Algoritmo memético para encontrar reglas Golomb cercanamente óptimas basado en construcciones y métodos algebraicos

Monografía para optar por el título de Ingeniero de Sistemas
Modalidad: Trabajo de Investigación

Santiago De La Cruz Idarraga
104615020911

Director: PhD. Siler Amador Donado.

Codirector: MSc. Ember Ubeimar Martínez Flor.

Codirector: PhD. Carlos Andrés Martos Ojeda.

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Sistemas
Línea de Investigación: Inteligencia Computacional
Popayán, marzo de 2024

Agradecimientos

Me gustaría iniciar agradeciendo a mi madre, mi compañera de vida y persona más importante en mi vida, gracias por enseñarme el valor del estudio, por enseñarme el valor de la vida, por enseñarme el valor de los momentos, pero, sobre todo gracias por formarme como persona y profesional hasta el día de hoy, te amo infinitamente madre.

Agradecerle a mi padre, mi consejero, mi amigo, mi persona a seguir, mi cómplice. Gracias por apoyarme en cada paso que di desde pequeño, por enseñarme a manejar bicicleta, por apoyarme en el deporte, por apoyarme en las cosas que siempre vi imposible, te amo padre.

Agradecerles a mis directores de grado, profe Siler, gracias por todas las enseñanzas y por esa manera tan particular de dar consejos profe, me sirvieron mucho en mi desarrollo personal. Al profe Carlos, por la dedicación que me brindo de inicio a fin, siempre tuviste la mejor pedagogía y disposición para explicarme las cosas, tu apoyo sin duda fue indispensable para lograr los objetivos del proyecto, gracias Carlos. Al profe Ember le agradezco su manera tan particular de enseñar y de apoyarme, gracias por enseñarme las materias que terminaron inspirando mi futuro profesional. Pero sobretodo, agradecerles a los tres por la paciencia que me tuvieron a lo largo de este proyecto, fue indispensable para sacar esto adelante.

Agradecerles a mis amigos, en especial a Jacobo por haber estado desde el inicio de este proceso hasta el final y por haber confiado en ese joven llenos de ilusiones con ganas de comerse el mundo, has sido una de mis motivaciones más grandes para llegar a ser la persona que soy hoy en día, gracias hermano. A Jhon Dennis por haberme escuchado en los momentos más difíciles y darme esos consejos realistas que tanto necesitaba en su momento, pero sobre todo por inspirarme a luchar por lo que me apasiona, gracias hermano. A Luisa por haber sido como una hermana para mí, por brindarme ese plato de comida al lado de su familia cuando tanto lo necesite, a vos, gracias. A María Paula, mi compañera y hermana de guerra, le agradezco mucho a la vida por haber cruzado mi camino con el tuyo, fuiste mi inspiración y unas de mis motivaciones para sacar esto adelante, gracias haber crecido conmigo, estoy muy orgulloso de vos.

Agradecerles a los que ya no están y quizás tengan la oportunidad de leer eso, gracias por haber sido parte de mi desarrollo personal y profesional, gracias por el apoyo que me brindaron a lo largo de tanto tiempo y ver las cosas que yo nunca pude ver en mí, fueron mi motivación principal para iniciar este proyecto, pero sobre todo para a lograrlo y sacarlo adelante.

Tabla de contenido

Índice de tablas

Índice de figuras

Capítulo 1. Introducción.

En este capítulo se presenta la motivación, problemática, justificación, aportes, objetivos y estrategia de investigación para este trabajo. Adicionalmente, se presenta la estructura de la monografía en donde se resume el contenido de cada uno de los capítulos desarrollados a lo largo del proyecto.

1.1. Planteamiento del problema.

Las reglas Golomb nacen de un problema de ingeniería en 1952 por Wallace C. Babcock [1] cuando estudiaba la interferencia de intermodulación en canales de radio. Posteriormente, motivado por el trabajo de Babcock, el profesor Solomon W. Golomb formalizo el concepto de reglas Golomb [2] describiendo la idea de Babcock más detallada y realizando una investigación sistemática sobre el tema y los problemas que este conlleva.

Desde sus orígenes las reglas Golomb han tenido aplicaciones en diversos campos de la ingeniería como radiocomunicaciones [1], cristalografía de rayos X [2], redes informáticas [33, 34], diseño de circuitos [3], mapeo geográfico [4], fase de modulación de pulsos [5], asignación de frecuencia portadora [6], asignación de canales WDM ópticos [35], códigos ortogonales [7], integración a muy gran escala (VLSI) [8], teoría de códigos [9], arreglos lineales [10, 11], radioastronomía [12, 13], diseño de antenas para misiones de radar [2], aplicaciones de sonar y misiones de la NASA en astrofísica [12, 13, 14], ciencias planetarias y terrestres [12, 13, 14].

Estas reglas se definen como un conjunto de enteros positivos con la particularidad de que todas las diferencias positivas de dos elementos del conjunto son distintas, es decir, si tomamos dos números del conjunto y los restamos entre sí, siempre obtendremos un resultado diferente. Matemáticamente se tienen las siguientes definiciones.

Reglas Golomb. Una regla Golomb es un conjunto de ***n*** números enteros no negativos ordenados definidos de la siguiente manera:

$$A = \{a_i, a_{i+1}, \dots, a_n\}$$

$$Regla\ Golomb\ cercanamente\ óptima \rightarrow A' = \{0,\ 1,\ 4,\ 8\}$$

$$D = \{1\ ,3\ ,2\ ,4\ ,5\ ,6\ \}$$

$$a_i < a_{i+1} \ \rightarrow 1 \leq i \leq n$$

con la propiedad de que todas las diferencias de la forma ***d*** tienen única solución.

$$d = a_j - a_i \ \rightarrow 1 \leq i < j \leq n$$

Por otro lado, cada elemento de la regla Golomb es conocido como “marca” y la mayor de las diferencias de dos elementos se conoce como longitud de la regla y es denotado por

$$l(A) = máxA - mínA$$

así mismo, el número ***n*** de elementos del conjunto es conocido como “orden de la regla” o “número de marcas”.

Ejemplo

El conjunto $A = \{0,1,4,6\}$ es una regla Golomb de orden 4 y longitud 6. Adicionalmente, la figura 1 nos muestra el triángulo de diferencias de la regla Golomb evidenciando cada una de las diferencias que tiene la regla. En particular, todas las diferencias del triángulo deben ser diferentes para que el conjunto A se considere una regla Golomb [55].

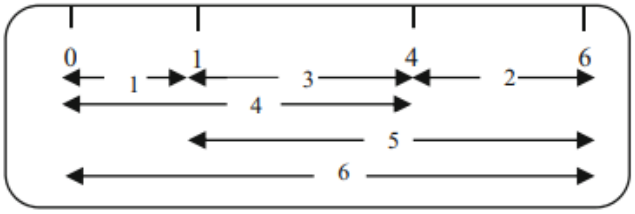


Figura 1. Regla Golomb óptima de 4 marcas.

Fuente: Adaptado y traducido de [34].

Debido a que pueden existir múltiples reglas Golomb para un número determinado de marcas, la calidad de las reglas se medirá en términos de sus longitudes, por ejemplo, si la longitud de una regla A es más corta que la *longitud* de una regla B, entonces la regla A será más óptima que B:

$$A = \{0, 1, 4, 6\} \rightarrow l(A) = \mathbf{6}$$

$$B = \{0, 1, 4, 9\} \rightarrow l(B) = \mathbf{9}$$

$$\boldsymbol{l(A) < l(B)}$$

El problema principal sucede cuando se necesita encontrar la regla Golomb con la longitud más corta posible para un numero de marcas fijo. Esta regla se le conoce como regla Golomb óptima (OGR) y encontrarla se convierte en un problema de optimización difícil sobre todo cuando se aumenta el número de marcas [3], es por eso, que en la literatura se le considera como un problema NP-completo [15, 16, 17]. Hasta el momento, no existe una formula o solución exacta que encuentre OGR, la única forma de probar o encontrar una OGR es utilizando métodos de búsqueda exhaustivos [18]. La complejidad del problema es tan grande que desde el año 2000 la organización distributed.net [19] ha implementado una búsqueda exhaustiva con la ayuda de una red distribuida para encontrar reglas Golomb óptimas de 23 a 28 marcas, tardando aproximadamente 8,5 años solo en encontrar la OGR para 28 marcas [20]. La razón es que el tiempo necesario para encontrar una OGR de manera exhaustiva aumenta exponencialmente con el tamaño del problema [18]. Por otro lado, si una regla no es óptima pero su longitud es cercana a la óptima, se le denominará como una regla de Golomb cercanamente óptima [36]. Así mismo, si no se conoce la OGR para un numero de marcas determinado (n), existirán reglas Golomb candidatas que cumplan la conjetura de Erdos [21] en donde la longitud de la regla debe ser menor a n cuadrado, estas reglas también se les denominara como reglas Golomb cercanamente óptimas: $A' = \{0, 1, 4, 8\}$.

Hasta la fecha, en la literatura no se conoce ningún algoritmo eficaz para encontrar reglas Golomb óptimas. Existen aproximaciones clásicas o exactas como Bose, Singer, Ruzsa [22, 23, 24], a su vez, también existen metaheurísticas que abordan el problema de OGR como Genetic algorithm, Bat algortihm, Big-bang-Big crunch, Multi-objective firefly, algoritmos inspirados en la naturaleza [17, 31, 32], entre otros, en donde serán mencionados posteriormente en el estado del arte.

Generalmente, las aproximaciones exactas abordan el problema desde su naturaleza matemática utilizando métodos y conceptos matemáticos para encontrar reglas Golomb en corto tiempo, es por esto que también se les conoce como métodos algebraicos. Por otro lado, las metaheurísticas reportadas por la literatura se han enfocado en resolver el problema de una forma más computacional, proponiendo formas hibridas de metaheurísticas, versiones multi-objetivo de algoritmos, entre otros. A pesar de que estas metaheurísticas han obtenido buenos resultados en el problema de OGR ninguna ha abordado el problema teniendo en cuenta su naturaleza matemática ni han logrado obtener buenos resultados para reglas Golomb por encima de 21 marcas. Es aquí donde los algoritmos meméticos juegan un papel importante, ya que este tipo de algoritmos aprovecha el conocimiento y la naturaleza del problema para obtener buenos resultados en problemas de optimización NP-completo [25, 26, 27].

Por esta razón surge la siguiente pregunta de investigación: **¿Cómo encontrar reglas Golomb cercanamente óptimas hasta 28 marcas, mediante un algoritmo memético incorporando conceptos y métodos matemáticos?** Responder a esta pregunta no es fácil, es por esto, que el presente trabajo de investigación propone realizar un algoritmo memético que incorpore métodos matemáticos para facilitar la búsqueda de reglas Golomb óptimas o cercanamente óptimas y que su vez, el trabajo quede a la disposición de la comunidad científica en el ámbito local, regional, nacional e internacional para así seguir aportando conocimiento al problema.

1.2. Aportes del proyecto.

Investigación:

- Revisión de la literatura existente con relación a metaheurísticas que encuentran reglas Golomb óptimas o cercanamente óptimas.
- Se propone un algoritmo memético que incorpore construcciones y métodos algebraicos para encontrar reglas Golomb óptimas o cercanamente óptimas y así generar nuevo conocimiento en el área de inteligencia computacional.
- Evaluación del algoritmo memético propuesto mediante la comparación de las reglas Golomb encontradas con las reglas conocidas.

Innovación:

- Contribuir con un algoritmo memético que incorpore métodos algebraicos para encontrar reglas Golomb. Este código quedará disponible para la comunidad académica, científica e industrial y puede ser utilizado para encontrar reglas Golomb que pueden ser usadas en diferentes áreas de aplicación.

1.3. Objetivos.

En esta sección, se presentarán los objetivos del proyecto de investigación y estarán divididos en dos sub secciones, objetivo general y objetivos específicos.

1.3.1. Objetivo general

OG. Proponer un algoritmo memético para encontrar reglas Golomb cercanamente óptimas hasta 28 marcas incorporando construcciones algebraicas, traslaciones y cotas.

1.3.2. Objetivos específicos.

- **OE1**: Caracterizar las metaheurísticas sugeridas por la comunidad científica que encuentran reglas Golomb cercanamente óptimas por medio de un mapeo sistemático de la literatura.
- **OE2.** Implementar un algoritmo memético que integre las construcciones Bose, Singer y Ruzsa, así como traslaciones y cotas matemáticas para encontrar reglas Golomb cercanamente óptimas.
- **OE3.** Evaluar los resultados del algoritmo memético propuesto mediante la comparación de las soluciones encontradas contra las 28 reglas Golomb óptimas conocidas para determinar si son cercanamente óptimas.

1.4. Metodología.

1.4.1. Metodología para el desarrollo del proyecto.

Para llevar a cabo la ejecución del proyecto propuesto, se utilizó el método Design Patterns for Research Methods: Iterative Field Research [28] donde se propone 4 etapas en la investigación: observación, identificación, desarrollo, evaluación. Teniendo en cuenta las etapas mencionadas, el desarrollo de esta propuesta se realizó mediante 4 iteraciones de investigación. A continuación, se describen los ciclos y las actividades que se llevaron a cabo de manera secuencial e incremental.

- **ITERACIÓN 1. Revisión del estado del arte [Tabla 1]:** Este proceso se llevó a cabo por medio de un mapeo sistemático de la literatura sobre metaheurísticas que encuentran Reglas Golomb óptimas o cercanamente óptimas.

Tabla 1. Iteración 1 – Estado del arte.

OBSERVACIÓN	IDENTIFICACIÓN	DESARROLLO	EVALUACIÓN
O1. Revisión de la literatura sobre metaheurísticas utilizadas para encontrar reglas Golomb cercanamente óptimas.	I1. Definir los objetivos de la búsqueda de la literatura.	D1. Ejecutar la búsqueda de la literatura.	E1. Evaluar el resultado de la búsqueda de la literatura.

- **ITERACIÓN 2. Construcción de la base conceptual [Tabla 2]:** Definición de los conceptos matemáticos que se abordaron en el proyecto de investigación.

Tabla 2. Iteración 2 – Base conceptual.

OBSERVACIÓN	IDENTIFICACIÓN	DESARROLLO	EVALUACIÓN
O2. Revisión de la literatura para identificar implementaciones de las construcciones.	I2. Definir las construcciones algebraicas. I3. Definir las cotas y las traslaciones matemáticas.	D2. Implementación de las construcciones algebraicas.	E2. Evaluar los resultados obtenidos por las construcciones algebraicas.

- **ITERACIÓN 3. implementación del algoritmo memético [Tabla 3]:** Se implementó un algoritmo memético que integra los conceptos matemáticos para encontrar reglas Golomb cercanamente óptimas.

Tabla 3. Iteración 3 – Desarrollo de la propuesta.

OBSERVACIÓN	IDENTIFICACIÓN	DESARROLLO	EVALUACIÓN
O3. Revisión del lenguaje de programación para realizar la implementación del algoritmo memético.	I4. Definir las características generales del algoritmo memético.	D3. Implementación del algoritmo memético.	

- **ITERACIÓN 4: Evaluación del algoritmo [Tabla 4]:** Se desarrolló mediante la comparación de los resultados del algoritmo con las reglas Golomb óptimas conocidas hasta la fecha.

Tabla 4. Iteración 4 – Evaluación de la propuesta.

OBSERVACIÓN	IDENTIFICACIÓN	DESARROLLO	EVALUACIÓN
O4. Revisión de la literatura para identificar las métricas de evaluación apropiadas.	I5. Definir el diseño de los experimentos.	D4. Ejecución de la propuesta. D5. Refinamiento de la propuesta mediante el ajuste de sus parámetros de configuración.	E3. Evaluar los resultados obtenidos por el algoritmo memético propuesto con las métricas de evaluación apropiadas.

1.4.2. Metodología para la construcción de la revisión bibliográfica.

Para el desarrollo del estado del arte se realizó un mapeo sistemático de la literatura con el objetivo de recopilar y caracterizar las investigaciones existentes que han aportado al área de investigación de nuestro interés. De esta manera, se tomó como referencia el protocolo propuesto por Petersen [29] y Kitchenham [30] y se definieron las siguientes actividades:

- Definición de objetivos de búsqueda y preguntas de investigación.
- Definir la búsqueda y estrategia de selección.
- Realizar la revisión de la literatura.
- Realizar el informe de la revisión de la literatura.

En el capítulo 2 se describe cada una de las actividades mencionadas con mayor detalle.

1.5. Estructura del documento.

A continuación, se presenta un breve resumen de los 6 capítulos que componen este trabaja de investigación.

- **Capítulo 1. Introducción.** En este capítulo se presenta el planteamiento del problema y su respectiva justificación que motivó a la realización de este proyecto de investigación, así como los objetivos y metodologías para llevar a cabo este proyecto.
- **Capítulo 2. Marco teórico y estado del arte.** En este capítulo son abordados conceptos clave para el proyecto de investigación. Así mismo, se presenta un mapeo sistemático de la literatura en donde se analizan las diferentes propuestas recientes que han abordado el problema de investigación.
- **Capítulo 3. Conceptos matemáticos.** Definición de conceptos matemáticos relevantes para la propuesta de investigación como construcciones algebraicas, cotas y traslaciones.

- **Capítulo 4. Algoritmo memético.** Definición e implementación del algoritmo memético propuesto integrando los conceptos matemáticos del capítulo 3.
- **Capítulo 5. Evaluación del algoritmo memético.** Evaluación de los resultados encontrados por el algoritmo memético propuesto.
- **Capítulo 6. Conclusiones y trabajos futuros.** En este capítulo se presentan las conclusiones obtenidas al finalizar el proyecto de investigación, así como sus posibles trabajos futuros.
- **Referencias bibliográficas.** Se presenta la lista de referencias citadas en el presente documento en formato IEEE.
- **Anexos.** Se dividen en:
  - **Anexo 1.** Url del repositorio en Gitlab con el código fuente del algoritmo memético.
  - **Anexo 2.** Reglas Golomb óptimas conocidas.

Capítulo 2. Revisión del estado del arte.

Este capítulo se encuentra conformado por dos secciones: (i) el marco teórico presentando los conceptos utilizados en el proyecto de investigación y (ii) el estado del estar realizado mediante un mapeo sistemático con el fin de identificar y caracterizar investigaciones relacionados con metaheurísticas que encuentran reglas Golomb óptimas o cercanamente óptimas.

2.1. Marco teórico.

A continuación, se presenta la definición de los conceptos que ayudarán a comprender el problema de investigación de este proyecto.

2.1.1. Metaheurística.

Es una técnica inteligente de optimización [77] que sirve para resolver problemas de optimización complejos con soluciones aproximadas a las óptimas en un tiempo de máquina razonable.

2.1.2. Algoritmo memético.

Son algoritmos provenientes del área de inteligencia artificial [73] y de la familia de las metaheurísticas, su función principal es resolver problemas de optimización complejos [25, 26, 27] con la particularidad de que el algoritmo conoce el contexto e información del problema de optimización. Con esta información, el algoritmo logra implementar operadores de búsqueda local y global para encontrar soluciones óptimas o cercanamente óptimas al problema de optimización.

2.1.3. Reglas g-Golomb.

Es una extensión del concepto de reglas Golomb presentado en [54] en donde se define el concepto de reglas g-Golomb como reglas que admiten un determinado número de repeticiones en las diferencias de las marcas. Recordemos que todas las diferencias de una regla Golomb son diferentes por definición (Ver ejemplo de figura 1), pero en el caso de las reglas g-Golomb, se acepta un número entero fijo de repeticiones en las marcas y estará definido por una variable denominada ***"g"***.

2.1.4. Reglas Golomb cercanamente óptimas.

son reglas Golomb casi casi tan buenas como las reglas Golomb óptimas conocidas hasta la fecha, es decir, son reglas Golomb con longitudes muy cercanas a las longitudes óptimas. Adicionalmente, estas reglas Golomb cumplen con la conjetura de Erdos [21] en donde la longitud de la regla debe ser menor al número de marcas cuadrado [3].

2.2. Estado del arte.

Un estudio de mapeo sistemático es un método que se utiliza para recopilar y categorizar las investigaciones existentes sobre un campo de interés o área de investigación [29]. De esta manera, para el diseño del presente mapeo sistemático se tomó como referencia el protocolo propuesto por Petersen et al. [29], además de los lineamientos planteados por Kitchenham [30], con lo cual, se definieron una serie de actividades mencionadas en la figura 2.

2.2.1. Protocolo de investigación.

(1) Definición de objetivos de búsqueda y preguntas de investigación.
(2) Definir la búsqueda y estrategia de selección.
(3) Realizar la revisión de la literatura.
(4) Realizar el informe de la revisión de la literatura.

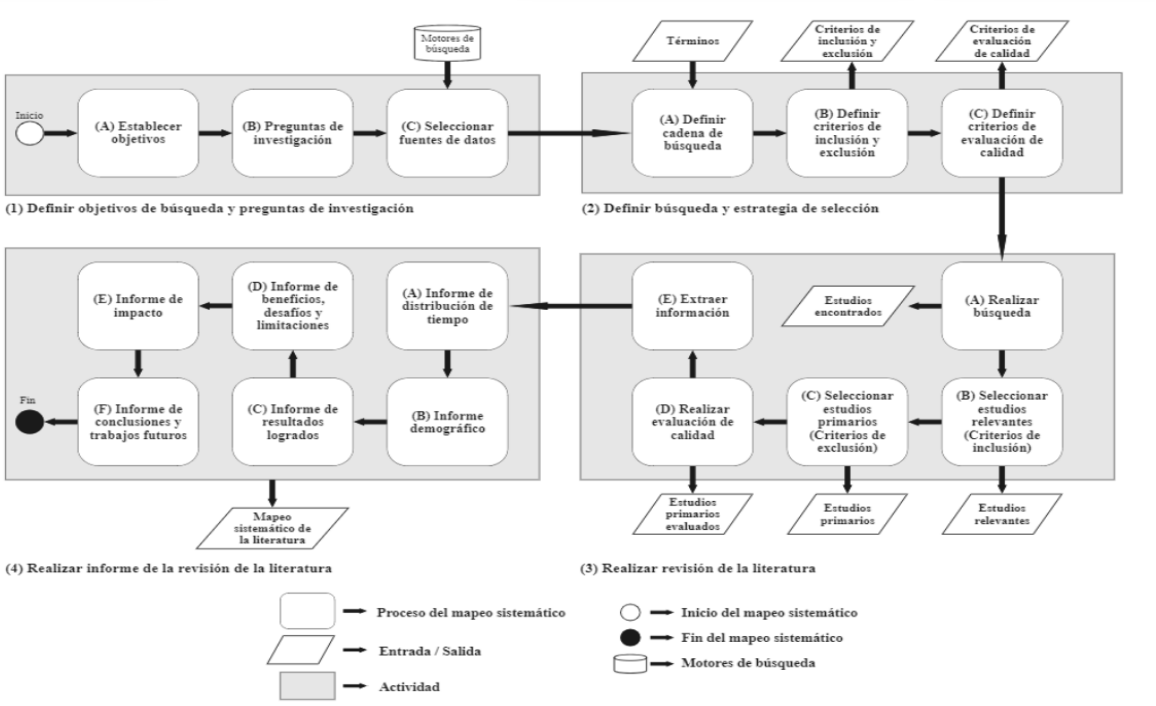


Figura 2. Proceso del mapeo sistemático.

Fuente: Elaboración propia.

En las siguientes secciones se presentará una descripción de cada una de las actividades mencionadas.

2.2.1.1. Definición de las preguntas de investigación.

Inicialmente, se definió un conjunto de objetivos de búsqueda (OB) con el propósito de dirigir adecuadamente el mapeo sistemático.

- **OB1.** Caracterizar la evidencia científica sugerida por la comunidad académica a partir de criterios temporales y de interés por parte de la misma comunidad para dimensionar el impacto científico que tienen las *metaheurísticas que encuentran reglas Golomb óptimas o cercanamente óptimas* en la literatura.

- **OB2.** Analizar las principales iniciativas científicas sobre las *metaheurísticas que encuentran reglas Golomb óptimas o cercanamente óptimas* evidenciados en la literatura actual mediante la definición de criterios demográficos y de calidad para clasificarlos con base en la relevancia de su aporte investigativo.
- **OB3.** Apoyar a académicos y otros interesados en investigar sobre las *metaheurísticas que encuentran reglas Golomb óptimas o cercanamente óptimas* mediante la recopilación de la información relevante como definiciones conceptuales, propuestas, resultados, limitaciones, procesos y validaciones para identificar el grado de desarrollo de las iniciativas con base en los resultados obtenidos.

A partir de los objetivos de búsqueda, se definieron 8 preguntas de investigación (PI) presentadas en la siguiente tabla, donde cada pregunta está mapeada a los objetivos planteados, junto a su respectiva motivación. Las preguntas de investigación planteadas en la tabla 5 buscan caracterizar la información encontrada sobre las metaheurísticas que encuentran reglas Golomb óptimas o cercanamente óptimas y permitir la identificación de las brechas existentes a nivel de investigación.

Tabla 5. Preguntas de investigación.

No.	Pregunta de investigación	Motivación	OB
PI1	¿Cuál es la distribución de tiempo de los estudios primarios?	La distribución de tiempo nos ayudará representar la tendencia de la literatura macro entre el año 2012 y 2022.	OB1
PI2	¿Cuáles son las revistas y conferencias más importantes en el área?	Identificar las conferencias y revistas científicas más relevantes en el área de investigación.	OB1
PI3	¿Cuál es la distribución local de los estudios primarios?	Identificar las Universidades que lideran las propuestas relacionadas con el tema de investigaciónx| de *metaheurísticas que encuentran reglas Golomb óptimas o cercanamente óptimas.*	OB2
PI4	¿Cuál es la distribución geográfica de los estudios primarios?	Determinar cuáles son los países que predominan en las propuestas de los artículos primarios.	OB2
PI5	¿Cuáles son los autores más citados en el área?	Identificar a los autores y trabajos más citados en el tema consultado.	OB1
PI6	¿Cuáles son los lenguajes de programación más importantes en el área?	Identificar los lenguajes de programación utilizados en los diferentes artículos primarios y con esto, identificar los lenguajes más relevantes en el área.	OB3
PI7	¿Qué resultados se han alcanzado con las propuestas realizadas?	Identificar cuáles fueron los resultados de los artículos primarios y con esto, identificar cual fue el alcance de las propuestas.	OB3
PI8	¿Qué beneficios y desafíos conlleva la investigación en el tema?	Determinar los beneficios y desafíos implicados cuando se aborda el problema de reglas Golomb.	OB3

Adicionalmente, se utilizó la estrategia PICOC: Population, Intervention, Comparison, Outcomes, Context [37] para definir los términos de la cadena de búsqueda. En la tabla 6 se describen los términos utilizados.

Tabla 6. Términos PICOC

PICOC	Término	Sinónimo / Término relacionado
Population	Problema de reglas Golomb	“Golomb ruler problem” OR “Golomb ruler”
Intervention	Algoritmos de optimización	"Metaheuristic" OR "Memetic Algorithm" OR "Hybrid"
Comparison	State of art	
Outcomes	Encontrar reglas Golomb cercanamente óptimas	"Optimal Golomb Ruler" OR "Marks" OR "Near optimal"
Context	Contexto académico	Academic context

2.2.1.2. Estrategia de búsqueda.

Para realizar la búsqueda de la información, se realizaron combinaciones entre las palabras clave identificadas por la estrategia PICOC y los operadores lógicos “AND” y “OR” generando la cadena básica de búsqueda presentada en la tabla 7:

Tabla 7. Cadena de búsqueda.

Cadena de búsqueda
(“Golomb ruler problem” OR “Golomb ruler”) AND ("Metaheuristic" OR "Memetic Algorithm" OR "Hybrid") AND ("Optimal Golomb Ruler" OR "Marks" OR "Near optimal")

Posteriormente, la cadena de búsqueda fue adaptada y aplicada sobre seis (6) bases de datos: Scopus, Science direct, Springer Link, Google scholar, IEEE Xplore y ACM Digital Library. Adicionalmente, se consideró una ventana de tiempo de diez (10) años, es decir, entre 2012 y 2022 y los siguientes filtros: (i) Artículos que hayan sido publicados en inglés, (ii) Estudios de tipo artículo científico o proceedings de eventos científicos.

2.2.1.3. Selección de artículos para inclusión y exclusión.

Inicialmente, para realizar la selección de los artículos relevantes, se tuvieron en cuenta los artículos encontrados que cumplieran con al menos un criterio de inclusión (CI) [Tabla 8]. Este proceso se realizó mediante una revisión en 2 niveles: (Nivel 1) revisión del título y (Nivel 2) revisión del resumen o abstract.

Tabla 8. Criterios de inclusión.

Id.	Criterios de inclusión (CI)
CI1	Estudios cuyo tema principal sean metaheurísticas que encuentran reglas Golomb óptimas o cercanamente óptimas.
CI2	Estudios que traten temas relacionados con metaheurísticas que encuentran reglas Golomb óptimas o cercanamente óptimas.

Después, los artículos relevantes fueron sometidos a una tercera revisión teniendo en cuenta los criterios de exclusión (CE) [Tabla 9] para seleccionar los artículos primarios: (Nivel 3) revisión de la introducción y conclusiones. Cabe resaltar que, en esta revisión, los artículos relevantes que cumplieron con al menos un CE fueron descartados de la investigación. Posteriormente, se realizó una revisión sobre el texto completo en los artículos primarios para continuar con los siguientes pasos del mapeo sistemático.

Tabla 9. Criterios de exclusión.

Id.	Criterios de exclusión (CE)
CE1	Estudios duplicados (considerando solo el más completo y reciente que se logre evidenciar).
CE2	Estudios donde se aborde superficialmente el tema de investigación.
CE3	Estudios donde no se aborden problemas relacionados con metaheurísticas que encuentran reglas Golomb óptimas o cercanamente óptimas.
CE4	Estudios con resultados poco claros o imparciales.

2.2.1.4. Criterios de evaluación de la calidad.

Con el propósito de medir la calidad de los estudios primarios seleccionados y determinar su grado de pertinencia con respecto al tema de investigación “metaheurísticas que encuentran reglas Golomb óptimas o cercanamente óptimas”, se construyó un cuestionario de 11 preguntas con un sistema de puntuación de tres valores (-1, 0, +1) como se describe en la tabla 10, con lo cual cada artículo puede obtener una calificación de calidad entre -11 y 11. De esta forma, se trabajó únicamente con los artículos primarios con una calificación superior a cero y los artículos que tuvieron una puntuación negativa fueron descartados de la investigación. En la tabla 15 se presentarán los resultados después de aplicar los criterios de calidad sobre cada estudio.

Tabla 10. Criterios de calidad.

No.	Criterio de calidad	Puntuacion de las respuestas		
		+1	0	-1
C1	El estudio se enfoca en investigar las *metaheurísticas que encuentran reglas Golomb óptimas o cercanamente óptimas.*	Sí	Parcialmente	No
C2	El estudio ofrece una descripción clara del problema de investigación abordado.	Sí	Parcialmente	No
C3	El estudio sigue un proceso de investigación estructurado y fundamentado.	Sí	Parcialmente	No
C4	El estudio proporciona una descripción clara de los estudios previos que han abordado el problema de investigación.	Sí	Parcialmente	No
C5	El estudio propone un conjunto de elementos a considerar para evaluar los resultados de su propuesta.	Sí	Parcialmente	No
C6	El estudio propone una forma de evaluarse con los estudios del estado del arte.	Sí	Parcialmente	No
C7	El estudio expone de manera clara y detallada los resultados obtenidos tras validar su propuesta.	Sí	Parcialmente	No
C8	El estudio presenta claramente las contribuciones de investigación hacia la academia, la industria y/o el gobierno.	Sí	Parcialmente	No
C9	El estudio describe claramente la discusión de las limitaciones del proceso de investigación realizado y del análisis de los resultados obtenidos.	Sí	Parcialmente	No
C10	El estudio describe claramente los trabajos futuros o alternativas de investigación.	Sí	Parcialmente	No
C11	El estudio ha sido citado por otros autores.	Ha sido citado por más de diez autores	Entre uno y diez autores	No ha sido citado hasta el momento

2.2.1.5. Extracción de datos.

En la tabla 11 se presenta una ficha resumen que sirvió para crear un resumen inicial de cada uno de los artículos primarios evaluados permitiendo así la estrategia de extracción de datos, esto ayudó a clasificar la información de una manera más fácil debido a que la ficha contiene los aspectos más importantes a considerar de cada artículo.

Tabla 11. Ficha resumen de los artículos.

Identificación			
Título:			
DOI:			Número de citaciones:
Fecha de publicación:	Conferencia y/o revista:		
Autores:			
Nombre	País	Universidad	Grupo de investigación
Resumen			
Descripción			
Tipo de investigación (Clasificación de Wieringa): ☐ Investigación de validación ☐ Investigación de evaluación ☐ Propuesta de solución ☐ Artículo filosófico o similar ☐ Documento de opinión ☐ Experiencia personal			
Metodología de investigación:			
Tipo de solución(es) ofrecida(s): ☐ Definición conceptual ☐ Causas, efectos, impactos y limitaciones ☐ Métodos o técnicas de evaluación ☐ Herramientas tecnológicas ☐ Validación en la industria ☐ Metodologías de documentación ☐ Otros			
Propuesta:			
Evaluación de la propuesta:			
Problemática abordada			
Elementos para la justificación			
Aspectos para destacar			

2.2.2. Ejecución de la búsqueda

Se realizaron 6 iteraciones, una por cada base de datos considerada para la búsqueda: Scopus, Science Direct, Springer Link, Google Scholar, IEEE Xplore y ACM digital library [38, 39, 40, 41, 42, 43], adicionalmente, esta búsqueda fue ejecutada el día 20 de junio del año 2022 teniendo en cuenta la ventana de tiempo de diez (10) años (Del 2012 al 2022). Sin embargo, debido a las diferentes configuraciones y parametrización que permite realizar cada base de datos, se tuvo la necesidad de adaptar la cadena básica de búsqueda. Es así como en la tabla 12 presentan las diferentes adaptaciones que sufrió la cadena original. Particularmente, el conteo detallado de todos los hallazgos se presenta en la tabla 13, donde además se cuantifican los artículos descartados en cada etapa.

.

Tabla 12. Adaptación de la cadena de búsqueda en las diferentes bases de datos.

No	Cadena de búsqueda	Motor de búsqueda
1	(“Golomb ruler problem”OR“Golom ruler”)AND("Metaheuristic"OR"Memetic Algorithm" OR"Hybrid")AND("Optimal Golomb Ruler"OR"Marks"OR"Near optimal")	Scopus
2	(“Golomb ruler problem”OR“Golomb ruler”)AND("Metaheuristic"OR"Memetic Algorithm" OR"Hybrid")AND("Optimal Golomb Ruler"OR"Marks"OR"Near optimal")	Science Direct
3	(“Golomb ruler problem” OR “Golomb ruler”) AND ("Metaheuristic" OR "Memetic Algorithm" OR "Hybrid") AND ("Optimal Golomb Ruler" OR "Marks" OR "Near optimal")	Springer Link
4	(“Golomb ruler problem”OR“Golomb ruler”)AND("Metaheuristic"OR"Memetic Algorithm" OR"Hybrid")AND("Optimal Golomb Ruler"OR"Marks"OR"Near optimal")	Google Scholar
5	(“Golomb ruler problem”OR“Golomb ruler”)AND("Metaheuristics"OR"Memetic Algorithm" OR"Construction"OR"Hybrid"OR"Evolutionary"OR"Genetic Algorithm")AND("Nearly optimal" OR"Optimal Golomb Ruler"OR"Marks"OR"Near optimal")	IEEE Xplore
6	(“Golomb ruler problem”OR“Golomb ruler”)AND("Metaheuristic"OR"Memetic Algorithm" OR"Hybrid")AND("Optimal Golomb Ruler"OR"Marks"OR"Near optimal") AND [Publication Date: (01/01/2012 TO 20/06/2022)]	ACM digital library

En la tabla 13, se puede evidenciar los 203 artículos encontrados en los diferentes motores de búsqueda después de aplicar las cadenas de búsqueda. Posteriormente, a los artículos encontrados se les aplico los criterios de inclusión para obtener los artículos relevantes del mapeo sistemático, es decir, los artículos encontrados que cumplieran con al menos un criterio de inclusión, serían considerados como artículos relevantes, de este filtrado se obtuvieron 72 artículos relevantes. Finalmente, a estos artículos se les aplico los criterios de exclusión para obtener los artículos primarios finales, es decir, los artículos relevantes que cumplieran con al menos un criterio de exclusión, no se tendrían en cuenta para la investigación, de esta manera, los artículos relevantes que no cumplieran con ningún criterio de exclusión, serían considerados como artículos primarios. De este proceso, se obtuvieron 13 artículos primarios.

Tabla 13. Resultados de la búsqueda.

No	Motor de búsqueda	Estudios encontrados	Estudios relevantes	Estudios primarios
1	Scopus	10	6	2
2	Springer Link	57	12	4
3	ACM Digital Library	2	0	0
4	Google Scholar	125	48	5
5	Science Direct	8	5	2
6	IEEE Xplore	1	1	0
	Total	**203**	**72**	**13**

Adicionalmente, en la siguiente tabla se muestran los artículos primarios que fueron seleccionados con sus respectivos motores de búsqueda en donde fueron encontrados.

Tabla 14. Artículos primarios.

Id	Artículo	Motor de búsqueda	Ref.
Ar1	Optimal Golomb ruler sequence generation for FWM crosstalk elimination: soft computing versus conventional approaches	Science Direct	[31]
Ar2	Nature-Inspired Hybrid Multi-objective Optimization Algorithms in Search of Near-OGRs to Eliminate FWM Noise Signals in Optical WDM Systems and their Performance Comparison	Springer Link	[32]
Ar3	A novel bat algorithm for channel allocation to reduce FWM crosstalk in WDM systems	Google Scholar	[44]
Ar4	Generation of optimal Golomb rulers for FWM crosstalk reduction: BB-BC and FA approaches	Google Scholar	[33]

Id	Artículo	Motor de búsqueda	Ref.
Ar5	Optimal Golomb ruler sequences generation for optical WDM systems: a novel parallel hybrid multi-objective bat algorithm	Scopus	[34]
Ar6	Improved Multi–Objective Firefly Algorithms to Find Optimal Golomb Ruler Sequences for Optimal Golomb Ruler Channel Allocation	Google Scholar	[45]
Ar7	Nature–inspired metaheuristic algorithms to find near–OGR sequences for WDM channel allocation and their performance comparison	Scopus	[36]
Ar8	A novel hybrid multi-objective BB-BC based channel allocation algorithm to reduce FWM crosstalk and its comparative study	Google Scholar	[46]
Ar9	Constraint-based local search for Golomb rulers	Springer Link	[47]
Ar10	Constraint-based search for optimal Golomb rulers	Springer Link	[48]
Ar11	A wind driven optimization based WDM channel allocation algorithm	Google Scholar	[49]
Ar12	Agent-based Evolutionary and Memetic Black-box Discrete Optimization	Science Direct	[50]
Ar13	Multi-objective Flower Pollination Algorithm and Its Variants to Find Optimal Golomb Rulers for WDM Systems	Springer Link	[35]

Posteriormente, teniendo en cuenta los criterios de calidad definidos en la tabla 10 es se asignó un puntaje entre -11 y 11 a cada artículo primario seleccionado. Es así como la tabla 15 presenta los resultados obtenidos tras aplicar los criterios de calidad a todos los artículos primarios seleccionados. El puntaje total de cada artículo se obtiene a partir de la suma de los valores individuales para cada aspecto evaluado.

Tabla 15. Listado de publicaciones con sus respectivos puntajes de calidad.

Artículo	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	Total
Ar7	1	1	1	1	1	1	1	1	0	1	1	10
Ar2	1	1	1	1	1	1	1	1	1	0	0	9
Ar5	1	1	1	0	1	1	1	1	0	1	1	9
Ar13	1	1	1	0	1	1	1	1	0	1	0	8
Ar10	1	1	1	0	1	0	1	1	0	-1	0	5
Ar6	1	-1	0	0	1	1	1	1	-1	1	0	4
Ar1	1	1	1	-1	1	0	1	0	-1	-1	1	3
Ar3	1	0	0	-1	1	1	1	0	-1	1	0	3
Ar8	1	0	0	-1	1	1	1	0	-1	-1	1	2
Ar4	1	0	0	-1	1	1	1	0	-1	-1	0	1
Ar9	1	1	1	0	1	-1	0	-1	-1	-1	0	0
Ar11	1	0	0	-1	0	-1	1	0	-1	-1	0	-2
Ar12	1	-1	-1	-1	1	-1	0	-1	-1	0	0	-4

En la tabla 16 se puede evidenciar que los artículos primarios evaluados A11 y A12 cuentan con un puntaje negativo (-2 y -4 respectivamente), estos artículos no se tuvieron en cuenta en el mapeo sistemático debido a que los contenidos de estos artículos no aportaron el conocimiento suficiente en nuestra investigación.

Tabla 16. Resultados evaluados.

No	Motor de búsqueda	Estudios encontrados	Estudios relevantes	Estudios primarios	Estudios primarios evaluados
1	Scopus	10	6	2	2
2	Springer Link	57	12	4	4
3	ACM Digital Library	2	0	0	0
4	Google Scholar	125	48	5	4
5	Science Direct	8	5	2	1
6	IEEE Xplore	1	1	0	0
	Total	**203**	**72**	**13**	**11**

De esta manera, la tabla 17 se muestra la descripción de los estudios primarios finales que serán tenidos en cuenta en las siguientes fases del mapeo sistemático.

Tabla 17. Artículos primarios evaluados.

Id	Artículo	NC	Motor de búsqueda	Año	Ref.
A1	Optimal Golomb ruler sequence generation for FWM crosstalk elimination: soft computing versus conventional approaches	15	Science Direct	2014	[31]
A2	Nature-Inspired Hybrid Multi-objective Optimization Algorithms in Search of Near-OGRs to Eliminate FWM Noise Signals in Optical WDM Systems and their Performance Comparison	1	Springer Link	2021	[32]
A3	A novel bat algorithm for channel allocation to reduce FWM crosstalk in WDM systems	6	Google Scholar	2016	[44]
A4	Generation of optimal Golomb rulers for FWM crosstalk reduction: BB-BC and FA approaches	9	Google Scholar	2016	[33]
A5	Optimal Golomb ruler sequences generation for optical WDM systems: a novel parallel hybrid multi-objective bat algorithm	16	Scopus	2016	[34]
A6	Improved Multi–Objective Firefly Algorithms to Find Optimal Golomb Ruler Sequences for Optimal Golomb Ruler Channel Allocation	1	Google Scholar	2016	[45]
A7	Nature–inspired metaheuristic algorithms to find near–OGR sequences for WDM channel allocation and their performance comparison	13	Scopus	2017	[36]
A8	A novel hybrid multi-objective BB-BC based channel allocation algorithm to reduce FWM crosstalk and its comparative study	11	Google Scholar	2015	[46]
A9	Constraint-based local search for Golomb rulers	6	Springer Link	2015	[47]
A10	Constraint-based search for optimal Golomb rulers	7	Springer Link	2017	[48]
A11	Multi-objective Flower Pollination Algorithm and Its Variants to Find Optimal Golomb Rulers for WDM Systems	1	Springer Link	2021	[35]

2.2.3. Análisis de los resultados.

A continuación, se presentan los resultados obtenidos en cada una de las preguntas de investigación definidas en este mapeo sistemático.

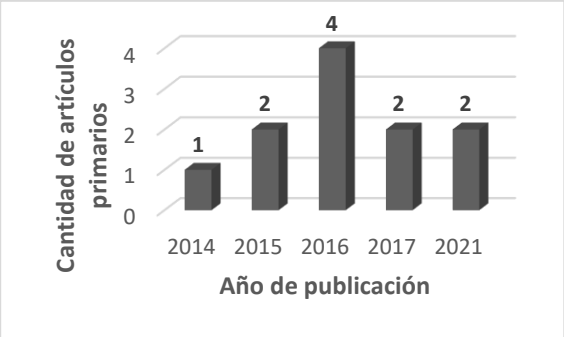
2.2.3.1. ¿Cuál es la distribución de tiempo de los estudios primarios?

En la figura 3(a) se puede apreciar los estudios primarios encontrados en una ventana de tiempo entre el 1 de enero de 2012 hasta el 30 de junio del 2022, en donde el primer año con un (1) artículo primario evaluado reportado fue en el año 2014 (A1). Por otro lado, en el año 2015, 2017 y 2021 se publicaron en cada uno de ellos dos (2) artículos primarios evaluados (A8, A9, A7, A10, A2 y A11 respectivamente), equivalente al 54.54% de los artículos primarios totales (Figura 3).

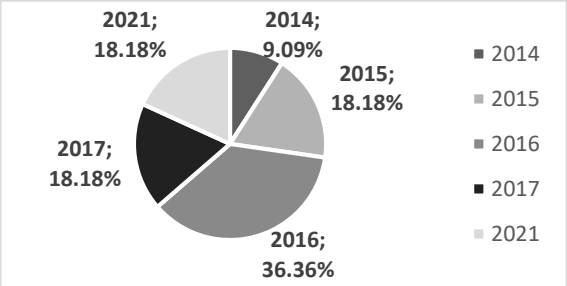
Por otro lado, teniendo en cuenta la figura 3(b), el año 2016 fue el año con más artículos primarios publicados con un total de (4) que equivalen al 36.36% (A3, A4, A5, A6) de todos los artículos. También se puede evidenciar en los años 2012, 2013, 2018, 2019, 2020 y 2022 no se encontraron estudios primarios sobre metaheurísticas que encuentran reglas Golomb óptimas o cercanamente óptimas.

Figura 3. Distribución de tiempo de los estudios primarios evaluados.

Fuente: Elaboración propia.



(a). *Cantidad de artículos primarios evaluados por año de publicación.*



(b). *Distribución porcentual de publicaciones por año.*

2.2.3.2. ¿Cuáles son las revistas y conferencias más importantes en el área?

La tabla 18 se muestra la agrupación de los artículos primarios evaluados según el evento en donde fueron publicados (Revista o Evento científico), así como el porcentaje aproximado (%) de los artículos primarios evaluados publicados en cada uno de los sitios, por ejemplo, para E1 se tiene un porcentaje de 18.185% que se calcula al dividir los dos (2) artículos publicados en E1 (A2, A5) por el total de artículos primarios (11) y luego multiplicando el resultado por cien (100).

Tabla 18. Listado de revistas y eventos con sus publicaciones.

Id	Sitio de publicación	Tipo	Artículos	%
E1	Journal of The Institution of Engineers (India)	R	A2, A5	18.185
E2	International Journal of Computer Applications	R	A3, A8	18.185
E3	Applied Soft Computing Journal	R	A1	9.09
E4	International Conference on Signal Processing and Communication (ICSC)	P	A4	9.09
E5	International Journal of Mathematical, Computational, Physical, Electrical and Computer Engineering	R	A6	9.09
E6	Journal Open Mathematics	R	A7	9.09
E7	International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research	P	A9	9.09
E8	Journal of Heuristics	R	A10	9.09
E9	Applications of Flower Pollination Algorithm and its Variants	P	A11	9.09

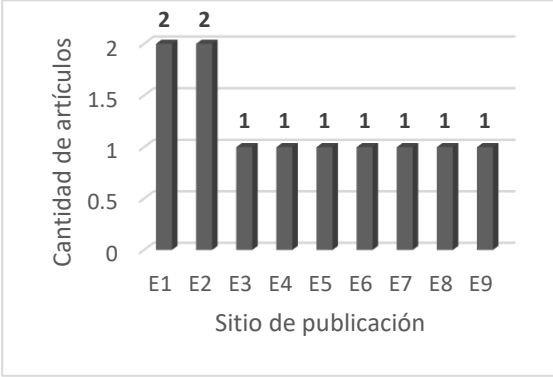
R: Revista, P: Proceedings de conferencias o eventos científicos. %: Porcentaje aproximado.

Por otro lado, la figura 4(a) nos muestra la cantidad de artículos primarios evaluados por sitio de publicación, en donde se destacan las revistas “Journal of The Institution of Engineers (India) y “International Journal of Computer Applications” por tener la mayor cantidad de artículos primarios evaluados publicados, dos (2) en cada una de ellas. Así mismo, se puede evidenciar que los sitios de publicación E3, E4, E5, E6, E7, E8 y E9 cuentan con un único (1) artículo primario evaluado publicado.

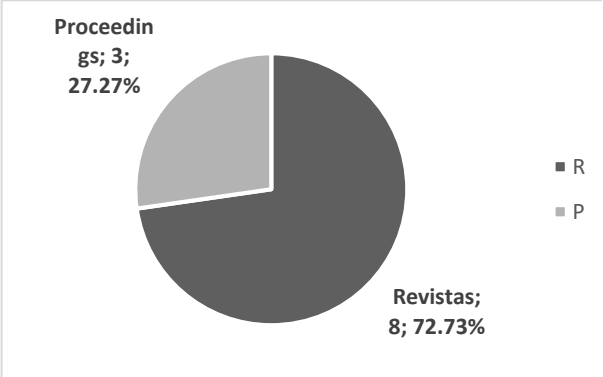
Adicionalmente, en la figura 4(b) se puede evidenciar que el 72.73% de los artículos fueron publicados en Revistas científicas y el otro 27.27% fueron publicados como Proceedings de eventos científicos.

Figura 4. Sitios de publicación de los estudios primarios evaluados.

Fuente: Elaboración propia.



(a). *Cantidad de artículos primarios evaluados por sitio de publicación.*



(b). *Distribución porcentual de artículos primarios evaluados por sitio de publicación*.

2.2.3.3. ¿Cuál es la distribución local de los estudios primarios?

La tabla número 19 representa las Universidades de los autores que participaron en los artículos primarios evaluados.

Tabla 19. Listado de publicaciones con sus Universidades.

Artículo primario	Universidad - Local
A3, A4, A6, A7, A8	PEC University of Technology
A2, A6, A11	Chandigarh University
A2, A11	University Institute of Engineering
A5, A11	Punjab Engineering College
A9, A10	Griffith University
A1	Institute of Science and Technology
A8	Seth Jai Parkash Mukand Lal Institute Of Engineering & Technology
A4	Jaypee Institute of Information Technology
A3	Haryana Engineering College

La figura 5 representa la cantidad de artículos según la Universidad en donde fueron publicados. La universidad con mayor cantidad de artículos primarios publicados es el “PEC University of Technology” con un total de cinco (5) artículos, seguido de “Chandigarh University” con tres (3) artículos publicados. Después, se evidencia que las universidades “Punjab Engineering College”, “University Institute of Engineering” y “Griffith University” cuentan con dos (2) artículos primarios publicados en cada una de ellas. Y así mismo, las universidades “Haryana Engineering College”, “Institute of Science and Technology”, “Seth Jai Parkash Mukand Lal Institute Of Engineering & Technology” y “Jaypee Institute of Information Technology" tienen un (1) artículo primario evaluado publicado.

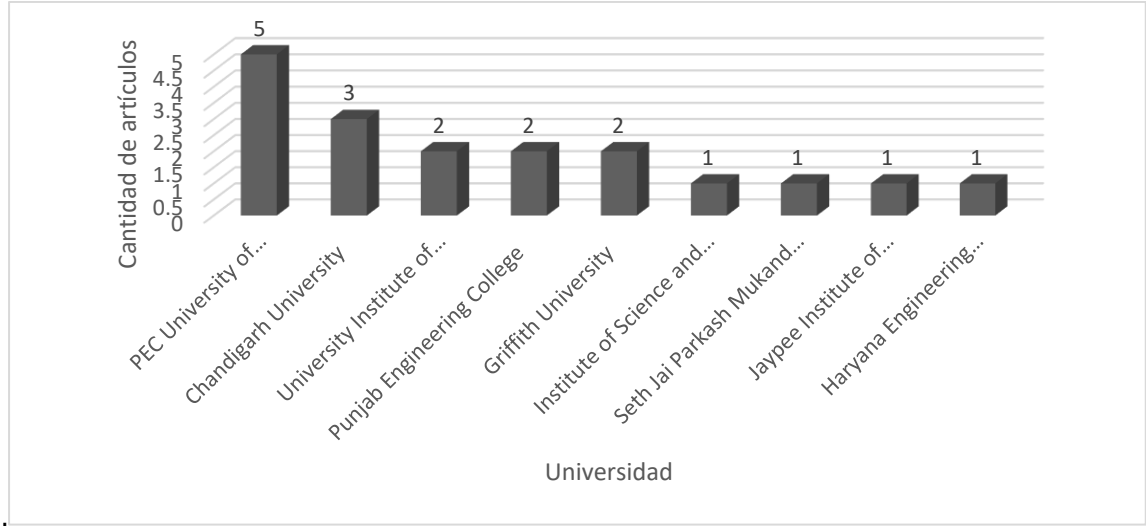


Figura 5. Cantidad de artículos primarios evaluados por Universidad.

Fuente: Elaboración propia.

2.2.3.4. ¿Cuál es la distribución geográfica de los estudios primarios?

La figura 6, y 7(b) muestran que las distribuciones geográficas de los artículos primarios provienen de dos (2) países ubicados en continentes diferentes, siendo India el país con la mayor contribución científica, aportando 9 artículos primarios evaluados que equivalen al 81.82% de los artículos primarios evaluados totales, seguido por Australia con dos (2) artículos equivalentes al 18.18% de los artículos.

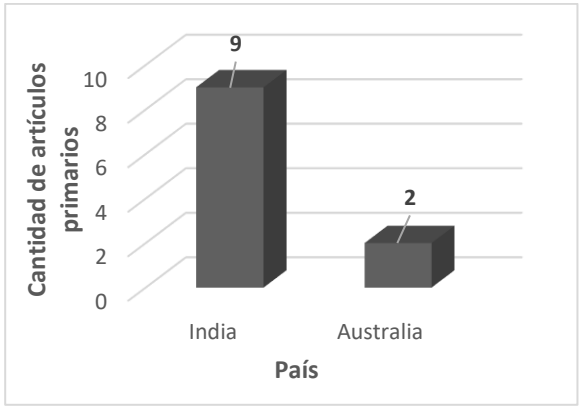
Figura 6. Distribución geográfica de los estudios primarios evaluados.

Fuente: Elaboración propia.

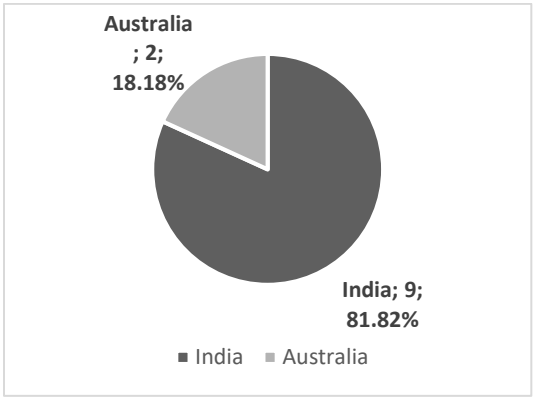
En la figura 7(a), se puede evidenciar la cantidad de artículos primarios evaluados en cada uno de los países en donde fueron publicados, destacando nuevamente la cantidad de artículos que aporta el país de India (9). Por otro lado, la figura 6(b), se puede evidenciar cual es el porcentaje de artículos primarios evaluados publicados en los diferentes países, destacando a India de nuevo con el 81.82% de los artículos primarios evaluados totales.

Figura 7. Países en donde fueron publicados los estudios primarios evaluados.

Fuente: Elaboración propia.



(a). *Cantidad de artículos primarios evaluados por país.*



(b). *Distribución porcentual de artículos primarios evaluados por país de publicación.*

2.2.3.5. ¿Cuáles son los autores más citados en el área?

Teniendo en cuenta la cantidad de citaciones mostradas (NC) en la tabla 17, se puede elaborar la tabla 20 en donde podemos apreciar en que artículos primarios evaluados ha participado cada uno de los autores, así como la cantidad de citaciones que ha tenido cada autor, el dato de esta columna se obtiene al sumar el número de citaciones de cada artículo mostrado en la tabla 17.

Tabla 20. Listado de autores con sus publicaciones y citaciones.

Id	Autor	Artículos	Citaciones
Au1	Shonak Bansal	A1, A2, A3, A4, A5, A6, A7, A8, A11	73
Au2	Neena Gupta	A5, A6, A7, A11	31
Au3	Arun Kumar Singh	A5, A6, A7, A11	31
Au4	Alam Polash	A9, A10	13
Au5	Hakim Newton	A9, A10	13
Au6	Abdul Sattar	A9, A10	13
Au7	Suruchi Bali	A8	11
Au8	Anil Kamboj	A8	11
Au9	Kuldeep Sharma	A4	9
Au10	Jyoti Vyas	A4	9
Au11	Shweta Sharma	A3	6
Au12	Ritu	A3	6
Au13	Prince Jain	A6	1

La figura 8 se destaca el artículo primario evaluado A5 por tener dieciséis (16) citaciones. También podemos ver que los artículos A8, A7 Y A1 obtuvieron más de diez (10) citaciones cada uno, seguido por los artículos A4 y A10 con nueve (9) y siete (7) citaciones respectivamente. Adicionalmente, se puede evidenciar que los artículos primarios evaluados con la menor cantidad de citaciones fueron el A2, A6 y A11 con una (1) citación cada uno.

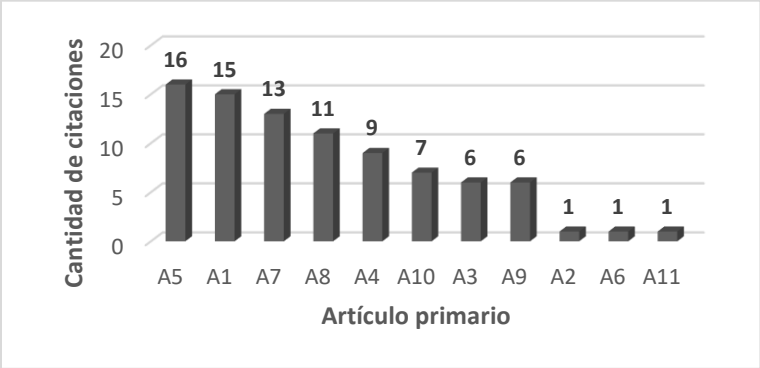


Figura 8. Número de citaciones por artículo primario evaluado.

Fuente: Elaboración propia.

Por otro lado, en la figura 9 se puede observar la cantidad de artículos primarios evaluados en los que ha participado cada uno de los autores. De esta manera, se puede destacar que el autor Shonak Bansal ha sido el autor con la mayor cantidad de artículos primarios publicados con un total de nueve (9), seguido por los autores Neena Gupta y Arun Kumar Singh publicando cuatro (4) artículos cada uno. Después, los autores Alam Polash, Hakim Newton y Abdul Sattar publicando dos (2) artículos cada uno en Australia. Asimismo, el resto de autores de la tabla 20 (Del Au7 al Au13) participaron en la publicación de un (1) artículo primario evaluados.

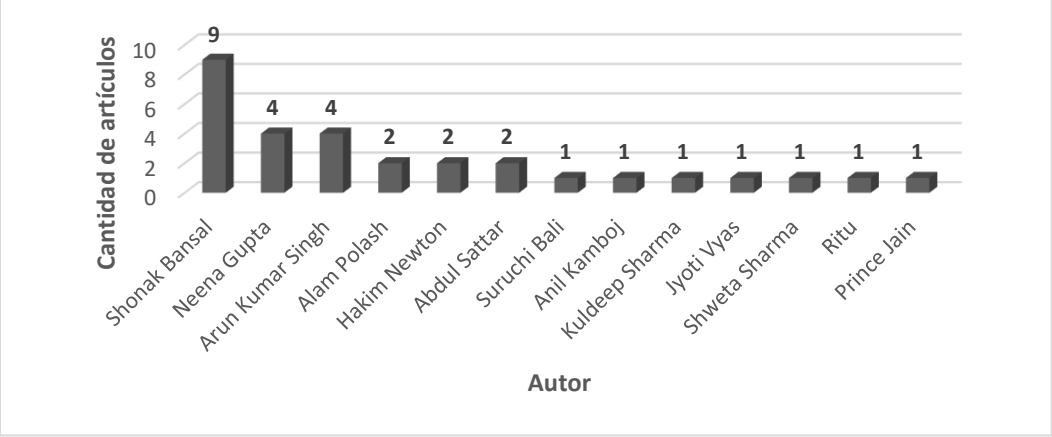


Figura 9. Número de artículos primarios evaluados por autor.

Fuente: Elaboración propia.

Por otra parte, en la figura 10 se puede evidenciar el número de citaciones que ha obtenido cada uno de los autores en los respectivos artículos en los que participo. Se puede destacar nuevamente que el autor Shonak Bansal fue el autor con mayor cantidad de citaciones con

un total de setenta y tres (73), seguido Neena Gupta y Arun Kumar Singh con treinta y un (31) citaciones cada uno. Seguido por Alam Polash, Hakim Newton y Abdul Sattar que consiguieron un total de 13 citaciones cada uno. También se puede notar que el autor Prince Jain obtuvo únicamente una citación (1) en su artículo.

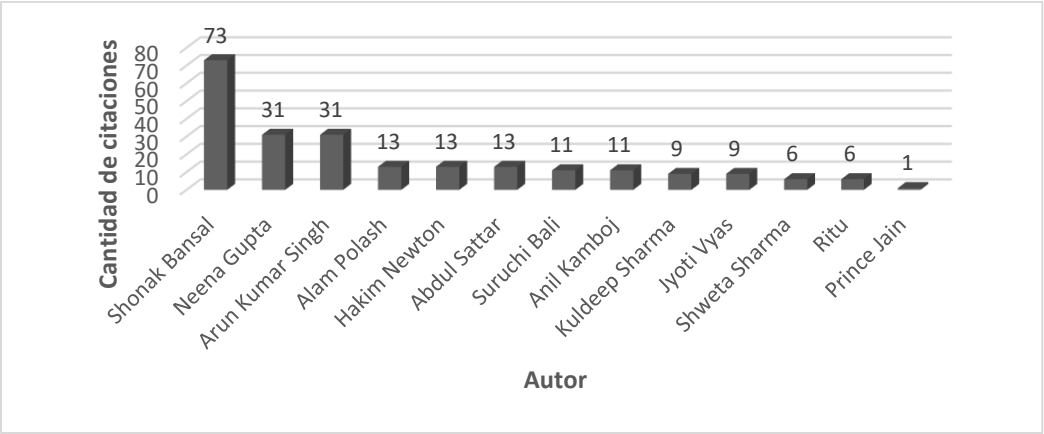


Figura 10. Número de citaciones por autor.

Fuente: Elaboración propia.

2.2.3.6. ¿Cuáles son los lenguajes de programación más importantes en el área?

En la tabla 21 se puede encontrar el lenguaje de programación en el que fueron implementadas las diferentes metaheurísticas de los artículos primarios evaluados.

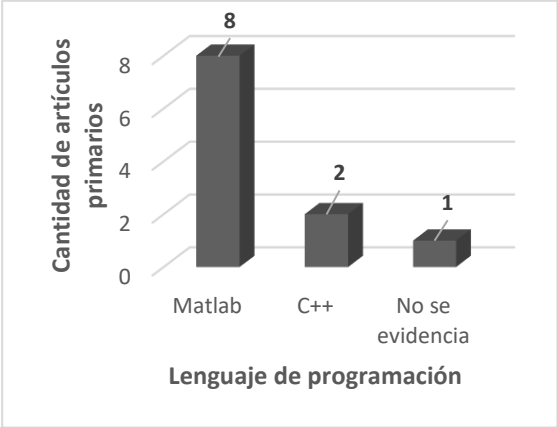
Tabla 21. Listado de publicaciones con sus respectivos lenguajes de programación.

Artículo primario	Lenguaje de programación
A1	Matlab
A2	No se evidencia
A3	Matlab
A4	Matlab
A5	Matlab
A6	Matlab
A7	Matlab
A8	Matlab
A9	C++
A10	C++
A11	Matlab

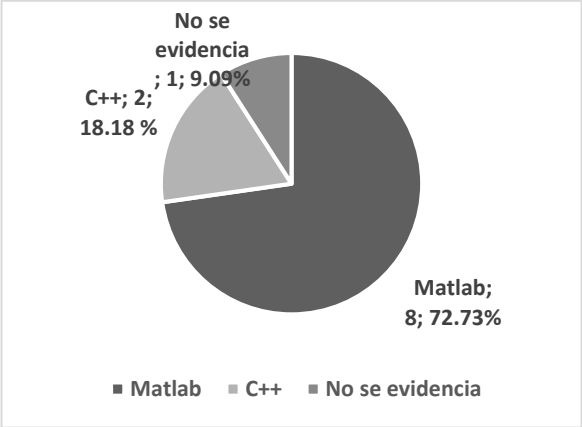
En la figura 11(a) se puede apreciar la cantidad de artículos primarios evaluados según el lenguaje de programación en el que fue implementado. Así mismo, en la figura 11(b) se puede apreciar que Matlab fue el lenguaje de programación más relevante ya que fue utilizado en nueve (8) artículos equivalentes al 72.73% de los artículos totales, seguido por C++ utilizado en dos (2) artículos primarios evaluados equivalentes al 18.18% de los artículos. Por otro lado, la figura 11(b) nos muestra que el 9.09% de los artículos primarios no evidencian o reportan el uso de algún lenguaje de programación para su propuesta.

Figura 11. Lenguajes de programación utilizados en los estudios primarios evaluados.

Fuente: Elaboración propia.



(a). *Número de artículos primarios evaluados por lenguaje de programación.*



(b). *Distribución porcentual de artículos primarios evaluados por lenguaje de programación.*

2.2.3.7. ¿Qué resultados se han alcanzado con las propuestas realizadas?

En la figura 12 se puede apreciar que los mejores resultados fueron alcanzados por los artículos primarios evaluados A5, A6, A7 y A11 encontrando reglas Golomb óptimas hasta veinte (20) marcas, cabe destacar que los artículos A5, A6 Y A11 proponen metaheurísticas multi-objetivo para resolver el problema de investigación. Seguido por el artículo A10 que trabajo reglas Golomb óptimas hasta diecinueve (19) marcas, los artículos A3 y A8 que encontraron reglas Golomb óptimas hasta diecisiete (17) marcas. También se puede resaltar que el artículo A1 obtuvo el resultado menos favorable en términos de reglas Golomb óptimas, encontrando reglas hasta 6 marcas.

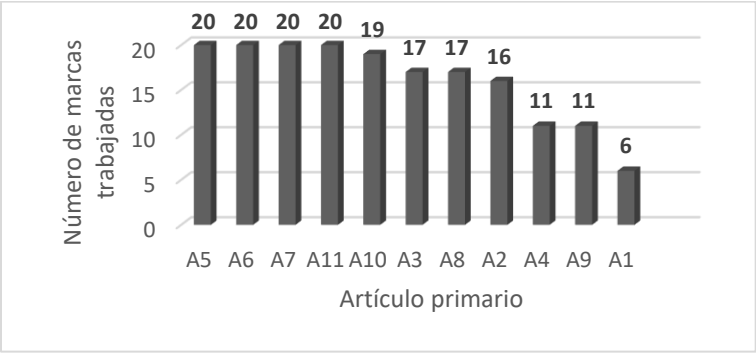


Figura 12. Número de marcas trabajas en términos de reglas Golomb óptimas por cada artículo primario.

Fuente: Elaboración propia.

Por otro lado, la figura 13 nos muestra que el artículo A1 fue el que obtuvo mejores resultados en términos de reglas Golomb cercanamente óptimas, encontrando hasta veintiuno (21) marcas, seguido por los artículos primarios A2, A3 Y A8 que encontraron reglas cercanamente óptimas hasta veinte (20) marcas, después el artículo A9 encontró

reglas hasta diecisiete (17) marcas y finalmente el artículo A4 encontró reglas Golomb cercanamente óptimas hasta trece (13) marcas. Adicionalmente, se puede evidenciar que los artículos A5, A6, A7, A10 y A11 no encontraron reglas Golomb cercanamente óptimas.

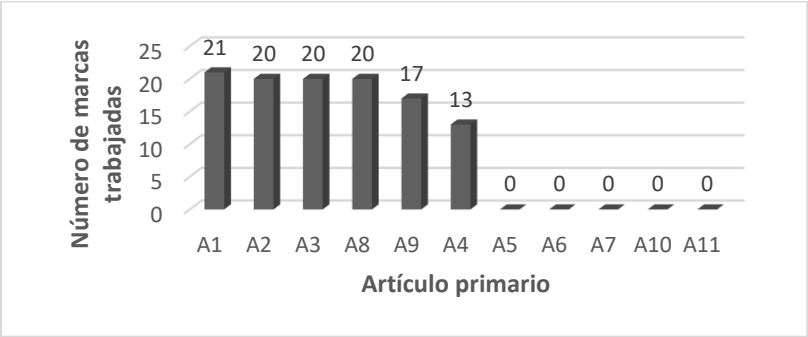


Figura 13. Número de marcas trabajas en términos de reglas Golomb cercanamente óptimas por cada artículo primario.

Fuente: Elaboración propia.

Por otra parte, la figura 14 nos muestra los resultados de los artículos primarios evaluados en términos de reglas Golomb óptimas y cercanamente óptimas destacando que los artículos que obtuvieron mejores resultados en términos de reglas óptimas (A5, A6, A7, A10 Y A11) no encontraron reglas Golomb cercanamente óptimas. Adicionalmente, se puede destacar que el artículo primario evaluado A1 no logró obtener los mejores resultados en términos de reglas Golomb óptimas (6) pero, aun así, logro extender su investigación para trabajar con reglas cercanamente óptimas de un orden superior hasta 21 marcas.

OGR: Reglas Golomb óptimas, N-OGR: Reglas Golomb cercanamente óptimas.

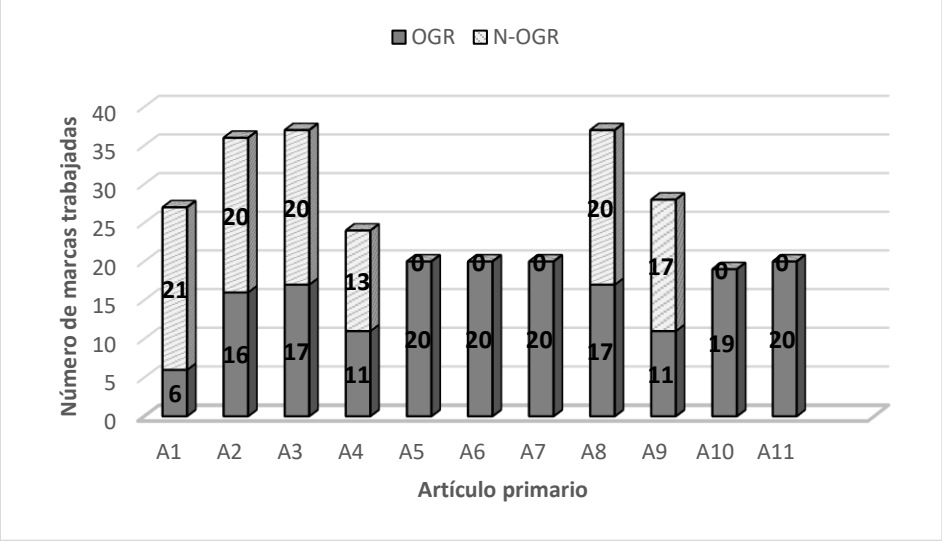


Figura 14. Número de marcas trabajas en términos de reglas Golomb óptimas y cercanamente por cada artículo primario.

Fuente: Elaboración propia.

2.2.3.8. ¿Qué beneficios y desafíos conlleva la investigación en el tema?

En la literatura analizada se evidencian algunos beneficios y desafíos relacionados al problema de investigación abordado por las diferentes propuestas.

Beneficios

- Las propuestas logran encontrar reglas Golomb óptimas y cercanamente óptimas en poco tiempo de máquina, teniendo en cuenta las limitaciones de máquina con las que contaban [Figura 14]. Este tiempo es corto si se compara con los resultados obtenidos por los algoritmos de fuerza bruta que han abordado el problema de investigación OGR [18].
- Se propone una metaheurística diferente para abordar el problema de investigación, cumpliendo con el teorema de No Free Lunch [51], en donde se menciona que, no se puede asegurar cuál es la mejor metaheurística existente para resolver un problema de optimización específico hasta que todas las metaheurísticas hayan sido estudiadas y comparadas experimentalmente en ese problema de optimización.

Desafíos

- La generación de reglas Golomb óptimas es un problema de optimización extremadamente desafiante [45, 36, 49] por eso el estado del arte ha trabajado reglas Golomb óptimas hasta 20 marcas [Figura 12] y reglas Golomb cercanamente óptimas hasta 21 marcas [Figura 13].
- A pesar de obtener buenos resultados en problemas de optimización con metaheurísticas teniendo en cuenta las limitaciones de máquina, este tipo de algoritmos no puede garantizar una solución óptima a un determinado número de marcas a menos que ya se conozca la regla Golomb óptima con ese mismo número de marcas para así, compararla con los resultados de la metaheurística y de esta manera, determinar si la longitud es óptima. Esto se debe a que la naturaleza de este tipo de algoritmos se encarga de encontrar soluciones aproximadas a las óptimas en un tiempo de máquina razonable [52, 53, 32].

2.2.4. Discusión.

2.2.4.1. Observaciones principales.

El objetivo de este mapeo sistemático fue identificar las principales iniciativas científicas relacionadas a las *metaheurísticas que encuentran reglas Golomb óptimas o cercanamente óptimas* y tras analizar los resultados del mapeo detalladamente, se pueden concluir las siguientes observaciones:

- Los mejores resultados fueron obtenidos por artículos primarios evaluados con propuestas de tipo multi-objetivo [Tabla 17, Figura 12], es decir, planteaban un algoritmo de optimización base y posteriormente, se diseñaba una versión multi-objetivo de este algoritmo para mejorar las reglas Golomb encontradas.
- Las versiones híbridas de los algoritmos propuestos tienden a mejorar los resultados de los artículos primarios [Tabla 17, Figura 12], esto se debe a que los autores aprovechan los diferentes parámetros de configuración de las metaheurísticas para explotar el espacio de búsqueda y así encontrar mejores soluciones al problema.

- Las metaheurísticas evidenciadas en los artículos primarios no tienen en cuenta la naturaleza matemática del problema de investigación para abordarlo, hay que recordar que este problema de investigación nace de un problema matemático [1, 22, 23, 24], y que, por esta razón, en esta área han abordado diferentes cualidades y elementos que pueden ser útiles para resolver el problema de reglas Golomb.

La importancia de este mapeo sistemático radica en la contribución con respecto a las metaheurísticas que encuentran reglas Golomb óptimas o cercanamente óptimas, debido a que los estudios primarios se enfocan en proponer algoritmos de optimización que abordan el tema de investigación. Gracias a esto, consideramos que este mapeo sistemático es de suma importancia para investigaciones futuras porque brinda una visión general de lo que ha trabajado la literatura con respecto a las metaheurísticas que encuentran reglas Golomb óptimas o cercanamente óptimas durante los últimos diez (10) años.

2.2.4.2. Limitaciones del mapeo sistemático.

A pesar de que el idioma inglés es ampliamente utilizado como el idioma principal para redactar y publicar artículos de investigación, restringir la búsqueda exclusivamente a este idioma podría llevar a descartar artículos que podrían ser relevantes en el tema de investigación redactados en diferentes idiomas. Por otro lado, la ventana de tiempo escogida para llevar a cabo el mapeo sistemático también se podría considerar como una limitación, ya que podrían existir artículos de investigación con más de diez (10) años de antigüedad y que podrían ser relevantes para la investigación.

2.2.4.3. Trascendencia para la investigación y la práctica.

Es fundamental destacar que las diferentes características y elementos del mapeo sistemático pueden tener un impacto y trascendencia significativa en investigaciones y prácticas futuras. Esto se debe a que los artículos primarios seleccionados generan gran valor al tema de investigación, ya que, proporcionan conceptos e ideas, tales como:

- Brindan información relacionada a los parámetros de configuración utilizados en las diferentes metaheurísticas, tales como, operadores de selección, mutación, población inicial, número de iteraciones. Destacando la importancia de escoger los parámetros ideales para obtener los mejores resultados.
- Los resultados de los artículos primarios evidencian la complejidad del problema de investigación y del esfuerzo que se necesita para abordarlo, esto se puede notar en el número de marcas trabajadas por las propuestas, ya que, el número de marcas trabajadas se encuentra por debajo de la mejor regla Golomb óptima conocida hasta el momento de hasta veintiocho (28) marcas. Los resultados demuestran que las mejores reglas Golomb encontradas por los artículos tienen una longitud de veinte (20) marcas para el caso de las reglas Golomb óptimas y veintiún (21) marcas para el caso de reglas Golomb cercanamente óptimas.

El mapeo sistemático nos proporciona una visión general de las métricas de evaluación utilizadas en la literatura para probar el rendimiento de los algoritmos. Esto nos da una idea de que métricas deberíamos utilizar si queremos proponer una metaheurística para abordar el problema de investigación.

Capítulo 3. Construcción de la base conceptual.

En este capítulo se presentarán los conceptos matemáticos relevantes para el desarrollo de la propuesta de investigación tales como construcciones algebraicas, cotas y traslaciones. Adicionalmente, estos conceptos fueron utilizados para realizar la creación del algoritmo memético buscando responder la pregunta de investigación definida en el planteamiento del problema.

3.1. Conceptos preliminares.

Antes de empezar a describir los conceptos matemáticos que fueron integrados en el algoritmo memético daremos un breve repaso a algunos de los conceptos relevantes sobre teoría de números que son necesarios para entender dichos conceptos. Adicionalmente, para entender el tema de teoría de números con mayor detalle, revisar los estudios de Apostolos Dimitromanolakis [3] y Carlos Martos [54] en donde se describe de una manera más formal el tema mencionado.

3.1.1. Números primos.

Un número primo (p) es un número entero mayor a uno (p > 1) con la particularidad de que tiene exactamente dos “divisores positivos”: Uno (1) y el numero primo en sí mismo (p), conocidos como divisores positivos [3]. Esto significa que, cuando se divide el número primo entre uno (1), el residuo será cero (0) y cuando se divide ese número primo entre sí mismo, el residuo también será cero (0). Así mismo, si dividimos el número primo entre cualquier número real, diferente a los “divisores positivos”, el resultado del residuo siempre será diferente de cero (0).

Por ejemplo, si tomamos el número primo siete (7) y lo dividimos entre 1 y el mismo (7) nos dará cero como residuo en ambas operaciones [Figura 15].

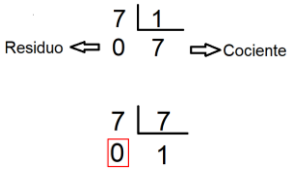


Figura 15. División entera.

Fuente: Elaboración propia.

En cambio, si dividimos el 7 con un número diferente, el residuo siempre será diferente de cero (0). Esta particularidad es la que caracteriza a los números primos de los demás números enteros [Figura 16].

Figura 16. División entera número primo.

Fuente: Elaboración propia.

Por otro lado, el número ocho (8) no es considerado como un número primo debido a que si lo dividimos por cuatro (4), obtendremos un residuo de cero (0) como resultado. Esto rompe la regla de los números primos, ya que el residuo solo puede ser cero cuando se divide el primo entre 1 y entre el mismo [Figura 17].

Figura 17. División entera número no primo.

Fuente: Elaboración propia.

Los números primos tienen un papel importante en las construcciones algebraicas debido a que estas funcionan con potencias primas, es decir, vamos a necesitar un número primo base y una potencia (r) en las construcciones para definir el número de marcas de la regla Golomb que queremos encontrar. La potencia prima se define en la figura 18, en donde "q", representara el número de marcas de la regla Golomb que queremos encontrar al utilizar las construcciones algebraicas [Figura 18].

$$q = p^r$$

Figura 18. Potencia prima.

Fuente: Elaboración propia.

3.1.2. Campos finitos.

Los campos finitos [60, 61], también conocidos como cuerpos finitos o campos de Galois [59], son estructuras algebraicas fundamentales en matemáticas y aplicaciones prácticas en diferentes áreas como en la criptografía [62], la teoría de códigos [63] y la teoría de números [64]. Un campo finito es un conjunto finito de elementos en el que se pueden realizar operaciones aritméticas, como suma, resta, multiplicación y división, de manera similar a los números reales o los números complejos.

Un campo finito se denota como:

$$GF(p)\ ;\ F(p)$$

Donde *p* es un número primo y hace referencia al número de elementos que tendrá el campo finito. Matemáticamente, el primo "p" hace referencia al grado de la extensión del campo [59] y determina los elementos posibles que este tendrá, por ejemplo, si p es igual a 7, entonces los elementos del campo estarían representados de la siguiente manera.

$$GF(7) = \{\ 0, 1, 2, 3, 4, 5, 6\ \}$$

Cabe mencionar, que los campos de Galois también se pueden trabajar con potencias primas en vez de números primos [3] de esta manera: $GF(p^r)$. Este es un detalle importante ya que, en el desarrollo de la propuesta de investigación, se utilizará el concepto de campos de Galois con potencias primas. Adicionalmente, todo campo de Galois con p elementos cuenta con al menos un generador conocido como elemento primitivo o raíz primitiva. Matemáticamente se define [3] en la figura 19 y será utilizado posteriormente por las construcciones algebraicas para generar reglas Golomb [Figura 19].

$$\theta^i \equiv 1 \, mod \, p$$

$$0 \leq i \leq p-1$$

Figura 19. Elemento primitivo.

3.2. Construcciones algebraicas.

En la actualidad, existen métodos o construcciones algebraicas que se encargan de encontrar reglas Golomb cercanamente óptimas. Estas construcciones utilizan métodos matemáticos relacionados a campos finitos para encontrar conjuntos B_2 [54], una equivalencia de las reglas Golomb. De esta forma, para el desarrollo de la propuesta de investigación, se implementaron las construcciones algebraicas recomendadas por la literatura [3, 54, 55] conocidas como Bose, Singer y Ruzsa debido a que estas logran encontrar reglas Golomb con una longitud menor a n^2 [3], donde n representa el número de marcas de la regla Golomb que queremos encontrar. En las siguientes secciones, se presentará la definición formal de cada construcción y al finalizar la construcción de Ruzsa se realizará un ejemplo para complementar los conceptos presentados.

3.2.1. Bose.

La construcción de Bose [3] aparece en 1962, cuando Bose prueba en [22] un teorema análogo al de Singer [23], logrando generar reglas Golomb a partir de un campo de Galois con extensión de grado 2. Vamos a presentar primero la construcción de Bose.

Inicialmente, se define un campo de Galois con extensión de grado 2 de la siguiente forma:

$$GF(q^2)$$

$$q = p^r$$

En donde

- ***p:*** Número primo que nos ayuda a encontrar el número de marcas.
- ***r:*** Exponente entero que nos ayuda a encontrar el número de marcas.
- ***q:*** Potencia prima que representa el número de marcas de la regla Golomb

Una vez se tenga definido el campo de Galois, se deben realizar las siguientes operaciones con sus elementos y con su respectivo elemento primitivo:

$$B_0(q,\theta) = \left\{ a_0 , (\theta^{q+1})^i , (\theta^{q+1})^{i+1} , \dots , (\theta^{q+1})^{q-2} \right\}$$

$$a \in GF(q^2)$$

$$0 \leq i \leq q-2$$

$$\theta \rightarrow Elemento \; primitivo \; de \; GF(q^2)$$

Después, se aplica el logaritmo en base θ (elemento primitivo) a los elementos generados en el paso anterior para encontrar las marcas finales de la regla Golomb que queremos encontrar con la construcción Bose.

$$B(q,\theta) = Log_\theta\ (\theta + B_0) = \{\ Log_\theta\ (\theta + a_i),\ Log_\theta\ (\theta + a_{i+1}), \dots,\ Log_\theta\ (\theta + a_q\ )\ \}$$

$$a\ \in\ B_0$$

$$0 \leq i \leq q$$

Finalmente, los elementos generados en el paso anterior deben ser ordenados para que tengan la estructura correcta de una regla Golomb.

$$B = \{a_i, a_{i+1}, \dots, a_q\}$$

$$a_i < a_{i+1}\ \rightarrow\ 1 \leq i \leq\ q$$

3.2.2. Singer.

La construcción de tipo Singer [23] apareció en el año 1938 en un problema análogo a las reglas Golomb. A diferencia de la construcción Bose, Singer se encarga de generar reglas Golomb a partir de un campo de Galois con extensión de grado 3.

Inicialmente, se define un campo de Galois con su respectiva extensión de la siguiente forma:

$$GF(q^3)$$

$$q = p^r$$

En donde

- ***p:*** Número primo que nos ayuda a encontrar el número de marcas.
- ***r:*** Exponente entero que nos ayuda a encontrar el número de marcas.
- ***q:*** Potencia prima que representa el número de marcas de la regla Golomb

Una vez se tenga definido el campo de Galois, se deben realizar las siguientes operaciones con sus elementos y con su respectivo elemento primitivo:

$$n_q = q^2 + q + 1 = \frac{q^3 - 1}{q - 1}$$

$$S_0(q,\theta) = \{\ a_0\ , \theta^{i\ n_q}, \theta^{(i+1)\ n_q}\ , \dots, \theta^{(q-2)\ n_q}\ \}$$

$$a \in GF(q^3)$$

$$0 \leq i \leq q - 2$$

$$\theta \rightarrow Elemento\ primitivo\ de\ GF(11^3)$$

Finalmente, se aplica el logaritmo en base θ (elemento primitivo) a los elementos generados en el paso anterior y posteriormente se aplica el modulo $\boldsymbol{n_q}$ a cada elemento para encontrar las marcas finales de la regla Golomb que queremos encontrar la construcción Singer.

$$S(q,\theta) = Log_\theta\ (\theta + S_0) = \{\ Log_\theta\ (\theta + a_i)\ mod\ n_q,\ Log_\theta\ (\theta + a_{i+1})\ mod\ n_q, \dots,\ Log_\theta\ (\theta + a_q)\ mod\ n_q\}$$

$$a\ \in\ S_0$$

Finalmente, los elementos generados en el paso anterior deben ser ordenados para que tengan la estructura correcta de una regla Golomb.

$$S = \{a_i, a_{i+1}, \dots, a_q\}$$

$$a_i < a_{i+1} \ \rightarrow 1 \leq i \leq \ q$$

$$0 \leq i \leq q$$

3.2.3. Ruzsa.

La construcción de tipo Ruzsa [24] apareció en el año 1993 en un problema análogo a las reglas Golomb y es la más sencilla de las 3 construcciones presentadas debido a que no requiere la generación de un campo de Galois para obtener el elemento primitivo, sino que obtiene el elemento primitivo a partir del número primo $\boldsymbol{p}$.

Matemáticamente, la construcción de Singer se define como:

$$R(p, \theta) = \left\{ \left(i\, p - \theta^i\, (p-1) \right) \ mod\ (p^2 - p) \right\}$$

$$1 \leq i \leq p-1$$

$$\theta \ \rightarrow \ E\text{lemento primitivo generado a partir de } p$$

En donde $\boldsymbol{\theta}$ es el elemento primitivo generado a partir del número primo $\boldsymbol{p}$**.** Finalmente, los elementos generados en el paso anterior deben ser ordenados para que tengan la estructura correcta de una regla Golomb.

$$R = \{a_i, a_{i+1}, \dots, a_{p-1}\}$$

$$a_i < a_{i+1} \ \rightarrow 1 \leq i \leq \ p-1$$

Cabe resaltar, que la construcción Ruzsa genera reglas Golomb con p-1 elementos, esto quiere decir, que si el número primo $\boldsymbol{q}$ es igual a 7, entonces la construcción generara reglas Golomb con **6** marcas, esto es algo que se tendrá en cuenta en la implementación del algoritmo.

Por ejemplo:

(1) Para obtener una regla Golomb de **4** marcas se define el siguiente número primo:

$$p = 5$$

(2) Se define el elemento primitivo θ a partir del número $\boldsymbol{p}$ y se realizan las respectivas operaciones:

$$\theta = 2$$

$$p^2 - p = 25 - 5 = 20$$

$$R(5,2) = \{\, 5 - 2(4), 10 - 4(4)\, , 15 - 8(4)\, , 20 - 16(4)\, \}\, mod\ 20$$

$$R(5,2) = \{-3\, , -6\, , -17\, , -44\, \}\, mod\ 20$$

$$R(5,2) = \{\, 17\, , 14\, , 3\, , 16\, \}$$

(3) Finalmente, se ordena el resultado anterior para obtener la regla Golomb de longitud 17 generada por la construcción Rusza.

$$\boldsymbol{R(5,2) = \{\, 3, 14, 16, 17 \,\}}$$

$$\boldsymbol{l = 17 - 3 = 14}$$

3.3. Traslaciones.

Es una propiedad matemática que proviene de la geometría y será utilizada en el proceso de generación de reglas Golomb como un componente fundamental para mejorar los resultados de nuestra propuesta. Matemáticamente, una traslación es una transformación que desplaza todos los puntos de un objeto o figura en la misma dirección y distancia [56]. La forma y el tamaño del objeto no cambian durante una traslación, solo su posición en el espacio. Análogamente, el objeto o figura en nuestro proyecto de investigación hace referencia a las reglas Golomb que queremos encontrar y al hacer uso de traslaciones en nuestra propuesta, mejoraremos las reglas Golomb obtenidas desplazando todas las marcas hacia una misma dirección.

Por otro lado, aplicando las traslaciones en el contexto de reglas Golomb [54, 55] tendremos lo siguiente, matemáticamente:

$$uA + t = \{u\, a_i + t, u\, a_{i+1} + t, \dots, u\, a_n + t\}$$

$$a \in A$$

$$m = p^2 - 1$$

$$1 \leq i \leq n$$

$$1 \leq t \leq m$$

En donde

- ***A:*** Regla Golomb.
- ***p:*** Representa el número de marcas de la regla Golomb.
- ***m***: Modulo de la regla Golomb.
- ***u:*** Primos relativos de ***m*** y hace referencia al desplazamiento que sufrirá cada una de las marcas en la regla Golomb.
- ***t:*** Hace referencia al desplazamiento que sufrirá cada una de las marcas en la regla Golomb y tendrá valores entre 1 y ***m***.
- ***n:*** Numero de marcas de la regla Golomb.
- ***i:*** Índice de la marca y tendrá valores entre 1 y ***n.***
- **a:** Marca o elemento de la regla Golomb.

Los primos relativos de m, también conocidos como primos entre sí, son números entre 2 y m-1 que no tienen ningún factor primo en común con *m*, excepto el 1 [57]. En otras palabras, dos números son primos relativos si su máximo común divisor (MCD) es igual a 1.

$$Primo\ relativo = MCD(m, i) = 1$$

$$2 \leq i \leq m - 1$$

Para el desarrollo de esta propuesta, los primos relativos se obtienen a partir de la función phi de Euler [3] que matemáticamente se define como:

$$\phi(m) = \{a < m \,\&\, MCD(m, a) = 1\}$$

Después de las primeras operaciones, se le debe aplicar el módulo ***m*** a cada una de las marcas obtenidas, el módulo ***m*** y sus operaciones se presentan de la siguiente manera:

$$A_t \; mod \; m = \{a_i \; mod \; m, a_{i+1} \; mod \; m, \dots, a_n \; mod \; m\}$$

$$m = \; p^2 - 1$$

En donde

- $\boldsymbol{A_t}$: Regla Golomb después de aplicar las primeras operaciones.
- **a:** Marca o elemento de la regla Golomb después de aplicar las primeras operaciones.

Luego, la regla obtenida debe ser ordenada y será nombrada como A_o.

$$A_o = \{a_i, a_{i+1}, \dots, a_n\}$$

$$a_i < a_{i+1} \;\; \rightarrow \; 1 \leq i \leq \; n$$

Por último, se debe aplicar una normalización a cada una de las marcas de la regla ordenada para obtener finalmente como resultado una regla Golomb mejorada, es decir, una regla Golomb con menor longitud. Por otro lado, la normalización en estadística [58] se refiere a la práctica de ajustar los valores de una variable para que sigan una escala común o estándar con el fin de facilitar análisis estadísticos. Aplicado al contexto de reglas Golomb, la normalización nos ayuda a trabajar con reglas en una misma escala común, obligando a cada una de las reglas a que tengan la escala de a_i, es decir, cada marca de la regla Golomb será restada por el primer elemento de la regla llamado a_i. Matemáticamente:

$$A - \; a_i = \{a_i - \; a_i, a_{i+1} - \; a_i, \dots, a_n - a_i\}$$

Por ejemplo:

(1) Teniendo en cuenta la regla Golomb definida como:

$$A = \{1, 10, 29, 31, 35, 36, 46\}; p = 7$$

$$p \;\; \rightarrow \; \text{Primo que representa el nú}mero \; de \; marcas.$$

(2) Se calcula el módulo:

$$M = 7^2 - 1 = 49 - 1 = \mathbf{48}$$

(3) Se obtiene el u calculando los primos relativos de M utilizando la función phi de Euler, para el caso $M = 48$, tenemos:

$$u = \{\, 5, 7, 11, 13, 17, 19, 23, 25, 29, 31, 35, 37, 41, 43, 47\}$$

(4) Se obtienen los valores de t

$$1 \leq t \leq 48$$

(5) Se aplican las operaciones con u y t para encontrar el desplazamiento. Para el caso $u = 5$ y $t = 1$

$$5 \; A + 1 = 5 \; \{1, 10, 29, 31, 35, 36, 46\} + 1$$

$$5\,A + 1 = \{5, 50, 145, 155, 175, 180, 230\} + 1$$

$$A_t = 5\,A + 1 = \{6, 51, 146, 156, 176, 181, 231\}$$

Cabe aclarar, que se debe aplicar las traslaciones con los diferentes valores de u y t a la regla Golomb A hasta encontrar un resultado con una longitud menor a la misma regla, por simplicidad, en este ejemplo se utilizaron los valores de u y t mencionados anteriormente.

(6) Se aplica el módulo al resultado anterior:

$$A_t\ mod\ 48 = \{6, 3, 2, 12, 32, 37, 39\ \}$$

(7) Se ordena el resultado anterior:

$$A_o = \{\ 2, 3, 6, 12, 32, 37, 39\ \}$$

(8) Finalmente, se normaliza el resultado anterior con el fin de trabajar con la misma escala de la regla Golomb inicial:

$$A_f = A_o - \ 1 = \{\ 1, 2, 5, 11, 31, 36, 38\ \}$$

(9) Si comparamos las longitudes de las reglas Golomb, se puede verificar que la traslación es una solución más óptima que la regla Golomb inicial ya que la longitud de la traslación es menor que la longitud de la regla Golomb inicial:

$$A = \{1, 10, 29, 31, 35, 36, \mathbf{46}\};\ l = 46$$

$$A_f = \{\ 1, 2, 5, 11, 31, 36, \mathbf{38}\ \};\ l_f = 38$$

$$l_f \leq l$$
$$38 \leq 46$$

En el capítulo 4 se discutirá de una manera más detallada el tema de traslaciones y también se hablara sobre cómo serán integradas en la implementación del algoritmo memético con el fin de mejorar las soluciones o reglas Golomb encontradas, cabe resaltar, que se tomara como referencia la propiedad de traslaciones definidas por Carlos Martos en [54, 55] para la implementación del algoritmo memético ya que en dichos proyectos de investigación, se aborda la propiedad de manera detallada planteando una definición y explicación clara sobre cómo aplicarlas en el contexto del problema de reglas Golomb.

3.4. Cotas.

Son una propiedad matemática en el contexto de reglas Golomb que sirve para determinar en qué rango de números enteros se encuentra la longitud óptima de un número determinado de marcas, es decir, las cotas representan los limites en donde se localiza la solución óptima de una regla Golomb en particular [3]. Estas cotas serán utilizadas en la implementación del algoritmo memético para reducir el campo de búsqueda de las soluciones óptimas.

3.4.1. Cota inferior.

La cota inferior [3] define el límite menor en el que debe existir la longitud óptima de una regla Golomb en particular, esto quiere decir que, la longitud óptima debe tener un valor de por lo menos:

$$\boldsymbol{l} > n^2 - 2n\sqrt{n} + \sqrt{n} - 2$$
$$vf \pm$$

Por ejemplo, si ***n*** es igual a 5, entonces la cota inferior estará definida como:

$$\boldsymbol{l} > 5^2 - 2(5)\sqrt{5} + \sqrt{5} - 2v\ \mathbf{2.8754}$$

Esto quiere decir, que la longitud de la regla Golomb para 5 marcas debe ser superior a ***2.8753.***

3.4.2. Cota superior.

La cota superior [3] define el límite mayor en el que debe existir la longitud óptima de una regla Golomb en particular, esto quiere decir que, la longitud óptima debe tener una longitud que no supere el siguiente valor:

$$\boldsymbol{l} < n^2$$

Por ejemplo, si ***n*** es igual a 5, entonces la cota superior estará definida como:

$$\boldsymbol{l} < 5^2$$

Esto quiere decir, que la longitud de la regla Golomb para 5 marcas debe ser inferior a ***25.*** Por otro lado, retomando el ejemplo de la cota inferior, la longitud óptima de la regla Golomb para 5 marcas tendrá un valor entre ***2.8753*** y ***25*** [Figura 20]***.*** Esto se puede verificar con el valor de la longitud de la regla Golomb óptima conocida para 5 marcas (Ver anexo 2).

$$G(5) = \{\ 0, 1, 4, 9, 11\ \}$$

$$l = 11$$

Figura 20. División entera número no primo.

Fuente: Elaboración propia.

La regla Golomb evidenciada en el anexo 2 muestra que la longitud óptima para 5 marcas es ***11*** demostrando el ejemplo anterior, ya que la longitud óptima se encuentre entre ***2.8753*** y ***25.*** De esta manera, se logra concluir la definición de la cota inferior y cota superior, la longitud de una regla Golomb óptima siempre estará definida en el siguiente rango:

$$n^2 - 2n\sqrt{n} + \sqrt{n} - 2 < \boldsymbol{l} < n^2$$

$$2.8753\ < \boldsymbol{l} < 25$$

Capítulo 4. Definición del algoritmo memético.

En este capítulo se presentará la definición del algoritmo memético integrando los componentes matemáticos presentados en el capítulo anterior tales como construcciones algebraicas, traslaciones y cotas. Este tipo de algoritmos forma parte de la familia de las metaheurísticas y son ideales para resolver problemas de optimización complejos ya que logran obtener buenos resultados en poco tiempo de máquina [25, 26, 27]. Por otro lado, este tipo de algoritmos se caracterizan por obtener buenos resultados debido a que conocen el contexto e información del problema y aprovechan ese conocimiento [72] para potenciar el espacio de búsqueda de las soluciones, en nuestro caso, la información del problema hace referencia a la naturaleza matemática del problema de investigación [21, 22, 23, 24] y será utilizada para mejorar los resultados de la propuesta. Adicionalmente, se plantea una metaheurística nueva en la literatura ya que según el teorema No-Free Lunch [51] no se puede asegurar cual es la mejor metaheurística existente para resolver un problema de optimización especifico hasta que todas las metaheurística hayan sido estudiadas y comparadas experimentalmente.

4.1. Lenguaje de programación.

Para la elección del lenguaje de programación utilizado en el desarrollo de la propuesta de investigación inicialmente se analizaron los lenguajes de programación utilizados en el estado del arte [Tabla 21, Figura 11], en donde se logra evidenciar que Matlab y C++ son los lenguajes de programación más relevantes en la literatura, sin embargo, para el desarrollo del algoritmo memético se utilizó el lenguaje de programación Python debido a que en la actualidad Python es el lenguaje de programación recomendado para desarrollar proyectos de inteligencia artificial dado sus buenos resultados en el área [65] por otro lado, Python también ofrece una gran variedad de librerías de código abierto que se pueden conectar con él y potenciar su funcionamiento, esto hace al lenguaje mucho más potente ya que permite programar los procesos del algoritmo memético integrando los conceptos matemáticos mencionados en el capítulo 3 con la ayuda de librerías o frameworks como DEAP [66], Galois [67], SymPy [68], entre otras. Adicionalmente, al ser librería de código abierto [69] nos permite reutilizar el código fuente y adaptarlo según nuestras necesidades de investigación [70, 71]. Cabe resaltar, que la razón principal para la elección de Python fue el conocimiento y la experiencia con la que ya contaba el equipo de desarrollo en este lenguaje de programación, así como experiencia en el uso del framework DEAP para implementar algoritmos evolutivos, esto ofreció una gran ventaja a Python sobre los demás lenguajes de programación disponibles ya que el equipo pudo reducir tiempos en la curva de aprendizaje y se centró en la implementación de la propuesta.

4.2. Entorno de desarrollo.

En esta sección se presentarán las características técnicas generales del proyecto relacionadas al entorno de desarrollo que se utilizaron para implementar la propuesta de investigación.

4.2.1. Características generales del equipo.

El equipo o computador donde se realizó los experimentos fue en un Dell con un procesador 11th Gen Intel(R) Core(TM) i7-1165G7 con 2.80GHz. También cuenta con una memoria RAM de 16GB y un disco de estado sólido de 500GB. Adicionalmente, el sistema operativo es Windows 10 Pro de 64 bits.

4.2.2. Librerías y framework.

- **DEAP** (Distributed Evolutionary Algorithms in Python) [66]: Es un framework escrito en Python diseñado específicamente para implementar algoritmos evolutivos. Su propósito principal es facilitar la creación, ejecución y análisis de algoritmos evolutivos distribuidos. La biblioteca DEAP proporciona una amplia gama de herramientas y componentes esenciales para la implementación eficiente de algoritmos evolutivos, incluyendo la representación de individuos, operadores evolutivos y herramientas estadísticas. Gracias al framework DEAP, se pueden abordar problemas de optimización complejos mediante la implementación de algoritmos evolutivos ya que es una herramienta robusta y bien documentada.
- **Galois [67]:** Librería en Python que extiende las funciones de Numpy para operar con campos finitos o campos de Galois en Python, un tema importante en el área de teoría de números. Esta biblioteca proporciona funcionalidades para operaciones aritméticas en campos finitos, como adición, multiplicación, inversión, entre otras.
- **SymPy [68]:** SymPy es una librería de para implementar matemáticas simbólicas en Python. Su objetivo principal es convertirse en un sistema de álgebra informática con todas las funciones manteniendo el código lo más simple posible para que sea comprensible y fácilmente extensible. SymPy está escrito completamente en Python.

4.2.3. Limitaciones.

Como se ha mencionado anteriormente, el problema de investigación abordado es un considerado un problema de optimización difícil de resolver [15, 16, 17], esto nos lleva a tener una serie de limitaciones para resolverlo y que se tuvieron en cuenta al momento de realizar la investigación.

- El computador utilizado y mencionado anteriormente puede tener recursos limitados en términos de potencia de procesamiento, memoria RAM y capacidad de almacenamiento. Esto podría limitar la escala de los experimentos o la complejidad de reglas Golomb que puede abordar.
- Los recursos limitados del computador podrían afectar la capacidad de realizar experimentos a gran escala o la exploración exhaustiva de espacios de búsqueda extensos.
- Existe cierta dependencia con el framework DEAP ya que se debe utilizar las funciones que el proporciona y en algunos casos implementar el código fuente de nuestra propuesta con una estructura específica para que se adapte al framework, esto puede aumentar el tiempo de desarrollo y pruebas.
- Se pueden tener problemas de almacenamiento ya que los experimentos de larga duración pueden generar grandes cantidades de datos. La capacidad limitada de almacenamiento en un computador puede ser una restricción.

4.3. Características generales de la implementación.

En esta sección se definirá la arquitectura del algoritmo memético que integrará los conceptos matemáticos para encontrar reglas Golomb cercanamente óptimas. Por otro, con esta integración se busca mejorar los resultados del algoritmo memético implementado las estrategias de intensificación y diversificación de los algoritmos evolutivos [73] con el objetivo de equilibrar la exploración del espacio de búsqueda en busca de soluciones óptimas y la explotación de áreas locales para mejorar las soluciones existentes. Estas estrategias son esenciales para el éxito de los algoritmos meméticos en la solución de problemas de optimización complejos.

4.3.1. Introducción.

En la figura 21 se puede evidenciar el diagrama de flujo del algoritmo memético integrando los diferentes conceptos matemáticos y aplicando las estrategias de intensificación y diversificación [72]. El algoritmo memético estará compuesto de los siguientes pasos y será explicado en detalle en las siguientes secciones:

1. **Generación de la población inicial:** En este paso se creará un conjunto inicial de soluciones candidatas que representan posibles soluciones al problema de investigación, en este caso en particular, las posibles soluciones representaran reglas Golomb candidatas a ser reglas óptimas o cercanamente óptimas.
2. **Función de evaluación:** También conocida como función objetivo, es un componente fundamental en los algoritmos meméticos. Esta función se encarga de cuantificar que tan buena es una solución candidata en términos del objetivo de optimización del problema. Por ejemplo, en un problema de minimización, donde el objetivo es reducir cierta medida de rendimiento como costos, errores o distancias, la función de evaluación asignaría valores más bajos a las soluciones que minimizan esta medida, en nuestro caso, la función de evaluación asignará valores más bajos a las reglas Golomb encontradas que tengan una longitud corta. El objetivo es encontrar soluciones que sean óptimas o satisfactorias, minimizando la función de evaluación. La función de evaluación orienta el proceso evolutivo, permitiendo que los algoritmos evolutivos identifiquen y seleccionen las soluciones más prometedoras para la generación de nuevas soluciones en cada iteración, buscando continuamente mejorar y converger hacia soluciones óptimas o satisfactorias.
3. **Operador de selección:** El componente de selección en los algoritmos meméticos resulta crucial para filtrar y dar prioridad a las soluciones candidatas con un gran potencial de mejora en las próximas iteraciones. Este componente se encarga de determinar qué soluciones candidatas serán elegidas en cada iteración para que puedan ser mejoradas en los siguientes pasos, en nuestro contexto, las soluciones candidatas hacen referencias a las reglas Golomb y serán mejoradas al reducir sus respectivas longitudes. En última instancia, el operador de selección juega un papel crucial en la mejora progresiva de las soluciones a lo largo de las iteraciones del algoritmo.
4. **Operador de cruce:** Este operador se encarga de combinar la información de dos soluciones candidatas de la población actual, también conocidas como soluciones padres, para generar una nueva solución candidata, esta solución se conoce cómo conoce como descendencia. El objetivo principal del operador de cruce es explorar

el espacio de búsqueda en busca de soluciones potencialmente mejores al combinar características prometedoras de los padres. Es aquí, donde se aplica el concepto de diversificación al explorar el espacio de búsqueda en busca de mejores soluciones.

5. **Operador de mutación:** Este operador se encarga de introducir variabilidad en las soluciones candidatas al alterar aleatoriamente parte de su información. A través de la mutación, se aplica la estrategia de intensificación explorando nuevas regiones del espacio de búsqueda que de otra manera no serían alcanzadas por el operador de cruce.
6. **Regeneración de la población:** En este paso se evalúa la población actual para determinar el porcentaje de reglas Golomb dentro de la población. Posteriormente, se compara el porcentaje con un umbral previamente definido para determinar si se debe re generar la población y así reemplazar los peores individuos de la población por reglas Golomb generadas a partir de las construcciones algebraicas. Este paso se relaciona con la estrategia de diversificación ya que pretende buscar mejores soluciones en el espacio de búsqueda global.
7. **Traslaciones:** Concepto matemático presentado en el capítulo anterior que será utilizado para mejorar las soluciones candidatas en cada iteración. Las traslaciones están directamente relacionadas con la estrategia de Intensificación de los algoritmos meméticos.
8. **Función de evaluación:** Se aplica nuevamente la función de evaluación mencionada en el paso 2 a toda la población actual con el objetivo de evaluar la calidad de las soluciones candidatas después de haber aplicado los diferentes operadores evolutivos y procesos de mejora mencionados en los pasos anteriores.
9. **Condición de parada:** Determina en que momento la ejecución del algoritmo memético debe llegar a su fin y estará determinada por un parámetro de entrada llamado “Número de iteraciones”, en caso de no haber llegado al número de iteraciones deseado, el algoritmo volverá a ejecutar todos sus procesos a partir del paso 2 con la población actual modificada y mejorada, es aquí donde se busca mejorar la población en cada una de las iteraciones y eventualmente mejorar el resultado final.
10. **Escoger el mejor individuo:** El paso final se encargará de escoger la mejor o mejores soluciones encontradas al finalizar el número de iteraciones. En este paso es donde el algoritmo memético muestra el resultado final y las reglas Golomb óptimas o cercanamente óptimas encontradas.

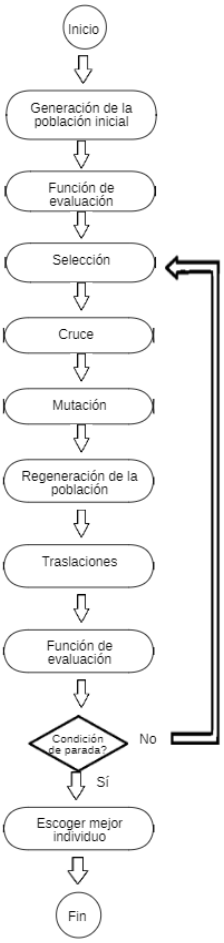


Figura 21. Diagrama de flujo del algoritmo memético

Fuente: Elaboración propia.

4.3.2. Representación del individuo.

Inicialmente, para abordar el problema de investigación, se debe definir la estructura de las soluciones candidatas al problema de investigación obtenidas por el algoritmo memético. Esta estructura se conoce como representación del individuo, en donde el individuo, también conocido como agente en algoritmos meméticos, estará representado por una lista de números enteros y actuará como una solución candidata al problema, en nuestro caso, el individuo hará referencia a una regla Golomb cercanamente óptima encontrada por el algoritmo memético. La representación del individuo es un componente crítico en la efectividad y rendimiento global del algoritmo, ya que afecta directamente la capacidad del algoritmo para explorar el espacio de búsqueda y encontrar soluciones óptimas o cercanamente óptimas al problema de investigación. En la figura 22 se puede evidenciar la representación del individuo para una regla Golomb de 7 marcas:

Figura 22. Representación del individuo.

Fuente: Elaboración propia.

4.3.3. Función de evaluación.

Como se mencionó en la sección 4.3.1, la función de evaluación se encarga de cuantificar que tan buena es una solución candidata en términos de objetivo de optimización del problema. En nuestro contexto en particular, la función evaluará cada una de las reglas Golomb representadas como individuos o agentes de la población dando como resultado un valor numérico que representará la calidad de cada una de las reglas encontradas. Por otro lado, al ser un problema de minimización, en donde se busca reducir la longitud de las reglas Golomb para encontrar resultados cercanamente óptimos, la función se encargará de penalizar con valores positivos los agentes que cumplan con un determinado de número de restricciones, es decir, si el agente cumple con una regla negativa en particular, la función le asignará un valor numérico positivo y lo alejará de nuestro objetivo de optimización. Al finalizar la evaluación de todas las restricciones presentadas en el siguiente ejemplo en cada una de las soluciones candidatas obtendremos un valor numérico positivo que será utilizado por el algoritmo memético para identificar las soluciones con mejor calidad. El conjunto de restricciones se puede evidenciar en el siguiente ejemplo y serán utilizados para medir la calidad de cada uno de los agentes de manera secuencial, por ejemplo, la función evaluará la primera restricción y dará como resultado una penalización inicial, posteriormente, evaluará las siguientes restricciones de manera secuencial e ira sumando el resultado de cada de ellas, así mismo, si un agente no cumple con ninguna restricción, no debería tener ninguna penalización positiva y estará más cerca de nuestro objetivo de optimización.

Restricciones:

1. **Número de diferencias:** esta restricción busca penalizar a los agentes o reglas que cuenten con un número determinado de diferencias, es decir, se busca penalizar agentes que no sean reglas Golomb y penalizar más a medida que la regla aumente su número de diferencias, este tipo de reglas se les conoce como reglas g-Golomb y son de gran utilidad en la literatura [54, 55]. La fórmula de la restricción es la siguiente:

$$\boldsymbol{p = l + ND}$$

   En donde:

   - $\boldsymbol{p}$**:** Hace referencia a la variable que almacenará el valor de la penalización.
   - $\boldsymbol{l}$**:** Longitud de la regla o agente.
   - $\boldsymbol{ND}$**:** Representa el número de diferencias que tiene el agente evaluado.

   Cabe aclarar, que esta penalización también tendrá en cuenta la longitud de la regla para realizar el cálculo debido a que nuestro objetivo de optimización se centra en minimizar la longitud de las reglas. Por ejemplo, se define un agente o regla g-Golomb de 5 marcas y se calcula su respectivo triángulo de diferencias [Figura 23]:

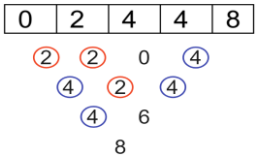


Figura 23. Aplicación de la primera restricción.

Fuente: Elaboración propia.

Se puede evidenciar en la figura anterior que la diferencia con valor 2 se repite 3 veces y la diferencia con valor 4 se repite 4 veces, esto quiere decir, que el $\boldsymbol{ND}$ estará definido por la suma de dichos valores. Aplicando la formula, tendríamos la siguiente penalización:

$$p = 9 + (3 + 4) = 9 + 7 = \mathbf{16}$$

El agente en este momento cuenta con una penalización de $\mathbf{16}$ y será utilizada en el ejemplo de la siguiente restricción.

2. **¿Es regla Golomb?:** Esta restricción se centra en penalizar de manera fuerte las soluciones candidatas que no sean reglas Golomb. Adicionalmente, la fórmula de esta restricción dependerá del valor de una variable $\boldsymbol{k}$ que nos servirá para penalizar a medida que aumente el número de marcas que queremos encontrar, esta parte en la fórmula es necesaria debido a que el problema aumenta exponencialmente a medida que aumenta el número de marcas [18] y la restricción debe tener más peso para esos casos. De esta manera, si la regla o agente evaluado no es una regla Golomb, se le aplicará la siguiente fórmula:

$$\boldsymbol{p} = \boldsymbol{p} + (\boldsymbol{n} * \boldsymbol{k})$$
$$k = 0.45 \ \rightarrow \ n < 8$$
$$k = 0.8 \ \rightarrow \ n \geq 8$$

En donde:

- $\boldsymbol{p}$**:** Hace referencia a la variable que almacena el valor de la penalización de la restricción anterior.
- $\boldsymbol{n}$**:** Número de marcas de la regla o agente.
- $\boldsymbol{k}$**:** Variable de penalización que tendrá el valor de 0.45 para reglas o agentes con un $\boldsymbol{n}$ inferior a 8 y tendrá el valor de 0.8 para reglas con un $\boldsymbol{n}$ superior o igual a 8.

Continuando con la regla del ejemplo anterior, se puede evidenciar que la regla no es Golomb debido a que cuenta con un número determinado de diferencias, es por esto que se debe penalizar con la formula presentada en esta restricción.

$$p = 16 + (5 * 0.45) = 16 + 2.25 = \mathbf{18.25}$$

El agente en este momento cuenta con una penalización de $\mathbf{18.25}$ y será utilizada en el ejemplo de la siguiente restricción.

3. **Elementos repetidos:** Esta restricción se encarga de penalizar de manera fuerte los agentes que cuenten con elementos repetidos, entre mayor sea el número de repeticiones, mayor será la penalización. De esta manera, si la regla o agente evaluado cuenta con elementos repetidos, se le aplicará la siguiente fórmula:

$$\boldsymbol{p = p + NR * n}$$

En donde:

- $\boldsymbol{p}$**:** Hace referencia a la variable que almacena el valor de la penalización de la restricción anterior.
- $\boldsymbol{NR}$**:** Representa el número de repeticiones que tiene la regla o agente.

Continuando con la regla del ejemplo anterior, obtenemos la figura 24:

Figura 24. Aplicación de la tercera restricción.

Fuente: Elaboración propia.

Como se puede evidencia en la figura anterior, el elemento con valor 5 de la regla evaluada cuenta con 2 repeticiones, esto quiere decir, que el $\boldsymbol{NR}$ tendrá el valor de 2 y se aplicará la formula presentada en esta restricción de la siguiente manera:

$$p = 18.25 + (2 * 5) = 18.25 + 10 = \mathbf{28.25}$$

Finalmente, obtenemos el valor final de la penalización después de aplicar todas las restricciones y esta hará referencia a la calidad del agente que estemos evaluando, para este ejemplo, la penalización final del agente **{ 1, 3 , 5 , 5 , 9 }** será igual a **28.25.**

4.3.4. Generación de la población inicial.

Como se mencionó en la sección 4.3.1, la generación de la población inicial se encarga de crear el conjunto inicial de individuos o agentes que representarán posibles soluciones al problema de investigación, es decir, reglas Golomb cercanamente óptimas. Este conjunto de soluciones candidatas será mejorado por el algoritmo memético de manera iterativa aplicando los diferentes operadores y procesos explicados en las siguientes subsecciones.

Por otro lado, la generación de la población inicial estará compuesta por dos subprocesos, generación de reglas Golomb y generación de reglas g-Golomb para aplicar la estrategia de diversificación de los algoritmos meméticos. En el primer subproceso se busca aplicar el conocimiento que se tiene en el problema de investigación para mejorar los resultados del algoritmo memético, la idea principal de este subproceso es generar reglas Golomb a partir de las construcciones algebraicas presentadas en la sección 3.2 y de esta forma, integrar los primeros conceptos matemáticos que nos ayudaran a abordar el problema desde su raíz matemática [1, 22, 23, 24]. Por otro lado, el segundo subproceso se encargará de generar reglas g-Golomb [55] a partir de un algoritmo de búsqueda local que será explicado a continuación.

Proceso de la generación:

Inicialmente, la generación recibe como parámetro de entrada el número de marcas de la regla Golomb que se desea encontrar y una variable llamada “GolombPercentage” que será utilizada por el algoritmo memético para definir el número de agentes de la población inicial

que serán reglas Golomb y reglas g-Golomb. Los valores de la variable GolombPercentage tendrá valores numéricos enteros entre 1 y 100 y definirá el porcentaje de agentes que serán reglas Golomb de la siguiente manera:

a. Se define el tamaño de la población con el número de agentes que tendrá la población a lo largo de la ejecución del algoritmo memético con la ayuda del número de marcas que se quiere encontrar $\boldsymbol{n}$ de la siguiente forma:

$$pop = n * 5$$

Por ejemplo, si se desea buscar la regla Golomb cercanamente óptima para 5 marcas, entonces el tamaño de la población se definirá como:

$$pop = 5 * 5 = \mathbf{25}$$

b. Se define el número de reglas Golomb que formarán parte de la población inicial a partir de la variable GolombPercentage y del tamaño de la población $\boldsymbol{pop}$ calculado anteriormente de la siguiente forma:

$$GR = pop * (\frac{GolombPercentage}{100})$$

Cabe aclarar, que se solamente se tendrá en cuenta el valor entero del resultado anterior para de trabajar de manera más fácil con las proporciones de los agentes de la población. Posteriormente, se calcula el número de reglas g-Golomb realizando la siguiente diferencia:

$$gGR = pop - GR$$

c. Se generan las reglas Golomb utilizando el GR y las construcciones algebraicas Bose, Singer y Rusza de la siguiente manera [Figura 25]:

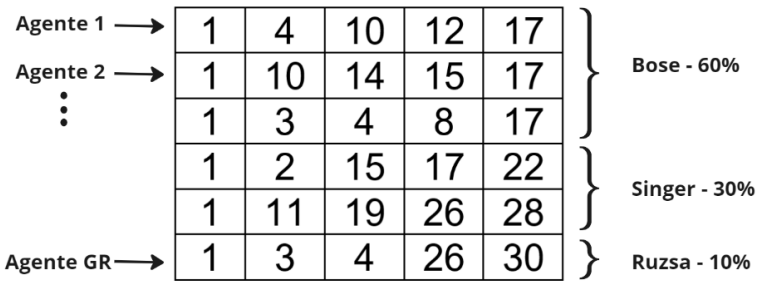


Figura 25. Generación de la población inicial.

Fuente: Elaboración propia.

Se puede evidenciar en la figura 25 que las reglas Golomb serán generadas proporcionalmente por cada una de las construcciones, es decir, el 60% de las reglas serán generadas por la construcción Bose, el 30% por la construcción Singer y el 10% por la construcción Ruzsa. Esto es debido a que, según la literatura [3, 54, 55] la construcción Bose en la mayoría de los casos obtiene mejores resultados que las demás construcciones, seguido por la construcción Singer que obtiene mejores resultados que Ruzsa.

d. Finalmente, se generan las reglas g-Golomb para completar la generación de la población inicial a partir de un algoritmo de búsqueda local. Este algoritmo será

presentado en el pseudocódigo de la figura 26 y estará compuesto por los siguientes procesos:

- **a.** Inicialmente, se genera una regla Golomb utilizando la construcción algebraica Bose y nos servirá como regla base para generar las reglas g-Golomb, es decir, a partir de la regla generada por Bose, se aplicará una búsqueda local para generar los agentes que hacen falta para completar la población inicial.
- **b.** Se crea un ciclo de $\boldsymbol{gGR}$ iteraciones para generar las reglas g-Golomb a partir de la búsqueda local y de la regla Golomb base. De manera iterativa, se creará cada una de las reglas g-Golomb teniendo como referencia la regla Golomb generada en el paso anterior.
- **c.** Se define un rango de números enteros que será utilizado posteriormente para definir el valor aleatorio que modificará cada marca de la regla Golomb base que se tiene como referencia y así obtener como resultado la nueva regla g-Golomb.
- **d.** Este paso es crucial para el buen funcionamiento del algoritmo memético ya que es aquí donde se integra el concepto matemático presentado en la sección 3.4 relacionada a las cotas de las reglas Golomb. Esta propiedad sirve para determinar en qué rango de números enteros se encuentra la longitud óptima de un número determinado de marcas y será utilizada por el algoritmo de búsqueda local para modificar la última marca de cada regla g-Golomb. Esto será de gran importancia para el algoritmo ya que le ayudará a reducir el espacio de búsqueda en donde se encuentran las soluciones óptimas y cercanamente óptimas.
- **e.** Se crea un ciclo hasta ***"n"*** iteraciones, donde ***"n"*** representa el número de marcas que queremos encontrar, para modificar aleatoriamente cada una de las marcas de la regla g-Golomb que se busca generar a partir del algoritmo de búsqueda local.
- **f.** Se define un condicional que nos ayudará a modificar únicamente el 80% de las marcas de la regla Golomb base.
- **g.** Se crea un condicional para modificar la marca de la regla Golomb base, en el caso de que la marca sea la última de la regla, se generará un número entero aleatorio entre el rango definido por las cotas matemáticas y reemplazará el valor de la marca que estemos modificando, y en el caso contrario, se generará un número entero aleatorio entre el rango definido en el paso **c** y será utilizado para reemplazar el valor de la marca que estemos modificando. Finalmente, al terminar todos los pasos mencionados anteriormente, obtendremos las reglas g-Golomb que formarán parte de la población inicial.

```
localSearchG
Entrada: una regla Golomb rule
Salida: nueva regla g-Golomb

sea size<-Longitud(rule)
sea low<-size
sea up<-size+2
sea topLimit<- getTopLimit(size)
sea lowerLimit<- getLowerLimit(size)
Para i<-0 Hasta size
    Si NumeroAleatorio(1,100) <= 80 Entonces
        Si i == (size-1) Entonces
            sea rule[i]<- NumeroAleatorio(lowerLimit, topLimit)
        Sino
            sea rule[i]<- NumeroAleatorio(low, up)
        Fin Si
    Fin si
Fin Para
Ordenar(rule)
Devolver (rule)


generateNgGolombRules
Entrada: número de reglas g-Golomb que se desean encontrar gGR y el
número de marcas que se está encontrando n
Salida: reglas g-Golomb gGRules

sea gGRules<-[gGR]
sea baseRule<- generateBoseRule(n)
sea  i<-0
Mientras i < gGR Hacer
    sea gGRules[i]<- localSearchG(baseRule)
    sea i<-i+1
FinMientras
Devolver(gGRules)
```

Figura 26. Pseudocódigo del algoritmo de búsqueda local.

Fuente: Elaboración propia.

4.3.5. Operadores meméticos.

En esta sección, se mostrará la forma en la que fueron implementados los operadores meméticos presentados en la sección 4.3.1 en el contexto de nuestra propuesta de investigación.

4.3.5.1. Operador de selección.

Para seleccionar los agentes candidatos que participaran en los siguientes operadores se utilizar un operador de selección existente con el objetivo de reducir tiempos de implementación en funcionalidades que ya existen y concentrarse en desarrollar la propuesta de investigación. Por otro lado, al utilizar un operador de selección existente, ayudaremos a la calidad del código debido a que estos operadores normalmente ya se encuentran testeados y depurados en la literatura. De esta forma, se utiliza el operador de selección **Stochastic Universal Sampling (SUS)**, un método utilizado en algoritmos meméticos para seleccionar agentes de una población basándose en sus valores de fitness (función de evaluación) de una manera uniforme pero espaciada a lo largo de una línea en el espacio de búsqueda. Este método promueve una exploración más equilibrada del espacio de búsqueda y ayuda a evitar la convergencia prematura hacia óptimos locales, ya que todos los individuos tienen la oportunidad de ser seleccionados independientemente de su posición en la escala de fitness. Para revisar en detalle Adicionalmente, para entender el operador de selección SUS con mayor detalle, revisar las definiciones presentadas por el framework DEAP [74, 75] en donde se describe de una manera más formal el tema mencionado. Cruzarse

4.3.5.2. Operador de cruce.

Este operador tomará dos agentes de la población actual seleccionada en el operador de selección y realizará una serie de pasos para generar un nuevo agente con características de los dos agentes seleccionados. El conjunto de pasos y pseudocódigo se puede evidenciar en la figura 27:

a. Se crea un ciclo de 50 iteraciones para crear el nuevo agente a partir de los agentes padres. Adicionalmente, se cuenta con este número de iteraciones para potenciar el espacio de búsqueda del algoritmo memético para encontrar dicho agente nuevo con las mejoras características posibles.
b. Se llama al subproceso **cxUniformGetGolomb** para generar el nuevo agente a partir de los agentes padres.
c. Se genera un número decimal aleatorio entre 0.4 y 0.8 y servirá para determinar el porcentaje de las marcas de los agentes padres que serán seleccionados para realizar el cruce y así generar el nuevo agente. Esto quiere decir que, solamente entre el 40% y 80% de las marcas participarán en el operador de cruce.
d. Se copia el agente padre 1 en la variable **agent3** y luego se crea un ciclo de **size** iteraciones, en donde size hace referencia al número de marcas que estamos encontrando, para generar cada una de las marcas del nuevo agente **agent3**.
e. Se define un condicional que nos ayudará a seleccionar las marcas de los agentes padres que formarán parte del posible cruce teniendo como referencia el porcentaje mencionado en el paso **c** y posteriormente estas marcas serán utilizadas para crear la marca del nuevo agente.
f. Se define la marca que formará parte del nuevo agente a partir de una serie de condiciones que serán mencionadas seguidamente. Se puede evidenciar que todas las marcas de los agentes padres participarán en el cruce a excepción de la primera y última marca. También, se valida si la marca **i+1** del agente1 es mayor que la marca **i** del agente2 para asegurar que al realizar el cruce, el nuevo agente conserve el orden en sus elementos. Finalmente, se cruzan las marcas de los agentes padres para generar la marca del agente nuevo. Al finalizar el ciclo mencionado en el paso **d,** obtendremos el nuevo agente cada una de sus marcas.
g. Se valida el agente nuevo generado en los pasos anteriores con la función **EsReglaGolomb** para determinar si el agente es una regla Golomb. En caso de que sea una regla Golomb, se terminará el ciclo del paso **a** y en caso contrario, seguirá buscando un nuevo agente hasta que el ciclo del paso **a** termine su número de iteraciones.
h. Finalmente, se valida con la función **EsReglaGolomb** el agente final generado después de terminar el número de iteraciones del ciclo del paso **a.** En caso de que no sea una regla Golomb, se retornará como resultado los agentes padres, ya que, en este caso el nuevo agente no contará con la calidad suficiente para ingresar en la población actual.

```
cxUniformGetGolomb
Entrada: primer agente padre que participará en el cruce agent1 y
segundo agente padre que participará en el cruce agent2.
Salida: hijos provenientes del cruce

sea indpb<- NumeroDecimalAleatorio(0.4, 0.8)
sea size<- Longitud(agent1)
sea agent3<- agent1
Para i<-0 Hasta size
    Si NumeroDecimalAleatorio(0,1) <= indpb Entonces:
        Si (i>0 && i<size-1) && (agent1[i+1] > agent2[i]) Entonces
            sea agent3[i]<- agent2[i]
        Fin Si
    Fin Si
Fin Para
Devolver([agent3, agent3])


cxUniformGolomb
Entrada: primer agente padre que participará en el cruce agent1 y
segundo agente padre que participará en el cruce agent2.
Salida: hijos provenientes del cruce

sea agent<-[]
sea i<-0
Mientras i < 50 Hacer:
    sea agent<- cxUniformGetGolomb(agent1, agent2)
    Si EsReglaGolomb(agent)==true Entonces
        RomperCiclo
    Fin Si
        sea i<-i+1
FinMientras

Si EsReglaGolomb(agent)==False Entonces
    sea agent<- [agent1, agent2]
Fin Si
Devolver(agent)
```

Figura 27. Pseudocódigo del operador de cruce.

Fuente: Elaboración propia.

4.3.5.3. Operador de mutación.

Este operador tomará el agente generado en el cruce anterior y realizará una serie de pasos para generar un nuevo agente con una serie de mutaciones. El conjunto de pasos y pseudocódigo se puede evidenciar en la figura 28:

- **a.** Se crea un ciclo de 1000 iteraciones para crear el agente mutado a partir del **agent** generado en el cruce anterior. Adicionalmente, se cuenta con este número de iteraciones para potenciar el espacio de búsqueda del algoritmo memético para encontrar dicho agente mutado con las mejoras características posibles.
- **b.** Se copia el agente del cruce anterior en la variable **mutatedRule**
- **c.** Se obtienen 4 números enteros aleatorios que harán referencia a las posiciones de las marcas dentro del agente que serán mutadas. Cabe aclarar que, al ser posiciones aleatorias el agente puede tener cualquier número de marcas positivas y el operador seguirá funcionando.
- **d.** En este paso, se realiza la mutación de las marcas que se encuentran en las posiciones obtenidas en el paso anterior. La mutación se encargará de sumar o restar aleatoriamente el valor de **1** al valor de la marca original que se esté mutando.
- **e.** Se valida el agente mutado con la función **EsReglaGolomb** para determinar si el agente es una regla Golomb. En caso de que sea una regla Golomb, se terminará

el ciclo del paso **a** y en caso contrario, seguirá buscando un nuevo agente mutado hasta que el ciclo del paso **a** termine su número de iteraciones.

```
mutUniformIntGolomb
Entrada: agente que será mutado agent
Salida: agente mutado mutatedRule

sea size<-Longitud(agent)
sea i<-0
Mientras i <= 1000 Hacer:
    sea mutatedRule<- agent
    sea posI<- NumeroAleatorio(1, size-1)
    sea posJ<- NumeroAleatorio(1, size-1)
    sea posK<- NumeroAleatorio(1, size-1)
    sea posL<- NumeroAleatorio(1, size-1)
    sea mutatedRule[posI]<- mutatedRule[posI] + NumeroAleatorio(-1, 1)
    sea mutatedRule[posJ]<- mutatedRule[posJ] + NumeroAleatorio(-1, 1)
    sea mutatedRule[posK]<- mutatedRule[posK] + NumeroAleatorio(-1, 1)
    sea mutatedRule[posL]<- mutatedRule[posL] + NumeroAleatorio(-1, 1)
    Ordenar(mutatedRule)
    Si EsReglaGolomb(mutatedRule)==true Entonces
        RomperCiclo
    Fin Si
    sea i<-i+1
FinMientras
Devolver(mutatedRule)
```

Figura 28. Pseudocódigo del operador de mutación.

Fuente: Elaboración propia.

4.3.6. Regeneración de la población.

Para iniciar el proceso de la regeneración de la población mencionado en la sección 4.3.1, el algoritmo memético recibe como parámetro de entrada la variable llamada **RegeneratePopPercentage** y será utilizada para definir el umbral (threshold) en el que se debe regenerar la población actual. El proceso de la regeneración se puede evidenciar en la figura 29 y será explicado en detalle a continuación:

1. Se identifica el porcentaje de reglas Golomb que se encuentran en la población actual.
2. Se compara el porcentaje anterior con el threshold para determinar si se debe regenerar la población actual, si el porcentaje es menor que el threshold, se regenerará la población actual reemplazando los agentes que tengan peor evaluación fitness por mejores agentes generados por las construcciones algebraicas Bose, Singer y Ruzsa
3. El número de agentes nuevos estarán determinados por la diferencia entre GolombPercentage y el porcentaje calculado en el paso 1.
4. Al finalizar el reemplazo, el porcentaje de reglas Golomb de la población actual será igual a la variable GolombPercentage y tendrá una población más robusta en términos de mejores soluciones. En esta regeneración se busca aplicar la estrategia de diversificación a lo largo de las iteraciones para buscar nuevos espacios de búsqueda teniendo en cuenta como referencia los resultados mejorados de la población actual.

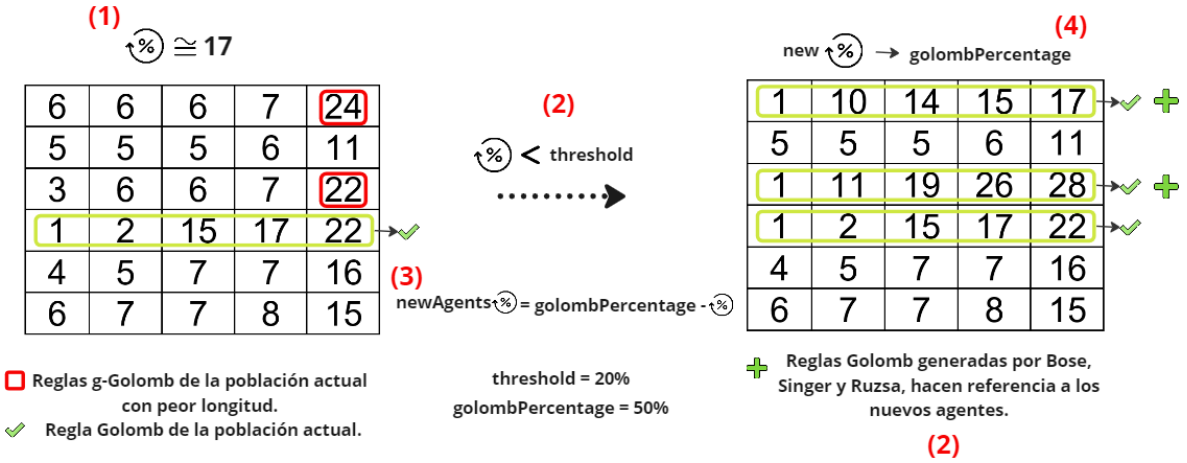


Figura 29. Regeneración de la población.

Fuente: Elaboración propia.

Cabe aclarar, que para el ejemplo de la figura 29 se utilizaron los valores de 20% para el threshold y el 50% para el GolombPercentage como ejemplo.

4.3.7. Traslaciones.

Posteriormente, un determinado número de agentes será mejorado utilizando el concepto matemático de traslaciones presentado en la sección 3.3. Este número será determinado gracias a un parámetro de entrada llamado “TranslationsPercentage” y al igual que GolombPercentage tendrá valores numéricos enteros entre 1 y 100 haciendo referencia al porcentaje de agentes que deben ser mejorados [Figura 30].

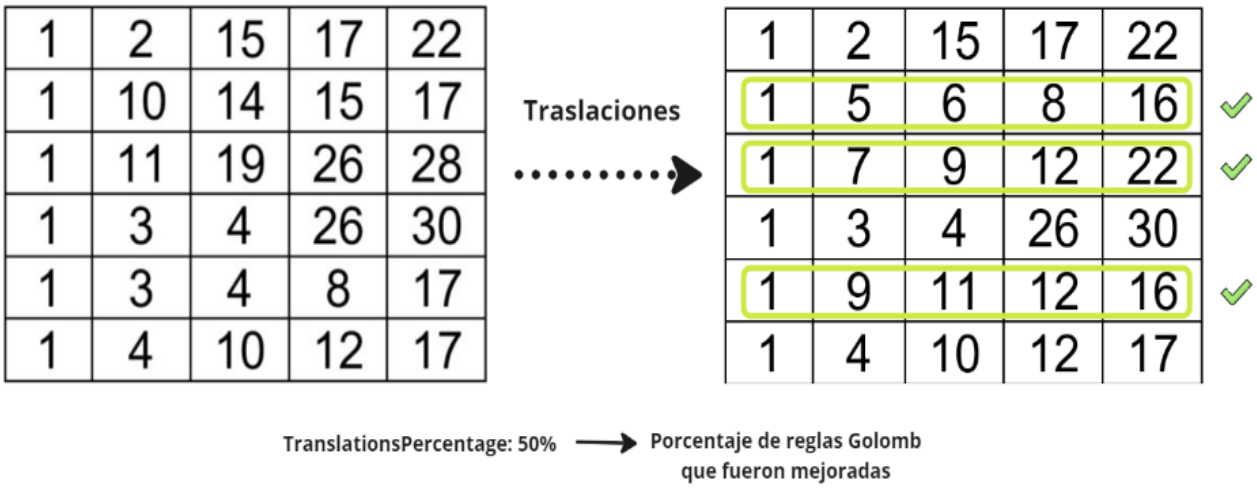


Figura 30. Aplicación de traslaciones.

Fuente: Elaboración propia.

En la figura 30 se puede evidenciar el resultado final de la mejora después de aplicar el concepto de traslaciones en la población actual de ejemplo, las reglas Golomb que pasaron por el proceso de traslaciones lograron reducir sus respectivas longitudes.

4.3.8. Criterios para terminar la ejecución del algoritmo memético.

Como se evidencio en la figura 21, la propuesta de investigación ejecutará los operadores meméticos, la regeneración de la población, las traslaciones y la función de evaluación de manera iterativa para obtener mejores resultados en la solución del problema de investigación. Dicho proceso, se realizará por medio de un ciclo de 1000 iteraciones y le será de utilidad para el algoritmo memético ya que podrá explorar los diferentes espacios de búsqueda que tiene el problema de investigación. Este número de iteraciones es considero como el criterio para terminar la ejecución de la propuesta de investigación, una vez termine el número de iteraciones, el algoritmo memético mostrará como resultado la o las mejores reglas Golomb cercanamente óptimas que pudo encontrar lo largo de todas las iteraciones.

Capítulo 5. Evaluación del algoritmo memético.

En este capítulo se muestra como se realizó la ejecución y evaluación del algoritmo memético propuesto para encontrar reglas Golomb cercanamente óptimas hasta 28 marcas comparando sus resultados con las reglas Golomb óptimas conocidas.

5.1. Simulación de los resultados.

En esta sección se presenta el rendimiento del algoritmo memético propuesto solucionando el problema de investigación [15, 16, 17] encontrando reglas Golomb cercanamente óptimas hasta 28 marcas, así como su comparación con las reglas Golomb óptimas existentes. El algoritmo memético se ejecutó 100 veces para encontrar cada una de las marcas y se validó en el equipo especificado en la sección 4.2.1 para obtener tener los resultados mostrados en la tabla 23.

5.2. Parametrización de la propuesta.

En esta sección se presentarán los diferentes parámetros de entrada que necesita el algoritmo memético para encontrar reglas Golomb cercanamente óptimas [Tabla 22]. Como se mencionó en el capítulo 3, la propuesta de investigación necesita un determinado grupo de variables iniciales para iniciar su ejecución, la primera de ellas se llama **marks**, haciendo referencia al número de marcas que se quiere encontrar y los posibles valores de esta variable serán números enteros mayores que cero**,** la segunda se llama **GolombPercentage** y está relacionada al número de agentes o soluciones candidatas de la población inicial que serán reglas Golomb y reglas g-Golomb. Los valores de la variable GolombPercentage tendrá valores numéricos enteros entre 1 y 100 y definirá el porcentaje de agentes que serán reglas Golomb. La tercera variable se conoce como **TranslationsPercentage** y al igual que GolombPercentage tendrá valores numéricos enteros entre 1 y 100 y representará el porcentaje de agentes que deben ser mejorados mediante el concepto de traslaciones en cada una de las iteraciones del algoritmo memético. La cuarta variable se llama **RegeneratePopPercentage** y será utilizada para definir el umbral (threshold) en el que se debe regenerar la población en cada iteración del algoritmo memético. Esta variable será utilizada en el paso de la propuesta conocido como "Regeneración de la población" y tendrá valores numéricos enteros entre 1 y 100.

Tabla 22. Parametrización del algoritmo memético.

Parámetro de entrada	Posibles valores
Marks	Número entero mayor a cero
GolombPercentage	Números enteros en el rango [1,100]
TranslationsPercentage	Números enteros en el rango [1,100]
RegeneratePopPercentage	Números enteros en el rango [1,100]

Esta sección está relacionada con la fase de identificación y desarrollo evidenciada en la tabla 4 ya que los parámetros de entrada relacionados a porcentajes mostrados en la tabla anterior deben ser ajustados con valores ideales con el objetivo de mejorar los resultados de la propuesta de investigación y dichos valores dependerán directamente del número de marcas que se requiera encontrar.

5.3. Reglas Golomb óptimas y cercanamente óptimas encontradas.

En esta sección se presentará los resultados finales de la propuesta de investigación evidenciando las reglas Golomb óptimas y cercanamente óptimas encontradas hasta 28 marcas. La propuesta fue ejecutada 100 veces con una semilla aleatoria para encontrar cada una de las reglas Golomb evidenciadas en la tabla 23, es decir, para encontrar las reglas Golomb de 5 marcas, el algoritmo fue ejecutado 100 veces y así con todas las respectivas marcas. Por otro lado, las reglas Golomb óptimas mencionadas a continuación se encuentran definidas en el Anexo 2 y el "***n"*** mencionado hace referencia al número de reglas Golomb que estamos evaluando. En las siguientes gráficas se mostrarán los resultados finales de las ejecuciones mencionadas para cada una de las marcas trabajadas evidenciando con un color diferente las reglas Golomb y g-Golomb encontradas por la propuesta de investigación.

Para el caso de ***n=2*** se puede evidenciar que la longitud 2 fue la longitud más encontrada ya que el algoritmo memético logro encontrar 55 reglas Golomb con esa longitud [Figura 31]. También se puede evidenciar que la propuesta logró encontrar la longitud (1) óptima 30 veces.

Regla Golomb óptima para 2 marcas: [0, 1]

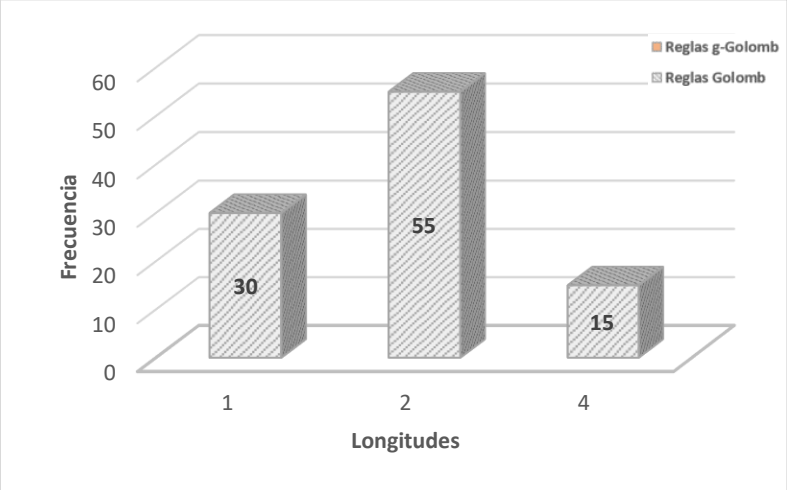


Figura 31. Resultados de la propuesta para 2 marcas.

Fuente: Elaboración propia.

Para el caso de ***n=3*** se puede evidenciar que la longitud óptima fue la longitud más encontrada logrando encontrar hasta 68 reglas Golomb óptimas [Figura 32]. También podemos apreciar que la peor longitud encontrada fue 6 y apareció en 2 ocasiones.

Regla Golomb óptima para 3 marcas: [0, 1, 3]

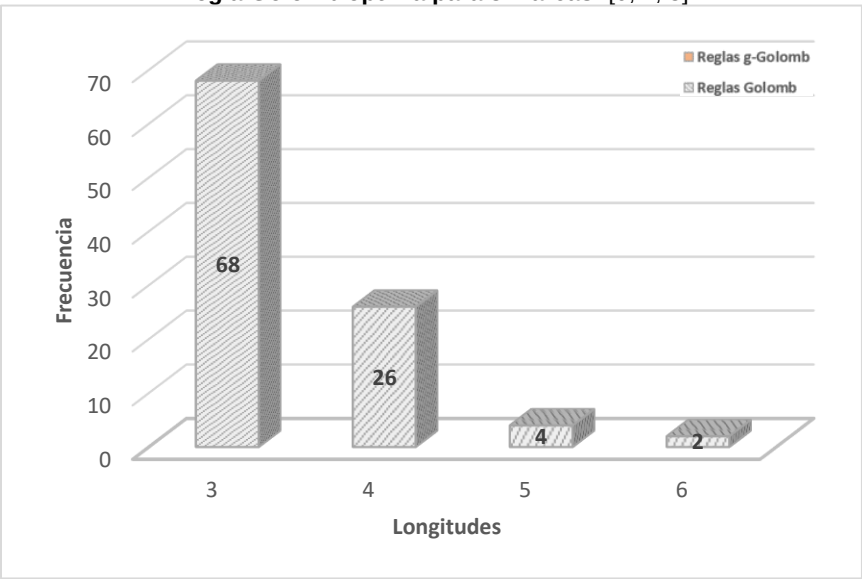


Figura 32. Resultados de la propuesta para 3 marcas.

Fuente: Elaboración propia.

Para el caso de ***n=4*** se puede evidenciar que la longitud 6 fue la longitud que más veces se encontró con 60 repeticiones y es considerada como la regla Golomb óptima para 4 marcas [Figura 33]. Adicionalmente, se puede evidenciar que el algoritmo memético logró encontrar una regla con longitud de 5 menor a la óptima (6), estas reglas son consideradas como reglas g-Golomb y no forman parte del objetivo de investigación principal.

Regla Golomb óptima para 4 marcas: [0, 1, 4, 6] ; [0, 2, 5, 6]

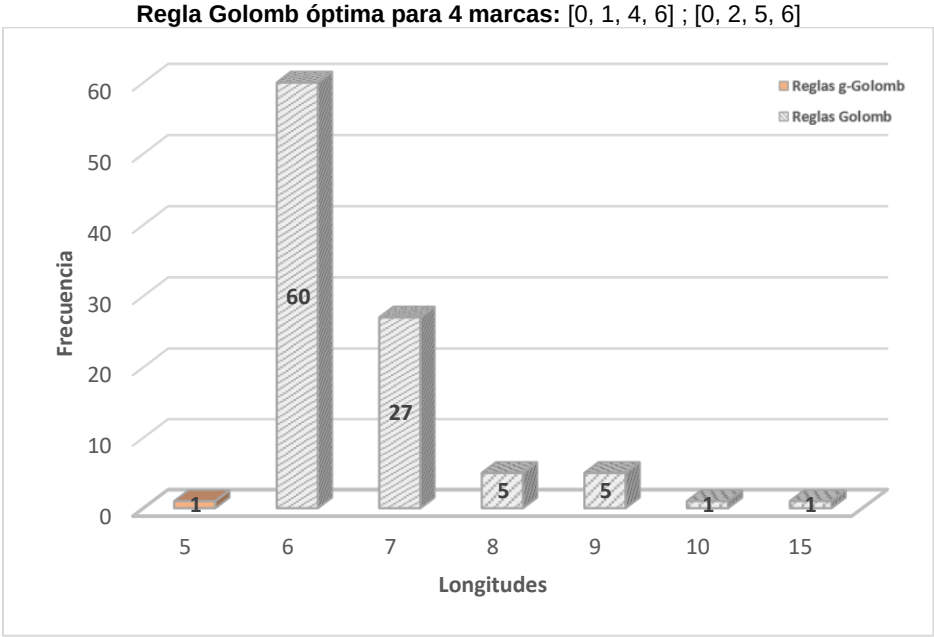


Figura 33. Resultados de la propuesta para 4 marcas.

Fuente: Elaboración propia.

Para el caso de ***n=5*** se puede evidenciar que el algoritmo memético logra encontrar 65 reglas Golomb con la longitud óptima 11 [Figura 34]. También logra encontrar reglas g-Golomb de longitud 8, 9 y 10.

Regla Golomb óptima para 5 marcas: [0, 2, 7, 10, 11] ; [0, 2, 7, 8, 11] ; [0, 3, 4, 9, 11] [0, 1, 4, 9, 11]

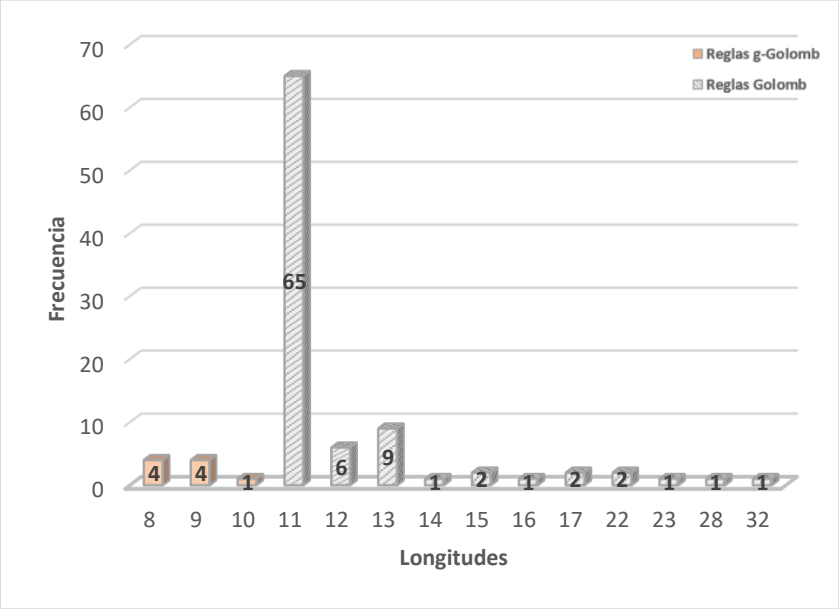


Figura 34. Resultados de la propuesta para 5 marcas.

Fuente: Elaboración propia.

Para el caso de ***n=6*** se puede evidenciar que la longitud óptima (17) fue la más encontrada con un total de 71 repeticiones [Figura 35]. También se puede apreciar que el algoritmo memético logra encontrar reglas g-Golomb con longitud 13, 14 y 15.

Regla Golomb óptima para 6 marcas: [0, 2, 7, 13, 16, 17] ; [0, 1, 8, 12, 14, 17] ; [0, 1, 4, 10, 15, 17]

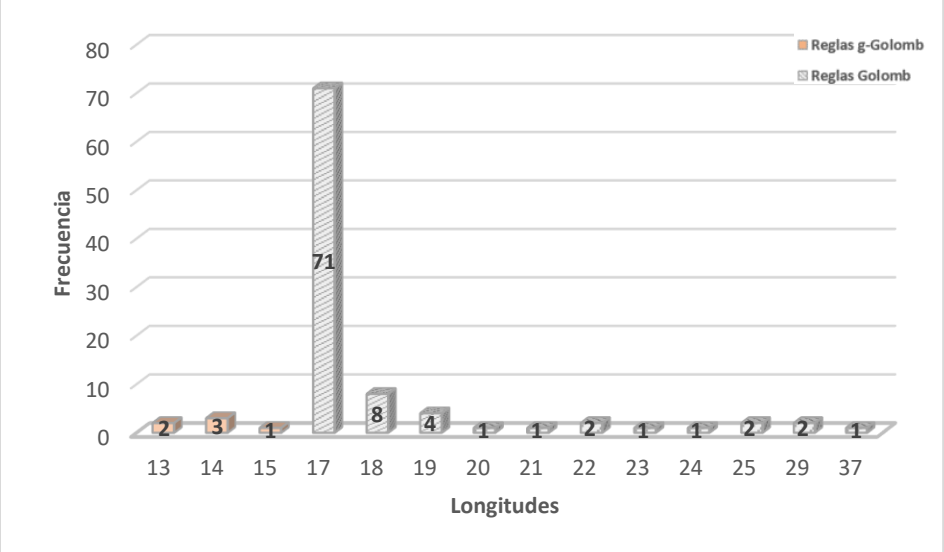


Figura 35. Resultados de la propuesta para 6 marcas.

Fuente: Elaboración propia.

Para el caso de ***n=7*** se puede evidenciar que la longitud que más veces se encontró fue 25 y es considerada como la longitud óptima para este número de marcas [Figura 36]. También logra obtener reglas Golomb cercanamente óptimas de longitud 26, 27 y 28.

Regla Golomb óptima para 7 marcas: [0, 2, 6, 9, 14, 24, 25] ; [0, 4, 9, 15, 22, 23, 25] ;[0, 1, 7, 11, 20, 23, 25]

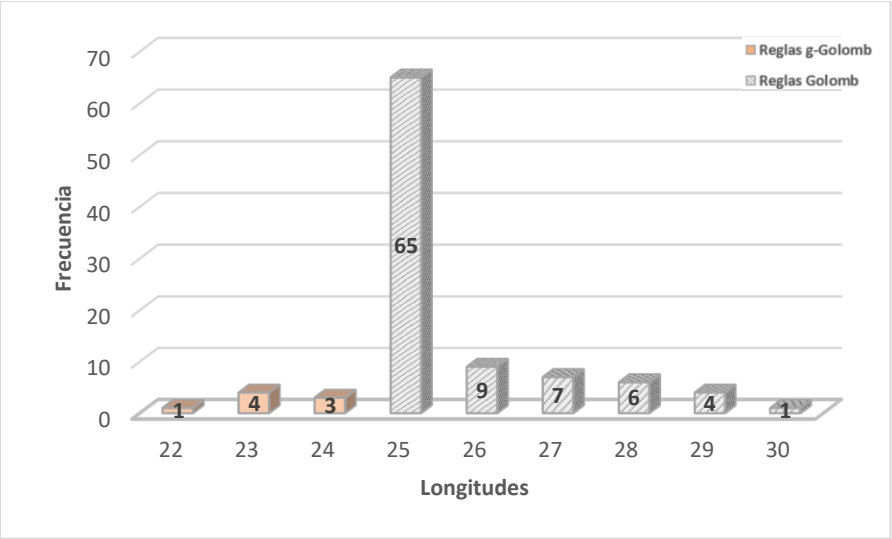


Figura 36. Resultados de la propuesta para 7 marcas.

Fuente: Elaboración propia.

Para el caso de ***n=8*** se puede evidenciar que las longitudes 37, 35 y 40 fueron las longitudes que más veces se encontraron con un total de 16, 16 y 15 repeticiones respectivamente [Figura 37]. También se puede evidenciar que la longitud óptima (34) solo fue encontrada 4 veces.

Regla Golomb óptima para 8 marcas: [0, 1, 4, 9, 15, 22, 32, 34]

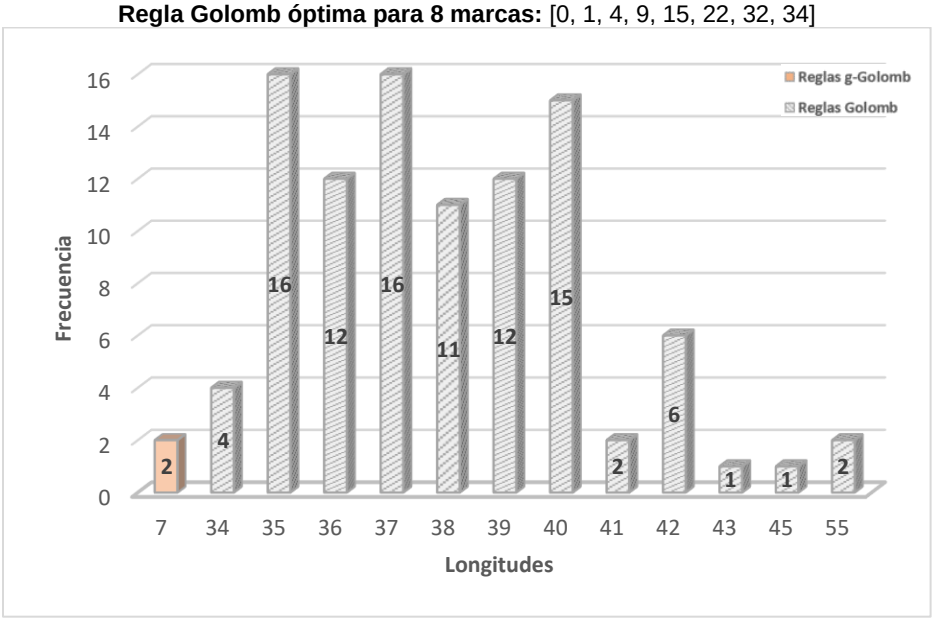


Figura 37. Resultados de la propuesta para 8 marcas.

Fuente: Elaboración propia.

Para el caso de ***n=9*** se puede evidenciar que la longitud más corta encontrada fue 51 apareciendo en 21 ocasiones seguida por las longitudes 64, 61 y 56 apareciendo en 15, 15 y 10 ocasiones respectivamente [Figura 38]. También se puede evidenciar que la propuesta de investigación logra encontrar reglas g-Golomb con longitud 6 y 16.

Regla Golomb óptima para 9 marcas: [0, 1, 5, 12, 25, 27, 35, 41, 44]

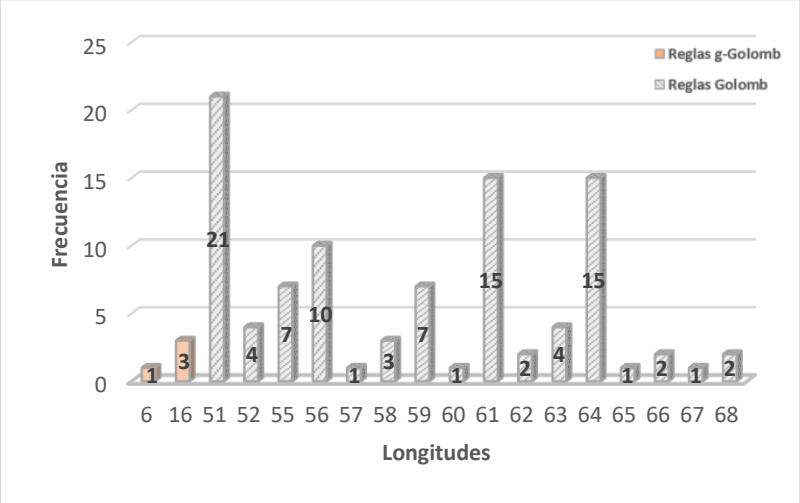


Figura 38. Resultados de la propuesta para 9 marcas.

Fuente: Elaboración propia.

Para el caso de ***n=10*** se puede evidenciar que las longitudes más cortas encontradas y con mayor número de repeticiones fueron las longitudes 71 y 73 apareciendo en 43 y 40 ocasiones respectivamente [Figura 39]. Adicionalmente, se logra evidenciar que se encontró una regla g-Golomb de longitud 21.

Regla Golomb óptima para 10 marcas: [0, 1, 6, 10, 23, 26, 34, 41, 53, 55]

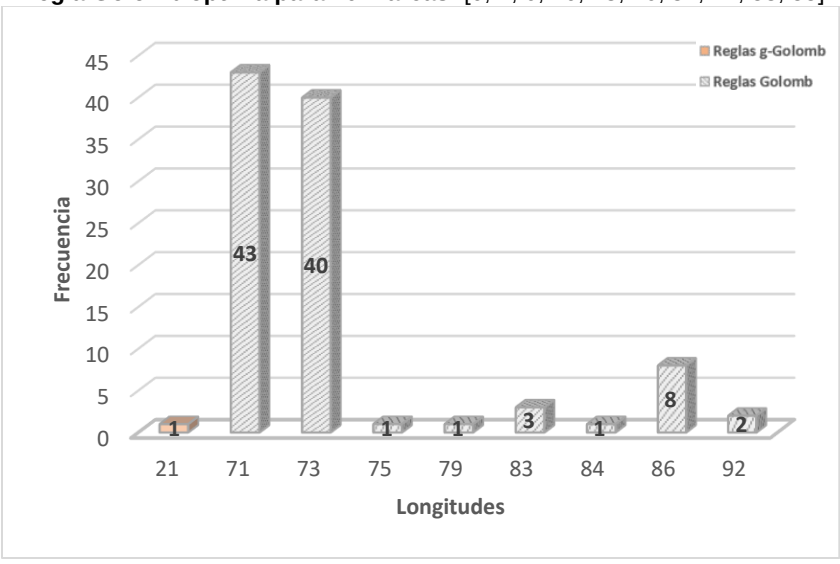


Figura 39. Resultados de la propuesta para 10 marcas.

Fuente: Elaboración propia.

Para el caso de ***n=11*** se puede evidenciar que la propuesta de investigación logra encontrar la longitud 74 en 98 ocasiones y solo en una ejecución logra encontrar la longitud óptima 72 [Figura 40].

Regla Golomb óptima para 11 marcas: [0, 1, 9, 19, 24, 31, 52, 56, 58, 69, 72]

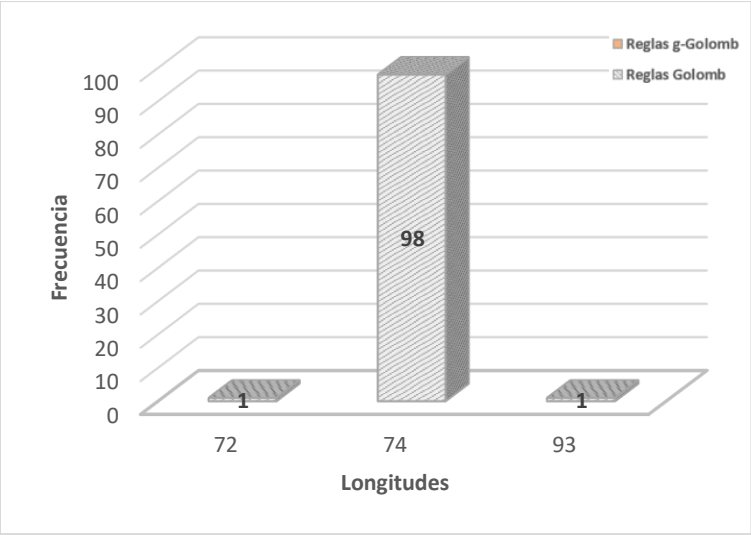


Figura 40. Resultados de la propuesta para 11 marcas.

Fuente: Elaboración propia.

Para el caso de ***n=12*** se puede evidenciar que la mejor longitud encontrada fue 101 con un total de 90 repeticiones seguida por las longitudes 109, 115 y 139 con un total de 5, 2 y 2 repeticiones respectivamente [Figura 41]. Adicionalmente, el algoritmo logra encontrar una regla g-Golomb de longitud 5.

Regla Golomb óptima para 12 marcas: [0, 2, 6, 24, 29, 40, 43, 55, 68, 75, 76, 85]

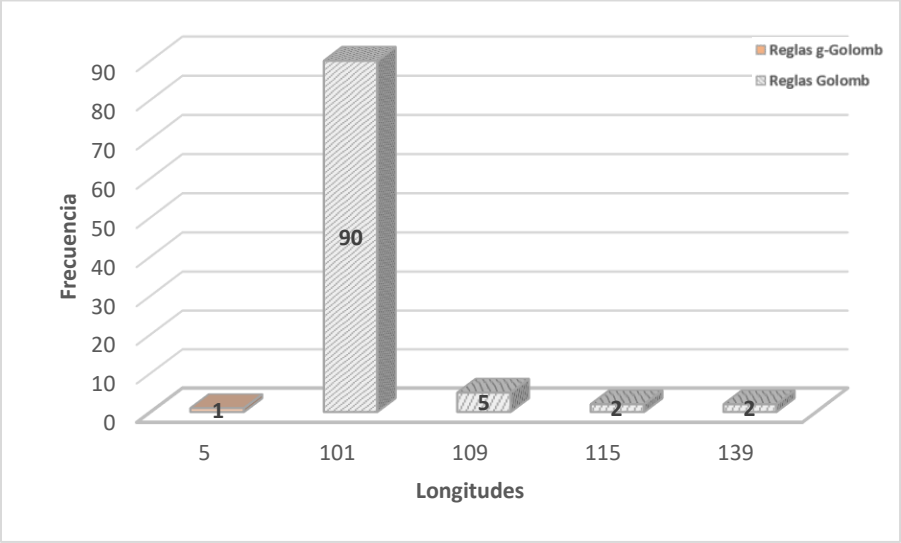


Figura 41. Resultados de la propuesta para 12 marcas.

Fuente: Elaboración propia.

Para el caso de ***n=13*** se puede evidenciar que la mejor longitud encontrada fue 111 con un total de 67 repeticiones [Figura 42]. También podemos apreciar que el algoritmo memético logra encontrar una regla g-Golomb con una longitud de 52.

Regla Golomb óptima para 13 marcas: [0, 2, 5, 25, 37, 43, 59, 70, 85, 89, 98, 99, 106]

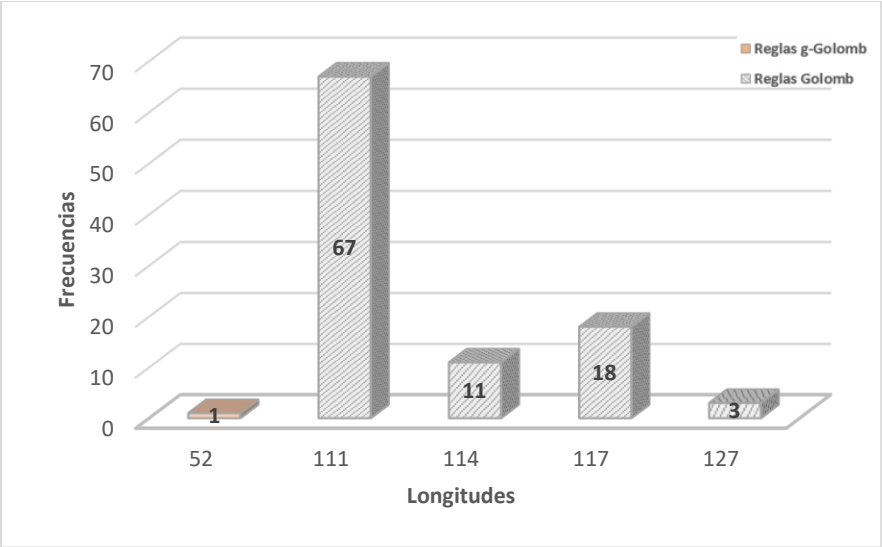


Figura 42. Resultados de la propuesta para 13 marcas.

Fuente: Elaboración propia.

Para el caso de ***n=14*** se puede evidenciar que el algoritmo memético logra encontrar 96 veces la longitud 160, siendo este, el mejor resultado obtenido para este número de marcas [Figura 43].

Regla Golomb óptima para 14 marcas: [0, 4, 6, 20, 35, 52, 59, 77, 78, 86, 89, 99, 122, 127]

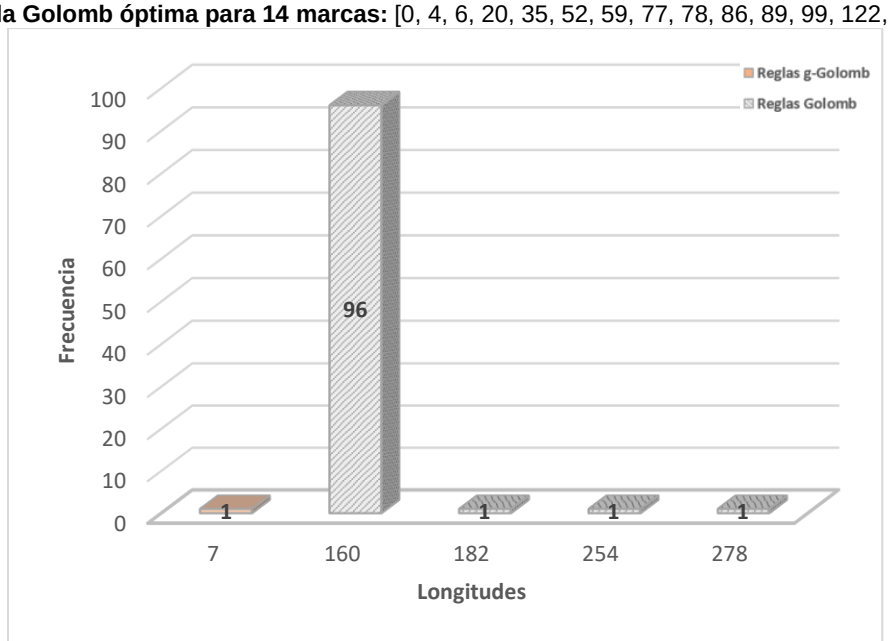


Figura 43. Resultados de la propuesta para 14 marcas.

Fuente: Elaboración propia.

Para el caso de ***n=15*** se puede evidenciar que la mejor longitud encontrada fue 166 apareciendo en 83 ocasiones [Figura 44]. Adicionalmente, la propuesta de investigación logra encontrar longitudes de orden superior, de hasta 200, pero no sin superar las 6 ocasiones en cada una de ellas.

Regla Golomb óptima para 15 marcas: [0, 4, 20, 30, 57, 59, 62, 76, 100, 111, 123, 136, 144, 145, 151]

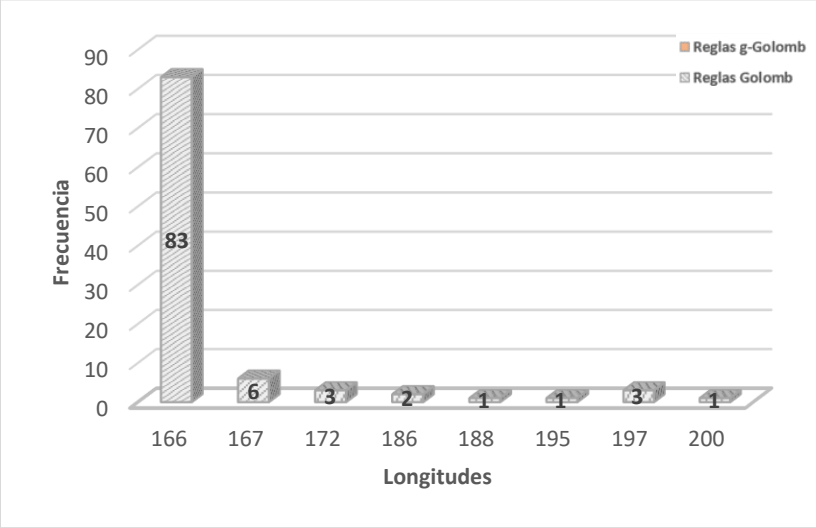


Figura 44. Resultados de la propuesta para 15 marcas.

Fuente: Elaboración propia.

Para el caso de ***n=16*** se puede evidenciar que la longitud 194 es la que más se logra encontrar apareciendo en 85 ocasiones y a su vez, es la mejor longitud que el algoritmo logra encontrar para este número de marcas [Figura 45].

Regla Golomb óptima para 16 marcas: [0, 1, 4, 11, 26, 32, 56, 68, 76, 115, 117, 134, 150, 163, 168, 177]

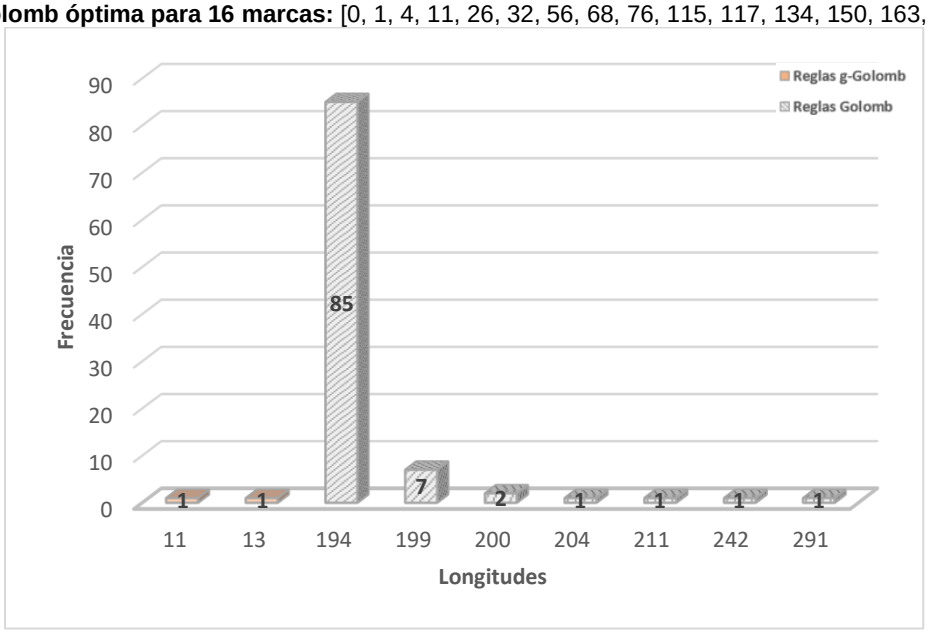


Figura 45. Resultados de la propuesta para 16 marcas.

Fuente: Elaboración propia.

Para el caso de ***n=17*** se puede evidenciar que el algoritmo memético logra encontrar la longitud 19 considerada como la longitud óptima [Figura 46]. También podemos evidenciar que la longitud que más veces se logra encontrar es 211 apareciendo 51 veces y la longitud que menos aparece es 218 en 4 ocasiones.

Regla Golomb óptima para 17 marcas: [0, 5, 7, 17, 52, 56, 67, 80, 81, 100, 122, 138, 159, 165, 168, 191, 199]

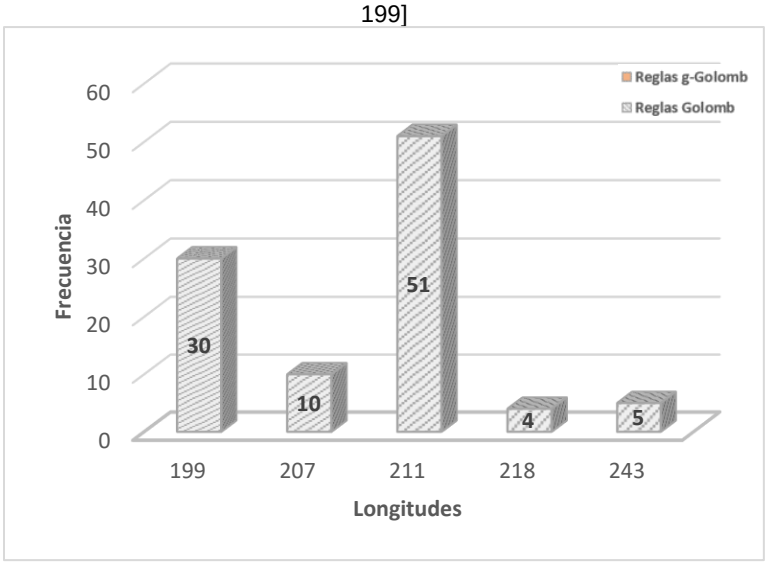


Figura 46. Resultados de la propuesta para 17 marcas.

Fuente: Elaboración propia.

Para el caso de ***n=18*** se puede evidenciar que la mejor longitud encontrada fue 245 apareciendo 51 veces seguido de la longitud 247 apareciendo 31 veces [Figura 47]. Adicionalmente, también se obtienen reglas g-Golomb de longitud 7 y 39.

Regla Golomb óptima para 18 marcas: [0, 2, 10, 22, 53, 56, 82, 83, 89, 98, 130, 148, 153, 167, 188, 192, 205, 216]

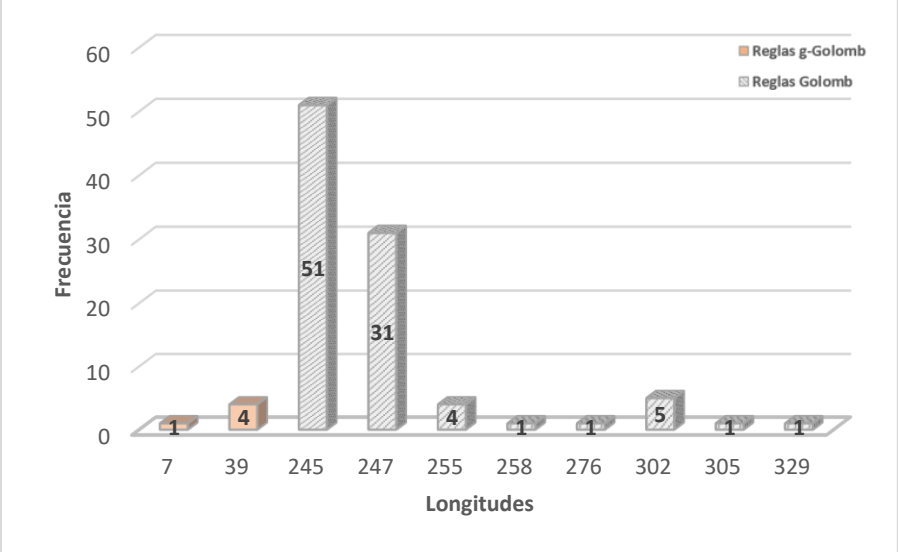


Figura 47. Resultados de la propuesta para 18 marcas.

Fuente: Elaboración propia.

Para el caso de ***n=19*** se puede evidenciar que la mejor longitud encontrada es 257 en donde se aparece en 82 ocasiones seguida de la longitud 318 apareciendo en 9 ocasiones [Figura 48].

Regla Golomb óptima para 19 marcas: [0, 1, 6, 25, 32, 72, 100, 108, 120, 130, 153, 169, 187, 190, 204, 231, 233, 242, 246]

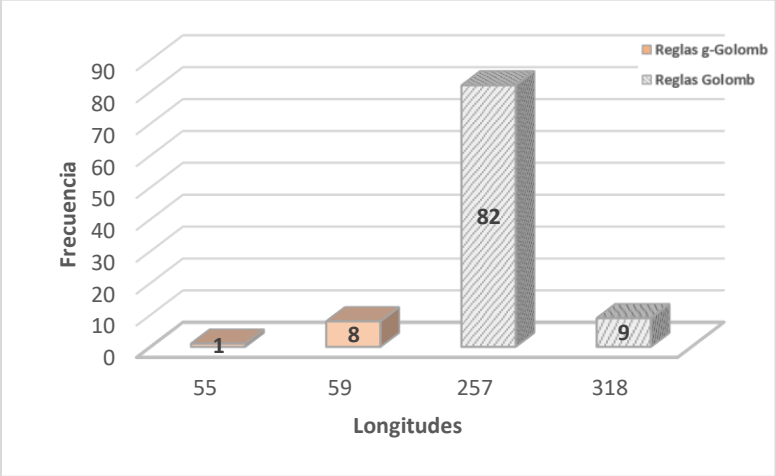


Figura 48. Resultados de la propuesta para 19 marcas.

Fuente: Elaboración propia.

Para el caso de ***n=20*** se puede evidenciar que en 60 ocasiones se encuentra la longitud 339 siendo esta la longitud más corta encontrada para este número de marcas [Figura 49]. También logra encontrar una regla g-Golomb de longitud 7.

Regla Golomb óptima para 20 marcas: [0, 1, 8, 11, 68, 77, 94, 116, 121, 156, 158, 179, 194, 208, 212, 228, 240, 253, 259, 283]

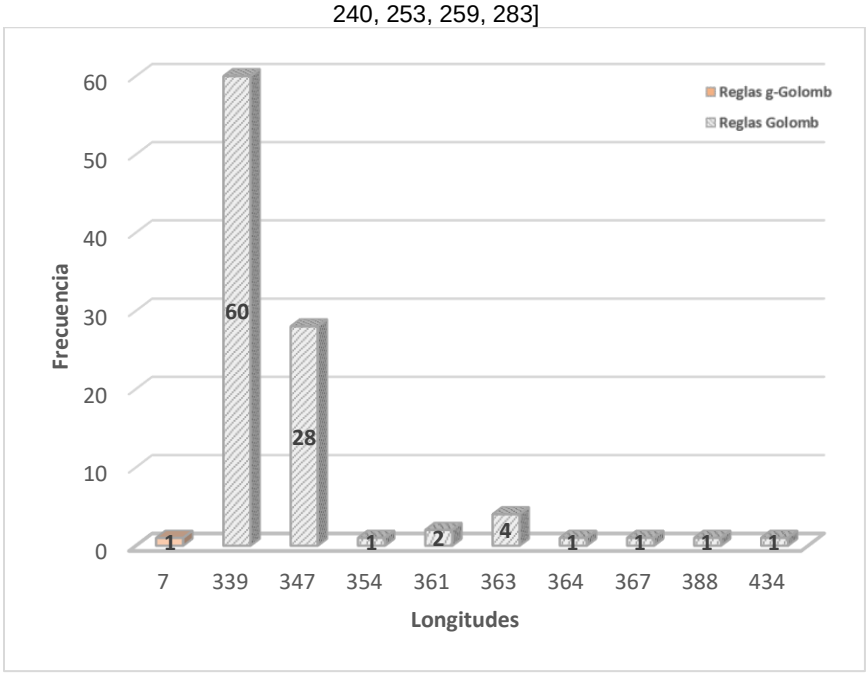


Figura 49. Resultados de la propuesta para 20 marcas.

Fuente: Elaboración propia.

Para el caso de ***n=21*** se puede evidenciar que en 78 ocasiones la propuesta de investigación logra encontrar la longitud 355 siendo esta la más corta encontrada [Figura 50]. También se logra encontrar las longitudes 368, 390, 445 y 486 sin superar las 9 repeticiones en cada una de ellas.

Regla Golomb óptima para 21 marcas: [0, 2, 24, 56, 77, 82, 83, 95, 129, 144, 179, 186, 195, 255, 265, 285, 293, 296, 310, 329, 333]

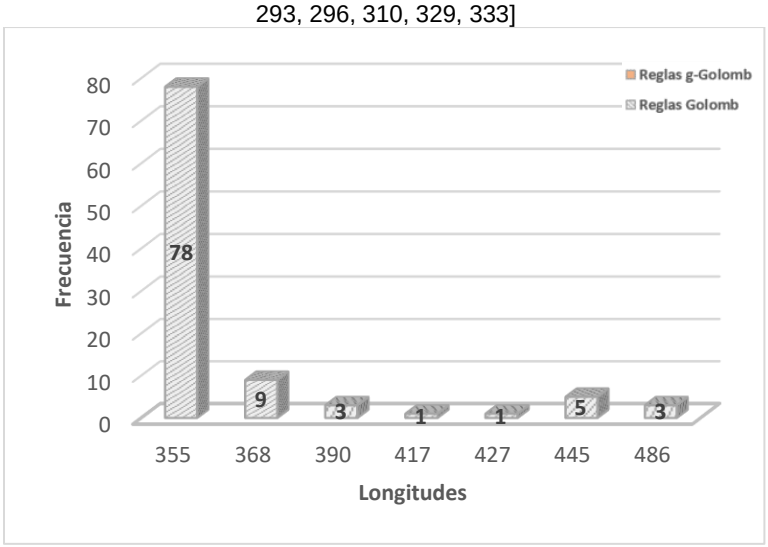


Figura 50. Resultados de la propuesta para 21 marcas.

Fuente: Elaboración propia.

Para el caso de ***n=22*** se puede evidenciar que el algoritmo memético logra encontrar la longitud 326 considerada como la óptima en 82 ocasiones [Figura 51]. El resto de longitudes encontradas no superan las 6 ocasiones en cada una de ellas.

Regla Golomb óptima para 22 marcas: [0, 1, 9, 14, 43, 70, 106, 122, 124, 128, 159, 179, 204, 223, 253, 263, 270, 291, 330, 341, 353, 356]

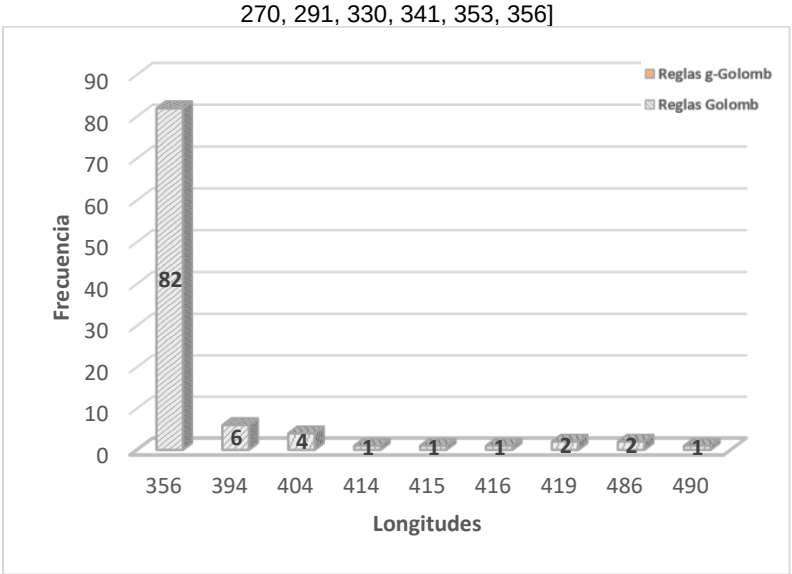


Figura 51. Resultados de la propuesta para 22 marcas.

Fuente: Elaboración propia.

Para el caso de ***n=23*** se puede evidenciar que la propuesta de investigación solo logra encontrar 2 longitudes diferentes, la longitud 388 y 419 con 98 y 2 repeticiones respectivamente, siendo la longitud 388 la más corta encontrada [Figura 52].

Regla Golomb óptima para 23 marcas: [0, 3, 7, 17, 61, 66, 91, 99, 114, 159, 171, 199, 200, 226, 235, 246, 277, 316, 329, 348, 350, 366, 372]

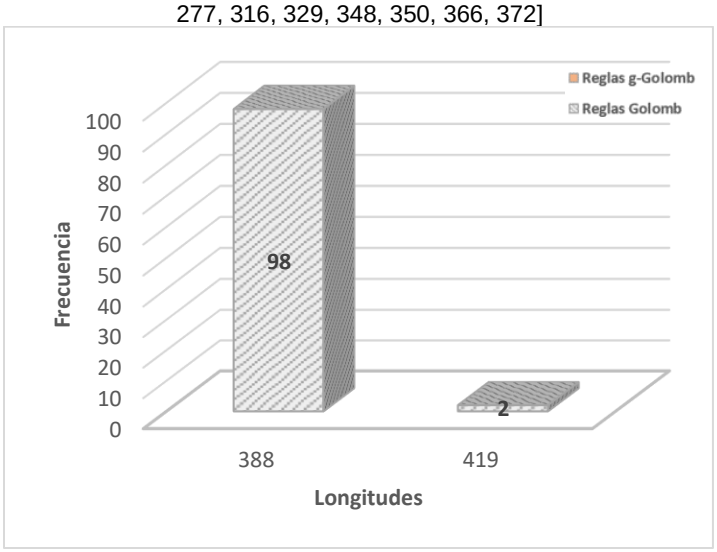


Figura 52. Resultados de la propuesta para 23 marcas.

Fuente: Elaboración propia.

Para el caso de ***n=24*** se puede evidenciar que el algoritmo memético logra encontrar 3 longitudes diferentes para este número de marcas, las longitudes 503, 517 y 526 aparecen en 65, 20 y 15 ocasiones respectivamente [Figura 53].

Regla Golomb óptima para 24 marcas: [0, 9, 33, 37, 38, 97, 122, 129, 140, 142, 152, 191, 205, 208, 252, 278, 286, 326, 332, 353, 368, 384, 403, 425]

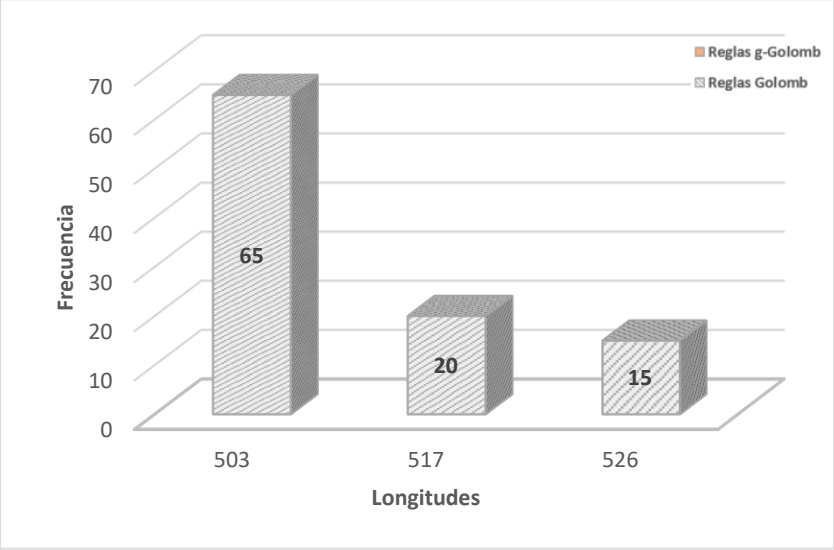


Figura 53. Resultados de la propuesta para 24 marcas.

Fuente: Elaboración propia.

Para el caso de ***n=25*** se puede evidenciar que la longitud que más se repite es la 507 apareciendo en 68 ocasiones [Figura 54]. También se puede evidenciar que logró encontrar un total de 28 reglas g-Golomb para este número de marcas.

Regla Golomb óptima para 25 marcas: [0, 12, 29, 39, 72, 91, 146, 157, 160, 161, 166, 191, 207, 214, 258, 290, 316, 354, 372, 394, 396, 431, 459, 467, 480]

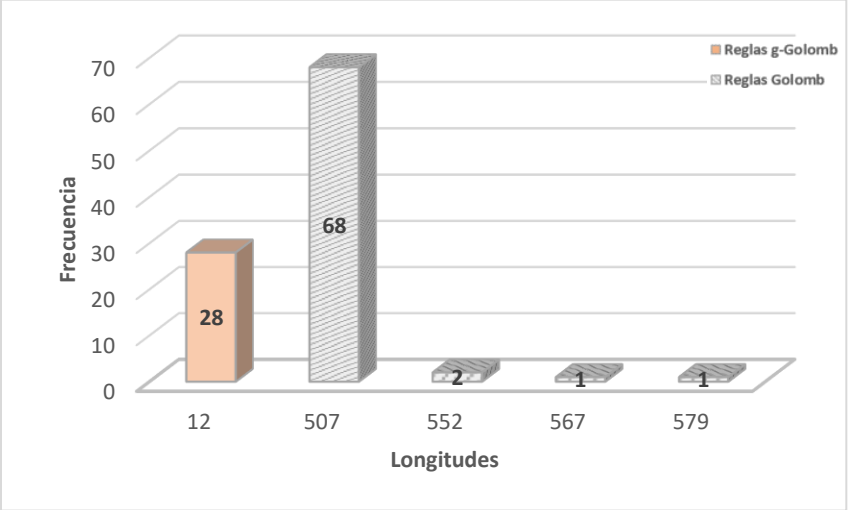


Figura 54. Resultados de la propuesta para 25 marcas.

Fuente: Elaboración propia.

Para el caso de ***n=26*** se puede evidenciar que la mejor longitud encontrada es 572 apareciendo en 82 ocasiones seguida de las longitudes 590, 583 y 632 apareciendo en 10, 5 y 3 ocasiones respectivamente [Figura 55].

Regla Golomb óptima para 26 marcas: [0, 1, 33, 83, 104, 110, 124, 163, 185, 200, 203, 249, 251, 258, 314, 318, 343, 356, 386, 430, 440, 456, 464, 475, 487, 492]

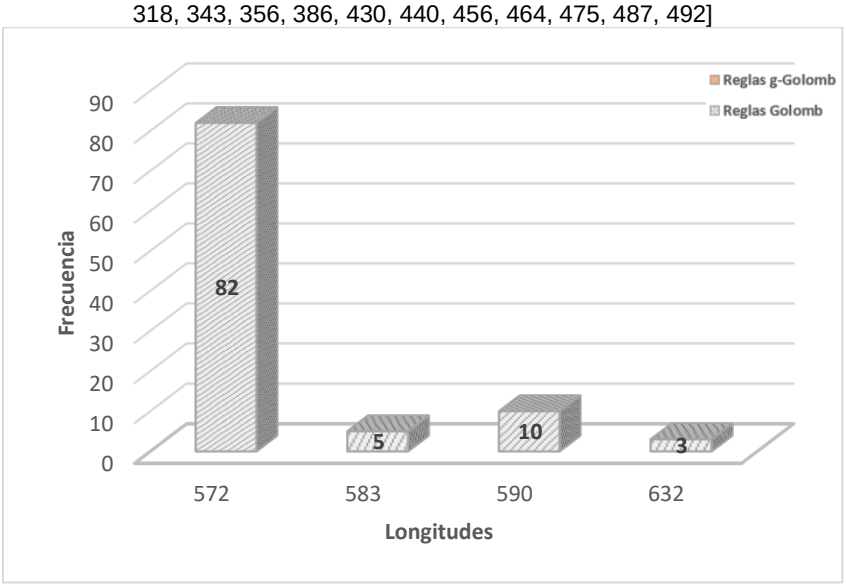


Figura 55. Resultados de la propuesta para 26 marcas.

Fuente: Elaboración propia.

Para el caso de ***n=27*** se puede evidenciar que la propuesta de investigación encuentra 8 longitudes diferentes para este número de marcas siendo la longitud 596 la más corta encontrada con 32 apariciones, seguida de las longitudes 614, 632, 627 y 624 apareciendo en 18, 14, 12 y 10 ocasiones [Figura 56].

Regla Golomb óptima para 27 marcas: [0, 3, 15, 41, 66, 95, 97, 106, 142, 152, 220, 221, 225, 242, 295, 330, 338, 354, 382, 388, 402, 415, 486, 504, 523, 546, 553]

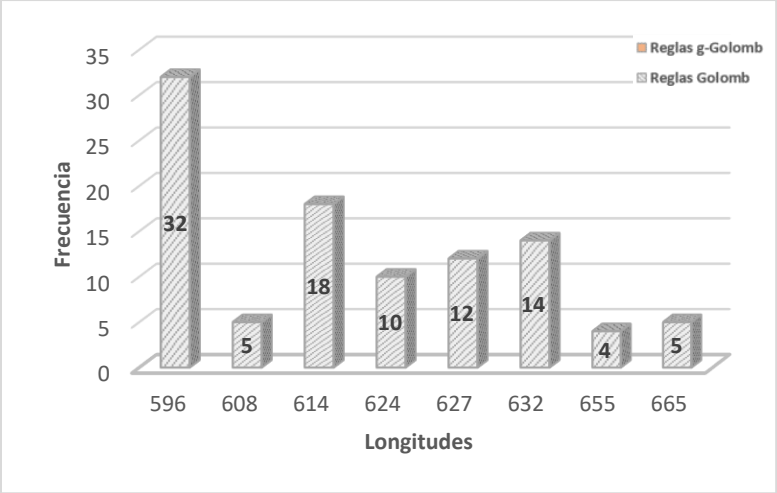


Figura 56. Resultados de la propuesta para 27 marcas.

Fuente: Elaboración propia.

Para el caso de ***n=28*** se puede evidenciar que la mejor longitud encontrada por el algoritmo memético para este número de marcas fue 666 apareciendo en 73 ocasiones [Figura 57]. También se puede evidenciar que las longitudes con menos apariciones fueron 704 y 697 con tan solo una aparición.

Regla Golomb óptima para 28 marcas: [0, 3, 15, 41, 66, 95, 97, 106, 142, 152, 220, 221, 225, 242, 295, 330, 338, 354, 382, 388, 402, 415, 486, 504, 523, 546, 553, 585]

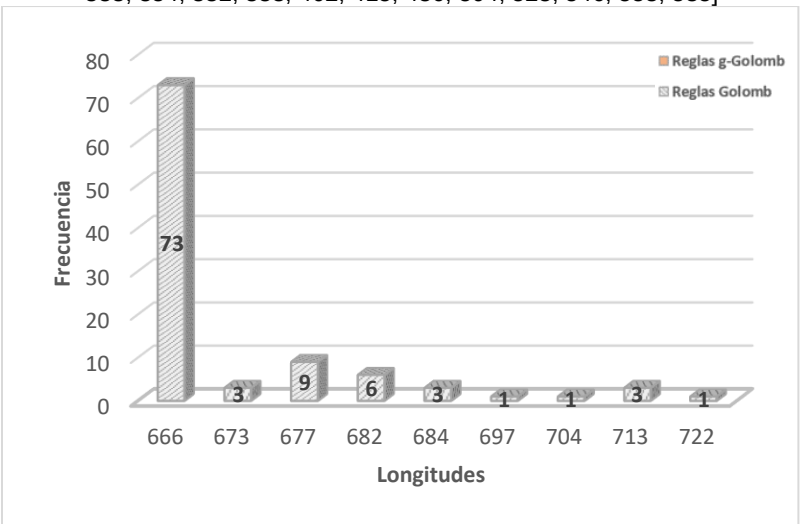


Figura 57. Resultados de la propuesta para 28 marcas.

Fuente: Elaboración propia.

Con la ayuda de los resultados mostrados anteriormente, se crea la tabla 23 en donde se presenta los diferentes resultados con los diferentes parámetros de entrada que utilizo el algoritmo memético para encontrar las reglas Golomb óptimas y cercanamente óptimas.

Tabla 23. Resultados del algoritmo memético.

M	P	L	L_p	Regla Golomb	¿O?	Tiempo de CPU
1	GP = 50 TP = 50 RP = 20	0	1	[0]	Sí	0.63
2	GP = 20 TP = 50 RP = 20	1	2	[0, 1]	Sí	2.748
3	GP = 20 TP = 50 RP = 20	3	3.4	[0, 1, 3] [0, 2, 3]	Sí	3.006
4	GP = 20 TP = 50 RP = 20	6	6.64	[0, 1, 4, 6] [0, 2, 5, 6]	Sí	3.972
5	GP = 50 TP = 50 RP = 20	11	12.03	[0, 2, 7, 10, 11] [0, 2, 7, 8, 11] [0, 3, 4, 9, 11] [0, 1, 4, 9, 11]	Sí	4.794
6	GP = 50 TP = 50 RP = 20	17	17.87	[0, 1, 4, 10, 12, 17] [0, 1, 4, 10, 15, 17] [0, 1, 8, 11, 13, 17] [0, 1, 8, 12, 14, 17] [0, 5, 7, 13, 16, 17] [0, 4, 6, 9, 16, 17] [0, 2, 7, 13, 16, 17] [0, 3, 5, 9, 16, 17]	Sí	7.602
7	GP = 80 TP = 50 RP = 20	25	25.48	[0, 2, 6, 9, 14, 24, 25] [0, 4, 9, 15, 22, 23, 25] [0, 1, 7, 11, 20, 23, 25] [0, 1, 11, 16, 19, 23, 25] [0, 2, 3, 10, 16, 21, 25] [0, 1, 4, 10, 18, 23, 25] [0, 2, 5, 14, 18, 24, 25] [0, 2, 7, 15, 21, 24, 25] [0, 3, 4, 12, 18, 23, 25] [0, 2, 7, 13, 21, 22, 25]	Sí	7.872
8	GP = 43 TP = 43 RP = 4	34	37.52	[0, 1, 4, 9, 15, 22, 32, 34]	Sí	11.496
9	GP = 28 TP = 59 RP = 22	51	56.33	[0, 3, 12, 28, 32, 43, 45, 50, 51]	No	11.976
10	GP = 67 TP = 65 RP = 41	71	73.53	[0, 8, 18, 23, 30, 51, 55, 57, 68, 71]	No	15.192
11	GP = 83 TP = 98 RP = 89	72	74.17	[0, 1, 9, 19, 24, 31, 52, 56, 58, 69, 72]	Sí	15.864
12	GP = 36 TP = 82 RP = 59	101	101.48	[0, 8, 10, 32, 49, 58, 61, 65, 76, 95, 96, 101]	No	29.364
13	GP = 13 TP = 45 RP = 6	111	112.3	[0, 3, 11, 38, 40, 47, 62, 72, 88, 92, 93, 105, 111] [0, 6, 18, 19, 23, 39, 49, 64, 71, 73, 100, 108, 111]	No	31.031

M	P	L	L_p	Regla Golomb	¿O?	Tiempo de CPU
14	GP = 9 TP = 31 RP = 54	160	160.81	[0, 19, 28, 50, 58, 60, 65, 113, 117, 133, 134, 146, 157, 160] [0, 9, 15, 25, 49, 62, 66, 101, 122, 127, 129, 130, 149, 160] [0, 11, 30, 31, 33, 38, 59, 94, 98, 111, 135, 145, 151, 160] [0, 13, 14, 37, 41, 84, 93, 99, 110, 115, 118, 148, 150, 160] [0, 10, 12, 42, 45, 50, 61, 67, 76, 119, 123, 146, 147, 160] [0, 3, 14, 26, 27, 43, 47, 95, 100, 102, 110, 132, 141, 160]	No	34.257
15	GP = 89 TP = 45 RP = 82	166	168.42	[0, 3, 14, 26, 27, 43, 47, 95, 100, 102, 110, 132, 141, 160, 166]	No	36.889
16	GP = 16 TP = 7 RP = 7	194	192.55	[0, 2, 12, 47, 51, 62, 75, 76, 95, 117, 133, 154, 160, 163, 186, 194]	No	40.389
17	GP = 80 TP = 50 RP = 20	199	208.88	[0, 5, 7, 17, 52, 56, 67, 80, 81, 100, 122, 138, 159, 165, 168, 191, 199] [0, 8, 31, 34, 40, 61, 77, 99, 118, 119, 132, 143, 147, 182, 192, 194, 199]	Sí	62.179
18	GP = 70 TP = 31 RP = 20	245	240.13	[0, 5, 24, 31, 71, 99, 107, 119, 129, 152, 168, 186, 189, 203, 230, 232, 241, 245]	No	74.198
19	GP = 65 TP = 69 RP = 76	257	244.63	[0, 2, 15, 48, 52, 59, 73, 124, 129, 141, 147, 163, 171, 190, 216, 225, 226, 254, 257] [0, 3, 31, 32, 41, 67, 86, 94, 110, 116, 128, 133, 184, 198, 205, 209, 242, 255, 257]	No	79.065
20	GP = 80 TP = 50 RP = 20	339	341.44	[0, 13, 37, 39, 45, 102, 106, 172, 183, 190, 200, 219, 231, 234, 254, 287, 309, 314, 330, 339]	No	90.631
21	GP = 78 TP = 5 RP = 58	355	366.99	[0, 3, 15, 26, 65, 86, 93, 103, 133, 152, 177, 197, 228, 232, 234, 250, 286, 313, 342, 347, 355]	No	130.187
22	GP = 80 TP = 50 RP = 20	356	367.17	[0, 3, 15, 26, 65, 86, 93, 103, 133, 152, 177, 197, 228, 232, 234, 250, 286, 313, 342, 347, 355, 356]	Sí	145.733
23	GP = 80 TP = 50 RP = 20	388	388.62	[0, 32, 33, 41, 46, 75, 102, 138, 154, 156, 160, 191, 211, 236, 255, 285, 295, 302, 323, 362, 373, 385, 388]	No	153.611
24	GP = 80 TP = 50 RP = 20	503	509.25	[0, 15, 31, 43, 107, 124, 134, 142, 145, 191, 228, 260, 289, 296, 340, 366, 406, 428, 429, 448, 453, 462, 501, 503]	No	173.446
25	GP = 49 TP = 38 RP = 20	507	370.62	[0, 15, 31, 43, 107, 124, 134, 142, 145, 191, 228, 260, 289, 296, 340, 366, 406, 428, 429, 448, 453, 462, 501, 503, 507]	No	197.274

M	P	L	L_p	Regla Golomb	¿O?	Tiempo de CPU
26	GP = 39 TP = 9 RP = 20	572	576.15	[0, 15, 19, 25, 43, 64, 91, 108, 159, 166, 200, 220, 222, 233, 236, 262, 321, 333, 368, 448, 485, 486, 494, 517, 567, 572]	No	199.298
27	GP = 51 TP = 24 RP = 19	596	617.21	[0, 6, 32, 43, 60, 84, 124, 191, 218, 220, 267, 277, 323, 332, 346, 394, 414, 429, 433, 445, 467, 503, 533, 575, 578, 583, 596]	No	224.924
28	GP = 94 TP = 81 RP = 15	666	671.36	[0, 48, 87, 107, 122, 123, 160, 191, 201, 203, 225, 253, 286, 342, 350, 397, 442, 448, 451, 455, 469, 474, 518, 584, 609, 626, 655, 666]	No	275.780

***M**: Número de marcas; **P**: Parametrización; **GP**: GolombPercentage; **TP**: TranslationsPercentage*

***RP**: RegeneratePopPercentage; **L**: Longitud de la mejor regla encontrada;*
L_p*: Promedio aproximado de las longitudes encontradas en las 100 ejecuciones;*

***Tiempo de CPU:** Tiempo de máquina en segundos que tardo el algoritmo en encontrar la regla en particular.*

En la tabla anterior se puede evidenciar que el algoritmo memético logra encontrar reglas Golomb óptimas para las primeras 8 marcas, también logra encontrar las reglas óptimas para 11, 17 y 22 marcas. De igual manera, se logra apreciar en la tabla 23 que el algoritmo memético logró encontrar más de una regla Golomb óptima para las marcas 3, 4, 5, 6, 7 y 17 respectivamente. También se puede evidenciar que los valores de la parametrización varían dependiendo de la marca que se desee encontrar esto con el objetivo de potenciar el espacio de búsqueda en cada una de las marcas trabajadas. Por otro lado, comparando los resultados con las reglas Golomb encontradas por el estado del arte [Figura 14], se puede evidenciar que la propuesta de investigación logra superar el límite de 21 marcas trabajadas encontrando reglas Golomb cercanamente óptimas hasta 28 marcas.

Adicionalmente, se puede evidenciar en la tabla anterior el tiempo de máquina que le tomó al algoritmo memético en encontrar cada una de las reglas Golomb óptimas y cercanamente óptimas. A pesar de que el objetivo general de este proyecto de investigación se centra en la calidad de los resultados finales, es decir, las longitudes de las reglas y no en el tiempo de máquina que se necesitó para encontrarlas, se puede evidenciar que, a medida que aumenta el número de marcas el tiempo de máquina aumenta linealmente y esto está relacionado directamente con la complejidad del problema [15, 16 y 17] mencionado en el capítulo 1.

5.4. Comparación de los resultados con las reglas Golomb óptimas conocidas.

En esta sección se presentará la comparación de los resultados de la propuesta de investigación con las reglas Golomb óptimas conocidas hasta la fecha [Anexo 2] en términos de longitudes de las reglas. La tabla 24 reúne los resultados del algoritmo memético propuesto con las 28 reglas Golomb óptimas conocidas hasta el momento y con la ayuda del error relativo se puede evidenciar que tan alejadas estuvieron las longitudes obtenidas por la propuesta de las longitudes óptimas.

Tabla 24. Comparación de los resultados con las reglas Golomb óptimas conocidas.

M	L_e	L_o	E_r	Regla Golomb encontrada	Regla Golomb óptima
1	0	0	0	[0]	[0]
2	1	1	0	[0, 1]	[0, 1]
3	3	3	0	[0, 1, 3] **[0, 2, 3]**	[0, 1, 3]
4	6	6	0	[0, 1, 4, 6] **[0, 2, 5, 6]**	[0, 1, 4, 6]
5	11	11	0	[0, 1, 4, 9, 11] [0, 2, 7, 8, 11] [0, 3, 4, 9, 11] **[0, 2, 7, 10, 11]**	[0, 1, 4, 9, 11] [0, 2, 7, 8, 11] [0, 3, 4, 9, 11]
6	17	17	0	[0, 1, 4, 10, 12, 17] [0, 1, 4, 10, 15, 17] [0, 1, 8, 11, 13, 17] [0, 1, 8, 12, 14, 17] **[0, 5, 7, 13, 16, 17]** **[0, 4, 6, 9, 16, 17]** **[0, 2, 7, 13, 16, 17]** **[0, 3, 5, 9, 16, 17]**	[0, 1, 4, 10, 12, 17] [0, 1, 4, 10, 15, 17] [0, 1, 8, 11, 13, 17] [0, 1, 8, 12, 14, 17]
7	25	25	0	[0, 1, 7, 11, 20, 23, 25] [0, 1, 11, 16, 19, 23, 25] [0, 2, 3, 10, 16, 21, 25] [0, 1, 4, 10, 18, 23, 25] [0, 2, 7, 13, 21, 22, 25] **[0, 2, 6, 9, 14, 24, 25]** **[0, 4, 9, 15, 22, 23, 25]** **[0, 2, 5, 14, 18, 24, 25]** **[0, 2, 7, 15, 21, 24, 25]** **[0, 3, 4, 12, 18, 23, 25]**	[0, 1, 7, 11, 20, 23, 25] [0, 1, 11, 16, 19, 23, 25] [0, 2, 3, 10, 16, 21, 25] [0, 1, 4, 10, 18, 23, 25] [0, 2, 7, 13, 21, 22, 25]
8	34	34	0	[0, 1, 4, 9, 15, 22, 32, 34]	[0, 1, 4, 9, 15, 22, 32, 34]
9	51	44	0.1590	[0, 3, 12, 28, 32, 43, 45, 50, 51]	[0, 1, 5, 12, 25, 27, 35, 41, 44]
10	71	55	55	[0, 8, 18, 23, 30, 51, 55, 57, 68, 71]	[0, 1, 6, 10, 23, 26, 34, 41, 53, 55]
11	72	72	0	[0, 1, 9, 19, 24, 31, 52, 56, 58, 69, 72]	[0, 1, 9, 19, 24, 31, 52, 56, 58, 69, 72] [0, 1, 4, 13, 28, 33, 47, 54, 64, 70, 72]
12	101	85	0.1882	[0, 8, 10, 32, 49, 58, 61, 65, 76, 95, 96, 101]	[0, 2, 6, 24, 29, 40, 43, 55, 68, 75, 76, 85]

M	L_e	L_o	E_r	Regla Golomb encontrada	Regla Golomb óptima
13	111	106	0.0471	[0, 3, 11, 38, 40, 47, 62, 72, 88, 92, 93, 105, 111] [0, 6, 18, 19, 23, 39, 49, 64, 71, 73, 100, 108, 111]	[0, 2, 5, 25, 37, 43, 59, 70, 85, 89, 98, 99, 106]
14	160	127	0.2598	[0, 19, 28, 50, 58, 60, 65, 113, 117, 133, 134, 146, 157, 160] [0, 9, 15, 25, 49, 62, 66, 101, 122, 127, 129, 130, 149, 160] [0, 11, 30, 31, 33, 38, 59, 94, 98, 111, 135, 145, 151, 160] [0, 13, 14, 37, 41, 84, 93, 99, 110, 115, 118, 148, 150, 160] [0, 10, 12, 42, 45, 50, 61, 67, 76, 119, 123, 146, 147, 160] [0, 3, 14, 26, 27, 43, 47, 95, 100, 102, 110, 132, 141, 160]	[0, 4, 6, 20, 35, 52, 59, 77, 78, 86, 89, 99, 122, 127]
15	166	151	0.0993	[0, 3, 14, 26, 27, 43, 47, 95, 100, 102, 110, 132, 141, 160, 166]	[0, 4, 20, 30, 57, 59, 62, 76, 100, 111, 123, 136, 144, 145, 151]
16	194	177	0.0960	[0, 2, 12, 47, 51, 62, 75, 76, 95, 117, 133, 154, 160, 163, 186, 194]	[0, 1, 4, 11, 26, 32, 56, 68, 76, 115, 117, 134, 150, 163, 168, 177]
17	199	199	0	[0, 5, 7, 17, 52, 56, 67, 80, 81, 100, 122, 138, 159, 165, 168, 191, 199] **[0, 8, 31, 34, 40, 61, 77, 99, 118, 119, 132, 143, 147, 182, 192, 194, 199]**	[0, 5, 7, 17, 52, 56, 67, 80, 81, 100, 122, 138, 159, 165, 168, 191, 199]
18	245	216	0.134	[0, 5, 24, 31, 71, 99, 107, 119, 129, 152, 168, 186, 189, 203, 230, 232, 241, 245]	[0, 2, 10, 22, 53, 56, 82, 83, 89, 98, 130, 148, 153, 167, 188, 192, 205, 216]
19	257	246	0.0447	[0, 2, 15, 48, 52, 59, 73, 124, 129, 141, 147, 163, 171, 190, 216, 225, 226, 254, 257] [0, 3, 31, 32, 41, 67, 86, 94, 110, 116, 128, 133, 184, 198, 205, 209, 242, 255, 257]	[0, 1, 6, 25, 32, 72, 100, 108, 120, 130, 153, 169, 187, 190, 204, 231, 233, 242, 246]
20	339	283	0.1978	[0, 13, 37, 39, 45, 102, 106, 172, 183, 190, 200, 219, 231, 234, 254, 287, 309, 314, 330, 339]	[0, 1, 8, 11, 68, 77, 94, 116, 121, 156, 158, 179, 194, 208, 212, 228, 240, 253, 259, 283]
21	355	333	0.066	[0, 3, 15, 26, 65, 86, 93, 103, 133, 152, 177, 197, 228, 232, 234, 250, 286, 313, 342, 347, 355]	[0, 2, 24, 56, 77, 82, 83, 95, 129, 144, 179, 186, 195, 255, 265, 285, 293, 296, 310, 329, 333]
22	356	356	0	**[0, 3, 15, 26, 65, 86, 93, 103, 133, 152, 177, 197, 228, 232, 234, 250, 286, 313, 342, 347, 355, 356]**	[0, 1, 9, 14, 43, 70, 106, 122, 124, 128, 159, 179, 204, 223, 253, 263, 270, 291, 330, 341, 353, 356]
23	388	372	0.043	[0, 32, 33, 41, 46, 75, 102, 138, 154, 156, 160, 191, 211, 236, 255, 285, 295, 302, 323, 362, 373, 385, 388]	[0, 3, 7, 17, 61, 66, 91, 99, 114, 159, 171, 199, 200, 226, 235, 246, 277, 316, 329, 348, 350, 366, 372]

24	503	425	0.1835	[0, 15, 31, 43, 107, 124, 134, 142, 145, 191, 228, 260, 289, 296, 340, 366, 406, 428, 429, 448, 453, 462, 501, 503]	[0, 9, 33, 37, 38, 97, 122, 129, 140, 142, 152, 191, 205, 208, 252, 278, 286, 326, 332, 353, 368, 384, 403, 425]
25	507	480	0.0562	[0, 15, 31, 43, 107, 124, 134, 142, 145, 191, 228, 260, 289, 296, 340, 366, 406, 428, 429, 448, 453, 462, 501, 503, 507]	[0, 12, 29, 39, 72, 91, 146, 157, 160, 161, 166, 191, 207, 214, 258, 290, 316, 354, 372, 394, 396, 431, 459, 467, 480]
26	572	492	0.1626	[0, 15, 19, 25, 43, 64, 91, 108, 159, 166, 200, 220, 222, 233, 236, 262, 321, 333, 368, 448, 485, 486, 494, 517, 567, 572]	[0, 1, 33, 83, 104, 110, 124, 163, 185, 200, 203, 249, 251, 258, 314, 318, 343, 356, 386, 430, 440, 456, 464, 475, 487, 492]
27	596	553	0.0777	[0, 6, 32, 43, 60, 84, 124, 191, 218, 220, 267, 277, 323, 332, 346, 394, 414, 429, 433, 445, 467, 503, 533, 575, 578, 583, 596]	[0, 3, 15, 41, 66, 95, 97, 106, 142, 152, 220, 221, 225, 242, 295, 330, 338, 354, 382, 388, 402, 415, 486, 504, 523, 546, 553]
28	666	585	0.1384	[0, 48, 87, 107, 122, 123, 160, 191, 201, 203, 225, 253, 286, 342, 350, 397, 442, 448, 451, 455, 469, 474, 518, 584, 609, 626, 655, 666]	[0, 3, 15, 41, 66, 95, 97, 106, 142, 152, 220, 221, 225, 242, 295, 330, 338, 354, 382, 388, 402, 415, 486, 504, 523, 546, 553, 585]

$\mathbf{L_e}$: Longitud de la regla Golomb encontrada por la propuesta de investigación.

$\mathbf{L_o}$: Longitud de la regla Golomb óptima conocida hasta fecha.

$\mathbf{E_r}$: Error relativo aproximado entre $\mathbf{L_o}$ y $\mathbf{L_e}$

En la tabla anterior se puede evidenciar el análisis comparativo entre las reglas Golomb encontradas por la propuesta de investigación con las reglas Golomb óptimas conocidas hasta la fecha. Está análisis se realizará comparando las 28 longitudes de las reglas Golomb óptimas conocidas con las longitudes de las reglas Golomb encontradas por el algoritmo memético. Revisando la figura 58 y los resultados de la tabla anterior en términos de reglas Golomb cercanamente óptimas, se puede evidenciar que a partir de la marca número 8, exceptuando la 11, 17 y 22, la propuesta de investigación logra encontrar reglas Golomb cercanamente óptimas hasta 28 marcas, esto quiere decir que, el algoritmo memético obtiene resultados óptimos al problema de investigación en marcas de orden inferior a 8 pero para encontrar marcas de orden superior en donde el problema de investigación aumenta su complejidad [15, 16 y 17], la propuesta solo logra encontrar reglas Golomb cercanamente óptimas. Asimismo, el algoritmo memético encuentra con éxito un total de 6 reglas Golomb cercanamente óptimas para 14 marcas, siendo este el número de marcas en donde el algoritmo memético logró potenciar en mayor medida el espacio de búsqueda del problema de optimización [Figura 58].

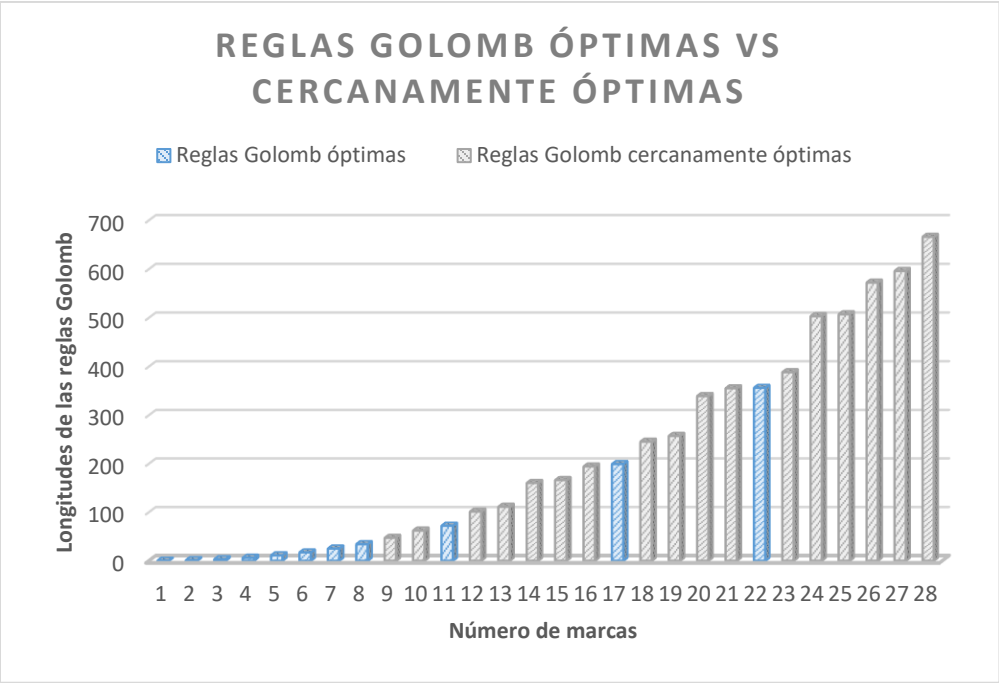


Figura 58. Reglas Golomb óptimas Vs cercanamente óptimas.

Fuente: Elaboración propia.

También se puede apreciar en la figura 59 la distribución porcentual de las reglas Golomb óptimas y cercanamente óptimas por cada una de las marcas estudiadas. La propuesta de investigación logra encontrar reglas Golomb cercanamente óptimas para 17 marcas de las 28 marcas estudiadas que representan aproximadamente el 61% de las marcas trabajadas y para el caso de las reglas Golomb óptimas, la propuesta logra encontrar 11 marcas que representan aproximadamente el 39% de las reglas trabajadas.

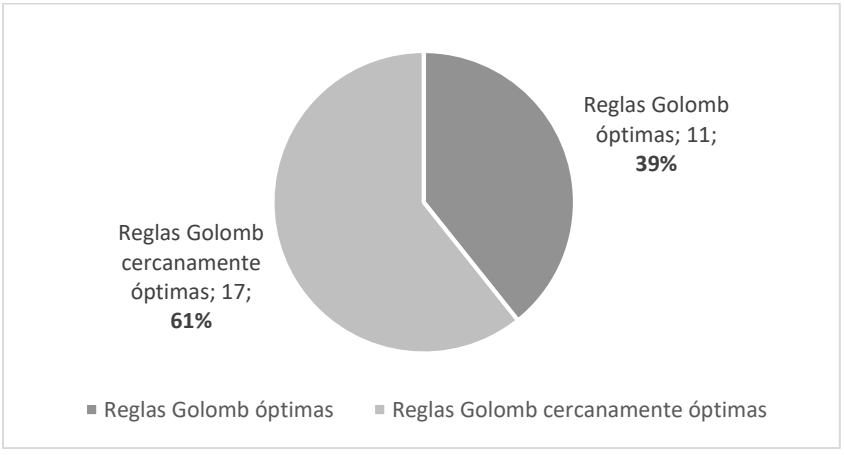


Figura 59. Distribución porcentual de reglas Golomb por número de marcas encontradas.

Fuente: Elaboración propia.

Por otro lado, se puede evidenciar que el algoritmo memético logra encontrar reglas Golomb óptimas para un determinado número de marcas que no estaban identificadas en la literatura generando de esta manera, nuevo conocimiento en el área de investigación. Estas

reglas conservan la longitud óptima conocida, pero cuentan con una combinación diferente de marcas y serán presentadas en la tabla 25:

Tabla 25**.** Nuevas reglas Golomb óptimas.

M	L_o	Regla Golomb óptima no identificada por la literatura	Regla Golomb óptima conocida
3	3	[0, 2, 3]	[0, 1, 3]
4	6	[0, 2, 5, 6]	[0, 1, 4, 6]
5	11	[0, 2, 7, 10, 11]	[0, 1, 4, 9, 11] [0, 2, 7, 8, 11] [0, 3, 4, 9, 11]
6	17	[0, 5, 7, 13, 16, 17] [0, 4, 6, 9, 16, 17] [0, 2, 7, 13, 16, 17] [0, 3, 5, 9, 16, 17]	[0, 1, 8, 12, 14, 17] [0, 1, 4, 10, 15, 17] [0, 1, 4, 10, 12, 17] [0, 1, 8, 11, 13, 17]
7	25	[0, 2, 6, 9, 14, 24, 25] [0, 4, 9, 15, 22, 23, 25] [0, 2, 5, 14, 18, 24, 25] [0, 2, 7, 15, 21, 24, 25] [0, 3, 4, 12, 18, 23, 25]	[0, 1, 7, 11, 20, 23, 25] [0, 1, 4, 10, 18, 23, 25] [0, 1, 11, 16, 19, 23, 25] [0, 2, 3, 10, 16, 21, 25] [0, 2, 7, 13, 21, 22, 25]
17	199	[0, 8, 31, 34, 40, 61, 77, 99, 118, 119, 132, 143, 147, 182, 192, 194, 199]	[0, 5, 7, 17, 52, 56, 67, 80, 81, 100, 122, 138, 159, 165, 168, 191, 199]
22	356	[0, 3, 15, 26, 65, 86, 93, 103, 133, 152, 177, 197, 228, 232, 234, 250, 286, 313, 342, 347, 355, 356]	[0, 1, 9, 14, 43, 70, 106, 122, 124, 128, 159, 179, 204, 223, 253, 263, 270, 291, 330, 341, 353, 356]

De esta manera, se comprueba que el algoritmo memético propuesto logra encontrar reglas Golomb cercanamente óptimas hasta 28 marcas y también logra encontrar reglas Golomb óptimas que no habían sido identificadas por la literatura hasta el anteriormente.

Capítulo 6. Conclusiones y trabajos futuros.

En este capítulo se presentarán los resultados finales del trabajo de investigación distribuidos en dos secciones; en la primera sección se presenta un análisis del cumplimiento de los objetivos de investigación y en la segunda sección, se presentan las conclusiones y trabajos futuros de la investigación.

6.1. Análisis de los objetivos de investigación.

A continuación, se presenta el análisis de cada uno de los objetivos de investigación relacionando los capítulos donde se evidencia el cumplimiento del trabajo realizado.

6.1.1. Objetivos específicos.

- **OE1**. Caracterizar las metaheurísticas sugeridas por la comunidad científica que encuentran reglas Golomb cercanamente óptimas por medio de un mapeo sistemático de la literatura.
  Para dar cumplimiento a este objetivo, se realizó un mapeo sistemático acerca de las metaheurísticas que encuentran reglas Golomb óptimas o cercanamente óptimas en la literatura (Ver capítulo 2) con el fin de recopilar y categorizar las investigaciones existentes en el área de investigación [15, 16, 17].
- **OE2**. Implementar un algoritmo memético que integre las construcciones Bose, Singer y Ruzsa, así como traslaciones y cotas matemáticas para encontrar reglas Golomb cercanamente óptimas.
  A partir de los resultados obtenidos en el objetivo específico anterior, se identificaron las brechas existentes en el área de investigación y se plantea una propuesta que involucre conceptos matemáticos para resolver el problema de investigación. Dichos conceptos son presentados en el capítulo 3 de manera detallada. Posteriormente, en el capítulo 4 se define el algoritmo memético integrando los conceptos matemáticos de Bose, Singer, Ruzsa, traslaciones y cotas para encontrar reglas Golomb cercanamente óptimas. De esta manera, se garantiza el cumplimiento del OE2.
- **OE3.** Evaluar los resultados del algoritmo memético propuesto mediante la comparación de las soluciones encontradas contra las 28 reglas Golomb óptimas conocidas para determinar si son cercanamente óptimas.
  En el capítulo 5, se presentan las reglas Golomb obtenidas por el algoritmo memético propuesto y posteriormente se comparan con las 28 reglas Golomb óptimas conocidas hasta la fecha para determinar si los resultados son cercanamente óptimos.

6.1.2. Objetivo general.

OG. Proponer un algoritmo memético para encontrar reglas Golomb cercanamente óptimas hasta 28 marcas incorporando construcciones algebraicas, traslaciones y cotas.

Al cumplir con los 3 objetivos específicos planteados anteriormente, se cumple con el objetivo general de manera correcta. En el capítulo 4 se presenta el algoritmo memético

propuesto incorporando los conceptos de Bose, Singer, Ruzsa, traslaciones y cotas para encontrar reglas Golomb cercanamente óptimas.

6.2. Conclusiones.

- El mapeo sistemático realizado en este proyecto de investigación permitió identificar los trabajos relacionados al problema de investigación que sirvieron como base para el desarrollo del algoritmo memético propuesto ya que propusieron temáticas de gran valor como: (i) evidencia de brechas existentes en el área de investigación y (ii) muestra los resultados alcanzados por la literatura hasta la fecha para resolver el problema de investigación.
- Se definió la base conceptual matemática con la que trabajo el algoritmo memético propuesto para encontrar reglas Golomb cercanamente óptimas: (i) Construcciones algebraicas conocidas como Bose, Singer y Rusza que abordan el problema de investigación desde una perspectiva matemática, (ii) Concepto de Traslaciones que sirve para mejorar la longitud de las reglas Golomb y (iii) Propiedad matemática llamada Cotas (Ver sección 3.4).
- El algoritmo memético propuesto es un algoritmo de optimización que busca resolver el problema de investigación presentado en [15, 16, 17] encontrando reglas Golomb óptimas y cercanamente óptimas hasta 28 marcas. Adicionalmente, el algoritmo logra integrar en su funcionamiento interno los conceptos matemáticos presentados en la base conceptual y también utiliza una serie de parámetros de entrada con valores dinámicos que dependerán del número de marcas de la regla que deseamos encontrar con el objetivo de encontrar reglas Golomb cercanamente óptimas. Este algoritmo puede ser de gran utilidad para la comunidad científica que esté interesada en abordar el problema de investigación ya que podrá generar reglas Golomb cercanamente óptimas para cualquier número de marcas.
- El algoritmo memético inicialmente se ejecutó 100 veces en cada una de las 28 marcas estudiadas para encontrar reglas Golomb cercanamente óptimas y posteriormente los resultados de la propuesta de investigación fueron evaluados con las 28 reglas Golomb óptimas conocidas hasta la fecha para determinar si los resultados obtenidos por el algoritmo son cercanamente óptimos.
- Consideramos que el algoritmo memético propuesto es un gran aporte en el área de investigación debido a que, según los resultados alcanzados por el estado del arte no se evidencian estudios o metaheurísticas que implementen métodos o técnicas algebraicas para abordar el problema desde su naturaleza matemática. También se superó el número de 21 marcas cercanamente óptimas reportadas por el estado del arte logrando encontrar hasta 28 marcas cercanamente óptimas. Adicionalmente, a pesar de haber tenido limitaciones con los recursos computacionales del equipo de desarrollo, el algoritmo memético logra encontrar reglas Golomb óptimas para 3, 4, 5, 6, 7, 17 y 22 marcas que no habían sido identificadas por la literatura hasta la fecha actual.
- Teniendo en cuenta la pregunta de investigación planteada en la sección 1.1: ¿Cómo encontrar reglas Golomb cercanamente óptimas hasta 28 marcas, mediante un algoritmo memético incorporando conceptos y métodos matemáticos?, se obtuvo como resultado la implementación de un algoritmo memético que incorpora los conceptos matemáticos de Bose, Singer, Ruzsa, traslaciones y cotas presentado en

el capítulo 4 en donde se busca aprovechar el buen rendimiento de los algoritmos meméticos en resolver problemas de optimización y el conocimiento de métodos algebraicos que abordan el problema desde su naturaleza matemática para encontrar reglas Golomb hasta 28 marcas.

6.3. Trabajos futuros.

a. **Actualización del mapeo sistemático:** teniendo en cuenta que el mapeo sistemático acerca de las metaheurísticas que encuentran reglas Golomb óptimas o cercanamente óptimas se realizó a inicios de la investigación, se sugiera realizar una actualización con el fin de identificar nuevos trabajos que proponen metaheurísticas para abordar el problema de investigación.
b. **Realizar la evaluación de la propuesta hasta 150 marcas:** el algoritmo memético propuesto fue evaluado mediante la comparación con las 28 reglas Golomb óptimas conocidas [Anexo 2], sin embargo, es pertinente evaluar la propuesta de investigación con las 150 reglas Golomb cercanamente óptimas conocidas hasta la fecha [76] para identificar aspectos que contribuyan en la mejora de la propuesta.
c. **Crear una nueva versión del algoritmo memético para trabajar con reglas g-Golomb [54]:** La propuesta de investigación actual se puede adaptar para crear una nueva versión del algoritmo memético en donde se aborde el problema de investigación de reglas g-Golomb planteado en [54] para encontrar reglas cercanamente óptimas con un número determinado de diferencias.

Referencias bibliográficas

[1] W. C. Babcock, “Intermodulation interference in Radio Systems,” Bell System Technical Journal, vol. 32, no. 1, pp. 63–73, 1953.

[2] G. S. Bloom and S. W. Golomb, “Applications of numbered undirected graphs,” Proceedings of the IEEE, vol. 65, no. 4, pp. 562–570, Apr. 1977.

[3] A. Dimitromanolakis, ‘‘Analysis of the Golomb ruler and the Sidon set problems, and determination of large, near-óptimal Golomb rulers,’’ M.S. thesis, Dept. Electron. Comput. Eng., Tech. Univ. Crete, Chania, Greece, Jun. 2002.

[4] A. Dollas, W. T. Rankin and D. McCracken, "A new algorithm for Golomb ruler derivation and proof of the 19 mark ruler," in IEEE Transactions on Information Theory, vol. 44, no. 1, pp. 379-382, Jan. 1998.

[5] R. Gagliardi, J. Robbins, and H. Taylor, “Acquisition sequences in PPM Communications (Corresp.),” IEEE Transactions on Information Theory, vol. 33, no. 5, pp. 738–744, 1987.

[6] M. Atkinson, N. Santoro, and J. Urrutia, “Integer sets with distinct sums and differences and carrier frequency assignments for nonlinear repeaters,” IEEE Transactions on Communications, vol. 34, no. 6, pp. 614–617, 1986.

[7] J.-G. Zhang and A. B. Sharma, “Notes on use of strict optical orthogonal codes to design unequal channel spacing (UCS) frequency sequences for DWDM systems with reduced FWM Crosstalk,” Optics Communications, vol. 281, no. 22, pp. 5574–5579, 2008.

[8] P. Lavoie, D. Haccoun, and Y. Savaria, “New VLSI architectures for fast soft-decision threshold decoders,” IEEE Transactions on Communications, vol. 39, no. 2, pp. 200–207, 1991.

[9] J. Robinson and A. Bernstein, “A class of binary recurrent codes with limited error propagation,” IEEE Transactions on Information Theory, vol. 13, no. 1, pp. 106–113, 1967.

[10] F. Biraud, E. Blum, and J. Ribes, “On optimum synthetic linear arrays with application to radioastronomy,” IEEE Transactions on Antennas and Propagation, vol. 22, no. 1, pp. 108–109, 1974.

[11] E. J. Blum, J. C. Ribes and F. Biraud, "Some new possibilities of optimum synthetic linear arrays for radioastronomy", Astronomy and Astrophysics, vol. 41, no. 3-4, pp. 409-411, July, 1975.

[12] N. Memarsadeghi, R. D. Joseph, J. C. Kaufmann, and B. S. Lee, “Golomb patterns, astrophysics, and citizen science games,” IEEE Access, vol. 10, pp. 76125–76135, 2022.

[13] NASA Computational Case Study: Golomb Rulers and Their Applications: https://encompass.gsfc.nasa.gov/caseStudies/golomb.pdf

[14] Project educational NASA computational and scientific studies (enCOMPASS): https://encompass.gsfc.nasa.gov/cases.html

[15] M. Sorge, “Algorithmic Aspects of Golomb Ruler Construction,” B.S. Thesis, Dept. Comput, Univ. Friedrich Schiller, Germany, 2010.

[16] C. Meyer and P. A. Papakonstantinou, “On the complexity of constructing Golomb rulers,” Discrete Applied Mathematics, vol. 157, no. 4, pp. 738–748, 2009.

[17] S. W. Soliday, A. Homaifar & G. L. Lebby, "Genetic Algorithm Approach to the Search for Golomb Rulers", In ICGA, pp. 528-535, July, 1995.

[18] W. Rankin, ‘‘Óptimal Golomb rulers: An exhaustive parallel search implementation,’’ M.S. thesis, Dept. Electr. Eng., Duke Univ., Durham, NC, USA, Dec. 1993.

[19] Distributed.net, Project OGR. http://www.distributed.net/ogr

[20] Distributed.net, OGR-28 Completion: https://blogs.distributed.net/2022/11/23/03/28/bovine/

[21] P. Erdös and P. Turán, “On a problem of sidon in additive number theory, and on some related problems,” Journal of the London Mathematical Society, vol. s1-16, no. 4, pp. 212–215, 1941.

[22] R. C. Bose and S. Chowla, “Theorems in the additive theory of Numbers,” Commentarii Mathematici Helvetici, vol. 37, no. 1, pp. 141–147, 1962.

[23] J. Singer, "A Theorem in Finite Projective Geometry and Some Applications to Number Theory", Transactions of the American Mathematical Society, vol. 43, no. 3, pp. 377-385, 1938.

[24] I. Z. Ruzsa, "Solving a linear equation in a set of integers I.", Acta arithmetica, vol. 65, no. 3, pp. 259-282, 1993.

[25] B. Keshanchi and N. J. Navimipour, “Priority-based task scheduling in the Cloud Systems using a memetic algorithm,” Journal of Circuits, Systems and Computers, vol. 25, no. 10, p. 1650119, May 2016.

[26] Z. Zhang, Z. Li, X. Qiao, and W. Wang, “An efficient memetic algorithm for the minimum load coloring problem,” Mathematics, vol. 7, no. 5, p. 475, May 2019.

[27] C.-K. Ting and C.-C. Liao, “A memetic algorithm for extending Wireless Sensor Network Lifetime,” Information Sciences, vol. 180, no. 24, pp. 4818–4833, 2010.

[28] K. S. Pratt, "Design Patterns for Research Methods: Iterative Field Research", In AAAI Spring Symposium: Experimental Design for Real, No. 1994, pp. 1-7, 2009.

[29] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, “Systematic mapping studies in software engineering,” Electronic Workshops in Computing, 2008.

[30] B. Kitchenham, O. Pearl Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, “Systematic literature reviews in software engineering – A systematic literature review,” Information and Software Technology, vol. 51, no. 1, pp. 7–15, 2009.

[31] S. Bansal, “Óptimal Golomb ruler sequence generation for FWM crosstalk elimination: Soft Computing versus conventional approaches,” Applied Soft Computing, vol. 22, pp. 443–457, 2014.

[32] S. Bansal, “Nature-inspired hybrid multi-objective optimization algorithms in search of near-ogrs to eliminate FWM noise signals in optical WDM systems and their performance comparison,” Journal of The Institution of Engineers (India): Series B, vol. 102, no. 4, pp. 743–769, 2021.

[33] J. Vyas, S. Bansal, and K. Sharma, “Generation of óptimal golomb rulers for FWM crosstalk reduction: BB-BC and FA approaches,” 2016 International Conference on Signal Processing and Communication (ICSC), 2016.

[34] S. Bansal, A. K. Singh, and N. Gupta, “Optimal Golomb ruler Sequences Generation for optical WDM systems: A novel parallel hybrid multi-objective bat algorithm,” Journal of The Institution of Engineers (India): Series B, vol. 98, no. 1, pp. 43–64, 2016.

[35] S. Bansal, N. Gupta, and A. K. Singh, “Multi-objective flower pollination algorithm and its variants to find optimal Golomb rulers for WDM Systems,” Springer Tracts in Nature-Inspired Computing, pp. 171–196, 2021.

[36] S. Bansal, N. Gupta, and A. K. Singh, “Nature–inspired metaheuristic algorithms to find near–OGR sequences for WDM channel allocation and their performance comparison,” Open Mathematics, vol. 15, no. 1, pp. 520–547, 2017.

[37] A. S. C. Modesto, R. M. Figueiredo, C. S. Ramos, L. Santos, E. Venson, G.V. Pedrosa, “Organizational strategies for end-user development—a systematic literature mapping”, Informatics 2021, vol. 8, pp.15, 2021.

[38] Scopus preview - Scopus - Welcome to Scopus, https://www.scopus.com/ (accessed Feb. 4, 2024).

[39] Sciencedirect.com | Science, Health and medical journals, https://www.sciencedirect.com/ (accessed Feb. 4, 2024).

[40] Access Research Journals, articles, books and more | springerlink, Springer Link, https://link.springer.com/ (accessed Feb. 4, 2024).

[41] Google Scholar, https://scholar.google.com/ (accessed Feb. 4, 2024).

[42] IEEE Xplore: https://ieeexplore.ieee.org/ (accessed Feb. 4, 2024).

[43] ACM Digital Library: https://www.acm.org/publications/digital-library (accessed Feb. 4, 2024).

[44] R. R., S. Bansal, and S. Sharma, “A novel Bat Algorithm for channel allocation to reduce FWM crosstalk in WDM Systems,” International Journal of Computer Applications, vol. 136, no. 4, pp. 33–42, 2016.

[45] S. Bansal, P. Jain, A. K. Singh and N. Gupta, "Improved Multi–Objective Firefly Algorithms to Find Óptimal Golomb Ruler Sequences for Óptimal Golomb Ruler Channel Allocation", International Journal of Physical and Mathematical Sciences, vol. 10, no. 7, pp. 350-357, 2016.

[46] S. Bali, S. Bansal, and A. Kamboj, “A novel hybrid multi–objective BB–BC based channel allocation algorithm to reduce FWM crosstalk and its comparative study,” International Journal of Computer Applications, vol. 124, no. 12, pp. 38–45, 2015.

[47] M. M. Polash, M. A. Newton, and A. Sattar, “Constraint-based local search for Golomb rulers,” Integration of AI and OR Techniques in Constraint Programming, pp. 322–331, 2015.

[48] M. M. Polash, M. A. Newton, and A. Sattar, “Constraint-based search for óptimal Golomb rulers,” Journal of Heuristics, vol. 23, no. 6, pp. 501–532, 2017.

[49] P. K. Singh and N. Gupta, "A wind driven optimization based WDM channel allocation algorithm", Int J Innov Res Sci Technol, vol. 5, no. 7, pp. 12073-12080, 2016.

[50] M. Kowol, K. Pietak, M. Kisiel-Dorohinicki, and A. Byrski, “Agent-based evolutionary and memetic black-box discrete optimization,” Procedia Computer Science, vol. 108, pp. 907–916, 2017.

[51] D. H. Wolpert and W. G. Macready, “No free lunch theorems for optimization,” IEEE Transactions on Evolutionary Computation, vol. 1, no. 1, pp. 67–82, 1997.

[52] L. Araujo, "How evolutionary algorithms are applied to statistical natural language processing," Artificial Intelligence Review, vol. 28, pp. 275-303, 2007.

[53] J. Brownlee, "Clever algorithms: nature-inspired programming recipes", vol. 436, pp. 454

[54] C. A. Martos, “CONJUNTOS Bh Y REGLAS g-GOLOMB CORTAS”, Ph.D. Thesis, Departamento de Matemáticas, Universidad del Valle, Cali, Colombia, 2019.

[55] C. A. Martos, “REGLAS g-GOLOMB”, M.S. thesis, Departamento de Matemáticas, Universidad del Cauca, Cali, Colombia, 2015.

[56] W. F. Osgood, & W. C. Graustein, “Plane and solid analytic geometry”, The Macmillan Company, p. 330, 1922.

[57] R. Kwiatek & G. Zwara, “The divisibility of integers and integer relatively primes”, Formalized Mathematics, vol. 1, no. 5, pp. 829-832, 1990.

[58] L. Huang et al., “Normalization techniques in training dnns: Methodology, analysis and Application,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 45, no. 8, pp. 10173–10196, Aug. 2023. doi:10.1109/tpami.2023.3250241.

[59] M. Sanz, “POLINOMIOS IRREDUCIBLES SOBRE CUERPOS FINITIOS”, B.Sc, Thesis, Facultad de ciencias, Universidad de Cantabria, Cantabria, España, Jun. 2021.

[60] R. Lidl, H. Niederreiter, “Introduction to finite fields and their applications”, Cambridge University Press, 1986.

[61] G.L. Mullen, D. Panario, “Handbook of finite fields”, CRC Press, 2013.

[62] E. Savas and C. Koc, “Finite Field Arithmetic for Cryptography,” IEEE Circuits and Systems Magazine, vol. 10, no. 2, pp. 40–56, 2010. doi:10.1109/mcas.2010.936785.

[63] M. I. Hidayat, Irwansyah and I. G. Wardhana, “A construction of generalized quasi-cyclic codes over finite field using gray map,” 7TH INTERNATIONAL CONFERENCE ON MATHEMATICS: PURE, APPLIED AND COMPUTATION: Mathematics of Quantum Computing, Dec. 2022. doi:10.1063/5.0114988.

[64] L. K. Hua and H. S. Vandiver, “Characters over certain types of rings with applications to the theory of equations in a finite field,” Proceedings of the National Academy of Sciences, vol. 35, no. 2, pp. 94–99, 1949. doi:10.1073/pnas.35.2.94.

[65] L. Gianfagna & A. D. Cecco, “Explainable AI with Python”, Cham: Springer, 2021.

[66] DEAP documentation - DEAP 1.4.1 documentation, https://deap.readthedocs.io/en/master/index.html (accessed Feb. 4, 2024).

[67] BASIC usage - Galois documentation, https://galois.readthedocs.io/en/v0.0.21/basic-usage.html (accessed Feb. 4, 2024).

[68] SymPy, https://www.sympy.org/en/index.html (accessed Feb. 4, 2024).

[69] P. Currion, C. de Silva, & B. Van de Walle, “Open source software for disaster management,” Communications of the ACM, vol. 50, no. 3, pp. 61–65, 2007. doi:10.1145/1226736.1226768.

[70] DEAP License, https://github.com/deap/deap?tab=LGPL-3.0-1-ov-file#readme (accessed Feb. 4, 2024).

[71] Galois License, https://github.com/mhostetter/galois?tab=MIT-1-ov-file#readme (accessed Feb. 4, 2024).

[72] F. Neri and C. Cotta, “Memetic algorithms and Memetic Computing Optimization: A literature review”, Swarm and Evolutionary Computation, vol. 2, pp. 1–14, Feb. 2012. doi:10.1016/j.swevo.2011.11.003.

[73] A. E. Eiben and J. E. Smith, “Introduction to Evolutionary Computing”, Heidelberg: Springer, 2016.

[74] DEAP documentation, deap.tools.selStochasticUniversalSampling, https://deap.readthedocs.io/en/master/api/tools.html#deap.tools.selStochasticUniversalSampling (accessed Feb. 19, 2024).

[75] DEAP documentation, Source code for selStochasticUniversalSampling, https://deap.readthedocs.io/en/master/_modules/deap/tools/selection.html#selStochasticUniversalSampling (accessed Feb. 19, 2024).

[76] Table of lengths of shortest known Golomb rulers, https://web.archive.org/web/20180416223506/http://www.research.ibm.com/people/s/shearer/grtab.html (accessed Feb. 19, 2024).

[77] K. Hussain, M. N. Mohd Salleh, S. Cheng, and Y. Shi, “Metaheuristic Research: A comprehensive survey”, Artificial Intelligence Review, vol. 52, no. 4, pp. 2191–2233, Jan. 2018. doi:10.1007/s10462-017-9605-z

Anexos.

Anexo 1. Repositorio con el código fuente del algoritmo memético:

Anexo 2. Reglas Golomb óptimas conocidas.