina el compromiso entre la exploración y la explotación del algoritmo.

Los primeros gráficos de convergencia corresponden a la imagen 100075, la cual es segmentada en 5 grupos. En la Figura 7.4 se muestra el gráfico de convergencia usando el GGA-CGT-RGB-IS+IE, donde se pueden observar las tres mediciones mencionados anteriormente. En la Figura 7.5 se muestra el gráfico de convergencia correspondiente a la versión inicial del algoritmo.

Para la primera imagen, se puede observar que el comportamiento de ambos algoritmos es muy similar, aunque la forma en que varía la distancia intra-cluster de la mejor solución tiene más saltos. Esto puede sugerir que el algoritmo no pudo encontrar una mejor solución tan rápido y la mejor solución disponible en el grupo élite en determinada generación desaparecía. De forma que estos saltos se pueden atribuir a esa desaparición de la mejor solución del grupo élite sin que tuviera oportunidad de ser reemplazada por una mejor. Esto no ocurre con la versión inicial donde los saltos que se observan corresponden a una mejora en la calidad de la solución encontrada.

Con la segunda imagen, que tiene 54 segmentos, se observa una diferencia más notable en el comportamiento algorítmico entre ambas versiones. En las Figuras 7.6 y 7.7 se puede observar el comportamiento de ambas versiones para la imagen con 54 segmentos. Claramente se observa que la versión inicial tiende a oscilar más en las tres mediciones mostradas, mientras que la versión con Item elimination parece ser más estable a lo largo de las 100 generaciones. Con respecto a lo que se discutió para la primer imagen, se puede observar que la línea de la mejor solución para la versión inicial del algoritmo tiende a oscilar y la mejor solución del grupo élite se pierde al no encontrar algo mejor en el proceso evolutivo. Esto no ocurre con la nueva versión, donde el comportamiento es más suave y la calidad de la mejor solución inicial mejora y, en el peor de los casos, se mantiene.

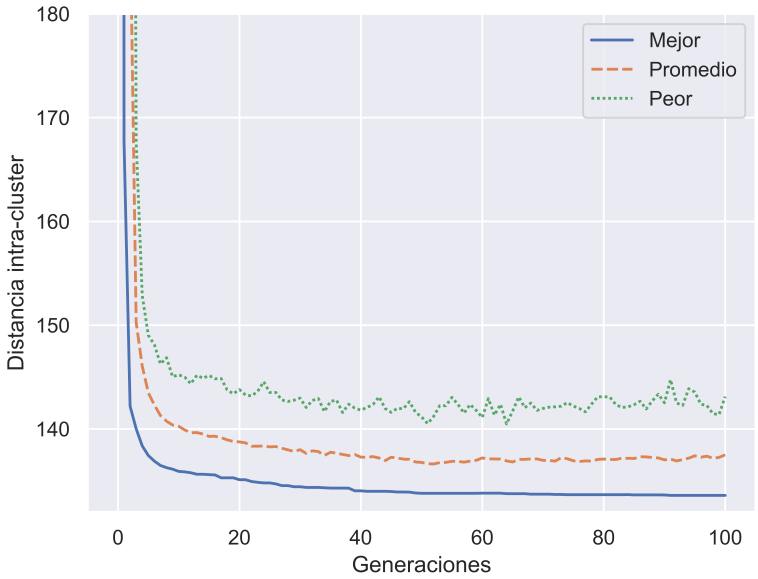



Figura 7.6: Gráfico de convergencia de GGA-CGT-RGB-IS+IE para la imagen 376001 de BSDS500 con 54 segmentos.

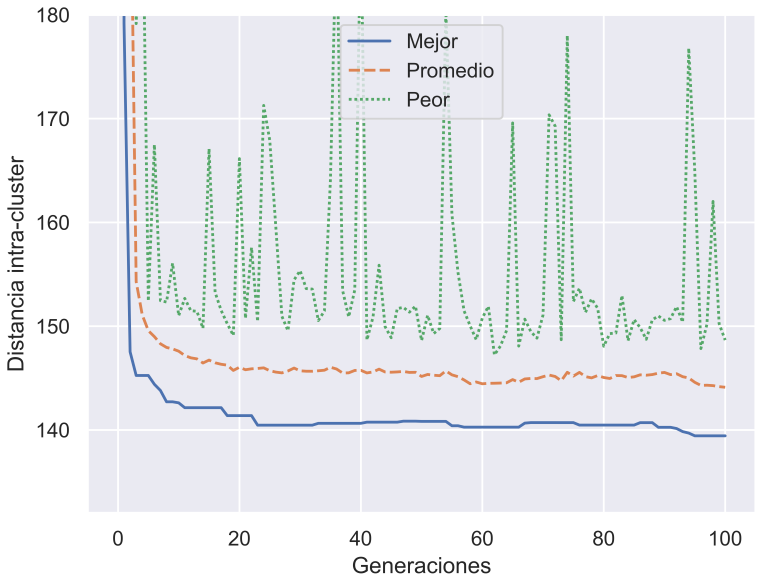



Figura 7.7: Gráfico de convergencia de GGA-CGT-RGB-IS para la imagen 376001 de BSDS500 con 54 segmentos.

Con la imagen de 54 segmentos se pueden observar más diferencias entre el comportamiento de estas versiones del algoritmo. Por un lado, las soluciones de la nueva versión parecen ser más diversas en cuanto a que su gráfico muestra una mayor separación entre cada una de las métricas mostradas y oscilan mucho más. Mientras que en el gráfico de

la nueva versión la diferencia entre la mejor y la peor no es tan alta y se mantiene más estable (oscila menos). Sin embargo, el comportamiento de esta nueva versión puede ser deseable, ya que no es tan útil mantener soluciones con muy mala calidad en la población. Es decir, el nuevo operador es responsable de mantener la diversidad en la población, pero no lo hace tan aleatoriamente como el operador de Elimination, sino que también se busca tener soluciones con una buena calidad en general.

# Capítulo 8

# Conclusiones y trabajo futuro

En esta sección se presentan las conclusiones de este trabajo. Además, los aspectos que no estuvieron dentro de los alcances de este proyecto, se muestran como propuestas a desarrollar para trabajo futuro.

## 8.1. Conclusiones

En este trabajo se exploró si un GGA podría tener un buen desempeño para la segmentación de imágenes en RGB y obtener soluciones competitivas con las del estado del arte, teniendo como objetivo principal el diseño de un operador de mutación a través de un estudio experimental para mejorar el desempeño del algoritmo con respecto a la versión inicial, GGA-CGT-RGB-IS, algoritmo propuesto en este trabajo como resultado de la adaptación del GGA-CGT. De esta forma, la nueva versión, GGA-CGT-RGB-IS+IE, tiene un nuevo operador de mutación que proporciona mejores resultados con respecto a la calidad de las soluciones y la rapidez en cuanto al número de generaciones tomadas para encontrar una buena solución. Como producto de este estudio se diseñaron dos operadores de mutación, uno con un enfoque basado en grupos y otro con un enfoque basado en elementos. Sin embargo, el primero operador de mutación diseñado no fue el mejor para este problema, pues Item elimination obtuvo mejores resultados. Ambos operadores fueron adaptados por primera vez para el problema de la segmentación de imágenes. Con respecto a Item Elimination, Rossi et al. (2010) lo utilizaron por primera vez para un problema dis-

tinto y emplearon dos valores de umbral: uno constante y otro variable. A diferencia de la propuesta en este trabajo que emplea un solo valor constante, pues fue el que proporcionó mejores resultados. La configuración de este parámetro tuvo un impacto importante en el desempeño del algoritmo, el cual fue más evidente con el banco de imágenes de control. Durante esta configuración se pudo observar que este operador juega un papel importante tanto para generar soluciones diversas como para mejorarlas. Esto depende del valor de umbral utilizado, pues el operador puede ser muy disruptivo si este umbral es muy alto, o puede hacer cambios más pequeños con un valor más bajo.

En el desarrollo de este trabajo, se propuso el primer GGA para la segmentación de imágenes en RGB, para el cual se observó que existía cierta similitud con el GGA-CGT con respecto a los operadores de cruza y mutación empleados. Con esto surgió el GGA-CGT-RGB-IS, un nuevo GGA adaptado del GGA-CGT que resuelve el 1D-BPP.

Con base en lo anterior, se realizó un estudio experimental de operadores de mutación del estado del arte orientados a grupos, los cuales fueron adaptados y aplicados por primera vez al problema de la segmentación de imágenes. El estudio no se limitó a la adaptación de estos operadores, si no que se exploraron diferentes estrategias y configuraciones de los operadores. Gracias a este estudio, fue posible diseñar una nueva versión del algoritmo, llamada GGA-CGT-RGB-IS, que permitió mejorar de manera significativa el desempeño del algoritmo al incluir el operador adecuado según las características del problema.

Se propuso un nuevo banco de imágenes de control para probar los operadores de mutación. Estas imágenes tienen un número de segmentos específico que hace que se llegue a un óptimo global en el que la distancia intra-cluster es de 0. Dicho banco de imágenes permitió observar el comportamiento de los operadores y decidir qué elementos abonaban en el desempeño de estos en cuanto a la calidad de las soluciones encontradas, como en el número de generaciones que se requerían para alcanzar el óptimo global, por lo que este banco de imágenes fue de gran utilidad para conseguir los objetivos planteados.

Durante los experimentos realizados con los distintos operadores de mutación se observó que la heurística BF es una excelente auxiliar para mejorar el desempeño del operador de mutación para el problema de la segmentación de imágenes. Esta heurística se emplea para reparar las soluciones producidas en la mutación. Uno de los posibles y más comunes

escenarios que ocurren durante la mutación es que existan píxeles que no estén asignados (píxeles perdidos) a alguno de los grupos existentes. Así, esta heurística asigna los píxeles perdidos al mejor grupo posible, lo que representa una oportunidad para generar una buena solución después de haber aplicado la mutación. Los operadores que no necesitaban reparar soluciones y, por lo tanto, no hicieron uso de la heurística BF, tuvieron un mal desempeño con la segmentación del banco de imágenes de control. También se observó que se requería un operador más disruptivo para mejorar el desempeño del algoritmo. En este caso, Item elimination y Elimination fueron los mejores operadores. El primero resulta ser más disruptivo que el segundo, pues los píxeles que se eliminan pueden pertenecer a cualquiera de los segmentos que están en la solución, en lugar de solo seleccionarlos de una cantidad pequeña de segmentos de la solución, como en el caso de Elimination. Se pudo observar que el comportamiento de este operador se preservaba cuando se utilizaba dentro del algoritmo completo, pues los resultados eran mejores.

En el caso del operador de mutación diseñado, el cual está basado en el de Elimination, durante la Fase 1 no se observó alguna tendencia que indicara que entre más segmentos de la solución se eliminaran, mejores resultados. De esta forma, se observó que eliminar el 20 % de los segmentos proporcionaba mejores resultados que eliminando el 40 %, 60 % u 80 %. En la Fase 2 se observó que eliminar los peores grupos es una buena estrategia de selección de grupos, la cual puede ser mejorada en combinación con la estrategia de selección aleatoria. Donde se tratan de eliminar de manera equitativa tanto grupos malos como aleatorios. Sin embargo, el uso de esta estrategia depende del número de segmentos en los que se vaya a dividir la imagen, pues si se tienen 5 segmentos o menos, es imposible eliminar 10 % de peores grupos y 10 % de grupos aleatorios. Esto se debe a que solo se puede trabajar a nivel de grupos y el 20 % de 5 segmentos corresponde solo a 1 segmento, por lo que únicamente se elimina un grupo de los peores. Ante estos casos, la estrategia deja de funcionar como estrategia de selección *peor-aleatorio* y solo actúa como *peor*.

Con respecto al operador de mutación que mejores resultados proporcionó, Item elimination, no existen muchos parámetros por configurar. Durante los experimentos de la Fase 1 se observó que entre mayor fuera el umbral, más rápido encontraba los óptimos el operador. El nivel de disrupción que se necesitó para que el desempeño fuera superior fue

muy alto. Con esto, el operador hacía un mayor uso de la heurística BF, lo cual favorecía el acomodo de los píxeles, encontrando soluciones con mejor calidad. El uso de esta heurística también es un elemento clave en la mejora del desempeño, pues resultó ser una estrategia de asignación de píxeles muy buena. En el caso del banco de imágenes de control, esta combinación de elementos favoreció altamente la convergencia del algoritmo al óptimo global, donde la diferencia en el desempeño entre la nueva versión, GGA-CGT-RGB-IS+IE fue mucho más significativa comparada con la versión inicial. En el caso de las imágenes de prueba de BSDS500, las mejoras no fueron tan notables, aunque se pudo observar un comportamiento similar.

En cuanto al costo computacional, el operador de Item elimination es más exigente, sobre todo porque utiliza un umbral muy alto. Pese a que se obtenían muy buenos resultados desde que se establecía el umbral en un 0.4 (superando a Elimination), se decidió trabajar con 0.8 porque existía todavía un margen de mejora considerable entre estos valores. Podríamos decir que un umbral de 0.4 habría sido suficiente, pero resultó de interés trabajar con este valor más alto y observar los resultados obtenidos. A pesar de que es más costoso emplear un umbral más alto, esto se compensa si consideramos que el porcentaje de mutación disminuyó considerablemente con respecto al valor inicial. Al inicio se tenía 80 % y en la nueva versión se redujo al 30 %.

Desde luego que es difícil tener un algoritmo que obtenga resultados buenos siempre con cualquier instancia. Y con el problema de la segmentación de imágenes esto no es la excepción. Algo que se pudo observar con las imágenes con las que se realizaron los experimentos es que, cuando se tiene un número de segmentos bajo (entre 2 y 11), el algoritmo en su versión inicial, el cual emplea el operador de mutación de Elimination es mejor que la nueva versión, aunque la diferencia no es significativa. Así, cuando se tiene una gran cantidad de segmentos, la nueva versión del algoritmo tiene un desempeño notablemente mejor que la versión inicial.

De esta forma, se logró cumplir con todos los objetivos planteados en este documento. A continuación se resumen las principales contribuciones de este trabajo de investigación:

- La principal contribución de este trabajo se centra en el área del diseño de operadores de variación, específicamente de operadores de mutación para GGAs.

- Se adaptó exitosamente el GGA-CGT para un nuevo problema de agrupación como el de la segmentación de imágenes, el cual fue abordado como un problema de clustering y de agrupación, dando como resultado un nuevo algoritmo llamado GGA-CGT-RGB-IS.

- Con base en el diseño experimental planteado, se realizó un estudio de operadores de mutación de agrupación para el problema de la segmentación de imágenes en RGB que permitió diseñar un operador de mutación que mejoró el desempeño de este algoritmo en cuanto a la calidad de las soluciones y el número de generaciones, cumpliendo así con el objetivo general.

- El desempeño del GGA-CGT-RGB-IS mejoró con respecto a la versión inicial al segmentar las imágenes de control en un 49.46 % (±21.78 %) en promedio en cuanto al número de generaciones tomadas para alcanzar el óptimo global, utilizando el operador de mutación de Item elimination, dando como resultado una nueva versión llamada GGA-CGT-RGB-IS+IE.

- Con las imágenes de prueba de BSDS500, la mejora en el desempeño de esta nueva versión varía con respecto al número de segmentos, donde se observa un mejor desempeño con imágenes que tienen más segmentos. Con imágenes de 2 a 14 segmentos, el promedio de mejora es del 0.57 % (±1.54 %). Para imágenes de 15 a 24 segmentos, la mejora es más evidente con 5.78 % (±5.24 %). Para el grupo de imágenes con segmentos de 25 a 54 se consiguió una mejora del 16.44 % (±7.67 %).

- Aunque el GGA-CGT-RGB-IS+IE es mejor que K-Means en 25 de las 171 imágenes con diferencia significativa, su desempeño es muy similar al de K-Means con respecto a la calidad de las soluciones encontradas. El RPD promedio de la nueva versión es de 0.0076 (±0.0203).

## 8.2. Trabajo futuro

Durante el desarrollo de este trabajo de investigación se identificaron algunos aspectos que requerían mayor tiempo de exploración y experimentación. Sin embargo, estos estu-

vieron fuera del alcance de los objetivos planteados para este trabajo durante el margen de tiempo establecido. A continuación, se mencionan algunas de estas áreas de oportunidad para trabajo futuro:

- Utilizar más instancias de prueba para la segmentación de imágenes. Especialmente con imágenes que tienen más segmentos, pues es donde se observó que el operador de Item elimination tiene un mejor desempeño. Sin embargo, debido a las características de las imágenes de prueba de BSDS500, el número de segmentos de la mayoría estaba por debajo de 15 segmentos.
- Explorar diferentes espacios de color para observar si se pueden obtener mejores resultados.
- Aplicar más formas de validar la segmentación realizada desde el punto de vista del clustering.
- Explorar el efecto del operador de Elimination cuando se tienen pocos grupos, ya que no es posible eliminar el porcentaje establecido.
- Estudiar otros operadores de mutación para GGAs que se encuentren en el estado del arte para poder explorar el comportamiento de los opreadores y su efecto en el proceso evolutivo del GGA.
- Estudiar otras estrategias para generar la población inicial que permitan al algoritmo encontrar mejores soluciones con un menor número de generaciones.
- Probar el algoritmo con una versión mejorada del operador de cruza y realizar una calibración de parámetros para esta nueva versión.
- Desarrollar una versión multi-objetivo del algoritmo para optimizar más objetivos, como el número de segmentos que debe tener la imagen, de forma que este no sea un parámetro más del algoritmo.
- Analizar otras aplicaciones del algoritmo para este problema. Este algoritmo podría adaptarse para resolver otros problemas que también sean de clustering.

# Bibliografía


Abdoun, O., Abouchabaka, J., and Tajani, C. (2012). Analyzing the performance of mutation operators to solve the travelling salesman problem. *arXiv preprint arXiv:1203.3099*.

Agustı, L., Salcedo-Sanz, S., Jiménez-Fernández, S., Carro-Calvo, L., Del Ser, J., Portilla-Figueras, J. A., et al. (2012). A new grouping genetic algorithm for clustering problems. *Expert Systems with Applications*, 39(10):9695–9703.

Aljarah, I., Faris, H., and Mirjalili, S. (2021). *Evolutionary data clustering: Algorithms and applications*. Springer.

Amador-Larrea, S. (2022). Un estudio de operadores genéticos de cruza para el Problema de Empacado de Objetos en Contenedores. Master's thesis, Universidad Veracruzana, Instituto de Investigaciones en Inteligencia Artificial.

Amador-Larrea, S., Quiroz-Castellanos, M., Hoyos-Rivera, G. d. J., and Mezura-Montes, E. (2022). An experimental study of grouping crossover operators for the bin packing problem. *Computación y Sistemas, An international Journal of computing science and applications.*, 26:663–682.

Arbelaez, P., Maire, M., Fowlkes, C., and Malik, J. (2010). Contour detection and hierarchical image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 33(5):898–916.

Awad, M., Chehdi, K., and Nasri, A. (2007). Multicomponent image segmentation using a genetic algorithm and artificial neural network. *IEEE Geoscience and remote sensing letters*, 4(4):571–575.

Awad, M., Chehdi, K., and Nasri, A. (2009). Multi-component image segmentation using a hybrid dynamic genetic algorithm and fuzzy c-means. *IET image processing*, 3(2):52–62.

Blum, C. and Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM computing surveys (CSUR)*, 35(3):268–308.

Csurka, G., Volpi, R., and Chidlovskii, B. (2021). Unsupervised domain adaptation for semantic image segmentation: a comprehensive survey. *arXiv preprint arXiv:2112.03241*.

Dao, P. B. (2022). On wilcoxon rank sum test for condition monitoring and fault detection of wind turbines. *Applied Energy*, 318:119209.

Eiben, A. E., Smith, J. E., et al. (2003). *Introduction to evolutionary computing*, volume 53. Springer.

Engelbrecht, A. P. (2007). *Computational intelligence: an introduction*. John Wiley & Sons.

Falkenauer, E. (1996). A hybrid grouping genetic algorithm for bin packing. *Journal of heuristics*, 2(1):5–30.

Ghosh, S., Das, N., Das, I., and Maulik, U. (2019). Understanding deep learning techniques for image segmentation. *ACM computing surveys (CSUR)*, 52(4):1–35.

Gonzalez, R. C. (2009). *Digital image processing*. Pearson education india.

Halder, A., Pradhan, A., Dutta, S. K., and Bhattacharya, P. (2016). Tumor extraction from mri images using dynamic genetic algorithm based image segmentation and morphological operation. In *2016 International Conference on Communication and Signal Processing (ICCSP)*, pages 1845–1849. IEEE.

Halder, A., Pramanik, S., and Kar, A. (2011). Dynamic image segmentation using fuzzy c-means based genetic algorithm. *International Journal of Computer Applications*, 28(6):15–20.

Hassanien, A. E. and Oliva, D. A. (2017). *Advances in soft computing and machine learning in image processing*, volume 730. Springer.

Holland, J. H. (1992). Genetic algorithms. *Scientific american*, 267(1):66–73.

Hong, T.-P., Lin, F.-S., and Chen, C.-H. (2012). Using the group genetic algorithm for attribute clustering. In *2012 IEEE Congress on Evolutionary Computation*, pages 1–5. IEEE.

Hong, T.-P., Wang, H.-S., and Chen, W.-C. (2000). Simultaneously applying multiple mutation operators in genetic algorithms. *Journal of heuristics*, 6(4):439–455.

Hruschka, E. R., Campello, R. J., Freitas, A. A., et al. (2009). A survey of evolutionary algorithms for clustering. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 39(2):133–155.

Lee, J. (2004). *A first course in combinatorial optimization*, volume 36. Cambridge University Press.

Lim, S. M., Sultan, A. B. M., Sulaiman, M. N., Mustapha, A., and Leong, K. Y. (2017). Crossover and mutation operators of genetic algorithms. *International journal of machine learning and computing*, 7(1):9–12.

Luke, S. (2009). *Essentials of metaheuristics*.

Martin, D., Fowlkes, C., Tal, D., and Malik, J. (2001). A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423.

Maulik, U. and Bandyopadhyay, S. (2003). Fuzzy partitioning using a real-coded variable-length genetic algorithm for pixel classification. *IEEE Transactions on geoscience and remote sensing*, 41(5):1075–1081.

Michalewicz, Z. and Fogel, D. B. (2013). *How to solve it: modern heuristics*. Springer Science & Business Media.

Minaee, S., Boykov, Y., Porikli, F., Plaza, A., Kehtarnavaz, N., and Terzopoulos, D. (2021). Image segmentation using deep learning: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3523–3542.

Mutingi, M. and Mbohwa, C. (2017). Grouping genetic algorithms. *Advances and Applications. Switzerland: Springer International Publishing*, 243.

Nakane, T., Bold, N., Sun, H., Lu, X., Akashi, T., and Zhang, C. (2020). Application of evolutionary and swarm optimization in computer vision: a literature survey. *IPSJ Transactions on Computer Vision and Applications*, 12(1):1–34.

Omran, M., Engelbrecht, A. P., and Salman, A. (2005). Particle swarm optimization method for image clustering. *International Journal of Pattern Recognition and Artificial Intelligence*, 19(03):297–321.

Papadimitriou, C. H. and Steiglitz, K. (1998). *Combinatorial optimization: algorithms and complexity*. Courier Corporation.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Quiroz-Castellanos, M., Cruz-Reyes, L., Torres-Jimenez, J., Gómez, C., Huacuja, H. J. F., and Alvim, A. C. (2015). A grouping genetic algorithm with controlled gene transmission for the bin packing problem. *Computers & Operations Research*, 55:52–64.

Ramos-Figueroa, O. (2022). *Efficient Heuristic Strategies for the Parallel-Machine Scheduling Problem with Unrelated Machines and Makespan Minimization*. PhD thesis, Universidad Veracruzana, Artificial Intelligence Resarch Institute.

Ramos-Figueroa, O., Quiroz-Castellanos, M., Mezura-Montes, E., and Cruz-Ramírez, N. (2023). An experimental study of grouping mutation operators for the unrelated parallel-machine scheduling problem. *Mathematical and Computational Applications*, 28(1):6.

Ramos-Figueroa, O., Quiroz-Castellanos, M., Mezura-Montes, E., and Kharel, R. (2021). Variation operators for grouping genetic algorithms: A review. *Swarm and Evolutionary Computation*, 60:100796.

Ramos-Figueroa, O., Quiroz-Castellanos, M., Mezura-Montes, E., and Schütze, O. (2020). Metaheuristics to solve grouping problems: A review and a case study. *Swarm and Evolutionary Computation*, 53:100643.

Rossi, A., Singh, A., and Sevaux, M. (2010). A metaheuristic for the fixed job scheduling problem under spread time constraints. *Computers & operations research*, 37(6):1045–1054.

Rousseeuw, P. J. (1987). Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65.

Rumsey, D. J. (2007). *Intermediate statistics for dummies*. John Wiley & Sons.

Saha, S., Seal, D. B., Ghosh, A., and Dey, K. N. (2016). A novel gene ranking method using wilcoxon rank sum test and genetic algorithm. *International Journal of Bioinformatics Research and Applications*, 12(3):263–279.

Stone, M. (2016). *A field guide to digital color*. CRC Press.

Sultana, F., Sufian, A., and Dutta, P. (2020). Evolution of image segmentation using deep convolutional neural network: A survey. *Knowledge-Based Systems*, 201:106062.

Tao, W.-B., Tian, J.-W., and Liu, J. (2003). Image segmentation by three-level thresholding based on maximum fuzzy entropy and genetic algorithm. *Pattern Recognition Letters*, 24(16):3069–3078.

Turi, R. H. (2001). *Clustering-based colour image segmentation*. PhD thesis, Monash University.

Welch, W. J. (1982). Algorithmic complexity: three np-hard problems in computational statistics. *Journal of Statistical Computation and Simulation*, 15(1):17–25.

Yu, X. and Gen, M. (2010). *Introduction to evolutionary algorithms*. Springer Science & Business Media.

Yusoh, Z. I. M. and Tang, M. (2012). Clustering composite saas components in cloud computing using a grouping genetic algorithm. In *2012 IEEE Congress on Evolutionary Computation*, pages 1–8. IEEE.

# Apéndice A

# Estadísticas completas de los resultados de la segmentación de imágenes.

Tabla A.1: Estadísticas por imagen y por algoritmo para la segmentación del banco de imágenes de control.

| **Imagen** | **Segmentos** | **GGA-CGT-RGB-IS** | | | **GGA-CGT-RGB-IS+IE** | | |
|---|---|---|---|---|---|---|---|
| | | **Mejor** | **Media** | **Peor** | **Mejor** | **Media** | **Peor** |
| *0_5* | 5 | 1 | 1.93 | 3 | 2 | 2.03 | 3 |
| *1_5* | 5 | 1 | 2.07 | 3 | 2 | 2.03 | 3 |
| *0_6* | 6 | 1 | 1.9 | 3 | 2 | 2.07 | 3 |
| *1_6* | 6 | 1 | 1.87 | 4 | 2 | 2.1 | 3 |
| *0_7* | 7 | 1 | 2.37 | 4 | 2 | 2.6 | 4 |
| *1_7* | 7 | 1 | 2.23 | 3 | 2 | 2.1 | 3 |
| *0_8* | 8 | 2 | 2.8 | 5 | 2 | 2.63 | 3 |
| *1_8* | 8 | 2 | 3.2 | 5 | 2 | 2.8 | 4 |
| *0_9* | 9 | 2 | 3.23 | 5 | 2 | 2.47 | 3 |
| *1_9* | 9 | 2 | 3.2 | 5 | 2 | 2.63 | 4 |
| *0_10* | 10 | 3 | 3.87 | 6 | 2 | 3 | 4 |
| *1_10* | 10 | 2 | 3.6 | 5 | 2 | 2.8 | 4 |
| *0_11* | 11 | 2 | 4.17 | 6 | 2 | 3.17 | 4 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| *1_11* | 11 | 3 | 4.1 | 6 | 2 | 3.03 | 4 |
| *0_12* | 12 | 2 | 4.5 | 9 | 2 | 3.23 | 4 |
| *1_12* | 12 | 3 | 4.67 | 7 | 2 | 3.3 | 5 |
| *0_13* | 13 | 3 | 4.8 | 9 | 2 | 3.23 | 4 |
| *1_13* | 13 | 2 | 5.2 | 10 | 2 | 3.57 | 4 |
| *0_14* | 14 | 3 | 4.83 | 8 | 2 | 3.23 | 4 |
| *1_14* | 14 | 3 | 5.17 | 8 | 2 | 3.5 | 4 |
| *0_15* | 15 | 3 | 5.53 | 10 | 2 | 3.6 | 6 |
| *1_15* | 15 | 4 | 6.33 | 10 | 2 | 3.83 | 5 |
| *0_16* | 16 | 4 | 7.2 | 14 | 2 | 3.67 | 5 |
| *1_16* | 16 | 4 | 6.93 | 10 | 3 | 4.33 | 6 |
| *0_17* | 17 | 4 | 7.03 | 10 | 3 | 4.63 | 7 |
| *1_17* | 17 | 5 | 7.83 | 13 | 3 | 4.93 | 8 |
| *0_18* | 18 | 4 | 6.8 | 9 | 3 | 4.4 | 5 |
| *1_18* | 18 | 5 | 7.93 | 11 | 4 | 4.97 | 6 |
| *0_19* | 19 | 6 | 8.27 | 12 | 4 | 4.97 | 6 |
| *1_19* | 19 | 6 | 8.17 | 11 | 4 | 5 | 6 |
| *0_20* | 20 | 6 | 9.7 | 15 | 4 | 5.6 | 7 |
| *1_20* | 20 | 5 | 9.9 | 15 | 3 | 5.23 | 6 |
| *0_21* | 21 | 6 | 9.43 | 14 | 3 | 4.87 | 6 |
| *1_21* | 21 | 6 | 9.8 | 16 | 4 | 5.9 | 9 |
| *0_22* | 22 | 6 | 9.57 | 14 | 4 | 5.4 | 7 |
| *1_22* | 22 | 7 | 12.33 | 21 | 4 | 6.07 | 9 |
| *0_23* | 23 | 7 | 12.37 | 20 | 4 | 5.97 | 7 |
| *1_23* | 23 | 7 | 15.43 | 29 | 4 | 6.67 | 10 |
| *0_24* | 24 | 5 | 10.53 | 14 | 4 | 5.7 | 7 |
| *1_24* | 24 | 7 | 13.17 | 25 | 5 | 6.47 | 8 |
| *0_25* | 25 | 6 | 18.3 | 36 | 5 | 7.87 | 13 |
| *1_25* | 25 | 8 | 18.77 | 32 | 6 | 7.83 | 12 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| *0_26* | 26 | 9 | 15.13 | 23 | 5 | 6 | 7 |
| *1_26* | 26 | 9 | 26.27 | 47 | 5 | 8.3 | 11 |
| *0_27* | 27 | 12 | 20.13 | 44 | 5 | 7.17 | 11 |
| *1_27* | 27 | 10 | 20.6 | 35 | 5 | 7.7 | 10 |
| *0_28* | 28 | 8 | 13.73 | 30 | 4 | 6.17 | 9 |
| *1_28* | 28 | 9 | 14.57 | 22 | 3 | 6.27 | 8 |
| *0_29* | 29 | 11 | 23.37 | 44 | 6 | 8.57 | 14 |
| *1_29* | 29 | 8 | 13.63 | 18 | 5 | 6.8 | 9 |
| *0_30* | 30 | 9 | 19.37 | 35 | 5 | 7.43 | 10 |
| *1_30* | 30 | 11 | 16.63 | 33 | 4 | 6.87 | 9 |
| *0_31* | 31 | 11 | 29.14 | 70 | 5 | 7.37 | 9 |
| *1_31* | 31 | 10 | 18.87 | 38 | 5 | 7.37 | 10 |
| *0_32* | 32 | 7 | 14.47 | 25 | 4 | 6.17 | 8 |
| *1_32* | 32 | 10 | 34 | 78 | 5 | 8.07 | 10 |
| *0_33* | 33 | 12 | 30.13 | 80 | 6 | 7.43 | 9 |
| *1_33* | 33 | 12 | 27.43 | 72 | 5 | 7.17 | 9 |
| *0_34* | 34 | 8 | 14.73 | 28 | 4 | 5.93 | 7 |
| *1_34* | 34 | 8 | 17.53 | 36 | 5 | 6.5 | 8 |
| *0_35* | 35 | 9 | 16.93 | 26 | 4 | 6.4 | 9 |
| *1_35* | 35 | 8 | 17.07 | 32 | 6 | 7.37 | 8 |
| *0_36* | 36 | 8 | 18.2 | 43 | 5 | 6.53 | 8 |
| *1_36* | 36 | 14 | 28.97 | 80 | 5 | 7.47 | 10 |
| *0_37* | 37 | 10 | 17.9 | 41 | 5 | 6.6 | 9 |
| *1_37* | 37 | 14 | 22.27 | 52 | 5 | 7.33 | 9 |
| *0_38* | 38 | 10 | 15.27 | 23 | 4 | 6.07 | 7 |
| *1_38* | 38 | 12 | 17.8 | 27 | 5 | 6.7 | 8 |
| *0_39* | 39 | 10 | 14.6 | 24 | 5 | 6.17 | 7 |
| *1_39* | 39 | 9 | 24.03 | 37 | 6 | 8.17 | 10 |
| *0_40* | 40 | 9 | 13.4 | 17 | 5 | 6.23 | 8 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| *1_40* | 40 | 9 | 17.47 | 30 | 5 | 7.3 | 10 |
| *0_41* | 41 | 11 | 18.4 | 38 | 6 | 7.17 | 9 |
| *1_41* | 41 | 11 | 20.8 | 40 | 6 | 7.63 | 9 |
| *0_42* | 42 | 6 | 10.9 | 18 | 4 | 5.07 | 6 |
| *1_42* | 42 | 10 | 16.97 | 35 | 5 | 6.87 | 9 |
| *0_43* | 43 | 7 | 11.23 | 19 | 4 | 5.5 | 7 |
| *1_43* | 43 | 11 | 17.37 | 34 | 6 | 7.03 | 8 |
| *0_44* | 44 | 6 | 13.6 | 22 | 5 | 6.83 | 9 |
| *1_44* | 44 | 16 | 43.97 | 91 | 6 | 8.63 | 12 |
| *0_45* | 45 | 9 | 14.9 | 24 | 5 | 6.47 | 8 |
| *1_45* | 45 | 16 | 27.53 | 58 | 6 | 7.97 | 10 |
| *0_46* | 46 | 9 | 15.5 | 24 | 5 | 6.53 | 8 |
| *1_46* | 46 | 15 | 46.92 | 95 | 6 | 8.27 | 11 |
| *0_47* | 47 | 8 | 11.97 | 17 | 5 | 5.87 | 7 |
| *1_47* | 47 | 13 | 39.43 | 94 | 6 | 8.43 | 10 |
| *0_48* | 48 | 8 | 11.43 | 18 | 4 | 5.43 | 7 |
| *1_48* | 48 | 25 | 49.81 | 95 | 7 | 9.37 | 12 |
| *0_49* | 49 | 5 | 10.7 | 20 | 4 | 5.53 | 6 |
| *1_49* | 49 | 25 | 48.04 | 80 | 8 | 9.8 | 12 |
| *0_50* | 50 | 6 | 9.47 | 13 | 3 | 5.2 | 7 |
| *1_50* | 50 | 30 | 58.29 | 95 | 8 | 9.5 | 12 |
| *0_51* | 51 | 7 | 11.07 | 20 | 4 | 5.47 | 7 |
| *1_51* | 51 | 17 | 48.55 | 100 | 7 | 9.23 | 11 |
| *0_52* | 52 | 6 | 9.23 | 13 | 4 | 4.93 | 6 |
| *1_52* | 52 | 19 | 36.7 | 74 | 7 | 8.43 | 10 |
| *0_53* | 53 | 6 | 8.23 | 10 | 3 | 4.73 | 6 |
| *1_53* | 53 | 28 | 58.41 | 96 | 7 | 9.9 | 13 |
| *0_54* | 54 | 6 | 8.17 | 11 | 3 | 4.47 | 6 |
| *1_54* | 54 | 18 | 40.28 | 82 | 6 | 9.3 | 12 |

Tabla A.2: Estadísticas de RPD por imagen y por algoritmo para la segmentación de las imágenes de prueba de BSDS500.

| **Imagen** | **Segmentos** | **GGA-CGT-RGB** | | | **GGA-CGT-RGB-IS+IE** | | |
|---|---|---|---|---|---|---|---|
| | | **Mejor** | **Media** | **Peor** | **Mejor** | **Media** | **Peor** |
| *12074* | 2 | 0.01114084 | 0.01429019 | 0.01915057 | 0.0064591 | 0.00635922 | 0.00612624 |
| *16052* | 2 | 0.0312293 | 0.0311992 | 0.03114943 | 0.03123119 | 0.03119793 | 0.03114535 |
| *28096* | 2 | 0.03101039 | 0.01899144 | -0.2348485 | 0.03100879 | 0.01898115 | -0.234866 |
| *41025* | 2 | 0.01988197 | 0.02109833 | 0.02273142 | 0.02195598 | 0.02060045 | 0.01980245 |
| *80099* | 2 | 0.03107936 | 0.01534191 | -0.1337029 | 0.03074333 | 0.01497938 | -0.1340322 |
| *94079* | 2 | 0.02706746 | 0.02617682 | 0.02607414 | 0.02707657 | 0.0259177 | 0.02533306 |
| *100098* | 2 | 0.02984551 | 0.02989919 | 0.03003646 | 0.02988264 | 0.02981492 | 0.02978051 |
| *130034* | 2 | 0.03123233 | 0.03117906 | 0.03112397 | 0.03123233 | 0.03117882 | 0.03112294 |
| *134008* | 2 | 0.02391981 | 0.02200795 | 0.02104433 | 0.02394717 | 0.02169325 | 0.02036144 |
| *135037* | 2 | 0.02949026 | 0.02993421 | 0.0302792 | 0.02997936 | 0.02997936 | 0.02997936 |
| *317080* | 2 | -0.0123461 | -0.004703 | 0.00071446 | -0.0085817 | -0.0087552 | -0.0093262 |
| *43083* | 3 | 0.0183311 | 0.01994898 | 0.02127751 | 0.0191787 | 0.0191474 | 0.01900763 |
| *60079* | 3 | 0.02409049 | 0.02476767 | 0.02664606 | 0.02722305 | 0.02654248 | 0.02581587 |
| *87065* | 3 | 0.03064159 | 0.03036448 | 0.03014367 | 0.03081753 | 0.0304882 | 0.0302022 |
| *106025* | 3 | 0.02432641 | 0.02495115 | 0.02640206 | 0.02500595 | 0.02500595 | 0.02500595 |
| *113016* | 3 | 0.02856542 | 0.02841745 | 0.02821098 | 0.02876424 | 0.0285127 | 0.02804806 |
| *134052* | 3 | 0.0217421 | 0.0216718 | 0.0211965 | 0.02372504 | 0.02303846 | 0.02163516 |
| *138032* | 3 | 0.02571999 | 0.02644863 | 0.02838985 | 0.02873379 | 0.02867791 | 0.02857507 |
| *176039* | 3 | 0.02805581 | 0.0282317 | 0.02865754 | 0.02827591 | 0.02816872 | 0.02804695 |
| *178054* | 3 | 0.03111106 | 0.03109701 | 0.03109366 | 0.03110904 | 0.03108099 | 0.03106334 |
| *183087* | 3 | 0.02812092 | 0.02783324 | 0.02820659 | 0.02950932 | 0.02888166 | 0.02807554 |
| *187083* | 3 | 0.0307022 | 0.03066316 | 0.03060225 | 0.03076819 | 0.03066559 | 0.03053552 |
| *216066* | 3 | 0.02941732 | 0.02548036 | -0.0095 | 0.02992642 | 0.02580606 | -0.0093545 |
| *299091* | 3 | -0.0254996 | -0.023258 | -0.0188213 | -0.0233504 | -0.0239296 | -0.0243925 |
| *311081* | 3 | 0.03005459 | 0.0298888 | 0.02970345 | 0.03030138 | 0.0299949 | 0.0296535 |
| *22013* | 4 | 0.02910353 | 0.02925301 | 0.02921994 | 0.02922094 | 0.02918941 | 0.02887659 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| *23084* | 4 | 0.02080029 | -0.0032591 | -0.0560425 | 0.02176319 | -0.0034376 | -0.0582262 |
| *24004* | 4 | 0.03093258 | 0.03097194 | 0.0310219 | 0.0309863 | 0.03095085 | 0.03091894 |
| *26031* | 4 | 0.00814348 | 0.00577594 | 0.00067357 | 0.00837373 | 0.00494038 | -0.0026016 |
| *28075* | 4 | -0.0167794 | -0.0140434 | -0.0118018 | -0.0158156 | -0.0160312 | -0.0177739 |
| *41004* | 4 | 0.02603005 | 0.02615201 | 0.02628278 | 0.02624563 | 0.02535292 | 0.02429644 |
| *55067* | 4 | 0.02975605 | 0.03015217 | 0.0304551 | 0.02983537 | 0.02983537 | 0.02983537 |
| *66075* | 4 | 0.00483677 | 0.00621442 | 0.00775374 | 0.00510249 | 0.00510249 | 0.00510249 |
| *100080* | 4 | 0.02953928 | -0.0026828 | -0.1391164 | 0.02982778 | -0.0025466 | -0.1391123 |
| *108041* | 4 | 0.02854722 | 0.0283133 | 0.0274743 | 0.02907761 | 0.02850671 | 0.02717442 |
| *108073* | 4 | 0.01898604 | 0.01892374 | 0.01410312 | 0.01985575 | 0.01902812 | 0.01266101 |
| *135069* | 4 | 0.02806833 | 0.00575668 | -0.2091768 | 0.02954936 | 0.00688911 | -0.208625 |
| *188091* | 4 | 0.01241834 | 0.01170768 | 0.00366761 | 0.014025 | 0.01196196 | 0.00239961 |
| *207056* | 4 | 0.02322622 | 0.02337108 | 0.0172115 | 0.02363603 | 0.02306704 | 0.01576299 |
| *227040* | 4 | 0.02863305 | 0.02088957 | 0.00884512 | 0.0287964 | 0.02093713 | 0.00880922 |
| *271031* | 4 | 0.02886183 | 0.02796862 | 0.02777577 | 0.02892668 | 0.02747219 | 0.02652916 |
| *311068* | 4 | 0.01980766 | -0.0062905 | 0.00261998 | 0.02006791 | -0.0085962 | -0.0146851 |
| *35008* | 5 | 0.02472351 | 0.02173138 | 0.0204787 | 0.02494386 | 0.02129048 | 0.01926044 |
| *42078* | 5 | -0.0019453 | -0.0031096 | -0.0015892 | -0.0006656 | -0.0030128 | -0.0043667 |
| *48055* | 5 | 0.02986183 | 0.00817285 | -0.0149149 | 0.03011134 | 0.00817289 | -0.0151006 |
| *71046* | 5 | 0.02874716 | -0.0021884 | -0.1259973 | 0.02920788 | -0.0023322 | -0.1264524 |
| *92059* | 5 | 0.03008283 | 0.03023202 | 0.0294075 | 0.03024744 | 0.0301659 | 0.02905181 |
| *100075* | 5 | 0.0286354 | 0.02080816 | -0.0183898 | 0.02889059 | 0.02079025 | -0.0187838 |
| *109034* | 5 | 0.02883104 | 0.02825996 | 0.02834713 | 0.02913298 | 0.02820065 | 0.02722926 |
| *113044* | 5 | 0.02669151 | 0.02654046 | 0.02336958 | 0.02770637 | 0.02671621 | 0.02235308 |
| *122048* | 5 | 0.00378953 | 0.00345136 | 0.00127857 | 0.00440301 | 0.00302406 | -0.0009392 |
| *124084* | 5 | 0.02495442 | 0.00584673 | -0.0785486 | 0.02519871 | 0.00545246 | -0.0800802 |
| *147021* | 5 | 0.02126278 | 0.02003316 | 0.01259166 | 0.02391669 | 0.02073871 | 0.01039331 |
| *157036* | 5 | 0.0306892 | 0.03021778 | 0.03022287 | 0.03068805 | 0.03011314 | 0.02995656 |
| *163062* | 5 | 0.02761515 | 0.02752871 | 0.02771173 | 0.02814654 | 0.02777197 | 0.02631984 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| *189003* | 5 | 0.01482124 | 0.01317794 | 0.00712151 | 0.01641304 | 0.01174158 | -0.0006746 |
| *238011* | 5 | 0.02342395 | 0.00655356 | -0.1674057 | 0.0243241 | 0.00520364 | -0.1702256 |
| *254033* | 5 | 0.02391014 | -0.0480233 | -0.1762842 | 0.02466737 | -0.0478441 | -0.1769049 |
| *326038* | 5 | 0.01900988 | 0.01844326 | 0.01448271 | 0.02281908 | 0.02059794 | 0.01304708 |
| *374020* | 5 | 0.03071166 | -0.0026676 | -0.0420098 | 0.03068275 | -0.0062921 | -0.0456633 |
| *12003* | 6 | 0.03068404 | 0.02103172 | 0.01699605 | 0.03067191 | 0.02089922 | 0.0167003 |
| *24063* | 6 | 0.02387742 | -0.1318503 | -0.2853844 | 0.02447633 | -0.1319182 | -0.2860186 |
| *33066* | 6 | 0.02866057 | 0.01701222 | -0.0321425 | 0.0291866 | 0.01721665 | -0.0323393 |
| *42044* | 6 | 0.01736998 | 0.01891605 | 0.02189427 | 0.0179476 | 0.0179476 | 0.0179476 |
| *56028* | 6 | 0.03042995 | 0.02005636 | -0.0180521 | 0.03062625 | 0.02004833 | -0.0193593 |
| *61060* | 6 | 0.02700191 | 0.0248085 | 0.0180562 | 0.02734506 | 0.02466801 | 0.01732288 |
| *65010* | 6 | 0.02688125 | 0.01822994 | 0.00658822 | 0.02808846 | 0.0177803 | 0.00490203 |
| *67079* | 6 | 0.02828907 | 0.01223891 | -0.0025238 | 0.02948712 | 0.01283895 | -0.0023427 |
| *95006* | 6 | 0.02461467 | 0.00059019 | -0.009388 | 0.0251152 | -0.0003555 | -0.0137214 |
| *103041* | 6 | -0.0063847 | -0.0263851 | -0.1038163 | -0.0049769 | -0.026338 | -0.1053043 |
| *105053* | 6 | 0.0197092 | -0.0012855 | -0.0707981 | 0.02321172 | 0.00067646 | -0.0705048 |
| *147062* | 6 | 0.02451924 | 0.02124644 | -0.0686743 | 0.02515121 | 0.0208106 | -0.0703095 |
| *155060* | 6 | 0.02818913 | 0.02346117 | -0.079891 | 0.02837704 | 0.02327987 | -0.0807552 |
| *159029* | 6 | 0.02724252 | 0.0146117 | -0.0134201 | 0.02737136 | 0.01449463 | -0.014367 |
| *159091* | 6 | 0.02997857 | 0.02943949 | 0.02935837 | 0.03029433 | 0.02939138 | 0.02891887 |
| *173036* | 6 | 0.00698819 | 0.00860326 | 0.00497923 | 0.0094046 | 0.00869485 | 0.00195698 |
| *247085* | 6 | 0.03033088 | 0.02606242 | 0.01359237 | 0.0303773 | 0.02594742 | 0.01331231 |
| *323016* | 6 | 0.02090341 | -0.0867155 | -0.1995217 | 0.02202444 | -0.0865779 | -0.2009938 |
| *353013* | 6 | 0.01185319 | 0.00551902 | -0.0856671 | 0.01262453 | 0.00573096 | -0.0860981 |
| *388016* | 6 | 0.02410644 | 0.01752939 | -0.0167168 | 0.02513261 | 0.01266764 | -0.0228344 |
| *2092* | 7 | 0.02864672 | -0.0211094 | -0.149885 | 0.02956647 | -0.0207606 | -0.1504379 |
| *46076* | 7 | 0.02542167 | -0.0173963 | -0.0481311 | 0.0283868 | -0.0160404 | -0.0481597 |
| *61086* | 7 | 0.03097225 | 0.01943798 | -0.0297849 | 0.03098453 | 0.01934721 | -0.0300552 |
| *66039* | 7 | 0.00970962 | 0.00314394 | -0.0647305 | 0.01133398 | 0.00324129 | -0.0675099 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| *105019* | 7 | 0.01619347 | -0.0203294 | -0.0456132 | 0.01900862 | -0.0188595 | -0.0459199 |
| *164074* | 7 | 0.02573496 | 0.02644822 | 0.02667761 | 0.02607151 | 0.02595612 | 0.02380567 |
| *183055* | 7 | 0.02108762 | 0.01697586 | 0.00720705 | 0.02223461 | 0.01707962 | 0.00537888 |
| *187003* | 7 | 0.02581491 | 0.02506648 | 0.0233276 | 0.02672565 | 0.02489561 | 0.02116055 |
| *236017* | 7 | 0.02461478 | 0.02142559 | 0.0168415 | 0.02523657 | 0.02088609 | 0.01348583 |
| *242078* | 7 | 0.02906685 | 0.00930356 | -0.0495922 | 0.03012019 | 0.00981314 | -0.0495809 |
| *254054* | 7 | 0.02986438 | 0.02058732 | -0.0029696 | 0.03010141 | 0.02063406 | -0.0035046 |
| *260081* | 7 | 0.02814892 | 0.02746449 | 0.02232099 | 0.02844532 | 0.02743343 | 0.02054627 |
| *309004* | 7 | 0.02950617 | 0.02631029 | 0.01315878 | 0.02972577 | 0.02579945 | 0.00958847 |
| *368078* | 7 | 0.030048 | 0.01427749 | -0.0134736 | 0.02996144 | 0.01152067 | -0.0164926 |
| *20008* | 8 | 0.02350741 | 0.0070925 | -0.0754897 | 0.0252631 | 0.00740641 | -0.07644 |
| *35058* | 8 | 0.02962957 | 0.01346344 | -0.034546 | 0.02958852 | 0.01267696 | -0.0361476 |
| *112082* | 8 | 0.02894272 | 0.01112207 | -0.0446125 | 0.03006922 | 0.01124766 | -0.0453264 |
| *117054* | 8 | 0.02773587 | 0.00925438 | -0.0022634 | 0.02858336 | 0.00921708 | -0.0067966 |
| *138078* | 8 | 0.02091278 | -0.0109304 | -0.0862802 | 0.02358911 | -0.0104216 | -0.0893586 |
| *156079* | 8 | 0.02277054 | 0.02116255 | 0.00516926 | 0.0232019 | 0.01997094 | 0.00080812 |
| *176035* | 8 | 0.0242982 | 0.00692181 | -0.0558576 | 0.02453427 | 0.00651392 | -0.0574425 |
| *187029* | 8 | 0.02854761 | -0.0037188 | -0.077995 | 0.02905812 | -0.0039604 | -0.0787678 |
| *187071* | 8 | 0.02979479 | 0.01730597 | 0.00335539 | 0.03006246 | 0.01728082 | 0.0031602 |
| *198023* | 8 | 0.02102085 | 0.00336228 | -0.0981988 | 0.02257422 | 0.00244717 | -0.1014254 |
| *209070* | 8 | 0.02820677 | 0.02359665 | 0.00080613 | 0.02820871 | 0.02330644 | 3.14E-05 |
| *249061* | 8 | 0.02926956 | 0.00277721 | -0.1322704 | 0.02986741 | 0.00291373 | -0.1328794 |
| *249087* | 8 | 0.02996059 | 0.02192171 | 0.01865899 | 0.02999445 | 0.02140415 | 0.01416498 |
| *310007* | 8 | 0.02847719 | -0.0125056 | -0.0513681 | 0.02903614 | -0.0124704 | -0.0520352 |
| *314016* | 8 | 0.02313467 | 0.0166293 | 0.00568754 | 0.02456013 | 0.01702759 | 0.00480878 |
| *35070* | 9 | 0.01554342 | 0.01394361 | 0.00060996 | 0.01710617 | 0.01302883 | -0.0042171 |
| *90076* | 9 | 0.0224975 | -0.0091337 | -0.0419929 | 0.02317904 | -0.0101003 | -0.0441382 |
| *118020* | 9 | 0.02503448 | 0.01323273 | -0.0409223 | 0.02648268 | 0.01310388 | -0.044106 |
| *144067* | 9 | 0.02796844 | 0.00448702 | -0.0579506 | 0.02834874 | 0.00390838 | -0.0600414 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| *170054* | 9 | 0.0223537 | 0.00702872 | -0.0261948 | 0.02291106 | 0.00680264 | -0.0306175 |
| *188063* | 9 | 0.02098041 | -0.0032865 | -0.029451 | 0.02352887 | -0.0028993 | -0.0314434 |
| *225017* | 9 | 0.02031066 | 0.00275974 | -0.0641321 | 0.02164084 | 0.00261004 | -0.0702252 |
| *246053* | 9 | 0.02483555 | 0.01827541 | 0.01055773 | 0.02615798 | 0.0173907 | 0.00393091 |
| *268002* | 9 | 0.01754458 | 0.01577216 | -0.0314608 | 0.01902772 | 0.01433759 | -0.0364009 |
| *277095* | 9 | 0.02889439 | 0.01157212 | -0.0496595 | 0.02926631 | 0.01094727 | -0.0528582 |
| *376020* | 9 | 0.02983978 | 0.01943212 | -0.0276605 | 0.02960479 | 0.01046793 | -0.0368191 |
| *35091* | 10 | 0.02838815 | 0.02513287 | 0.00450601 | 0.02911528 | 0.02481294 | 0.00245617 |
| *45077* | 10 | 0.02703655 | 0.01326964 | -0.0346844 | 0.02776386 | 0.01302838 | -0.0364784 |
| *54005* | 10 | 0.03035501 | 0.02807831 | 0.02404684 | 0.03057609 | 0.02749444 | 0.02220146 |
| *68077* | 10 | 0.02658978 | 0.01819106 | -0.0433075 | 0.02672885 | 0.01676479 | -0.0471779 |
| *106020* | 10 | 0.02866499 | 0.02586103 | -0.0106203 | 0.02895158 | 0.02463213 | -0.0157943 |
| *140075* | 10 | 0.02523497 | 0.00273533 | -0.0567708 | 0.02649903 | 0.00273626 | -0.0588709 |
| *145014* | 10 | 0.0278359 | 0.01934497 | -0.0070538 | 0.02813306 | 0.01925112 | -0.0076963 |
| *153077* | 10 | 0.01936189 | 0.00500273 | -0.0741211 | 0.02256429 | 0.00606973 | -0.0771046 |
| *153093* | 10 | 0.01912231 | -0.0154063 | -0.0671123 | 0.02261657 | -0.0140273 | -0.0680846 |
| *166081* | 10 | 0.02823545 | 0.01762681 | 0.00986017 | 0.02858831 | 0.01592895 | 0.00035494 |
| *176019* | 10 | 0.02820738 | 0.02582605 | 0.00619718 | 0.02788152 | 0.024455 | 0.00356231 |
| *181018* | 10 | 0.020112 | 0.01499211 | -0.0131646 | 0.0207186 | 0.01388808 | -0.0181339 |
| *187039* | 10 | 0.02184206 | 0.01083783 | -0.0232318 | 0.02451899 | 0.01089514 | -0.0271732 |
| *189011* | 10 | 0.0284797 | 0.02343742 | 0.00378892 | 0.02864959 | 0.02282613 | 0.00207587 |
| *246016* | 10 | 0.02936985 | 0.00476168 | -0.0631672 | 0.02987375 | 0.00439448 | -0.0654514 |
| *271008* | 10 | 0.02263703 | -0.0067665 | -0.0185706 | 0.02473526 | -0.0066609 | -0.0217872 |
| *372047* | 10 | 0.02858875 | -0.0166515 | -0.0963 | 0.02905432 | -0.0254373 | -0.1062519 |
| *374067* | 10 | 0.02471453 | 0.01287236 | -0.0482203 | 0.02537204 | 0.0092381 | -0.0527542 |
| *43070* | 11 | 0.01072696 | 0.00796981 | 0.01125583 | 0.012324 | 0.00627014 | 0.00282538 |
| *97017* | 11 | 0.02754387 | -0.0165292 | -0.1258268 | 0.02823266 | -0.017723 | -0.1322999 |
| *140055* | 11 | 0.01918611 | -0.0050683 | -0.0552614 | 0.01954691 | -0.0055102 | -0.0569248 |
| *145053* | 11 | 0.03042127 | 0.01870982 | 0.00517273 | 0.0304575 | 0.01801553 | 0.0019711 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| *198004* | 11 | 0.02769986 | 0.0049695 | -0.0327491 | 0.02778272 | 0.00448092 | -0.0340377 |
| *227046* | 11 | 0.02935356 | -0.0013816 | -0.0814432 | 0.02990102 | -0.0028962 | -0.0897165 |
| *385028* | 11 | 0.02768168 | 0.00948968 | -0.011188 | 0.02799144 | 0.00350278 | -0.0203328 |
| *15088* | 12 | 0.02977247 | 0.01872095 | 0.00778503 | 0.02991293 | 0.01673654 | 0.00208454 |
| *118035* | 12 | 0.01657698 | -0.0438868 | -0.1286852 | 0.01697463 | -0.044832 | -0.1337705 |
| *169012* | 12 | 0.03031767 | 0.02590762 | 0.01826781 | 0.02961751 | 0.0230275 | 0.01025646 |
| *172032* | 12 | 0.02657527 | -0.0371511 | -0.0743672 | 0.02689735 | -0.0385118 | -0.0792017 |
| *188005* | 12 | 0.03020695 | 0.00341915 | -0.0686365 | 0.03011734 | 0.0022748 | -0.0722834 |
| *293029* | 12 | 0.03055928 | 0.01766792 | -0.0054462 | 0.03053974 | 0.01636304 | -0.0113989 |
| *301007* | 12 | 0.02556604 | 0.00427465 | -0.0475265 | 0.02584007 | 0.00328018 | -0.0512362 |
| *8049* | 13 | 0.02811593 | 0.01796346 | -0.0359878 | 0.02808371 | 0.01602199 | -0.0405357 |
| *23025* | 13 | 0.02312795 | -0.0086795 | -0.0487019 | 0.02384836 | -0.009368 | -0.0521481 |
| *65132* | 13 | 0.03098773 | 0.0191419 | -0.0158888 | 0.02992922 | 0.01679135 | -0.0221556 |
| *76002* | 13 | 0.02818695 | -0.0115256 | -0.0911854 | 0.02736726 | -0.0142729 | -0.099964 |
| *113009* | 13 | 0.02240742 | 0.01574345 | 0.00279532 | 0.02164192 | 0.01319357 | -0.0073319 |
| *159045* | 13 | 0.01521953 | 0.00637019 | 0.00335596 | 0.01596505 | 0.00361702 | -0.0058926 |
| *161062* | 13 | 0.02076539 | -0.0393271 | -0.2550205 | 0.02119841 | -0.0404638 | -0.2574604 |
| *198054* | 13 | 0.02865296 | 0.00645955 | -0.0368112 | 0.0291522 | 0.0051466 | -0.0443649 |
| *292066* | 13 | 0.02032634 | 0.00153909 | -0.0185115 | 0.02192699 | 0.00083965 | -0.0248431 |
| *365073* | 13 | 0.0327668 | 0.04382657 | 0.02692576 | 0.02936281 | 0.02199989 | -0.0003403 |
| *8143* | 14 | 0.02866422 | 0.02634667 | 0.0164802 | 0.02843309 | 0.02359236 | 0.00872689 |
| *15004* | 14 | 0.03071226 | 0.01469075 | 0.0048314 | 0.03012845 | 0.0128342 | -0.0030852 |
| *22090* | 14 | 0.02445351 | -0.0187262 | -0.0552063 | 0.02804766 | -0.0188803 | -0.058229 |
| *181091* | 15 | 0.0242685 | 0.00037333 | -0.0360031 | 0.02549698 | -0.0024037 | -0.0434575 |
| *202012* | 15 | 0.01916856 | 0.00897963 | -0.0086491 | 0.01866696 | 0.00549714 | -0.0224514 |
| *216053* | 15 | 0.02531497 | 0.01620733 | -0.0024461 | 0.02481451 | 0.01397352 | -0.0099099 |
| *365025* | 15 | 0.02587294 | 0.01212065 | -0.0238067 | 0.02515743 | 0.00877832 | -0.0334095 |
| *22093* | 16 | 0.02835526 | 0.00507767 | -0.0315259 | 0.0254971 | -0.0016543 | -0.0411209 |
| *27059* | 16 | 0.02303537 | -0.011603 | -0.0567715 | 0.02277034 | -0.0142317 | -0.0641748 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| *59078* | 16 | 0.0176248 | 0.00085738 | -0.072885 | 0.01873681 | -0.0026768 | -0.0794077 |
| *126039* | 16 | 0.03062694 | 0.00265775 | -0.0574891 | 0.02887436 | -0.002552 | -0.0674646 |
| *163014* | 16 | 0.02181607 | -0.0172441 | -0.058511 | 0.02130335 | -0.0216357 | -0.0688955 |
| *216041* | 16 | 0.02163331 | -0.0151505 | -0.0690202 | 0.02186221 | -0.0182129 | -0.0766023 |
| *231015* | 16 | 0.03139138 | 0.02531011 | 0.01948128 | 0.02781214 | 0.01921501 | 0.0081093 |
| *232038* | 16 | 0.02091527 | -0.0149905 | -0.1914189 | 0.02236703 | -0.0180206 | -0.198735 |
| *239096* | 17 | 0.03076345 | 0.02449413 | 0.00379913 | 0.02868139 | 0.02060831 | -0.0036158 |
| *55075* | 18 | 0.01711923 | 0.00160168 | -0.0145253 | 0.01526455 | -0.004812 | -0.0285229 |
| *104022* | 18 | 0.02849485 | 0.02699153 | 0.02435794 | 0.02830457 | 0.01988237 | 0.01258257 |
| *181079* | 18 | 0.028817 | 0.01940239 | -0.0103888 | 0.02600973 | 0.01197632 | -0.0216843 |
| *196015* | 18 | 0.03017661 | 0.0228986 | 0.01240016 | 0.02680419 | 0.01434519 | -0.0012162 |
| *23080* | 20 | 0.02618055 | 0.01625617 | -0.0085336 | 0.0230731 | 0.00787549 | -0.0203894 |
| *274007* | 21 | 0.02750141 | 0.01740979 | -0.0048263 | 0.0227047 | 0.00543092 | -0.0216904 |
| *285036* | 21 | 0.02596909 | 0.00963315 | -0.0291423 | 0.01989367 | -0.0028032 | -0.0500091 |
| *368016* | 22 | 0.03713759 | 0.02336456 | -0.0118008 | 0.0191782 | -0.0100278 | -0.0488254 |
| *35010* | 23 | 0.02913881 | 0.01771557 | -0.0136398 | 0.0236316 | 0.00638086 | -0.0306991 |
| *25098* | 24 | 0.02742361 | 0.01767667 | 0.00341688 | 0.01982578 | 0.00284308 | -0.0208681 |
| *361084* | 24 | 0.02856519 | 0.029537 | 0.00110028 | 0.01931156 | -0.0045195 | -0.0385123 |
| *239007* | 25 | 0.02673854 | 0.01230758 | -0.0209005 | 0.0180697 | -0.001055 | -0.04173 |
| *245051* | 26 | 0.01707808 | -0.0192728 | -0.1188913 | 0.01277115 | -0.0307048 | -0.1365322 |
| *151087* | 27 | 0.0379785 | 0.02374125 | -0.0042159 | 0.02525605 | 0.00548806 | -0.0280398 |
| *78019* | 28 | 0.02221047 | 0.00831266 | -0.0467259 | 0.01184165 | -0.0096407 | -0.0715033 |
| *286092* | 28 | 0.02677544 | -0.0065833 | -0.0882315 | 0.01854326 | -0.0191805 | -0.1057758 |
| *253036* | 31 | 0.03286901 | 0.00377277 | -0.1687859 | 0.0167651 | -0.0213952 | -0.1970588 |
| *65074* | 33 | 0.03668931 | 0.01475357 | -0.0144981 | 0.02182224 | -0.0070172 | -0.0419099 |
| *65019* | 34 | 0.03709914 | 0.02967653 | -0.0011673 | 0.02060966 | 0.00429043 | -0.0334489 |
| *370036* | 41 | 0.06269746 | 0.06904229 | 0.06193436 | 0.01915933 | 0.0029068 | -0.01301 |
| *302003* | 43 | 0.044893 | 0.03088252 | -0.0097728 | 0.0253633 | 0.00252104 | -0.0465788 |
| *376001* | 54 | 0.04656476 | 0.03136052 | 0.00878424 | 0.01460359 | -0.0152631 | -0.0499157 |

# Apéndice B

# Resultados de las pruebas estadísticas de la segmentación de imágenes

En la primera columna de la Tabla B.1 se muestra el nombre de la imagen. Mientras que en la segunda se muestra el número de segmentos de cada imagen. En la tercera columna se muestra el *p-value* para cada imagen, donde se resalta en negritas el valor que es significativo (menor que 0.05).

Tabla B.1: Resultados de la prueba de Wilcoxon Rank-Sum para la segmentación de imágenes del banco de imágenes de control.

| **Imagen** | **Segmentos** | ***p-value*** |
|---|---|---|
| 0_5 | 5 | 0.529782 |
| 1_5 | 5 | 0.813005 |
| 0_6 | 6 | 0.300711 |
| 1_6 | 6 | 0.10547 |
| 0_7 | 7 | 0.166866 |
| 1_7 | 7 | 0.363222 |
| 0_8 | 8 | 0.468798 |
| 1_8 | 8 | 0.064595 |
| 1_9 | 9 | **0.005322** |
| 0_9 | 9 | **0.000129** |

| | | |
|---|---|---|
| 1_10 | 10 | **0.000879** |
| 0_10 | 10 | **0.000104** |
| 0_11 | 11 | **4.35E-05** |
| 1_11 | 11 | **5.86E-06** |
| 0_12 | 12 | **0.000213** |
| 1_12 | 12 | **9.85E-06** |
| 0_13 | 13 | **1.31E-07** |
| 1_13 | 13 | **1.86E-06** |
| 0_14 | 14 | **1.16E-07** |
| 1_14 | 14 | **1.07E-07** |
| 0_15 | 15 | **7.32E-07** |
| 1_15 | 15 | **2.79E-09** |
| 1_16 | 16 | **6.53E-08** |
| 0_16 | 16 | **8.1E-10** |
| 0_17 | 17 | **8.86E-09** |
| 1_17 | 17 | **2.39E-08** |
| 0_18 | 18 | **1.62E-09** |
| 1_18 | 18 | **6.12E-10** |
| 0_19 | 19 | **7.76E-11** |
| 1_19 | 19 | **5.77E-11** |
| 0_20 | 20 | **1.39E-10** |
| 1_20 | 20 | **1.94E-09** |
| 0_21 | 21 | **4.08E-11** |
| 1_21 | 21 | **6.52E-09** |
| 1_22 | 22 | **8.56E-11** |
| 0_22 | 22 | **1.15E-10** |
| 0_23 | 23 | **4.98E-11** |
| 1_23 | 23 | **2.74E-10** |
| 0_24 | 24 | **2.87E-10** |

| | | |
|---|---|---|
| 1_24 | 24 | **7.76E-11** |
| 0_25 | 25 | **1.35E-09** |
| 1_25 | 25 | **1.54E-10** |
| 0_26 | 26 | **2.87E-11** |
| 1_26 | 26 | **1.21E-10** |
| 0_27 | 27 | **2.87E-11** |
| 1_27 | 27 | **3.18E-11** |
| 0_28 | 28 | **4.73E-11** |
| 1_28 | 28 | **2.87E-11** |
| 1_29 | 29 | **7.03E-11** |
| 0_29 | 29 | **4.73E-11** |
| 0_30 | 30 | **3.69E-11** |
| 1_30 | 30 | **2.87E-11** |
| 0_31 | 31 | **2.87E-11** |
| 1_31 | 31 | **3.02E-11** |
| 0_32 | 32 | **6.37E-11** |
| 1_32 | 32 | **3.51E-11** |
| 0_33 | 33 | **2.87E-11** |
| 1_33 | 33 | **2.87E-11** |
| 0_34 | 34 | **2.87E-11** |
| 1_34 | 34 | **3.88E-11** |
| 1_35 | 35 | **6.07E-11** |
| 0_35 | 35 | **3.02E-11** |
| 0_36 | 36 | **3.02E-11** |
| 1_36 | 36 | **2.87E-11** |
| 0_37 | 37 | **2.87E-11** |
| 1_37 | 37 | **2.87E-11** |
| 0_38 | 38 | **2.87E-11** |
| 1_38 | 38 | **2.87E-11** |

| | | |
|---|---|---|
| 0_39 | 39 | **2.87E-11** |
| 1_39 | 39 | **5.49E-11** |
| 0_40 | 40 | **2.87E-11** |
| 1_40 | 40 | **3.34E-11** |
| 1_41 | 41 | **2.87E-11** |
| 0_41 | 41 | **2.87E-11** |
| 1_42 | 42 | **2.87E-11** |
| 0_42 | 42 | **3.88E-11** |
| 1_43 | 43 | **2.87E-11** |
| 0_43 | 43 | **3.18E-11** |
| 0_44 | 44 | **6.41E-10** |
| 1_44 | 44 | **2.87E-11** |
| 0_45 | 45 | **2.87E-11** |
| 1_45 | 45 | **2.87E-11** |
| 0_46 | 46 | **2.87E-11** |
| 1_46 | 46 | **2.87E-11** |
| 1_47 | 47 | **2.87E-11** |
| 0_47 | 47 | **2.87E-11** |
| 0_48 | 48 | **2.87E-11** |
| 1_48 | 48 | **2.87E-11** |
| 0_49 | 49 | **2.74E-10** |
| 1_49 | 49 | **2.87E-11** |
| 0_50 | 50 | **8.99E-11** |
| 1_50 | 50 | **2.87E-11** |
| 0_51 | 51 | **3.88E-11** |
| 1_51 | 51 | **2.87E-11** |
| 0_52 | 52 | **3.34E-11** |
| 1_52 | 52 | **2.87E-11** |
| 1_53 | 53 | **2.87E-11** |

| | | |
|---|---|---|
| 0_53 | 53 | **3.18E-11** |
| 0_54 | 54 | **3.18E-11** |
| 1_54 | 54 | **2.87E-11** |

En la primera columna de la Tabla B.2 se muestra el nombre de la imagen. Mientras que en la segunda se muestra el número de segmentos de cada imagen. En las siguientes columnas se muestran los *p-values* de las dos pruebas estadísticas realizadas por imagen. La primera prueba es la de Kruskal-Wallis, donde se resaltan en negritas los valores signficativos (menores a 0.05). La segunda prueba es la de Wilcoxon Rank-Sum, cuyos resultados se muestran en las últimas dos columnas. En estas se presentan los valores para dos pares de algoritmos: uno comparando el GGA-CGT-RGB-IS+IE con la versión inicial y otro con K-Means. De igual manera, se resalta en negritas el valor que es significativo (menor que 0.05) para cada imagen.

Tabla B.2: Resultados de las pruebas estadísticas para la segmentación de imágenes del banco de prueba de BSDS500.

| **Imagen** | **No. Segmentos** | **Kruskal-Wallis** | **GGA-CGT-RGB-IS+IE vs GGA-CGT-RGB-IS** | **GGA-CGT-RGB-IS+IE vs K-Means** |
|---|---|---|---|---|
| 317080 | 2 | **8.94E-10** | 0.120575 | **3.13E-07** |
| 100098 | 2 | **3.52E-14** | **0.029757** | **2.87E-11** |
| 94079 | 2 | **3.58E-16** | **3.39E-07** | **2.87E-11** |
| 12074 | 2 | **1.34E-14** | **0.004128** | **2.87E-11** |
| 80099 | 2 | **1.07E-10** | 0.11708 | **8.12E-09** |
| 28096 | 2 | **1.6E-12** | **0.020278** | **5.32E-10** |
| 135037 | 2 | **1.55E-16** | **9.9E-07** | **2.87E-11** |
| 134008 | 2 | **1.05E-13** | 0.407711 | **2.87E-11** |
| 41025 | 2 | **6.87E-16** | **1.93E-06** | **2.87E-11** |
| 130034 | 2 | **2.23E-14** | 0.056496 | **2.87E-11** |
| 16052 | 2 | **5.95E-14** | 0.15581 | **2.87E-11** |
| 311081 | 3 | **9.16E-18** | **9.44E-11** | **2.87E-11** |

| | | | | |
|---|---|---|---|---|
| 106025 | 3 | **4.37E-18** | **3.31E-10** | **2.87E-11** |
| 113016 | 3 | **8.29E-18** | **9.44E-11** | **2.87E-11** |
| 299091 | 3 | **3.94E-15** | **0.000114** | **2.87E-11** |
| 216066 | 3 | **2.4E-13** | **2.87E-11** | **1.02E-07** |
| 43083 | 3 | **2.11E-14** | **0.007451** | **2.87E-11** |
| 183087 | 3 | **6.2E-18** | **3.51E-11** | **2.87E-11** |
| 87065 | 3 | **6.04E-18** | **2.87E-11** | **2.87E-11** |
| 176039 | 3 | **2.82E-16** | **2.47E-07** | **2.87E-11** |
| 134052 | 3 | **5.59E-18** | **2.87E-11** | **2.87E-11** |
| 187083 | 3 | **2.43E-17** | **7.04E-10** | **2.87E-11** |
| 138032 | 3 | **4.29E-18** | **2.87E-11** | **2.87E-11** |
| 178054 | 3 | **9.13E-14** | 0.56922 | **2.87E-11** |
| 60079 | 3 | **9.08E-18** | **8.56E-11** | **2.87E-11** |
| 55067 | 4 | **2.45E-14** | 0.19073 | **2.87E-11** |
| 108041 | 4 | **8.49E-18** | **5.77E-11** | **2.87E-11** |
| 135069 | 4 | **5.25E-15** | **2.87E-11** | **8.12E-09** |
| 108073 | 4 | **1.28E-17** | **5.84E-10** | **2.87E-11** |
| 100080 | 4 | **1.01E-09** | **2.87E-11** | **6.56E-05** |
| 66075 | 4 | **6.93E-15** | **0.005445** | **2.87E-11** |
| 207056 | 4 | **4.17E-17** | **3.34E-09** | **2.87E-11** |
| 22013 | 4 | **2.87E-17** | **8.86E-09** | **2.87E-11** |
| 188091 | 4 | **1.16E-16** | **1.02E-07** | **2.87E-11** |
| 41004 | 4 | **1.13E-13** | 0.801555 | **2.87E-11** |
| 23084 | 4 | **4.76E-06** | **6.81E-09** | **0.007786** |
| 24004 | 4 | **1.5E-14** | **0.002439** | **2.87E-11** |
| 26031 | 4 | **2.55E-12** | **1.81E-05** | **1.48E-09** |
| 271031 | 4 | **4.71E-14** | **0.041325** | **2.87E-11** |
| 28075 | 4 | **3.39E-17** | **5.22E-09** | **2.87E-11** |
| 311068 | 4 | **9.93E-08** | **1.66E-07** | 0.351637 |

| | | | | |
|---|---|---|---|---|
| 227040 | 4 | **7.66E-18** | **5.77E-11** | **2.87E-11** |
| 42078 | 5 | **3.28E-12** | **1.15E-08** | **0.009674** |
| 157036 | 5 | **7.58E-14** | 0.287112 | **2.87E-11** |
| 113044 | 5 | **4.99E-17** | **3.06E-09** | **2.87E-11** |
| 163062 | 5 | **1.96E-17** | **5.32E-10** | **2.87E-11** |
| 238011 | 5 | **1.35E-05** | 0.198358 | **2.51E-05** |
| 100075 | 5 | **1.26E-08** | **2.23E-06** | **9.19E-06** |
| 71046 | 5 | **1.83E-07** | **3.45E-06** | **6.56E-05** |
| 374020 | 5 | **0.00014** | **4.01E-10** | 0.65738 |
| 147021 | 5 | **1.03E-16** | **1.93E-08** | **2.87E-11** |
| 109034 | 5 | **6.97E-16** | **1.33E-06** | **2.87E-11** |
| 92059 | 5 | **7.22E-15** | **0.00041** | **2.87E-11** |
| 48055 | 5 | **0.00069** | **1.8E-07** | 0.183321 |
| 326038 | 5 | **1.1E-17** | **9.44E-11** | **2.87E-11** |
| 189003 | 5 | **5.35E-13** | **0.006819** | **3.18E-11** |
| 122048 | 5 | **2.02E-13** | **7.48E-06** | **4.01E-10** |
| 35008 | 5 | **2.92E-16** | **4.99E-07** | **2.87E-11** |
| 124084 | 5 | **6.68E-05** | **1.02E-07** | **0.026578** |
| 254033 | 5 | **3.86E-06** | **1.54E-10** | **0.026578** |
| 155060 | 6 | **7.68E-15** | **5.77E-08** | **5.32E-10** |
| 61060 | 6 | **5.69E-16** | **8.51E-07** | **2.87E-11** |
| 33066 | 6 | **1.7E-09** | **7.03E-11** | **6.56E-05** |
| 323016 | 6 | **0.000376** | **4E-09** | 0.65738 |
| 159029 | 6 | **2.06E-08** | **0.005445** | **1.07E-06** |
| 159091 | 6 | **8.4E-15** | **0.000541** | **2.87E-11** |
| 56028 | 6 | **8.87E-09** | **9.9E-07** | **9.19E-06** |
| 103041 | 6 | **1.72E-17** | **3.01E-10** | **2.87E-11** |
| 105053 | 6 | **2.91E-05** | **2.87E-11** | 0.375044 |
| 65010 | 6 | **3.26E-15** | **5.1E-05** | **2.87E-11** |

| | | | | |
|---|---|---|---|---|
| 173036 | 6 | **7.3E-17** | **4.49E-08** | **2.87E-11** |
| 95006 | 6 | 0.245775 | **0.030887** | 0.72272 |
| 147062 | 6 | **8.91E-15** | **7.39E-08** | **5.32E-10** |
| 247085 | 6 | **7.66E-14** | 0.124151 | **2.87E-11** |
| 24063 | 6 | **9.62E-09** | **3.66E-09** | **6.56E-05** |
| 12003 | 6 | **6.41E-14** | 0.076041 | **2.87E-11** |
| 388016 | 6 | **3.75E-08** | **9.44E-08** | **6.56E-05** |
| 42044 | 6 | **3.98E-14** | 0.65738 | **2.87E-11** |
| 67079 | 6 | **2.44E-13** | **2.87E-11** | **1.02E-07** |
| 353013 | 6 | **1.9E-11** | **6.41E-10** | **1.07E-06** |
| 164074 | 7 | **5.48E-15** | **0.000276** | **2.87E-11** |
| 260081 | 7 | **3.85E-16** | **1.93E-06** | **2.87E-11** |
| 254054 | 7 | **2.17E-14** | **5.39E-07** | **5.32E-10** |
| 105019 | 7 | **4.15E-05** | **6.37E-11** | 0.383055 |
| 368078 | 7 | **7.24E-05** | **2.26E-10** | 0.375044 |
| 46076 | 7 | **1.67E-05** | **3.51E-11** | 0.183321 |
| 242078 | 7 | **4.34E-07** | **2.87E-11** | **0.007786** |
| 2092 | 7 | **1.38E-05** | **2.79E-09** | **0.026578** |
| 66039 | 7 | **1.83E-11** | **9.44E-08** | **1.02E-07** |
| 236017 | 7 | **1.39E-14** | **0.001557** | **2.87E-11** |
| 309004 | 7 | **1.28E-13** | 0.801555 | **2.87E-11** |
| 61086 | 7 | **5.07E-12** | **4.49E-05** | **8.12E-09** |
| 183055 | 7 | **1.8E-16** | **6.26E-08** | **2.87E-11** |
| 187003 | 7 | **1.81E-16** | **6.26E-08** | **2.87E-11** |
| 117054 | 8 | **0.005131** | **9.9E-07** | 1 |
| 187071 | 8 | **9.91E-15** | **0.00071** | **2.87E-11** |
| 198023 | 8 | **1.46E-08** | **1.06E-08** | **6.56E-05** |
| 138078 | 8 | **1.88E-05** | **1.15E-08** | 0.076041 |
| 20008 | 8 | **3.16E-05** | **5.39E-07** | **0.007786** |

| | | | | |
|---|---|---|---|---|
| 209070 | 8 | **8E-14** | 0.147373 | **2.87E-11** |
| 176035 | 8 | **8.38E-08** | **5.39E-07** | **6.56E-05** |
| 187029 | 8 | 0.494928 | 0.375044 | 0.375044 |
| 112082 | 8 | **1.12E-07** | **1.07E-06** | **6.56E-05** |
| 156079 | 8 | **1.67E-15** | **1.38E-05** | **2.87E-11** |
| 314016 | 8 | **4.15E-17** | **2.55E-09** | **2.87E-11** |
| 310007 | 8 | **9.11E-07** | **0.000145** | **6.56E-05** |
| 35058 | 8 | **2.04E-05** | **1.95E-07** | **0.007786** |
| 249087 | 8 | **1.19E-14** | **0.001086** | **2.87E-11** |
| 249061 | 8 | **1.87E-12** | **2.13E-09** | **1.02E-07** |
| 188063 | 9 | **1.21E-06** | **5.32E-10** | 0.399389 |
| 225017 | 9 | **2.8E-09** | **6.8E-08** | **9.19E-06** |
| 246053 | 9 | **1.04E-13** | 0.336558 | **2.87E-11** |
| 90076 | 9 | **0.006285** | **4.22E-05** | 0.178502 |
| 268002 | 9 | **5.43E-13** | **0.001086** | **5.32E-10** |
| 118020 | 9 | **2.77E-09** | **1.05E-05** | **1.15E-06** |
| 277095 | 9 | **3.74E-08** | **0.023696** | **1.07E-06** |
| 144067 | 9 | 0.120389 | **0.041325** | 0.183321 |
| 170054 | 9 | **1.18E-07** | **4.27E-06** | **3.71E-05** |
| 376020 | 9 | **1.08E-09** | **1.15E-10** | **0.001905** |
| 35070 | 9 | **3.01E-08** | **0.032054** | **8.51E-07** |
| 372047 | 10 | **2.2E-05** | **5.32E-10** | 0.076041 |
| 140075 | 10 | **7.73E-08** | **7.39E-08** | **7.43E-05** |
| 187039 | 10 | **1.64E-09** | **4.27E-06** | **7.89E-07** |
| 45077 | 10 | **0.025101** | 0.188237 | **0.026578** |
| 246016 | 10 | **0.031398** | 0.700686 | **0.024625** |
| 106020 | 10 | **1.11E-10** | 0.094793 | **8.12E-09** |
| 271008 | 10 | **1.38E-05** | **2.68E-07** | **0.007786** |
| 35091 | 10 | **7.36E-14** | 0.110329 | **2.87E-11** |

| | | | | |
|---|---|---|---|---|
| 54005 | 10 | **5.31E-14** | **0.045945** | **2.87E-11** |
| 374067 | 10 | **1.02E-12** | **5.32E-10** | **1.02E-07** |
| 145014 | 10 | **1.77E-14** | **3.39E-07** | **5.32E-10** |
| 176019 | 10 | **3.21E-15** | **4.79E-05** | **2.87E-11** |
| 68077 | 10 | **5.5E-12** | 0.554268 | **5.32E-10** |
| 166081 | 10 | **7.27E-14** | 0.10707 | **2.87E-11** |
| 153093 | 10 | **8.05E-07** | **1.15E-10** | **0.007786** |
| 153077 | 10 | **1.15E-12** | **1.49E-08** | **8.12E-09** |
| 189011 | 10 | **1.27E-13** | 0.813005 | **2.87E-11** |
| 181018 | 10 | **5.1E-11** | **0.01471** | **7.44E-09** |
| 140055 | 11 | **0.00852** | 0.790147 | **0.007786** |
| 145053 | 11 | **5.04E-17** | **3.06E-09** | **2.87E-11** |
| 227046 | 11 | **0.034355** | **0.002962** | 0.183321 |
| 385028 | 11 | **1.39E-06** | **8.12E-09** | **0.007786** |
| 97017 | 11 | 0.347934 | 0.124151 | 0.375044 |
| 43070 | 11 | **1.21E-13** | 0.636141 | **2.87E-11** |
| 198004 | 11 | **0.000926** | **0.001557** | **0.007786** |
| 301007 | 12 | 0.602424 | 0.442018 | 0.689761 |
| 118035 | 12 | **1.44E-06** | 0.28047 | **9.19E-06** |
| 15088 | 12 | **3.22E-15** | **4.79E-05** | **2.87E-11** |
| 293029 | 12 | **4.78E-15** | **1.63E-08** | **5.32E-10** |
| 169012 | 12 | **1.43E-17** | **1.69E-10** | **2.87E-11** |
| 172032 | 12 | **1.12E-06** | **1.26E-08** | **0.001905** |
| 188005 | 12 | **0.000248** | **1.95E-07** | 0.076041 |
| 159045 | 13 | **7.27E-08** | **8.01E-06** | **0.002002** |
| 365073 | 13 | **1.43E-17** | **2.87E-11** | **1.69E-10** |
| 113009 | 13 | **1.34E-13** | **4.49E-05** | **5.32E-10** |
| 161062 | 13 | **9.42E-11** | **0.049261** | **8.12E-09** |
| 65132 | 13 | **3.73E-13** | **5.23E-11** | **1.02E-07** |

| | | | | |
|---|---|---|---|---|
| 23025 | 13 | 0.099207 | 0.225389 | 0.076041 |
| 292066 | 13 | **0.004016** | **1.13E-05** | 0.188237 |
| 198054 | 13 | **0.045321** | **0.004128** | 0.183321 |
| 76002 | 13 | **1.99E-05** | **8.51E-07** | **0.007451** |
| 8049 | 13 | **1.3E-10** | 0.143287 | **8.12E-09** |
| 8143 | 14 | **4.31E-17** | **2.13E-09** | **2.87E-11** |
| 15004 | 14 | **3.41E-14** | **2.87E-11** | **1.02E-07** |
| 22090 | 14 | **0.001253** | **0.00071** | **0.026578** |
| 216053 | 15 | **3.77E-08** | 0.554268 | **2.89E-07** |
| 181091 | 15 | **0.018965** | **0.000107** | 0.383055 |
| 202012 | 15 | **8.28E-08** | **3.51E-11** | **0.026578** |
| 365025 | 15 | **8.88E-06** | **0.000388** | **0.000485** |
| 232038 | 16 | 0.140405 | 0.094793 | 0.183321 |
| 22093 | 16 | **1.78E-07** | **2.33E-09** | 0.060433 |
| 216041 | 16 | **0.002138** | **6.16E-05** | **0.044359** |
| 27059 | 16 | **0.008404** | **1.13E-05** | 0.976411 |
| 126039 | 16 | **1.87E-05** | **3.01E-10** | 0.178502 |
| 59078 | 16 | 0.097376 | 0.056496 | 0.15581 |
| 163014 | 16 | **0.000427** | **0.006819** | **0.001723** |
| 231015 | 16 | **3.01E-17** | **9.31E-10** | **2.87E-11** |
| 239096 | 17 | **5.08E-16** | **2.87E-11** | **8.86E-09** |
| 104022 | 18 | **2.36E-17** | **5.32E-10** | **2.87E-11** |
| 196015 | 18 | **1.56E-17** | **3.18E-11** | **1.86E-10** |
| 55075 | 18 | **0.013822** | **0.000367** | 0.450846 |
| 181079 | 18 | **3.41E-15** | **4.4E-10** | **6.24E-09** |
| 23080 | 20 | **1.54E-13** | **9.44E-11** | **5.39E-07** |
| 285036 | 21 | **3.18E-08** | **8.56E-11** | 0.870811 |
| 274007 | 21 | **1.5E-11** | **3.18E-11** | **0.000346** |
| 368016 | 22 | **3.03E-11** | **2.87E-11** | **0.02193** |

| | | | | |
|---|---|---|---|---|
| 35010 | 23 | **4.21E-14** | **2.87E-11** | **5.79E-05** |
| 361084 | 24 | **1.14E-12** | **2.87E-11** | 0.399389 |
| 25098 | 24 | **8.91E-12** | **2.87E-11** | **0.001205** |
| 239007 | 25 | **6.12E-10** | **2.87E-11** | 0.273934 |
| 245051 | 26 | **2.06E-08** | **3.18E-11** | **0.000292** |
| 151087 | 27 | **7.8E-13** | **2.87E-11** | **0.007129** |
| 78019 | 28 | **1.5E-09** | **2.87E-11** | 0.052775 |
| 286092 | 28 | **2.23E-06** | **2.87E-11** | 0.273934 |
| 253036 | 31 | **4.02E-09** | **2.87E-11** | 0.131552 |
| 65074 | 33 | **1.37E-10** | **2.87E-11** | **0.039876** |
| 65019 | 34 | **4.34E-13** | **2.87E-11** | **0.001144** |
| 370036 | 41 | **7.1E-15** | **2.87E-11** | **0.000309** |
| 302003 | 43 | **2.53E-13** | **2.87E-11** | **0.004969** |
| 376001 | 54 | **1.09E-11** | **2.87E-11** | 0.147373 |