PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

Tesis para obtener el título profesional de Ingeniero Informático

IMPLEMENTACIÓN DE UN ALGORITMO MEMÉTICO PARA LA DISTRIBUCIÓN DE ANTENAS WIFI EN ALMACENES DE GRANDES DIMENSIONES

AUTOR:

Rafael Jair Burgos Chuqui

ASESOR:

Asesor: Mg. Rony Cueva Moscoso

Lima, Febrero, 2024

INFORME DE SIMILITUD

Yo, …Rony Cueva
Moscoso……..……,
docente de la Facultad de ……Ciencias e Ingeniería……
de la Pontificia Universidad Católica del Perú, asesor(a) de la tesis/el trabajo de investigación titulado
…… IMPLEMENTACIÓN DE UN ALGORITMO MEMÉTICO PARA LA DISTRIBUCIÓN DE ANTENAS WIFI EN ALMACENES DE GRANDES
DIMENSIONES……..,
del/de la autor(a)/ de los(as) autores(as)
… Rafael Jair Burgos Chuqui…….,
…..……..……,
…..……..……,
dejo constancia de lo siguiente:

- El mencionado documento tiene un índice de puntuación de similitud de 13%. Así lo consigna el reporte de similitud emitido por el software *Turnitin* el 22/01/2024.
- He revisado con detalle dicho reporte y la Tesis o Trabajo de Suficiencia Profesional, y no se advierte indicios de plagio.
- Las citas a otros autores y sus respectivas referencias cumplen con las pautas académicas.

Lugar y fecha: …Lima, 29 de enero del
2024……

Apellidos y nombres del asesor / de la asesora: Cueva Moscoso Rony	
DNI: 09942265	Firma
ORCID: 0000-0003-4861-571X	

RESUMEN

En la actualidad, es indispensable tener un acceso rápido a la información en los almacenes con el fin de atender de forma óptima a las demandas del mercado. Esto es importante, porque el usar un almacén de empacado es necesario para tener un control del inventario, al mismo tiempo que se incrementa la productividad y se ahorra dinero al permitir que los clientes reciban sus productos a tiempo. Así como permitir una distribución de productos más eficientemente e incluso saber cuándo ya no se cuenta con stock.

Sin embargo, el uso de almacenes tiene que ser complementado con una efectiva comunicación entre los trabajadores. Por tanto, el uso de redes Wi-Fi en almacenes tiene el potencial de permitir la rápida automatización de procesos intensivos tales como la recepción, el desembarco, almacenamiento, conteo de órdenes, recogidas y empaquetamientos.

Debido a las razones mencionadas, se necesita tener una manera organizada de planificar el posicionamiento de antenas con el fin de evitar costos innecesarios y de aprovechar el mayor espacio posible.

Hay implementaciones con algoritmos clásicos para problemáticas similares, pero se desea aprovechar la posibilidad de aplicar conceptos más modernos para afrontar esta situación.

Es por esto que se plantea como objetivo el implementar un algoritmo memético para la optimización del posicionamiento de antenas Wi-Fi en un almacén rectangular de grandes dimensiones y los objetivos adicionales serán el apoyo para el desarrollo del proyecto.

El proyecto de tesis pertenece al tema de algoritmos de optimización (que a su vez es perteneciente al área de ciencias de la computación).

La solución permitirá determinar un conjunto de posiciones óptimas con el número de antenas Wi-Fi adecuadas para poder mejorar el uso de recursos sin perder señal.

Es posible que se tomen consideraciones respecto a la complejidad de la infraestructura del almacén debido a que no todos los almacenes poseen la misma. Asimismo, la atenuación de la señal inalámbrica en las paredes del almacén planteado y el cambio energético en la intensidad de las antenas son aspectos que no serán considerados ya que se encuentran más relacionados al área de Ingeniería Electrónica.

Con todo lo anterior mencionado, se busca verificar si realmente la implementación presentada logrará mejorar los resultados obtenidos mediante una implementación realizada con el algoritmo genético, que representa la implementación clásica de una solución a la problemática planteada.

DEDICATORIA

Durante el desarrollo del presente trabajo, se tuvo que realizar numerosas ejecuciones para las secciones de calibración de variables y experimentación numérica. A esto se le suma las ejecuciones adicionales por cualquier error cometido en el ingreso de datos. Es por esto, que se requirió el apoyo de distintos amigos que apoyaron mediante ejecuciones en sus computadoras y esto permitió finalizar el proyecto en el tiempo establecido y/o ayudando a relajar la tensión y el estrés mediante actividades recreativas durante el proyecto y toda la etapa universitaria. Siendo algunos de ellos: Jean Patrick "Patroclo" Sanchez, Luis "Ruiz" Hernandez, Luis "Panda" Moscoso, Mauricio "Maotzetung" Avila y Sebastián "Barney" Nuñez.

Del mismo modo, agradezco a mis padres Walter R. Burgos y Rosa Chuqui quienes me formaron para convertirme en la persona que soy el día de hoy durante sus vidas, a mis abuelos y abuelas Julia Freyre, Elena Fernandez y Walter E. Burgos quienes también fueron parte importante de mi formación, a mi hermana Tatiana "Tati" Burgos por apoyarme no solo con las ejecuciones sino también de muchas otras maneras durante el desarrollo del proyecto y durante mi vida en general estando siempre a mi lado. De manera similar, a mi buen amigo Daniel "Dani Craig" Zedano por prestarme su laptop de manera seguida, ser muy servicial y siempre brindar su apoyo y a mis amigos Bruno "Ipad" Laurente y José "Filo" Pajares quienes siempre tenían mucha iniciativa y fueron de apoyo durante mis años en la universidad.

En adición, agradezco a mis amigos Jorge Baca, Diego Ramirez, Paolo Patiño, Luis Rodrigo Fernandez y a todo el grupo "MidDevs" con quienes he compartido muchos eventos a lo largo de la carrera.

Asimismo, es necesario dar agradecimiento a mi padrino Sandro Burgos y su familia, así como a los amigos de mis padres (siendo algunos de ellos Sergio Segura, Dunia Dextre, Luis Reyes, Luis Fernandez, Lourdes Pilco, entre otros) y sus familias, a la Sra. Carmen Valenzuela y al Sr. Juan quienes siempre fueron de apoyo en las situaciones más inesperadas, a mis tías Lourdes y Liliana Freyre así como a la familia de mi abuela Julia Freyre. A mis tías Martha Cabrera, Azalia Chavez, Zoila Maco y a toda la familia de parte de mi padre. También a mis tíos familiares de mi madre, entre los cuales se encuentran: Jose Chuqui, Socorro Cabanillas, Onias Chuqui, Gil Chuqui, entre otros. De la misma manera, a todos mis primos, sobrinos y a toda mi familia de parte de mi madre en los departamentos de San Martín y Amazonas quienes siempre tuvieron la disposición de ayudar en todo momento.

A mi profesor asesor Mg. Rony Cueva por orientarme y guiarme para completar el proyecto, a todos mis profesores y jefes de practica quienes me corrigieron y formaron durante este viaje y a la universidad por darme las oportunidades necesarias para poder finalizar mi carrera y escribir este párrafo.

¡Muchas gracias a todos!

TABLA DE CONTENIDOS

INDICE DE TABLAS

INDICE DE FIGURAS

1. Problemática

1.1. Introducción

A continuación, se procede a detallar respecto a la problemática que busca dejar en evidencia el entorno, las causas y efectos relacionados al desarrollo generará este proyecto. Mediante el desarrollo de la solución propuesta se busca dejar conocimiento que pueda ser usado en el entorno actual por diversas entidades en sus almacenes.

1.1.1. Árbol de problemas

Se procede a presentar el árbol de problemas correspondiente al tema del proyecto en la Figura 1.

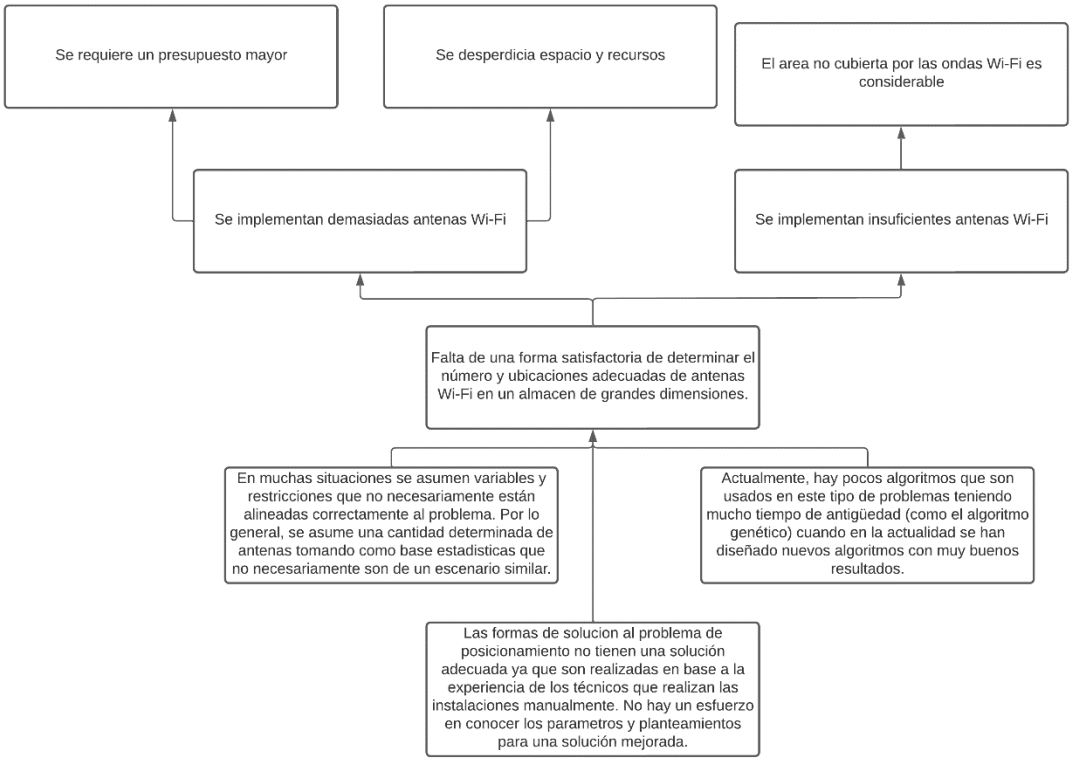


Figura 1. Árbol de problemas

1.1.2. Descripción de la problemática

En la actualidad las compañías buscan tener acceso rápido y preciso a diversos tipos de información necesario para atender las demandas del mercado para lo cual es necesaria una solución que agilice las operaciones y una manera de controlar los productos que manejan en conjunto con el uso de un almacén.

En diversas empresas de gestión de bienes, el usar un almacén de empacado (packing warehouse) es necesario para tener un control del inventario al mismo tiempo que se incrementa la productividad y se ahorra dinero al permitir que los clientes reciban sus productos a tiempo. Al tener un control del inventario se puede distribuir productos más eficientemente e incluso saber cuándo ya no se cuenta con stock, de modo que se pueden tomar decisiones operativas al respecto (Picincu, 2019). Sin embargo, el uso de almacenes tiene que ser complementado con una efectiva comunicación entre los trabajadores.

Por tanto, el uso de redes Wi-Fi en almacenes tiene el potencial de permitir la rápida automatización de procesos intensivos tales como la recepción, el desembarco, almacenamiento, conteo de órdenes, recogidas y empaquetamientos. Asimismo, pueden permitir un buen rendimiento en ambientes difíciles de almacenamiento a bajas temperaturas como refrigeradoras y congeladoras (*Why Is Warehouse WiFi So Important?*, 2020). Pero desgraciadamente, los emisores de señal Wi-Fi suelen ser puestos de maneras que se consideran adecuadas solamente a criterio del personal instalador y esto ocasiona que se coloquen más de lo necesario e incluso es posible que se deje un área sin cobertura debido a que es posible que se agrupen los emisores en zonas muy juntas y la señal estaría concentrada en una porción del terreno (Bhuwania et al., 2016).

La adecuada planificación del posicionamiento de antenas emisoras permitiría que los empleados de una zona amplia se mantengan comunicados o conectados a las aplicaciones que pertenezcan a la empresa donde trabajan en todo momento (evitar zonas sin cobertura). Adicionalmente, se evitaría gastar recursos innecesarios que son adicionales a las propias antenas (como instalaciones y compras) y abarcar la mayor cantidad de área posible reduciendo la atenuación entre paredes (debido a que las ondas de radio no atraviesan todos materiales de la misma manera) para evitar zonas sin conexión (Ting & Liao, 2010).

Por lo general, se suelen usar algoritmos populares, como implementaciones del algoritmo genético, teniendo algunos de ellos muchos años de uso con resultados que pueden mejorarse.

Este algoritmo tiene diversos problemas siendo el más notable el tener cierta tendencia a converger alrededor de un óptimo local (Taherdangkoo et al., 2013), siendo este término la denominación de una solución en la que los resultados obtenidos solo se quedan cerca de un resultado que aparenta ser el mejor de todo un conjunto de soluciones pero que no necesariamente lo es.

Asimismo, también es posible considerar estadísticas, variables o restricciones que no necesariamente son similares a este problema, debido a que pueden parecer útiles, pero al momento de analizar no aportan mucho a la solución. Esto es particularmente notable ya que durante el desarrollo de este documento se han visualizado algunos documentos que aparentan abarcar el problema de optimización de cobertura pero abarcan problemas cuyas variables y restricciones están orientados a la cobertura de un mismo objetivo por diferentes antenas (Sahoo & Sahoo, 2020).

Con todo lo anterior mencionado, en la búsqueda de evitar disponer de antenas innecesarias, además de otros recursos desperdiciados. El enfoque de este proyecto será la búsqueda de una forma de ubicarlas de manera adecuada para alcanzar la eficacia y eficiencia deseada.

1.1.3. Objetivos

En el presente subtítulo se procede a describir el objetivo general y los objetivos específicos que se buscan cumplir en este proyecto para afrontar la problemática.

1.1.3.1. Objetivo general: Implementar un algoritmo memético para la optimización del posicionamiento de antenas Wi-Fi en un almacén rectangular de grandes dimensiones.

1.1.3.2. Objetivos específicos:

- **O1:** Determinar los parámetros y restricciones que serán utilizados en la implementación del algoritmo aplicado al problema.
- **O2:** Adaptar un algoritmo memético para optimizar la cobertura en el área del almacén en general.
- **O3:** Comparar el algoritmo implementado con otro algoritmo de implementación similar de gran uso mediante una experimentación numérica.

1.1.4. Resultados esperados

Se definirá los resultados que se espera obtener mediante este proyecto. Asimismo, también se muestra qué objetivo se busca satisfacer.

- **R1:** Definición de los parámetros, variables y restricciones orientados a la problemática del problema. Relacionado con O1.
- **R2:** Definición de una función objetivo que permita la representación del grado de optimización alcanzado. Relacionado con O1.
- **R3:** Definición de funciones, métodos y clases a implementar para ser usadas con el algoritmo memético. Relacionado con O2.
- **R4:** Diseño y adaptación de un algoritmo memético que pueda resolver el problema propuesto. Relacionado con O2.
- **R5:** Adaptación de un algoritmo de gran uso (equivalente o similar) que permita resolver el problema propuesto (o problemas similares). Relacionado con O3.
- **R6:** Desarrollo de las pruebas de ambos algoritmos para encontrar errores y probar distintas configuraciones.
- **R7:** Diseño y desarrollo de la experimentación numérica que permita comparar el rendimiento del algoritmo propuesto con soluciones similares. Relacionado con O3.

1.1.5. Medios de verificación

En la tabla 1.1 se procede a relacionar los resultados esperados con los objetivos planteados especificando un medio de verificación con un indicador objetivamente verificable.

Tabla 1.1 Tabla de objetivos y resultados

Objetivo	Resultado	Medio de verificación	Indicador Objetivamente Verificable
O1	**R1:** Definición de los parámetros, variables y restricciones orientados a la problemática del problema.	Documento que contendrá la declaración de variables, restricciones y sus definiciones.	-Revisión especializada por un experto en el área de ciencias de la computación de la efectividad de las variables y parámetros elegidos. Debe presentar no más de un par de observaciones.
	R2: Definición de una función objetivo que permita la representación del grado de optimización alcanzado.	Documento en el cual se podrá apreciar la formulación y definición de la función objetivo	-Revisión especializada por un experto en el área de ciencias de la computación de qué tan bien se ajusta la función objetivo a la búsqueda de la solución del problema. Debe presentar no más de un par de observaciones. -Alto grado de optimización alcanzado (un 95% respecto a un ordenamiento aleatorio).
O2	**R3:** Definición de funciones, métodos y clases a implementar para ser usadas con el algoritmo memético.	Documento que contendrá las definiciones de las funciones, métodos y clases de la solución. Documento que contendrá el pseudocódigo de la solución.	-Revisión especializada por un experto en el área de ciencias de la computación de la eficiencia del uso de las funciones, métodos y clases planteados, así como el pseudocódigo. Debe presentar no más de un par de observaciones.
	R4: Diseño y adaptación de un algoritmo memético que pueda resolver el problema propuesto.	Documento que contendrá el código fuente de la solución.	-Revisión especializada por un experto en el área de ciencias de la computación de la efectividad del algoritmo para resolver el problema propuesto. Debe presentar no más de un par de observaciones. -Pruebas de funcionamiento exitosas, funcionamiento esperado

Objetivo	Resultado	Medio de verificación	Indicador Objetivamente Verificable
			al 100%
O3	**R5:** Adaptación o extracción de un algoritmo de gran uso (equivalente o similar) que permita resolver el problema propuesto (o problemas similares).	Documentación del algoritmo y el código fuente que se desarrollará/adaptará/ empleará.	-Revisión especializada por un experto en el área de ciencias de la computación del nivel de adaptación del algoritmo a comparar a la problemática. Debe presentar no más de un par de observaciones. -Pruebas de funcionamiento exitosas, funcionamiento esperado al 100%.
	R6: Desarrollo de las pruebas de ambos algoritmos para encontrar errores y probar distintas configuraciones.	Documento de pruebas que contendrá, las pruebas y sus resultados, así como errores corregidos.	-Revisión y validación especializada por un experto en el área de ciencias de la computación para medir el nivel de efectividad de las pruebas para obtener la mejor adaptación posible. Búsqueda de aprobación del especialista. Debe presentar no más de un par de observaciones. -Alto grado de errores, por lo menos superior al 80%, que pudieron ser solucionados o pocos errores encontrados durante las pruebas, en menos del 10% del desarrollo, con alta optimización alcanzada.
	R7: Diseño y desarrollo de la experimentación numérica que permita comparar el rendimiento del algoritmo propuesto con soluciones similares. Relacionado con O3.	Documento que contendrá el diseño de la experimentación numérica a detalle.	-Revisión especializada por un experto en el área de ciencias de la computación de la fidelidad de los resultados. Debe presentar no más de un par de observaciones.

Objetivo	Resultado	Medio de verificación	Indicador Objetivamente Verificable
			-Resultados óptimos, por lo menos un 20% de optimización alcanzada, respecto a la efectividad del algoritmo propuesto a comparación del algoritmo alternativo en las pruebas planteadas en la experimentación numérica.

1.2. Métodos y procedimientos

A continuación, se describirán aquellas herramientas y procedimientos que serán utilizados con el fin de obtener los resultados previamente mencionados. Se muestran en la tabla 1.2.

Tabla 1.2. Métodos y procedimientos

Resultado	Herramientas, metodologías, métodos y procedimientos
R1: Definición de los parámetros, variables y restricciones orientados a la problemática del problema.	- Documentación del algoritmo memético
R2: Definición de una función objetivo que permita la representación del grado de optimización alcanzado.	- Documentación del algoritmo memético
R3: Definición de funciones, métodos y clases a implementar para ser usadas con el algoritmo memético.	- Documentación del algoritmo memético
R4: Diseño y adaptación de un algoritmo memético que pueda resolver el problema propuesto.	- C# (.NET) - Metodología Scrum
R5: Adaptación o extracción de un algoritmo de gran uso (equivalente o similar) que permita resolver el problema propuesto (o problemas	- C# (.NET) - Metodología Scrum

Resultado	Herramientas, metodologías, métodos y procedimientos
similares).	
R6: Desarrollo de las pruebas de ambos algoritmos para encontrar errores y probar distintas configuraciones.	- C# (.NET) - Metodología Scrum - Pruebas de Caja Negra
R7: Diseño de la experimentación numérica que permita comparar el rendimiento del algoritmo propuesto con soluciones similares.	- RStudio - Prueba de Shapiro-Wilk - Prueba Z - Prueba de U Mann-Whitney

1.3. Definición de las metodología, herramientas y métodos usados

1.3.1. C# (.NET)

Lenguaje de programación orientado a objetos que se ejecutan en el entorno de programación conocido como .NET. Está basado en el lenguaje C y tiene numerosas librerías que permiten un desarrollo seguro (*Paseo por el lenguaje C#*, 2022).

Se uso este lenguaje debido a que su base en C y el ser orientado a objetos le da bastante flexibilidad y es más sencillo de manejar que otros lenguajes orientados a objetos.

1.3.2. RStudio

Entorno de desarrollo basado en R que permite el desarrollo de diversas pruebas estadísticas usando conjuntos de datos (*About*, 2022).

Se usarán diversas pruebas que se determinarán al momento de tener la data resultante de los algoritmos disponibles.

1.3.3. Scrum Framework

Metodología de enfoque ágil que se caracteriza por entregas continuas con tareas que se entregan por ciclos y ayuda a la gente a enfrentar problemas complejos (*WHAT IS SCRUM?*, 2022).

El uso de este marco de trabajo ayudará en la organización del desarrollo y la documentación.

1.3.4. Pruebas de Caja Negra

Tipo de prueba que se usan para evaluar la confiabilidad de una aplicación (Gao et al., 2003). Se basan en el hecho de que la interfaz es la única interacción entre el usuario y la aplicación (Gao et al., 2003).

1.3.5. Prueba de Shapiro-Wilk

Procedimiento estadístico para comprobación de normalidad en una muestra (Shapiro & Wilk, 1965).

Esta prueba se usará para verificar si es viable usar una prueba paramétrica o no paramétrica en función de la normalidad.

1.3.6. Prueba F

Prueba estadística apropiada para evaluar la equidad de varianzas (Dean et al., 2017).

La prueba será utilizada como prerrequisito para verificar si se puede realizar la prueba Z.

1.3.7. Prueba Z

Debido a que las medias y la desviación estándar pueden ser calculadas, es posible usar la prueba z si es que es posible asumir normalidad y si las varianzas son iguales (Sprinthall, 2014).

Esta prueba será usada en la sección de experimentación numérica principalmente si es que es posible asumir una tendencia a la normalidad en los conjuntos de resultados de las ejecuciones de los algoritmos. Para que esta prueba sea usada se habrá determinado normalidad en la prueba de Shapiro-Wilk.

1.3.8. Prueba de U Mann-Whitney

Prueba no paramétrica en la cual se realiza un análisis para comprobar mediante 2 variables con funciones acumulativas de distribución si una es mayor que la otra (Mann & Whitney, 1947).

Esta prueba será usada en la sección de experimentación numérica principalmente si no se halla una tendencia a la normalidad en los conjuntos de resultados de las ejecuciones de los algoritmos.

2. Marco conceptual

2.1. Introducción

Se procede a presentar conceptos que son necesarios para entender la situación bajo la cual se plantea este proyecto. Se explorarán conceptos relacionados a las antenas Wi-Fi así como el posicionamiento óptimo para después presentar elementos relacionados a la optimización. Se espera que la definición de estos conceptos permita una comprensión más detallada de los elementos y resultados abarcados en este proyecto.

2.2. Conceptos

2.2.1. Antena (Antenna)

Es un dispositivo metálico transmisor o receptor de señales que permite emitir ondas de radio (o recibirlas) del espacio (Rudolf F, 1999). Es usado en diversos tipos de telecomunicaciones. Algunos tipos de antenas son:

- Antenas Wi-Fi: las antenas que se usarán para emitir ondas de radio en redes de área local en el almacén.
- Antenas Omnidireccionales: Antenas que emiten en todas direcciones.
- Antenas Direccionales: Solo emiten señales en un sentido a la vez.

2.2.2. Wi-Fi

Red que permite la conexión de computadoras, consolas y celulares que provee comunicación flexible e inalámbrica a cada usuario, opera en diversas bandas de GHz (Morshed et al., 2009). Wi-Fi significa Wireless Fidelity y esta tecnología traslada la transmisión de frecuencias de radio que contiene datos entre la tarjeta instalada en una laptop (u otro dispositivo) y un punto de acceso inalámbrico (Singh, 2003).

2.2.3. Punto de acceso (Wireless Access Point)

Los puntos de acceso inalámbricos (WAP por sus siglas en inglés) son un tipo de transmisores radiales que representan nodos en una red local. Se utilizan las radiofrecuencias designadas por los estándares 802.11. En algunos casos pueden funcionar como routers (Chow & Bucknall, 2012).

2.2.4. Problemas de optimización combinatoria

Se encuentran encargados de la determinación de un arreglo u orden óptimo de los elementos del problema. Cuentan con un funcionamiento constructivo e iterativo y refleja la optimización mediante la mejora continua de una función de costo asociada al problema (Sait & Youssef, 1999). El problema que se busca solucionar en este proyecto es el problema de cobertura de conjuntos, pero algunos otros son:

- Problema de la mochila
- Problema de las n-reinas
- Problema del vendedor viajero

Este proyecto contemplará diversos puntos donde podrían presentarse distintos usuarios que serían el set de elementos cubiertos.

2.2.5. Problema de cobertura de conjuntos (Set Covering Problem)

Es un problema NP-hard de optimización perteneciente a los problemas de optimización combinatoria donde se busca cubrir un conjunto de elementos con el mínimo de grupos posible (Grossman & Wool, 1997). En este proyecto el área cubierta por cada antena Wi-Fi son los conjuntos de elementos y se busca cubrir toda el área del almacén.

2.2.6. Algoritmo

Se trata de un proceso computacional con entradas y salidas. Describe un proceso paso a paso para resolver un problema (Aho et al., 1974). Algunos algoritmos son:

- Algoritmo Genético: Es uno de los algoritmos más conocidos, combina diversas soluciones simulando el emparejamiento de genes para obtener nuevos resultados.
- Algoritmo Memético: Se explica en la siguiente sección.
- Algoritmo de Optimización de partículas: Un tipo de algoritmo que se centra en la gestión de partículas en un sistema.

2.2.7. Algoritmo Memético

Es un algoritmo que simula la evolución cultural de manera análoga a como el algoritmo genético lo hace con la evolución biológica. Comprenden el desarrollo de una población (posibles configuraciones del conjunto de puntos de acceso) que cooperan (intercambiando información o compitiendo) para generar nuevas soluciones. A diferencia del algoritmo genético, el algoritmo memético tiene más capacidad de adaptación y cada individuo realiza una búsqueda local (que es un proceso de búsqueda de soluciones nuevas con las soluciones que ya se tienen) (Moscato, 1989).

2.2.8. Almacén

Es un tipo de estructura que se usa para almacenar distintos tipos de bienes (Harris, 2006), su existencia se debe a la necesidad de almacenamiento comunitario de alimentos y otros bienes similares, su uso fue vital en la formación de proto-estados en la edad de bronce (Renfrew, 1972).

3. Estado del Arte

3.1. Introducción

En esta sección se procede a realizar la revisión sistemática para el proyecto siguiendo la metodología de búsqueda o identificación, evaluación o criterios de inclusión/exclusión y la selección de documentos resultantes (Verner et al., 2012).

3.2. Objetivos de revisión

Se procede a mencionar los objetivos de revisión que forman parte de la revisión sistemática del proyecto, siendo la revisión de tipo teórica debido al apoyo en documentos ya existentes y empírica por los resultados de las investigaciones:

- Recopilar parámetros y planteamientos que permitan la implementación de un algoritmo memético para la distribución de antenas wifi en almacenes de grandes dimensiones.
- Conocer los conceptos relacionados a la cobertura de antenas wifi en áreas de gran magnitud, así como las variables y restricciones que se pueden aplicar para determinar el posicionamiento óptimo.
- Conocer los algoritmos, y de qué tipo son, que hayan sido aplicados para resolver problemas de cobertura similares
- Determinar formas de validar y comparar algoritmos usados en problemas similares.

3.3. Preguntas de revisión

Mediante el método PICOC (Population, Intervention, Comparison, Outcome, Context) se determinan las preguntas de investigación necesarias para poder obtener los documentos necesarios a partir de la tabla 3.1:

Tabla 3.1 Tabla PICOC para preguntas de revisión

Elemento	Respuesta hacia	Descripción
Población (Population)	¿Quién/es?	Posicionamiento de antenas wifi en almacenes de gran tamaño.

Elemento	Respuesta hacia	Descripción
Intervención (Intervention)	¿Qué o cómo?	Uso de algoritmo memético para la optimización del posicionamiento de las antenas wifi.
Comparación (Comparison)	¿Comparado con qué?	Algoritmos de optimización más usados en problemas similares de uso de antenas y cobertura de áreas.
Salida (Outcome)	¿Qué se está tratando de lograr o mejorar?	Publicaciones o investigaciones que permitan mejorar el posicionamiento de las antenas para cubrir toda el área del almacén con la menor cantidad de antenas wifi posibles así como reducciones en el costo de su implementación.
Contexto (Context)	¿En qué tipo de organización / circunstancias?	Toma lugar en los ámbitos académico e industrial.

Y con el método ya establecido y los resultados del análisis, se estructuran las preguntas de investigación:

P1: ¿Qué algoritmos o soluciones se han propuesto para resolver el problema de cobertura máxima con el mínimo de emisores similares?
P2: ¿Qué variables y restricciones se definen en los algoritmos que son utilizados para resolver este tipo de problemas?
P3: ¿Qué métodos se emplearon en la evaluación de los resultados que han tenido los diferentes algoritmos propuestos?

3.4. Estrategias de búsqueda

3.4.1. Motores de búsqueda

Las bases de datos que se usarán para la recopilación de información en el proyecto son las siguientes debido a que, al realizar las pruebas del caso, fueron las que dieron mejores

resultados sin muchos duplicados:

- IEEE XPLORE
- Scopus

3.4.2. Cadenas de búsqueda a usar

Se usan los siguientes términos mostrados en la tabla 3.2 para las cadenas obtenidos de la metodología PICOC, los términos son relacionados a los elementos pertenecientes a lo determinado en el método PICOC en la tabla 3.2:

Tabla 3.2 Tabla PICOC para palabras clave

Value	Key Words
Population	Set Cover
	Antenna
	Wi-Fi
	Wireless
	Access Point
Intervention	Genetic Algorithm
	Memetic Algorithm
	Algorithm
	Heuristic
Outcome	Optimal
	Optimization
	Positioning
	Position

Se consideró genetic algorithm como keyword porque es el más usado para resolver problemas NP y el memetic algorithm también porque es aquel que se planea implementar.

De modo que se tienen las siguientes cadenas para cada motor:

Para el caso de IEEE Xplore se usa un filtro desde el año 2015

- **IEEE XPLORE:**

(“Optim*”) AND (“Position*” OR “Distribution”) AND (“Algorithm” OR “Genetic Algorithm” OR “Memetic Algorithm” OR "Heuristic") AND (“Antenna” OR “Wireless” OR (“Access*” AND “Point”)) AND (“Set Cover*”)

- **Scopus:**

TITLE-ABS-KEY ("Optim*") AND ("Position*" OR "Distribution") AND ("Algorithm" OR "Genetic Algorithm" OR "Memetic Algorithm" OR "Heuristic") AND ("Antenna" OR "Wireless" OR ("Access*" AND "Point")) AND ("Set Cover*") AND (LIMIT-TO (SUBJAREA , "COMP")) AND (LIMIT-TO (PUBYEAR , 2022) OR LIMIT-TO (PUBYEAR , 2021) OR LIMIT-TO (PUBYEAR , 2020) OR LIMIT-TO (PUBYEAR , 2019) OR LIMIT-TO (PUBYEAR , 2018) OR LIMIT-TO (PUBYEAR , 2017) OR LIMIT-TO (PUBYEAR , 2016))

3.4.3. Cantidad de documentos que existen en cada motor de búsqueda (antes de criterios de inclusión y exclusión)

En la tabla 3.3 se muestran los resultados obtenidos en los motores de búsqueda con las cadenas generadas antes de aplicar criterios:

Tabla 3.3 Resultados de la búsqueda

Motor de búsqueda	Resultados	Duplicados	Después de quitar duplicados
IEEE XPLORE	62	50	62
Scopus	280	50	230

3.5. Criterios de Inclusión y Exclusión

3.5.1. Criterios de Inclusión

- La fuente reporta el uso de algoritmos para encontrar problemas de posicionamiento óptimo o similares.
- La fuente detalla en el proceso de funcionamiento del algoritmo
- La fuente toma en consideración un problema similar de cobertura de conjuntos
- Fuentes de preferencia con año de publicación a partir del 2015 debido a que documentos de mucha antigüedad pueden contener información que ya no es válida y no se escogió un año posterior porque no hay tantos documentos que realmente se relacionen al tema.

3.5.2. Criterios de Exclusión

- La fuente es parte de literatura gris, haciéndola no confiable.
- La fuente no detalla lo suficiente en alguno de los algoritmos para resolver el problema.
- La fuente está en un idioma distinto al español o inglés.
- La fuente detalla la solución, pero desde el punto de vista de la electrónica en vez de ciencias de la computación.

3.5.3. Nuevos resultados considerando los criterios de inclusión y exclusión

Detalle de la búsqueda una vez aplicados los criterios mostrados en la tabla 3.4.

Tabla 3.4 Tabla de resultados de búsqueda con criterios de inclusión y exclusión

Motor de búsqueda	Resultados previos	Resultados aplicando los criterios que solo aparecen en el motor respectivo
IEEE XPLORE	62	4
Scopus	230	5

En realidad, salieron 61 resultados en IEEE XPLORE pero uno de ellos que se obtuvo sin considerar el filtro de año se considera material importante y falla únicamente en el filtro del año de publicación de IEEE XPLORE, de modo que se incluye adicionalmente.

Se presentan los resultados que se consideran relevantes en la tabla 3.5.

Tabla 3.5 Tabla de documentos obtenidos

ID	Título	Autores	Año
D1	AP deployment optimization in non-uniform service areas: a genetic algorithm approach	Zhi, Zicong;Wu, Jianghong;Meng, Xin;Yao, Mengqian;Hu, Qian;Tang, Zhenzhou	2019
D2	Dynamic maximal covering location problem for fire stations under uncertainty: soft-computing approaches	Hajipour, Vahida; Fattahi, Parvizb; Bagheri, Hasanc; Babaei Morad, Samaneh	2022
D3	Location Optimization of VTS Radar Stations Considering Environmental Occlusion and Radar Attenuation	Huang, Chuana; Lu, Jinga; Sun, Li-Qian	2022
D4	Maximizing Lifetime of Range-Adjustable Wireless Sensor Networks: A Neighborhood-Based Estimation of Distribution Algorithm	Zong-Gan Chen;Ying Lin;Yue-Jiao Gong;Zhi-Hui Zhan;Jun Zhang	2021
D5	Positioning WiFi access points using Particle Swarm Optimization	Anshu Bhuwania; Pritam Subba; Uttam Kumar Roy	2016
D6	Coverage planning for outdoor wireless LAN systems	M. Kamenetsky; M. Unbehaun	2001
D7	Heuristic Algorithm for Gateway Location Selection in Large Scale LoRa Networks	Krzysztof Grochla; Konrad Połys	2020
D8	Some Neighbourhood Approaches for the Antenna Positioning Problem	Larbi Benmezal; Belaid	2017

		Benhamou; Dalila Boughaci	
D9	Meta-heuristic algorithms to improve fuzzy C-means and K-means clustering for location allocation of telecenters under E-governance in developing nations	Gupta, Rajan; Muttoo, Sunil Kumar; Pal, Saibal K	2019

3.6. Formulario de extracción de datos

Este formulario será desarrollado para contestar las preguntas de revisión en base a las fuentes seleccionadas y la plantilla se muestra a continuación. El formulario completo se puede ver en el anexo A.

Tabla 3.6 Formulario de extracción de datos

Campo	Descripción	Pregunta
Identificador		-
Título		-
Autor(es)		-
Año de publicación		
Tipo de fuente	Artículo de investigación	-
Idioma		
Motor de búsqueda que se usó en su extracción		-
Link de consulta		-
Abstract	Introducción y explicación del contenido del material	
Solución propuesta a problemas de cobertura	Qué situación busca resolver este problema de optimización de posicionamiento para cobertura	P1

Algoritmos usados	Qué algoritmos meméticos y no meméticos se proponen para resolver el problema del documento.	P1
Uso de variables y restricciones	Qué variables o restricciones se proponen en el algoritmo a usar para el problema	P2
Resultado de la aplicación del algoritmo	Qué tan efectivo es el algoritmo propuesto en la solución del problema	P3
Método de prueba	Qué método se usó para evaluar el algoritmo	P3

3.7. Resultados de revisión

Se procede a responder las preguntas con los documentos que se cumplen los criterios de inclusión y exclusión:

3.7.1. ¿Qué propuestas se han presentado para resolver el problema de cobertura máxima con el mínimo de emisores similares?

En el desarrollo de soluciones de algoritmos de diversos tipos se encontró que el algoritmo genético es uno de los algoritmos más usados para elaborar soluciones que requieren optimizar algún resultado.

De esta manera, el algoritmo genético se puede aplicar a la búsqueda de optimización de posicionamiento de señales inalámbricas (Zhi et al., 2019). Sin embargo, en diversos documentos con problemas de este tipo, se encontró que PSO (Particle Swarm Optimization) es una solución muy eficaz (Hajipour et al., 2022), (Huang et al., 2022), (Bhuwania et al., 2016), pero ciertamente el algoritmo Murciélago y el algoritmo Ant Colony Optimization pueden dar muy buenos resultados de igual manera si se divide el área total en celdas y cada centro de estas es un punto candidato para ser la ubicación del emisor (Gupta et al., 2019).

Pero también se encontraron propuestas únicas para resolver problemas de este tipo (Grochla & Polys, 2020) que incluso pueden ser mejores que otros algoritmos ya existentes (Chen et al., 2021) y se pueden aplicar algoritmos únicos que tomen como base otros algoritmos ya existentes (Benmezal et al., 2017). También se puede enfrentar este problema usando el algoritmo Pruning pero agregando ciertas partes del algoritmo Neighorhood Search para mejorar el rendimiento (Kamenetsky & Unbehaun, 2002).

3.7.2. ¿Qué variables y restricciones se definen en los algoritmos que son utilizados para resolver este tipo de problemas?

En todos los casos se tiene una función objetivo que está en función del costo generado por la posición de cada emisor (que cambia en función de la situación. Puede tratarse de energía, atenuación, etc). De similar manera se tienen variables de estado asociadas aquellos elementos que se busca cubrir con los emisores. Algunos ejemplos pueden ser edificios, celulares, etc. Pero en el escenario que este proyecto busca enfocarse es la cobertura total de un área y uno de estos documentos cubre ese escenario (Zhi et al., 2019).

En algunos escenarios se proponen obstáculos que reducen la señal de los emisores (que para estos casos son antenas o puntos de acceso) con el tipo de material y la reducción de señal causada (Bhuwania et al., 2016), (Kamenetsky & Unbehaun, 2002).

3.7.3. ¿Qué métodos se emplearon en la evaluación de los resultados que han tenido los diferentes algoritmos propuestos?

Es adecuado mencionar que en todos casos se realizaron pruebas de experimentación numérica con distintos parámetros (con valores a criterio del experimentador) para determinar la eficacia

del algoritmo evaluado (o para compararlo contra otros algoritmos existentes propuestos) y siempre se ejecutan diversos números de centenas para las iteraciones que se buscan comparar. En algunos casos se tomaron en cuenta para la comparación la determinación de medidas estadísticas obtenidas mediante la repetida ejecución del algoritmo siendo estas la media, la desviación estándar, el mínimo, el máximo y el rango intercuartil de los resultados (Gupta et al., 2019). En otros casos simplemente se ejecutaron simulaciones de los algoritmos con enormes cantidades de iteraciones (Kamenetsky & Unbehaun, 2002). Y en el escenario más similar al del proyecto además se muestra los resultados antes (posicionamiento inicial aleatorio) y después de aplicado el posicionamiento incluso pudiendo visualizar el rango de alcance de cada emisor dentro del edificio (Bhuwania et al., 2016). Pero en el resto de documentos por lo general se limitan a mostrar gráficos de posición con emisores usados.

3.8. Conclusiones

La búsqueda y revisión de estos documentos permitió una perspectiva con más conocimiento del problema de posicionamiento encontrando varias situaciones similares donde se aplican algoritmos conocidos. También permitió considerar aspectos no planeados previamente como la atenuación de la señal por las paredes (Bhuwania et al., 2016) o el gasto energético por emisor. Asimismo, se pudo visualizar que tipo de métodos se emplean para verificar la efectividad del algoritmo propuesto y esto logró comprobar que un buen algoritmo de posicionamiento permite prescindir de emisores innecesarios.

4. Parámetros, variables, restricciones y Función Objetivo

A continuación, se presentan los parámetros, variables y restricciones que permitirán cumplir con el resultado esperado 1. Este resultado, así como los elementos que lo conforman, permitirá el avance y determinación de los demás resultados.

4.1. Parámetros

Debido a las características del ambiente, se han encontrado los siguientes parámetros que ayudarán a configurar las características de los algoritmos.

Parámetros globales: Parámetros que serán utilizados para toda la solución.

Tabla 4.1 Parámetros globales

Parámetro	Descripción
Largo Almacén	Longitud del almacén en metros
Ancho Almacén	Ancho del almacén en metros
Total Celdas Horizontal	Total de celdas utilizadas para delimitar la dimensión horizontal
Total Celdas Vertical	Total de celdas utilizadas para delimitar la dimensión vertical
Radio Antena	Radio de la señal de la antena
Peso del área	Peso asignado al componente del área cubierta en un almacén en la función objetivo
Peso del número de antenas	Peso asignado al componente del número de antenas en un almacén en la función objetivo
Máximo de antenas	Cantidad máxima de antenas que se pueden utilizar
Máximo de iteraciones	Total de las iteraciones a utilizar para la ejecución de un algoritmo durante su aplicación
Total Población	Total de almacenes a utilizar en la población
Porcentaje Mínimo Área	Porcentaje del total del área mínimo deseable a obtener

Parámetros exclusivos del algoritmo memético: Son parámetros que únicamente son utilizados por el algoritmo memético en su funcionamiento.

Tabla 4.2 Parámetros exclusivos del algoritmo memético

Parámetro	Descripción
Iteraciones de la Evolución	Número de veces en las cuales se procede con la evolución de un almacén.
Probabilidad de Mutación Memética	Porcentaje que determina la probabilidad de mutación en un algoritmo memético

Porcentaje Movimientos X	Porcentaje del máximo de distancia en celdas que se puede desplazar una antena en movimientos horizontales.
Porcentaje Movimientos Y	Porcentaje del máximo de distancia en celdas que se puede desplazar una antena en movimientos verticales.

Parámetros exclusivos del algoritmo genético adaptado: Son parámetros que únicamente son utilizados por el algoritmo genético en su funcionamiento.

Tabla 4.3 Parámetros exclusivos del algoritmo genético

Parámetro	Descripción
Probabilidad de Mutación	Porcentaje que determina la probabilidad de mutación en un algoritmo genético
Probabilidad de Cruce	Porcentaje que determina la probabilidad de cruce en un algoritmo genético

Parámetros exclusivos del cálculo del área: Parámetros cuyos valores son utilizados para el cálculo del área cubierta por las antenas presentes actualmente en un almacén.

Tabla 4.4 Parámetros exclusivos del cálculo del área

Parámetro	Descripción
Pixeles por celda	Número de pixeles por lado de una celda. La potencia al cuadrado determinará cuantos pixeles se encuentran en una celda.

Parámetros auxiliares: Parámetros cuyos valores dependen de los parámetros previos.

Tabla 4.5 Parámetros auxiliares

Parámetro	Descripción
Movimientos Celdas X	Movimientos horizontales en celdas que una antena puede realizar y es determinado a partir del porcentaje de movimientos X
Movimientos Celdas Y	Movimientos verticales en celdas que una antena puede realizar y es determinado a partir del porcentaje de movimientos Y
Área total Almacén	Área total del almacén que depende del largo y ancho en metros cuadrados
Largo Celda	Largo de una celda que depende del largo del almacén y el número de celdas por longitud horizontal en metros

Ancho Celda	Ancho de una celda que depende del ancho del almacén y el número de celdas por longitud vertical en metros
Largo Pixel	Largo de un pixel que depende del largo de una celda y el número de pixeles por lado en metros
Ancho Pixel	Ancho de un pixel que depende del ancho de una celda y el número de pixeles por lado en metros
Peso del área	Peso asignado al componente del área cubierta en un almacén en la función objetivo
Centro Y superior límite	Coordenada superior en el eje Y máxima donde es posible colocar una antena. Depende de las filas validas de colocación de antenas y el ancho de una celda.
Centro Y inferior límite	Coordenada inferior en el eje Y mínima donde es posible colocar una antena. Depende de las filas validas de colocación de antenas y el ancho de una celda.
Centro X izquierda límite	Coordenada orientada a la izquierda en el eje X mínima donde es posible colocar una antena. Depende del largo de una celda.
Centro X derecha límite	Coordenada orientada a la derecha en el eje X máxima donde es posible colocar una antena. Depende del largo de una celda.
Aproximado de celdas por radio X	Cantidad de celdas horizontales aproximadas que ocupa el radio de una antena. Su valor depende del radio de la antena y el largo de una celda
Aproximado de celdas por radio Y	Cantidad de celdas horizontales aproximadas que ocupa el radio de una antena. Su valor depende del radio de la antena y el ancho de una celda

4.1.1. Eje de coordenadas

Corresponden a las coordenadas en las cuales se encuentra representado el almacén y puede ser colocada una antena, estar presente una fila donde hay mercadería o no hay ningún elemento. La coordenada X representa el largo y la Y representa el ancho.

Representado por *XY*.

4.2. Variables

En esta sección se procede a especificar las variables que serán usadas en la resolución del problema previamente mencionado. Su uso permitirá evaluar el comportamiento del algoritmo.

4.2.1. Número de antenas en un almacén

Es el número de antenas a usar para la emisión de señales en un individuo de la población.

Representado por *AN*.

4.2.2. Colección de coordenadas de antenas en un almacén

Representa las coordenadas seleccionadas para el posicionamiento de antenas (entre otros atributos), de modo que estas puedan emitir las señales.

Representado por *AB*.

4.2.3. Matriz de recepción de señal por secciones del almacén

Corresponde al conjunto de coordenadas del almacén que están recibiendo señal.

Representado por *MNR*.

4.3. Restricciones

Finalmente, se presentan las restricciones planteadas para este problema cuyo uso permite establecer condiciones orientadas a cumplir los objetivos presentados previamente.

4.3.1. Área total del almacén

Indica el área total que corresponde al almacén de grandes dimensiones. Esta dimensión no debe resultar igual o menor a cero

$$Total\ Celdas\ Horizontal * Total\ Celdas\ Vertical > 0$$

4.3.2. Área límite del almacén

Indica el área válida en la cual se considera el análisis de las ondas en el almacén, esta área debe estar entre 0 y el largo y ancho definidos en los parámetros.

$$\forall Xi \in [0, Largo\ Almacén >, Xi \in XY$$
$$\forall Yi \in [0, Ancho\ Almacén >, Yi \in XY$$

4.3.3. Las antenas deben estar todas encendidas en simultáneo

Con esta restricción se busca que todas las *AN* antenas se encuentren encendidas.

$$\forall i \in AB, MNRi > 0$$

4.3.4. Porcentaje de área sin señal

Con esta restricción se busca establecer un porcentaje *d* máximo del área total del almacén que no tendrá señal.

$$\sum_{i,j} MNRij \geq d * Total\ Celdas\ Horizontal * \ Total\ Celdas\ Vertical, MNRij > 0$$

4.4. Función Objetivo

El propósito de la función objetivo a definir es reducir el número de antenas para evitar la instalación de antenas innecesarias, de esta manera se puede evitar los gastos innecesarios y demás problemas asociados al exceso de antenas. Se utilizan 2 pesos para determinar la importancia de la cantidad de antenas usadas y la cobertura alcanzada. El uso de| las restricciones previamente planteada apoya la búsqueda de optimización en la selección de antenas. Se toman como elementos principales el número de antenas y el área cubierta por la señal. (Bhuwania et al., 2016).

$$w1 * \frac{1}{AN} + w2 * (\frac{\sum_{i,j} MNRij}{Total\ Celdas\ Horizontal * \ Total\ Celdas\ Vertical})$$

4.5. Discusión

En esta sección se buscó mostrar aquellos temas que permitirán alcanzar los resultados esperados R1 y R2 que permitirán cumplir con el objetivo específico O1. Con estos elementos es posible tener un planteamiento general que será la base del diseño e implementación del algoritmo y el resto de objetivos. Esta sección fue validada y aprobada por un especialista en Ciencias de la Computación y se presenta su autorización en el anexo D.

5. Diseño de la solución

Este apartado del proyecto corresponde al planteamiento del diseño de la implementación del algoritmo para su desarrollo, el propósito principal es cumplir con el desarrollo de estructuras lógicas para el cumplimiento de las tareas del algoritmo.

5.1. Estructuras Principales

5.1.1. Objeto celda

Objeto que determina una ubicación en el almacén donde en su centro posiblemente se contendrá una antena, se almacena también las coordenadas de la base inferior izquierda de esta celda, así como las coordenadas de su centro y la posición de la antena que almacena de

ser el caso. Adicionalmente se tiene un parámetro que contiene una antena cercana que sirve para el Algoritmo Calculo de Áreas.

5.1.2. Objeto antena

Objeto que contendrá las coordenadas de la posición de la antena, así como las ubicaciones horizontal y vertical de la celda en la cual se encuentra la antena.

5.1.3. Objeto almacén

Objeto que contendrá los objetos antenas y celdas y tendrá métodos que permitirán la ejecución del algoritmo.

5.2. Estructuras Auxiliares

5.2.1. Matriz de celdas para las antenas

Matriz que dividirá el almacén en celdas iguales donde se puede colocar 1 antena. También será usado para el cambio de posiciones. Se usará el centro de estas celdas para colocar las antenas. Se utilizará esta estructura para todos los algoritmos basado en su proposición para el algoritmo genético adaptado de la bibliografía (Zhi et al., 2019). Se ejemplifica de la siguiente forma en la Figura 5.1:

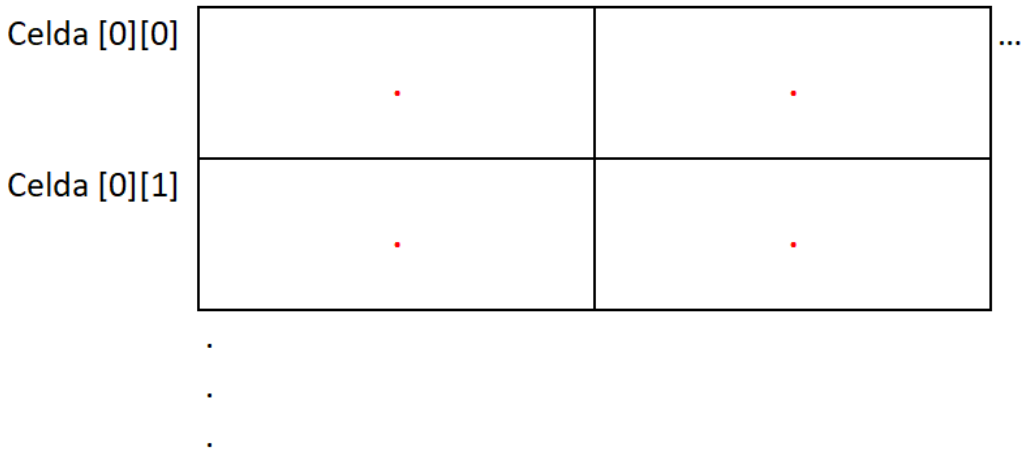


Figura 5.1. 4 celdas con antenas

5.2.2. Arreglos unidimensionales

Se tendrán arreglos unidimensionales que se encontrarán asignados a distintos elementos para tener un control rápido de estos y no estar supeditados a objetos superiores, los elementos que

tendrán estos arreglos son:

- **Arreglo de antenas:** Este arreglo existirá como atributo de un objeto almacén para tener identificadas las antenas que contiene y evitar explorar toda la matriz de celdas cada vez que se requiera información de las antenas.

- **Arreglo de filas candidatas:** Debido a que solo se pueden colocar las antenas en celdas donde no se encuentre mercadería del propio almacén, se tiene un arreglo conteniendo las posiciones de las filas de celdas sobre las cuales se puede realizar la instalación de (o que sea posible instalar) las antenas.

- **Población:** Arreglo de almacenes que funcionarán como individuos de ambos algoritmos, permite la gestión de los algoritmos sobre los almacenes con los que trabaja.

- **Filas Candidatas:** Arreglo de filas en las cuales se almacenan las posiciones de las celdas en las cuales es posible colocar una antena. Su propósito es evitar colocar antenas en zonas donde se encuentra la mercadería.

5.3. Pseudocódigo Algoritmo Memético

Se presenta el pseudocódigo que será representa el desarrollo de los métodos del algoritmo y la gestión de las estructuras mencionadas.

5.3.1. Cuerpo del Algoritmo Memético

Se visualiza su planteamiento en la Figura 5.2.

```
AlgoritmoMemetico()
BEGIN
    InicializarParametros(parametros)
    mejoresSoluciones.inicializar()
    Poblacion <- GenerarPoblacionInicial()
    WHILE(EvaluarCondicionSalida()) DO{
        FOR(i = 0 TO i = totalPoblacion - 1){
            EvolucionarIndividuo(Poblacion[i])
        }
        MutacionMemetica(Población)
        FOR(i = 0 TO i = totalPoblacion - 1){
            EvolucionarIndividuo(Poblacion[i])
        }
        Competición(Población)
        IF(EvaluacionDeConvergencia(Población) OR ultimaIteracion){
            mejoresSoluciones.agregar(ElegirMejorResultado(Poblacion))
            Poblacion <- GenerarPoblacionInicial()
        }
    }
    ElegirMejorResultado(mejoresSoluciones)
END
```

Figura 5.2. Cuerpo del algoritmo memético propuesto

- La primera acción consiste en inicializar los parámetros auxiliares en base a los

parámetros de entrada que se presentan, su definición se encuentra en el Anexo E.1.1.

- Se procede a la generación inicial de la población para poder comenzar su desarrollo, se presenta el detalle en el Anexo E.1.2.

- Posteriormente se realiza la evolución de cada individuo.

- Se procede a evaluar la mutación de un individuo (y a realizarla si cumple con las condiciones).

- Luego se permite otra evolución de un almacén para mejorar su fitness antes de realizar comparaciones.

- Se realiza una competición donde los almacenes con fitness inferiores son reemplazados por los superiores.

- Se evalúa si la población alcanzó una convergencia o si es la última iteración.

- En caso ser positivo se obtiene el mejor resultado a partir de aquel almacén que tenga mayor fitness. Se guarda la mejor solución en la lista de Mejores soluciones y se genera nuevamente la población para finalizar la iteración.

- Una vez alcanzada la condición de salida se obtiene el mejor resultado de la lista de Mejores soluciones para retornarlo.

5.3.2. Evolucionar individuo

Este método consiste principalmente en una implementación original basada en el concepto de búsqueda local mostrado en la Figura 5.2. Consiste en el concepto de verificar 4 movimientos (en 4 direcciones, 2 verticales y 2 horizontales) y escoger aquel que implique una menor cantidad de antenas en las cercanías para buscar cubrir más área del almacén. Si la posición candidata no es mejor a la actual entonces no se aplica el movimiento de la antena.

```
Procedure Local-Search-Engine (current)
begin
    repeat
        new ← GenerateNeighbor(current);
        if (F_g(new) ≺_F F_g(current)) then
            current ← new;
        endif
    until TerminationCriterion();
    return current;
end
```

Figura 5.3. Algoritmo de búsqueda local

Tomado de "A Gentle Introduction to Memetic Algorithms" por Moscato, P., & Cotta, C. (2003)

Se presenta el pseudocódigo en la Figura 5.4. y se explica su planteamiento:

```
EvolucionarIndividuo(pos)
BEGIN
    individuo = poblacion[pos]
    WHILE(Iteraciones_evolucion){
    {
        FOR(j = 0 TO j = totalAntenas - 1)
        {
            desplazamientoVertical <- Aleatorio(1, movimientosCeldasY + 1)
            desplazamientoHorizontal <- largoCelda * Aleatorio(1, movimientosCeldasX + 1)
            Antena evaluada <- individuo.antenas[j]

            antenasExteriores <- individuo.antenas.AntenasDiferentesDe(evaluada)

            posibilidades.inicializar()

            IF(evaluada.x < centroXCeldaDerechaLimite)
            {
                derechaX <- MINIMO(evaluada.x + desplazamientoHorizontal, centroXCeldaDerechaLimite)
                derechaY <- evaluada.y
                antenasDerecha <- calcularAntenasCercanas(derechaX, derechaY, antenasExteriores)
                posibilidades.Add(antenasDerecha, derechaX, derechaY)
            }

            IF(evaluada.x > centroXCeldaIzquierdaLimite)
            {
                izquierdaX <- MAXIMO(evaluada.x - desplazamientoHorizontal, centroXCeldaIzquierdaLimite)
                izquierdaY <- evaluada.y
                antenasIzquierda <- calcularAntenasCercanas(izquierdaX, izquierdaY, antenasExteriores)
                posibilidades.Add(antenasIzquierda, izquierdaX, izquierdaY)
            }

            IF(evaluada.y < centroYCeldaSuperiorLimite)
            {
                arribaX <- evaluada.x
                arribaY <- MINIMO(filasCandidatas.IndexOf(evaluada.celdaY) + desplazamientoVertical, filasCandidatas)
                antenasArriba <- calcularAntenasCercanas(arribaX, (arribaY + 0.5) * anchoCelda, antenasExteriores)
                posibilidades.Add(antenasArriba, arribaX, (arribaY + 0.5)*anchoCelda)
            }

            IF(evaluada.y > centroYCeldaInferiorLimite)
            {
                abajoX <- evaluada.x
                abajoY <- MAXIMO(filasCandidatas.IndexOf(evaluada.celdaY) - desplazamientoVertical, filasCandidatas)
                antenasAbajo <- calcularAntenasCercanas(abajoX, (abajoY + 0.5)*anchoCelda, antenasExteriores)
                posibilidades.Add(antenasAbajo, abajoX, (abajoY + 0.5)*anchoCelda)
            }

            nuevaAntena <- posibilidades.obtenerConjuntoConMenosAntenas()

            nuevoAlmacen <- individuo.crearCopia()

            nuevoAlmacen.reemplazarAntena(evaluada,nuevaAntena)

            nuevoFitness <- nuevoAlmacen.calcularFitness()

            IF(nuevoFitness > individuo.Fitness)
            {
                poblacion[pos] <- nuevoAlmacen
            }

        }
    }
END
```

Figura 5.4. Evolucionar Individuo

- Se procede a captar el almacén a evaluar.

- Se ingresa a una iteración de la cual no se saldrá hasta que se cumpla con el número de evoluciones representado por parámetro de Iteraciones de la Evolución.

- Se obtienen todas aquellas antenas que no son la antena evaluada.

- Se inicializa una colección que contendrá los datos de cada dirección.

- Para cada uno de los 4 movimientos posibles se evalúa que la antena no se encuentre en el límite de ese movimiento con el fin de evitar coger la antena evaluada como candidata.

- En caso pase la verificación:

    1. Se obtiene la coordenada que no cambiará con el movimiento (en caso de ser

vertical será aquella del eje X y para horizontales será la perteneciente al eje Y).

2. Se obtiene el mínimo/máximo valor entre aquella coordenada que se incrementará/disminuirá y el límite máximo/mínimo para evitar considerar antenas fuera del almacén.

3. Se obtiene el número de antenas que se encuentran alrededor de la posición candidata mediante un método detallado en el Anexo E.1.3.

4. Se almacena la información de esta configuración en la colección.

- Se obtiene el elemento con menor cantidad de antenas cercanas en la nueva posición.

- Se crea una copia del almacén mediante un método especificado en el Anexo E.1.4.

- Se reemplaza la antena evaluada en la copia por la nueva antena y se calcula su fitness mediante un método especificado en el Anexo E.1.5.

- Si la copia tiene mejor fitness que la configuración anterior, la copia reemplaza al almacén actual.

- Se repite el proceso hasta que se evalúe todas las antenas.

- Se repite el proceso hasta que se evalúe todos los almacenes.

5.3.3. Mutación Memético

Proceso por el cual se evalúa el ingreso de una antena a un almacén. Esta sección está altamente inspirada en la propuesta de mutación orientada al algoritmo genético presentado en la bibliografía que se visualiza en la Figura 5.4 (Zhi et al., 2019). La diferencia radica en la inserción de un almacén en caso de que no se cumpla con un área mínima que debe alcanzarse y si se cumple con la probabilidad de mutación.

Algorithm 4 Mutation operation

Input: Individual $\mathbb{I}_1$ and mutation probability Pr_m
Output: New Individual $\mathbb{I}_2$

```
i ← 1
while i ≤ n do
    r ← rand (0, 1)
    if r < Pr_m then
        I_2[i] ← overline(I_1[i])
    else
        I_2[i] ← I_1[i]
    end if
end while
```

Figura 5.4. Operación de mutación

Tomado de "AP Deployment Optimization in Non-Uniform Service Areas: A Genetic Algorithm Approach" por Zhi, Z., Wu, J., Meng, X., Yao, M., Hu, Q., & Tang, Z. (2019).

De modo que la versión generada para el algoritmo memético se presenta en la Figura 5.5.

```
MutacionMemetico()
{
    FOR(i = 0 TO totalPoblacion - 1)
    {
        IF(probabilidadAleatoria < probabilidadMutacionMemetico)
        {
            IF(((poblacion[i].AreaCubierta / areaTotalAlmacen) < porcentajeMinimoArea) AND (poblacion[i].antenas.numeroAntenas() < maxAntenas))
            {
                posiciones = ElegirPosicionesAleatorias(X,Y,filasCandidatas)
                WHILE(antenasEscogidas.contiene(posiciones)){
                    posiciones = ElegirPosicionesAleatorias(X,Y,filasCandidatas)
                }

                poblacion[i].celdas(posiciones).posicionAntena = poblacion[i].numeroAntenas()
                antenaNueva = crearAntena(posiciones)
                poblacion[i].antenas.Add(antenaNueva)

                poblacion[i].AreaCubierta = 0
                poblacion[i].calcularFitness()
            }
        }
    }
}
```

Figura 5.5. Mutación Memético

- Se procede a explorar todos los elementos de la población.

- Se genera una probabilidad aleatoria y se verifica si es menor a la probabilidad de mutación. En caso no lo sea se continua al siguiente almacén. Si lo es, se continúa con el flujo.

- Se verifica si el área cubierta es menor y si es que no se encuentra en el límite de antenas. En caso no lo sea se continua al siguiente almacén. Si lo es, se continúa con el flujo.

- Se procede a buscar posiciones aleatorias que pertenezcan al centro de una celda (dentro de las filas candidatas) que no hayan sido previamente elegidas para una antena.

- Se procede a guardar la posición de la nueva antena en la celda que corresponde.

- Se crea la nueva antena y se añade a las antenas del almacén.

- Se calcula el nuevo fitness.

5.3.4. Competencia

Esta sección se encuentra orientada a comparar el fitness y área cubierta de las distintas soluciones para actualizar la población. Se basa en la premisa propuesta en la bibliografía acerca de una competencia entre los individuos (Moscato, 1989). Se presenta su definición propuesta para esta adaptación del algoritmo memético en la Figura 5.6.

```
Competencia()
{
    porEvaluar <- LlenarConNumerosDesdeHasta(0, totalPoblacion)

    WHILE(porEvaluar.cantidad() >1)
    {

        posPorEvaluar1 <- Aleatorio(1, porEvaluar.totalAntenas-1)
        posRandom1 <- porEvaluar[posPorEvaluar1]
        porEvaluar.QuitarPosicion(posPorEvaluar1)
        posPorEvaluar2 <- Aleatorio(1, porEvaluar.totalAntenas-1)
        posRandom2 <- porEvaluar[posPorEvaluar2]
        porEvaluar.QuitarPosicion(posPorEvaluar2)
        porcentaje1 <- poblacion[posRandom1].AreaCubierta / areaTotalAlmacen
        porcentaje2 <- poblacion[posRandom2].AreaCubierta / areaTotalAlmacen

        IF(porcentaje1 > porcentajeMinimoArea AND porcentaje2 < porcentajeMinimoArea)
        {
            reemplazo(posRandom2, posRandom1)
        }
        ELSE IF(porcentaje1 < porcentajeMinimoArea AND porcentaje2 > porcentajeMinimoArea)
        {
            reemplazo(posRandom1, posRandom2)
        }
        ELSE IF(poblacion[posRandom1].Fitness < poblacion[posRandom2].Fitness)
        {
            reemplazo(posRandom1, posRandom2)

        }
        ELSE IF(poblacion[posRandom1].Fitness > poblacion[posRandom2].Fitness)
        {
            reemplazo(posRandom2, posRandom1)
        }

    }
}
```

Figura 5.6. Competencia

- Se inicializa una colección de posiciones de miembros de la población desde el primero hasta el último.

- Se inicia una secuencia de iteraciones que continua hasta que se hayan evaluado la mayor cantidad par de almacenes posible.

- Se seleccionan posiciones aleatorias que son retiradas de la colección de posiciones.

- Se compara el cumplimiento de la cobertura mínima de área solicitada como prioridad para elegir al ganador.

- En caso ambos o ninguno tengan la cobertura mínima solicitada, se evalúa el fitness de ambos para elegir al ganador.

- Se repite el ciclo hasta terminar la secuencia de iteraciones.

5.4. Pseudocódigo Algoritmo Genético

Se presenta el esquema general de un algoritmo genético adaptado al posicionamiento de antenas en un área que será utilizado para realizar comparaciones con el algoritmo memético planteado en el proyecto (Zhi et al., 2019). Se procede a detallar su funcionamiento:

```
AlgoritmoGenetico()
BEGIN
    InicializarParametros(parametros)
    Poblacion <- GenerarPoblacionInicial()
    WHILE(EvaluarCondicionSalida()) DO{
    {
        CruzamientoGenetico()
        MutacionGenetico()
    }
    ElegirMejorResultado(Población)
END
```

Figura 5.7. Cuerpo de un algoritmo genético

Adaptado de "AP Deployment Optimization in Non-Uniform Service Areas: A Genetic Algorithm Approach" por Zhi, Z., Wu, J., Meng, X., Yao, M., Hu, Q., & Tang, Z. (2019)

- La primera acción consiste en inicializar los parámetros auxiliares en base a los parámetros de entrada que se presentan, su definición se encuentra en el Anexo E.1.1.

- Se procede a la generación inicial de la población para poder comenzar su desarrollo, se presenta el detalle en el Anexo E.1.2.

- Se procede a realizar el cruzamiento entre miembros de la población.

- Se procede a evaluar la mutación de un individuo (y a realizarla si cumple con las condiciones).

- Se evalúa si es la última iteración.

Una vez alcanzada la condición de salida se obtiene el mejor resultado de la población para retornarlo.

5.4.1. Cruzamiento Genético

Para la obtención de una mejor generación se realiza el cruzamiento de las soluciones obtenidas, el procedimiento se realiza para todos los miembros de la población en este caso en pares seleccionados a partir de la operación de selección (Zhi et al., 2019). Se detalla su funcionamiento a continuación:

Algorithm 3 Crossover operation

Input: A pare of parents $\mathbb{I}_{p1}$ and $\mathbb{I}_{p2}$, Crossover Probability Pr_c

Output: Offsprings $\mathbb{I}_{x1}$ and $\mathbb{I}_{x2}$

1: $r \leftarrow$ rand $(0, 1)$
2: **if** $r < \mathrm{Pr}_c$ **then**
3: Randomly select the crossover points l_c from $\{1, 2, \ldots, 24n\}$
4: $\mathbb{I}_{x1} \leftarrow \mathbb{I}_{p1}[1 \ldots l_c] + \mathbb{I}_{p2}[l_c + 1 \ldots 24n]$
5: $\mathbb{I}_{x2} \leftarrow \mathbb{I}_{p2}[1 \ldots l_c] + \mathbb{I}_{p1}[l_c + 1 \ldots 24n]$
6: **end if**

Figura 5.8. Operación de cruzamiento

Tomado de "AP Deployment Optimization in Non-Uniform Service Areas: A Genetic Algorithm Approach" por Zhi, Z., Wu, J., Meng, X., Yao, M., Hu, Q., & Tang, Z. (2019)

Algorithm 2 Roulette wheel selection

Input: Population $\mathbb{P}$

Output: A selected individual.

1: **for** each $\mathbb{I}_i \in \mathbb{P}$ **do**
2: Calculate $F(\mathbb{I}_i)$, $\mathrm{Pr}_s(\mathbb{I}_i)$ and $\mathbb{C}_i$
3: **end for**
4: $r \leftarrow$ rand $(0, 1)$
5: $e \leftarrow 1$
6: **while** $\mathbb{C}_n < \mathrm{r}$ **do**
7: $e \leftarrow e + 1$
8: **end while**
9: $\mathbb{I}_i \leftarrow \mathbb{I}_e$

Figura 5.9. Operación de selección

Tomado de "AP Deployment Optimization in Non-Uniform Service Areas: A Genetic Algorithm Approach" por Zhi, Z., Wu, J., Meng, X., Yao, M., Hu, Q., & Tang, Z. (2019)

```
CruzamientoGenetico()
{
    descendencia.inicializar()
    orden.inicializar()
    probabilidadAcumulada.inicializar()

    fitTotal <- calcularSumaFitnessGenetico()

    probabilidadAcumuladaBucle <- 0
    FOR(i =0 TO totalPoblacion - 1)
    {
        double probabilidadFitness <- poblacion[i].Fitness/ fitTotal
        probabilidadAcumulada.Add(probabilidadAcumuladaBucle)
        probabilidadAcumuladaBucle +<- probabilidadFitness
    }
    restantes <- poblacion.Count()
    posCruce <- 1

    WHILE(restantes > 1)
    {
        indice <- probabilidadAleatoria
        index1 <- probabilidadAcumulada.seleccionarPrimeroQueSeaMenorOIgual(indice)
        AsegurarQueSeanDistintos(index1,index2)
        if (probabilidadAleatoria < probabilidadCruce)
        {
            posCruce <- Aleatorio(1, totalCeldasHorizontal)

            hijo1 <- CruceAlmacenesGenetico(poblacion[index1], poblacion[index2],posCruce)
            if(hijo1.antenas == 0 OR hijo1.antenas>limiteAntenas OR hijo1.areaCubierta/areaTotal<porcentajeAreaMinima){
                hijo1 <- poblacion[index1]
            }
            hijo2 <- CruceAlmacenesGenetico(poblacion[index2], poblacion[index1],posCruce)
            if(hijo2.antenas == 0 OR hijo2.antenas>limiteAntenas OR hijo2.areaCubierta/areaTotal<porcentajeAreaMinima){
                hijo2 <- poblacion[index2]
            }
        }
        else
        {
            hijo1 <- poblacion[index1].crearCopia()
            hijo2 <- poblacion[index2].crearCopia()
        }
        descendencia.Add(hijo1)
        descendencia.Add(hijo2)
        restantes -= 2
    }
    IF(restantes == 1)
    {
        Almacen ultimoHijo <- (sacarMejorSolucion(poblacion)).crearCopia()
        descendencia.Add(ultimoHijo)
    }
    poblacion <- descendencia
}
```

Figura 5.10. Cruzamiento Genético

Adaptado de "AP Deployment Optimization in Non-Uniform Service Areas: A Genetic Algorithm Approach" por Zhi, Z., Wu, J., Meng, X., Yao, M., Hu, Q., & Tang, Z. (2019)

- Se inicializa la descendencia.

- Se realiza la suma del fitness de todos los almacenes hasta el momento.

- A cada almacén se le determina el porcentaje que representan del fitness total y se acumula las probabilidades (asignándole la suma hasta el momento a cada almacén).

- Se determina una probabilidad aleatoria.

- Se determina un almacén aleatorio cuya probabilidad acumulada sea la primera que sea menor o igual a la probabilidad aleatoria.

- Se asegura que los 2 almacenes cogidos por selección sean distintos cambiando el segundo si es que ocurre esta posibilidad.

- En caso no se cumpla la probabilidad de cruzamiento se seleccionan los padres. En caso contrario se realiza la operación de cruzamiento en presentada en la Figura 5.8. que consiste en seleccionar en intercambiar las columnas que sean menor a un número aleatorio, obteniéndose 2 hijos.

- Se repite la operación hasta que se tenga una descendencia del tamaño de la población o solo quede un miembro.

- Si solo quedaba un almacén por cruzar, se selecciona aquel que tenga mejor fitness (priorizando si cubre el área mínima).

5.4.2. Mutación

El procedimiento de mutación es una adaptación del mostrado en la Figura 5.4. Se procede a detallar en su funcionamiento:

```
Mutacion()
{
    FOR(i = 0 TO totalPoblacion - 1)
    {
        IF(probabilidadAleatoria < probabilidadMutacion)
        {

            posiciones <- ElegirPosicionesAleatorias(X,Y,filasCandidatas)
            WHILE(antenasEscogidas.contiene(posiciones)){
                posiciones <- ElegirPosicionesAleatorias(X,Y,filasCandidatas)
            }

            celdaActual <- poblacion[i].celdas(posiciones)
            antenaPos <- celdaActual.posicionAntena

            IF(celdaActual.antena >= 0 AND poblacion[i] >= 2){
                poblacion[i].celdas(posiciones).posicionAntena <- -1
                poblacion[i].antenas.eliminar()

                FOR(a <- antenaPos TO poblacion[i].antenas.totalAntenas()){
                    antena <- poblacion[i].antenas[a]
                    poblacion[i].celdas(antena.coordenadas())
                }

            }ELSE{
                IF(antena.antenas >= maximoAntenas) CONTINUAR
                poblacion[i].celdas(posiciones).posicionAntena <- poblacion[i].numeroAntenas()
                antenaNueva <- crearAntena(posiciones)
                poblacion[i].antenas.Add(antenaNueva)
            }

            poblacion[i].AreaCubierta <- 0
            poblacion[i].calcularFitness()

        }
    }
}
```

Figura 5.11. Mutación Genética

Adaptada de "AP Deployment Optimization in Non-Uniform Service Areas: A Genetic Algorithm Approach" por Zhi, Z., Wu, J., Meng, X., Yao, M., Hu, Q., & Tang, Z. (2019)

- Se procede a explorar todos los elementos de la población.

- Se genera una probabilidad aleatoria y se verifica si es menor a la probabilidad de mutación. En caso no lo sea se continua al siguiente almacén. Si lo es, se continúa con el flujo.

- Se procede a buscar posiciones aleatorias que pertenezcan al centro de una celda (dentro de las filas candidatas) que no hayan sido previamente elegidas para una antena.

- Se procede a guardar la posición de la celda que corresponde.

- Se verifica si la celda posee una posición valida de antena, en caso no lo sea se crea la nueva antena y se añade a las antenas del almacén. Cabe destacar que se verifica si se tienen un mínimo de 2 antenas, de esta forma se evita tener almacenes con 0 antenas.

- En caso contrario se elimina la posición de la antena almacenada, se elimina la antena de las antenas de la población y se actualiza la posición de antena presente en las demás celdas que contienen antenas. Esta acción solo se realiza si el almacén no se encuentra en el máximo de antenas.

- Se calcula el nuevo fitness.

5.5. Pseudocódigo Algoritmo Calculo de Áreas

Durante la evaluación de este proyecto se encontró el problema de no tener una forma concreta y flexible para calcular el área de los círculos que representan la señal de las antenas dentro del almacén.

Tomando como base la división de una pantalla en pixeles iguales y la exploración de pixeles que tienen una etiqueta especifica así como la afirmación de que el área de componentes en la pantalla es la suma del área de los pixeles que los componen (Jain et al., 1995), se propone el siguiente algoritmo pero etiquetando cada pixel con la primera antena en la cual se encuentra que pertenece al área de su cobertura bajo la siguiente premisa para conseguir un área aproximada de las zonas en las cuales se alcanza señal por las antenas:

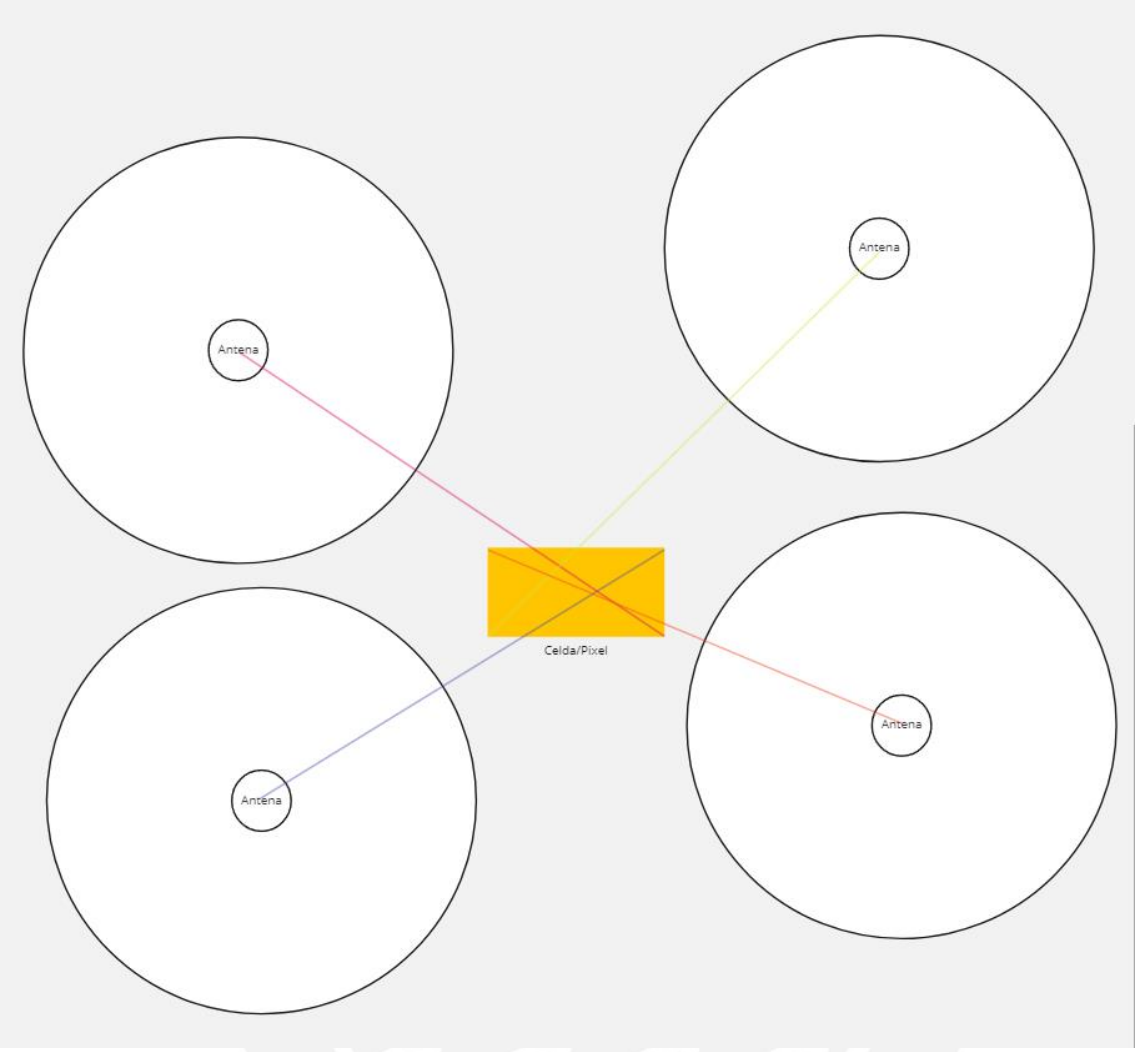


Figura 5.12. Ilustración de distancias entre una Celda/Pixel y las antenas del almacén

Se puede apreciar en la Figura 5.12. que el punto más lejano entre un rectángulo y un punto externo (para el caso, las posiciones de las antenas), siempre es aquel que tenga coordenadas opuestas al posicionamiento de las antenas y se encuentra en uno de los vértices (en caso de que una coordenada sea igual, serán 2 vértices del mismo lado de igual distancia hacia la antena).

De modo que, si se calcula la distancia entre el punto más lejano y la antena, se podrá saber si la celda/pixel se encuentra en el interior de la antena al compararla con el radio de esta y esto podría ser usado para el "etiquetado" del pixel o celda. El uso del conteo de celdas/pixeles que forman parte del área de una antena se inspira en el conteo de pixeles etiquetados en una pantalla que cumplen alguna característica (Jain et al., 1995).

Por tanto, se plantean las siguientes recursiones pertenecientes a la clase Almacén para el cálculo del área del almacén en la Figura 5.13 y se procede a explicar el flujo de la recursión.

La señal de las antenas será aproximada como un circulo. Se contó con la aprobación del especialista.

```
calcularArea()
{
    RETORNAR CalculoAreaFilaCelda(0)
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
CalculoAreaFilaCelda(columna)
{
    if (columna >= totalCeldasVertical) RETORNAR 0;
    preferencial <- -1
    if (columna > 0) preferencial <- celdas[columna - 1][0].antenaMasCercanaReciente;
    areaTotal <- CalculoPosicionCelda(0, columna, preferencial)

    RETORNAR areaTotal + CalculoAreaFilaCelda(columna + 1)
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
CalculoPosicionCelda(i, j, preferencial)
{
    if (i >= totalCeldasHorizontal) RETORNAR 0
    if (celdas[j][i].antena != -1) RETORNAR AreaCelda + CalculoPosicionCelda(i + 1, j, preferencial)
    ELSE
    {
        index <- antenaMasCercanaQueCubre(celdas[j][i].centroX, celdas[j][i].centroY, longCelda,
            anchoCelda,preferencial)
        preferencial <- index
        IF(index >= 0)
        {
            celdas[j][i].antenaMasCercanaReciente = index
            RETORNAR AreaCelda + CalculoPosicionCelda(i + 1, j, preferencial)
        }
        ELSE
        {
            double areaPixeles <- CalculoAreaFilaPixel(0,i*longCelda,j*anchoCelda)
            RETORNAR areaPixeles + CalculoPosicionCelda(i + 1, j, -1)
        }
    }
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
CalculoAreaFilaPixel(pixelAncho, distX, distY)
{
    if (pixelAncho >= PixelesXLado) RETORNAR 0

    areaTotal <- CalculoPosicionPixel( longPixel, anchoPixel,
            radioAntena,  0, pixelAncho, distX, distY, -1)

    RETORNAR areaTotal + CalculoAreaFilaPixel(pixelAncho + 1, distX, distY)
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
CalculoPosicionPixel(pixelLargoPos, pixelAnchoPos, distCeldaX, distCeldaY, preferencial2)
{
    if (pixelLargoPos >= PixelesXLado) RETORNAR 0

    indexP <- antenaMasCercanaQueCubre(distCeldaX + (longPixel*(pixelLargoPos+0.5)), distCeldaY + (anchoPixel*(pixelAnchoPos+0.5)),
        longPixel, anchoPixel, preferencial2)

    if (indexP >= 0)
    {
        preferencial2 <- indexP
        RETORNAR AreaPixel + CalculoPosicionPixel(pixelLargoPos+1, pixelAnchoPos,
            distCeldaX, distCeldaY, preferencial2)
    }
    ELSE
    {
        RETORNAR CalculoPosicionPixel(pixelLargoPos + 1, pixelAnchoPos, distCeldaX, distCeldaY, -1)
    }
}
```

Figura 5.13. Exploración recursiva de celdas y pixeles

- Se inicia el llamado desde el exterior al cálculo del área, lo cual llama al cálculo del área de una fila de celdas que se muestra con la posición cero (por ser la primera fila).

- **En el cálculo de una Fila de Celdas:** Se evalúa la condición de parada que verifica si esta es la última fila y se debe regresar. En caso no se cumpla, se debe inicializar un valor de la posición de la antena preferencial a ser revisada que es la última antena revisada por la celda superior. Si es la primera columna, no se evaluará ninguna (representado por la posición -1). Se envía la fila actual al método que realizará el cálculo de filas de celdas (inicializando en la celda horizontal inicial) y una vez esta retorne el área de la línea, esta se envía a la siguiente columna.

- **En el cálculo de una Posición de Celdas:** Se evalúa la condición de parada que verifica si esta es la última columna y se debe regresar. En caso no se cumpla, se verifica si es que se tiene una antena en esa celda (cosa que inmediatamente determina que la celda está cubierta y se le incrementa su área).

- En caso no se cumpla esta condición, se procede buscar la antena más cercana que cubre esta celda, enviando las coordenadas del centro y las dimensiones de la celda junto con la antena preferencial más cercana. Se usa este método que se encuentra mejor definido en el Anexo E.2.1.

- Si no se encuentra una antena, se procede a iniciar la exploración de pixeles enviando las coordenadas de la celda para calcular el área de esta que se encuentre dentro de alguna antena.

- Una vez calculada el área de la celda, se explora la siguiente celda horizontal.

- **En el cálculo de una Fila de pixeles:** Se evalúa la condición de parada que verifica si esta es la última fila de pixeles en la celda y se debe regresar. En caso no se cumpla, se inicializa y no se evaluará ninguna antena preferencial (representado por la posición -1). Se envía la fila actual al método que realizará el calculó de filas de pixeles (inicializando en el pixel horizontal inicial de la celda) y una vez esta retorne el área de la línea, esta se envía a la siguiente columna.

- **En el cálculo de una Posición de pixeles:** Se evalúa la condición de parada que verifica si esta es la última columna en la celda y se debe regresar.

- En caso no se cumpla esta condición, se procede buscar la antena más cercana que cubre este pixel, enviando las coordenadas del centro y las dimensiones del pixel junto con la antena preferencial más cercana. Se usa este método que se encuentra mejor definido en el Anexo E.2.1.

- Si no se encuentra una antena, no se incrementa el valor del área del pixel pasando a ser cero.

- Una vez calculada el área del pixel, se explora el siguiente pixel horizontal.

5.6. Discusión

En esta sección fueron presentados los elementos que son la base de la implementación del algoritmo del proyecto, fueron tomada como base de su diseño los conceptos base para el desarrollo de un algoritmo memético tales como la búsqueda local y cooperación para el desarrollo exhaustivo de soluciones candidatas (Moscato & Cotta, 2003). Se busca cumplir con el resultado esperado R3 del objetivo específico O2, así como dejar el planteamiento para cumplir con los resultados esperados R4 y R5 de los objetivos específicos O2 y O3 respectivamente. Esta sección fue validada y aprobada por un especialista en Ciencias de la Computación y se presenta su autorización en el anexo D.

6. Implementación

En este capítulo se explora la materialización en formato de software de los algoritmos planteados para resolver la problemática. El proyecto que contiene el código fuente se encuentra adjunto en el anexo F.

6.1. Implementación del Algoritmo Memético

La implementación del algoritmo memético se evidencia como el código fuente que toma como base el diseño presentado en el capítulo 5. Se toman las 4 clases diseñadas (Algoritmo, Almacén, Antena y Celda) para crear la abstracción que permite tener como entradas a los parámetros no auxiliares indicados en el capítulo anterior.

Los métodos más importantes son aquellos relacionados con el cálculo del área, la mutación memética, la evolución y la competencia.

La importancia del cálculo del área radica en que es una acción que se realiza en múltiples ocasiones a lo largo de la ejecución y su lentitud puede ocasionar un cuello de botella bastante ralentizador.

Es por esta misma razón que se optó por un flujo recursivo que permite determinar el área de la Celda solo en determinadas ocasiones.

El número de antenas presentes en el movimiento dentro de la evolución se inspiró en el concepto de acciones dentro de la búsqueda local que deben recurrir únicamente en mejoras al igual que en las artes marciales solamente se aprenden movimientos que permitan mejorar el desempeño en vez de perder el aprendizaje adquirido (Moscato, 1989).

Bajo este mismo concepto es que se aplica la mutación memética que permite incrementar el área cubierta en caso se detecte que la configuración actual no puede cubrir el área mínima.

6.2. Implementación del Algoritmo Genético

Por otro lado, la implementación del algoritmo genético tomó como base el planteamiento ya existente en la bibliografía (Zhi et al., 2019).

Su implementación realiza el uso de los mismos métodos y objetos que el algoritmo memético (exceptuando aquellos que fueron creados exclusivamente para el diseño del memético) ya que su implementación se encuentra en la misma clase Algoritmo que hace uso de su llamada.

6.3. Métodos, medios de verificación

Inicialmente se planteó el uso de un lenguaje de programación de menor nivel como C o C++, pero se optó por C# y el entorno .NET debido a que tiene buen rendimiento y su paradigma orientado a objetos que lo hace más sencillo de entender y de no perder el flujo del código. Por otro lado, el tesista constantemente realiza proyectos en este entorno, permitiendo un conocimiento de métodos y clases que ayudan a mejorar el rendimiento. Por otro lado, la

metodología SCRUM para distribuir el tiempo en conjunto con el cronograma planteado para el proyecto han apoyado enormemente a la correcta organización de los tiempos para el desarrollo adecuado el código fuente.

En ambos algoritmos se plantea el uso de la consola para presentar los datos correspondientes a la solución óptima y se muestra un ejemplo de salida en la Figura 6.1.

```
El fitness de esta solucion es: 100.75280713012478
Teniendo de area cubierta: 7528.07130124777
Cubriendo un %75.28071301247769 del area del almacen que es 10000
Teniendo la distribucion siguiente:
    Antena 0 con coordenadas(43.93939393939395,45.58823529411764)
```

Figura 6.1. Ejemplo de resultado

Donde se es posible visualizar aquellos elementos que son importantes respecto a la solución obtenida al margen del algoritmo los cuales son:

Fitness: Permite determinar el nivel de fiabilidad de la solución.

Área cubierta: Área en metros cuadrados que se logró cubrir con la configuración actual.

Porcentaje del área: Acompañada del área total del almacén, permite visualizar que parte del total se logró cubrir con la configuración actual.

Distribución de Antenas: Permite ver el número de antenas y las coordenadas en las cuales se recomienda su ubicación para conseguir los resultados planteados.

6.4. Interfaz gráfica

Con el fin de presentar e introducir los datos de manera más sencilla, flexible y ordenada se realizó una interfaz gráfica consistente en 2 pantallas:

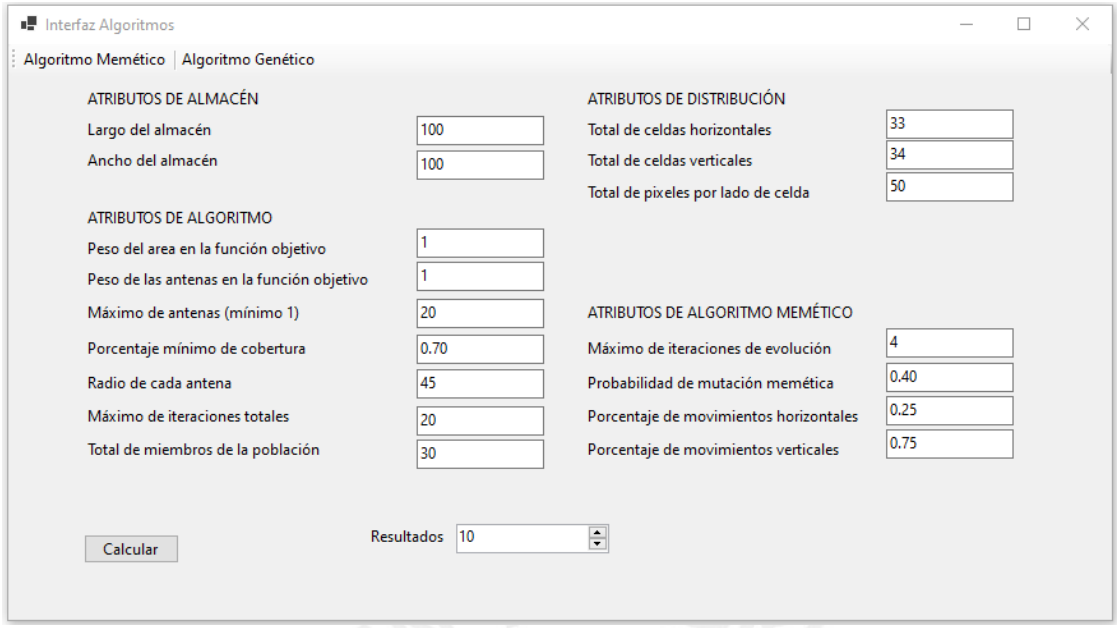


Figura 6.2. Pantalla de ingreso de parámetros para el algoritmo memético

En la Figura 6.2 se puede apreciar la pantalla para el ingreso de parámetros para el algoritmo memético. El algoritmo a elegir se puede seleccionar en la barra de herramientas y la sección de atributos cambiará en función del algoritmo seleccionado.

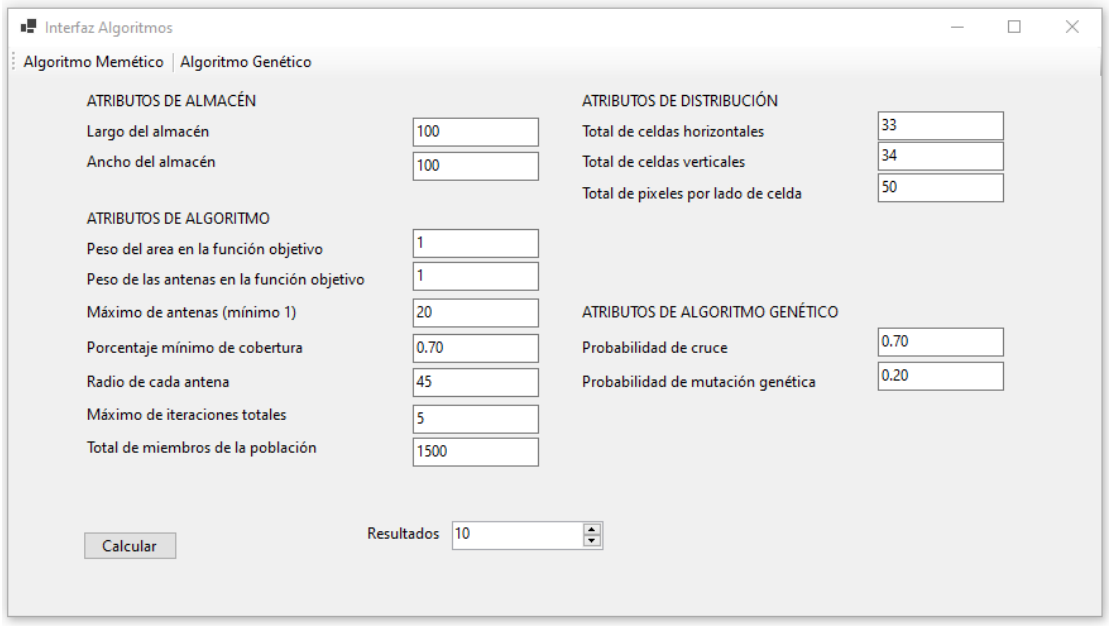


Figura 6.3 Pantalla de ingreso de parámetros para el algoritmo genético

En la Figura 6.3 se aprecia el cambio en la pantalla para la selección del algoritmo genético.

Para comenzar la ejecución se bloquean los elementos hasta la aparición de la pantalla de resultados.

Para ambos algoritmos, se agregó una cantidad de resultados a partir de las cuales se elegirá

aquel resultado con mejor fitness que cumpla con el área mínima

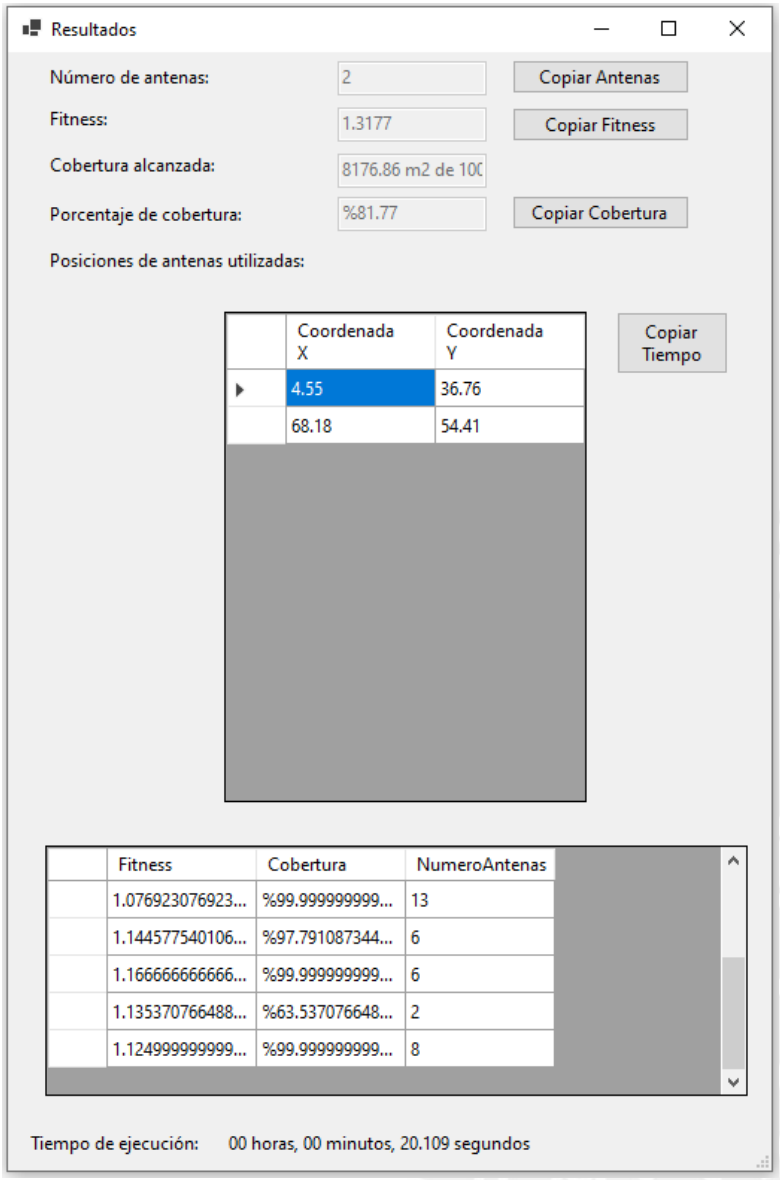


Figura 6.4. Pantalla de resultados para el algoritmo elegido

La pantalla de resultados se muestra en la Figura 6.4, aparecen los mismos elementos que en la presentación por consola, pero mostrando las coordenadas en celdas y mostrando el tiempo de ejecución total para la cantidad de resultados elegida con el algoritmo elegido. Siendo mostrados los atributos totales del mejor resultado que cumple con el área mínima, o aquel mejor resultado que no alcance a cumplir con el área mínima en su defecto en caso ninguno logre alcanzar esta área.

Se muestran todos los parámetros solo de la mejor solución, así como el fitness, cobertura y numero de antenas para todos los demás resultados obtenidos.

Para iniciar otra ejecución basta con cerrar la pantalla de resultados.

Es importante destacar que la ejecución por consola todavía es realizable cambiando el

proyecto inicial a ejecutarse en la solución de todos los proyectos en Visual Studio así como usando el archivo ejecutable que se encuentra en las carpetas de la solución.

6.5. Discusión

Tanto el modo de ejecución por consola como por interfaz gráfica fueron desarrollados teniendo en cuenta distintos objetivos. Siendo la fácil depuración e identificación de errores en el caso de la primera, así como el cambio sencillo de parámetros y la realización de múltiples ejecuciones para la segunda.

Originalmente la interfaz gráfica solo iba a realizar la ejecución de un resultado, pero la necesidad de múltiples resultados tanto para las etapas de calibración como en la toma de muestras de la experimentación volvieron necesaria esta funcionalidad a fin de cumplir con el cronograma planteado para su desarrollo.

Se utilizaron listas de C# con el fin de representar todos los arreglos (incluidos los de 2 dimensiones) debido a que cuentan con distintos métodos nativos que permiten su fácil acceso y control.

Se eligió presentar el área cubierta ya presentando el porcentaje de cobertura con el fin de visualizar rápidamente cuanta área se ha cubierto al margen del porcentaje que represente del área total.

La herramienta de depuración de Visual Studio fue muy útil para buscar y corregir anormalidades.

Con el desarrollo de esta busca cumplir con los resultados esperados R4 y R5 de los objetivos específicos O2 y O3 respectivamente. El código fuente y su funcionamiento fue validado y aprobado por un especialista en Ciencias de la Computación y se presenta su autorización en el anexo D.

7. Comparativa entre algoritmos

El propósito de este capítulo es presentar la comparación del desempeño de ambos algoritmos. Inicialmente se presentan las pruebas del funcionamiento. Seguido de la calibración de los parámetros con el fin de evaluar que parámetros se deben utilizar para la comparativa final. Esta comparación con los parámetros y resultados elegidos será empleada para el proceso de experimentación numérica de los datos obtenidos.

7.1. Pruebas de funcionamiento por caja negra

Con el fin de comprobar del funcionamiento inicial se realizaron pruebas con parámetros establecidos.

La información devuelta por las pruebas permite tener una primera impresión de la naturaleza de los resultados y logra comprobar que el programa está cumpliendo con su ejecución.

El detalle se presenta en el Anexo G.

7.2. Calibración de parámetros

Con el fin de obtener los parámetros más adecuados, se realiza su calibración. Esto es necesario para obtener resultados útiles que serán usados en la experimentación numérica.

Los valores empleados para la calibración se encuentran en el Anexo H. Fueron empleadas 30 resultados por cada valor evaluado en cada elemento de la calibración.

Algunos parámetros fueron elegidos usando información general así como la explicación del uso de las dimensiones del almacén así como el uso de un radio de 45 metros se encuentra en el Anexo G. Se presentan los parámetros fijos:

- El ancho y largo del almacén será de 100 metros de manera similar a lo propuesto en el escenario del algoritmo genético adaptado (Zhi et al., 2019).
- La longitud de celdas será de 33 celdas por largo y 34 por ancho. Esto fue conversado y decidido en consenso con el asesor con el fin de obtener celdas con un lado de aproximadamente de 3 metros en cada lado que se consideró una cantidad adecuada y fue sugerida por el especialista. Son 34 celdas de ancho con el fin de tener una última fila donde se pueda colocar antenas en el almacén.
- Se tiene 1 fila candidata seguida de 2 filas de mercadería hasta cumplir las 33 primeras filas. De esta manera se tiene un escenario similar de 429 celdas al del escenario propuesto de la bibliografía donde se usan 400 celdas de 5 metros de lado (Zhi et al., 2019).
- Se tienen 50 pixeles por lado, lo cual permite tener una precisión decente sin comprometer demasiado el tiempo de ejecución por el cálculo de áreas. Se tendría una totalidad de 2805000 pixeles para el cálculo del área lo cual es aceptable debido a que según el criterio del especialista sería suficiente una cantidad superior a 10000 pixeles.
- Se tiene un máximo de 20 antenas para visualizar su comportamiento, esta cantidad será variada durante la etapa de la experimentación numérica.
- Se tiene 45 metros de radio de señal para las antenas, esto es discutido en el Anexo G.
- Se iniciará con 10 iteraciones generales para ambos casos, 3 iteraciones evolutivas, 0.5 para

todas las probabilidades y porcentajes meméticos para verificar el comportamiento del algoritmo bajo circunstancias neutrales. Población inicial de 10 individuos en el caso del algoritmo memético debido a que durante la evaluación de la bibliografía se encontró que una población baja puede ayudar al desarrollo de algunos algoritmos (Haupt, 2000). Además de que el tiempo de ejecución en la población memética impacta de mayor manera. En el algoritmo genético se evaluará esta posibilidad, pero las calibraciones iniciales se llevarán a cabo con 100 individuos. La probabilidad de cruzamiento inicial será alta y la de mutación será baja ambas por recomendación del especialista en el caso del algoritmo genético.

- La data obtenida para el proceso de calibración se encuentra en el Anexo H.1.

Asimismo, se presenta un ejemplo de un almacén con las filas candidatas y de mercadería en la Figura 7.1.

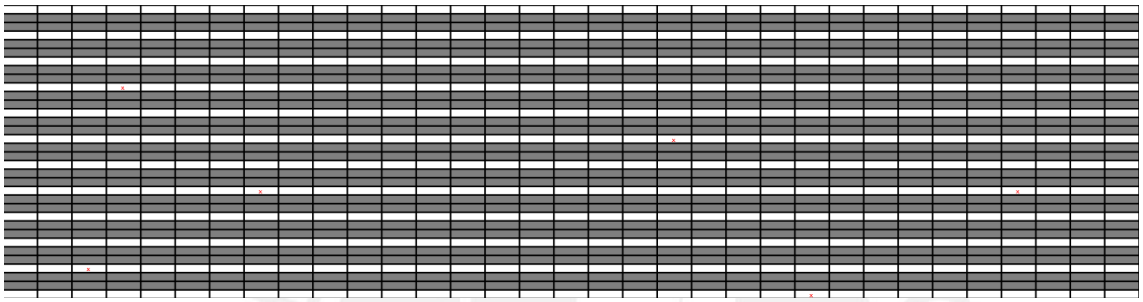


Figura 7.1. Almacén con filas de mercadería en gris, es en el resto donde se puede posicionar antenas

Se procede a presentar la calibración de los diferentes parámetros.

7.2.1. Porcentaje de movimientos X

Se procede a calibrar la el porcentaje de movimientos X en la Tabla 7.1.

Tabla 7.1. Porcentaje de movimientos X

Porcentaje de movimientos X	Algoritmo Memético	
	Fitness Promedio	Mejor Fitness
0.10	1.31733998	1.35634296
0.25	1.32449035	1.35634296
0.50	1.322638027	1.356342959
0.75	1.30498030	1.35634296

La cantidad de movimientos en el eje horizontal elegida será de 0.25 ya que tiene mejor fitness promedio y su fitness máximo es un empate en la mayoría de los casos.

7.2.2. Porcentaje de movimientos y

Se procede a calibrar la el porcentaje de movimientos Y en la Tabla 7.2.

Tabla 7.2. Porcentaje de movimientos Y

Porcentaje de movimientos Y	Algoritmo Memético	
	Fitness Promedio	Mejor Fitness
0.25	1.29808897	1.35634296
0.50	1.31582067	1.35531194
0.75	1.32845724	1.35634296
1.00	1.30012189	1.35555080

Se obtuvo un fitness promedio al explorar todas las filas verticales tanto por ambos lados y esto es representado por un porcentaje de movimientos vertical del 100%.

7.2.3. Probabilidad de cruzamiento

Se procede a calibrar la probabilidad de cruzamientos en la Tabla 7.3.

Tabla 7.3. Probabilidad de cruzamiento

Probabilidad de cruzamiento	Algoritmo Genético	
	Fitness Promedio	Mejor Fitness
0.60	1.184374678	1.274987522
0.70	1.203640716	1.323684135
0.80	1.194617929	1.325961141

Se inicio con un cruzamiento del 70% debido a que fue recomendación del especialista comenzar con un cruzamiento alto, lo cual fue un buen inicio ya que en el valor de 0.70 se tiene fitness promedio mayor que en los alrededores.

7.2.4. Probabilidades de mutación

Se procede a calibrar la probabilidad de mutaciones para los algoritmos memético y genético en la Tablas 7.4 y 7.5 respectivamente.

Tabla 7.4. Probabilidad de mutaciones para algoritmo memético.

Probabilidad de mutaciones	Algoritmo Memético	
	Fitness Promedio	Mejor Fitness
0.10	1.317165419	1.356342959
0.20	1.309947011	1.356342959
0.30	1.313240416	1.355311943
0.40	1.32553405	1.356342959
0.50	1.314309412	1.355550802
0.60	1.32071773	1.356342959

Tabla 7.5. Probabilidad de mutaciones para algoritmo genético

Probabilidad de mutaciones	Algoritmo Genético	
	Fitness Promedio	Mejor Fitness
0.10	1.1965571	1.2994228
0.20	1.2079756	1.2917501
0.30	1.1869896	1.3066446

En el caso de la mutación memética se procedió a evaluar con un valor de 0.50 porque durante las diversas ejecuciones iniciales se necesitaba de la mutación para ayudar a superar el área mínima. Pero se obtuvo que un valor de 0.40 obtiene un fitness promedio mayor.

Por otro lado, la mutación genética se inició con un valor de 0.20 ya que fue recomendación del especialista iniciar con valores bajos. De modo que un valor de 0.20 resulto ser acertado a comparación de los valores alternativos con un fitness promedio superior a las alternativas superiores e inferiores.

7.2.5. Iteraciones evolutivas

Se procede a calibrar la cantidad de iteraciones evolutivas en la Tabla 7.6.

Tabla 7.6. Iteraciones evolutivas

Iteraciones evolutivas	Algoritmo Memético	
	Fitness Promedio	Mejor Fitness
1	1.303151575	1.355311943
2	1.312625775	1.356342959
3	1.312481236	1.355550802
4	1.31707163	1.35531194
5	1.31307679	1.35531194

En el caso de las iteraciones evolutivas que determinan la cantidad de movimientos realizados antes de la mutación memética, así como después de esta (y antes de la competencia), se encontró que una cantidad de 4 movimientos es de mayor utilidad para el algoritmo al obtenerse el valor 2 con un fitness promedio superior a las demás opciones.

7.2.6. Iteraciones generales

Se procede a calibrar la cantidad de iteraciones generales para ambos algoritmos en las Tablas 7.7 y 7.8.

Tabla 7.7. Iteraciones generales para algoritmo memético

Iteraciones generales	Algoritmo Memético	
	Fitness Promedio	Mejor Fitness
5	1.282506	1.354785
10	1.311640	1.356343
15	1.322817	1.356343
20	1.334215	1.355312
25	1.325993	1.356343

Tabla 7.8. Iteraciones generales para algoritmo genético

Iteraciones generales	Algoritmo Genético	
	Fitness Promedio	Mejor Fitness
5	1.235309	1.330126
10	1.190525	1.281499
15	1.189147	1.335501
20	1.160628	1.267729

Se inicio la exploración en 10 iteraciones para ambos casos y se inició la exploración. Se puede observar que un aumento en las iteraciones mejora el fitness promedio para el caso del algoritmo memético siendo de manera inversa en el caso de su algoritmo de comparación. Para el algoritmo memético se alcanza el fitness promedio mayor en las 20 iteraciones y no difiere mucho del fitness promedio en las 25 iteraciones. Por lo cual se optó por detenerse en las 20 iteraciones.

En el caso del algoritmo genético se detuvo la exploración en 20 iteraciones debido a que tanto el fitness promedio mayor fue obtenidos con este valor y se puede apreciar que las iteraciones reducen el fitness.

7.2.7. Miembros de la población memética

Se procede a calibrar la cantidad de miembros de la población y se aprecian los resultados en la Tabla 7.9.

Tabla 7.9. Miembros de la población memética

Miembros de la población	Algoritmo Memético	
	Fitness Promedio	Mejor Fitness
5	1.323523	1.356343
10	1.335994	1.356343
15	1.339036	1.356343
20	1.345824	1.356343
25	1.348973	1.356343
30	1.353749	1.356343

En la revisión de la bibliografía se encontró que un valor cercano a 16 puede ser bueno para la población de un algoritmo genético con una tasa de mutación entre 5% y 20% (Haupt, 2000). Por tanto, se intentó evaluar con un valor cercano para estas circunstancias a pesar de tener un 40% de porcentaje de mutación. De modo que sí se obtuvo un buen resultado inicial, pero se encontró que aumentar la población sí beneficia al algoritmo. De modo que se escogió una población de 30 debido a que la variación entre los fitness promedio ya no es muy alta con respecto a los valores anteriores. Se puede observar que el mejor fitness es un empate en todos los casos.

7.2.8. Miembros de la población genética

Se procede a calibrar la cantidad de miembros de la población y se aprecian los resultados en la Tabla 7.10.

Tabla 7.10. Miembros de la población genética

Miembros de la población	Algoritmo Genético	
	Fitness Promedio	Mejor Fitness
15	1.228627	1.344685
30	1.172740	1.292671
65	1.274199	1.351012
100	1.227055	1.335041
500	1.281101	1.350346
1000	1.306154	1.347676
1500	1.318480	1.344400

En el caso del algoritmo genético, no se encontró un buen fitness promedio y máximo para el valor de 15 miembros poblacionales. En respuesta se intentó incrementar el número de miembros de la población y se tomó el valor de 1500 miembros como tamaño máximo de población obteniéndose un fitness promedio y máximo buenos, pero sin superar a la alternativa memética para este escenario.

7.3. Discusión

No se intentó incrementar la población hacia una cantidad mayor para ambos algoritmos debido a que el tiempo de ejecución en ambos casos ya es bastante alto con la calibración final. El tiempo obtenido en la calibración del algoritmo memético es mayor que el genético, pero durante la etapa de experimentación se encontrará que el tiempo de ejecución en el algoritmo genético se incrementa más rápido que en su contraparte memética. De modo que se eligió el valor de 1500 población genética.

Es posible que el tiempo de ejecución se reduzca en caso se realice solo una ejecución del programa a la vez. Sin embargo, para obtener los tiempos que se presentan en los anexos se utilizaron diversas ejecuciones simultaneas en distintos equipos tanto para esta sección como para las muestras usadas en la experimentación.

La validación de los valores iniciales por parte del especialista se presenta en el anexo D. A

raíz de los resultados obtenidos es posible comenzar a diseñar la experimentación numérica para la comparación final de resultados. Con esta selección y sin todavía evaluar mediante la experimentación, el algoritmo memético está presentando mejores resultados que su algoritmo de comparación. Con esta sección es posible cumplir con el resultado esperado R6 del objetivo específico O3.

8. Experimentación Numérica

Con las calibraciones realizadas se puede realizar el proceso de experimentación numérica. En este proyecto se realizará empleando pruebas estadísticas que permitirá evaluar de manera general los resultados generados por ambos algoritmos.

8.1. Muestras empleadas

Se obtuvieron 30 muestras para ambos algoritmos seleccionando el mejor valor fitness después de 10 ejecuciones con las cuales se busca realizar las pruebas estadísticas mencionadas iniciando con la prueba de normalidad de Shapiro-Wilk.

El tamaño del almacén se mantiene constante debido a que por lo general un almacén no cambia su tamaño y el tamaño es bastante estándar en base a almacenes grandes como fue mencionado anteriormente.

Se probo con los valores de 5, 10, 15, 20 y 25 para la cantidad de antenas máximas combinados con los valores de 47, 45, 43, 41, 39 y 37 metros de radio de todas las antenas con el fin de evaluar una atenuación en las antenas. Se presentan las muestras utilizadas en el Anexo H.2. Se presentan los datos generales de las muestras en la Tabla 8.1.

Tabla 8.1. Datos generales de las muestras

Algoritmo	Memético	Genético
Media Fitness	1.313148	1.301859
Mediana Fitness	1.307882	1.299885
Mejor Fitness	1.394879	1.394879
Desviación Estándar Fitness	0.05211745	0.0548569

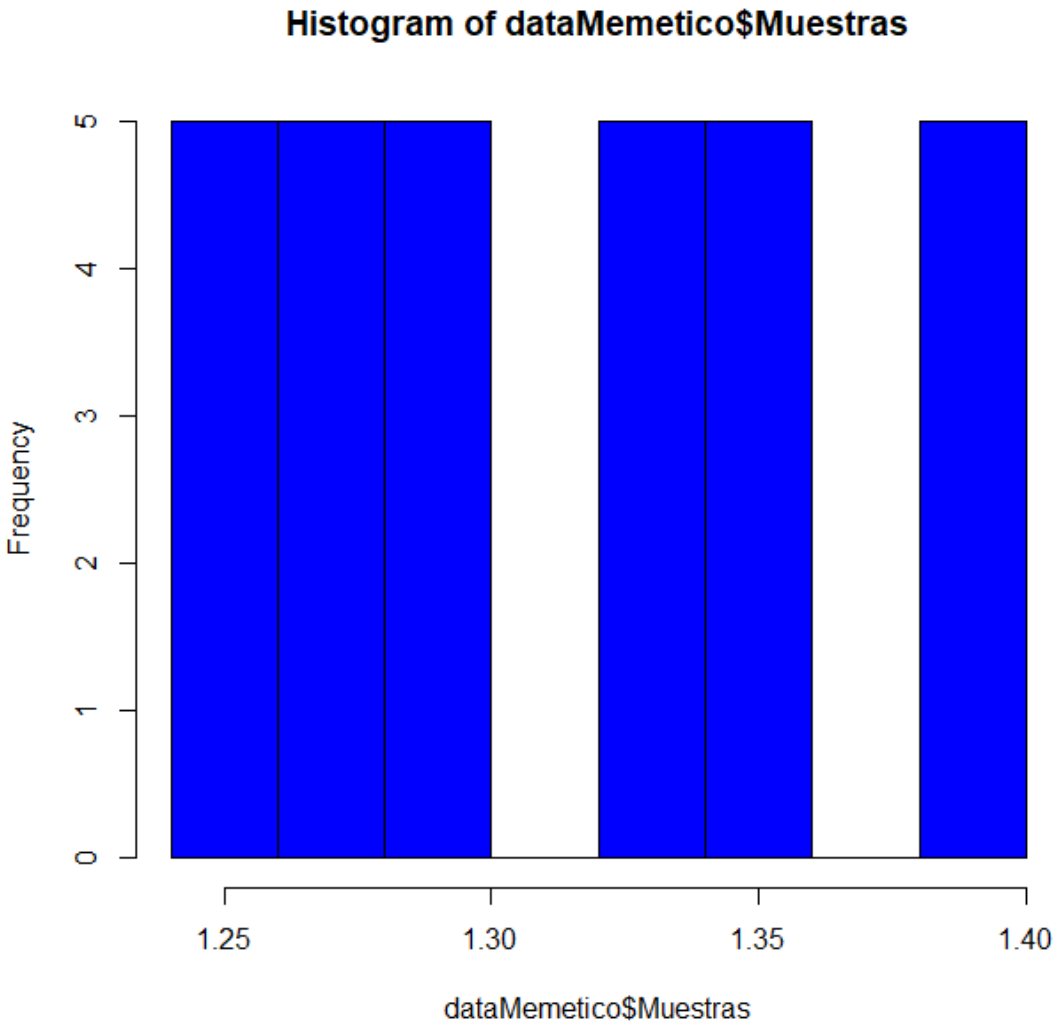


Figura 8.1. Histograma de los resultados del algoritmo memético

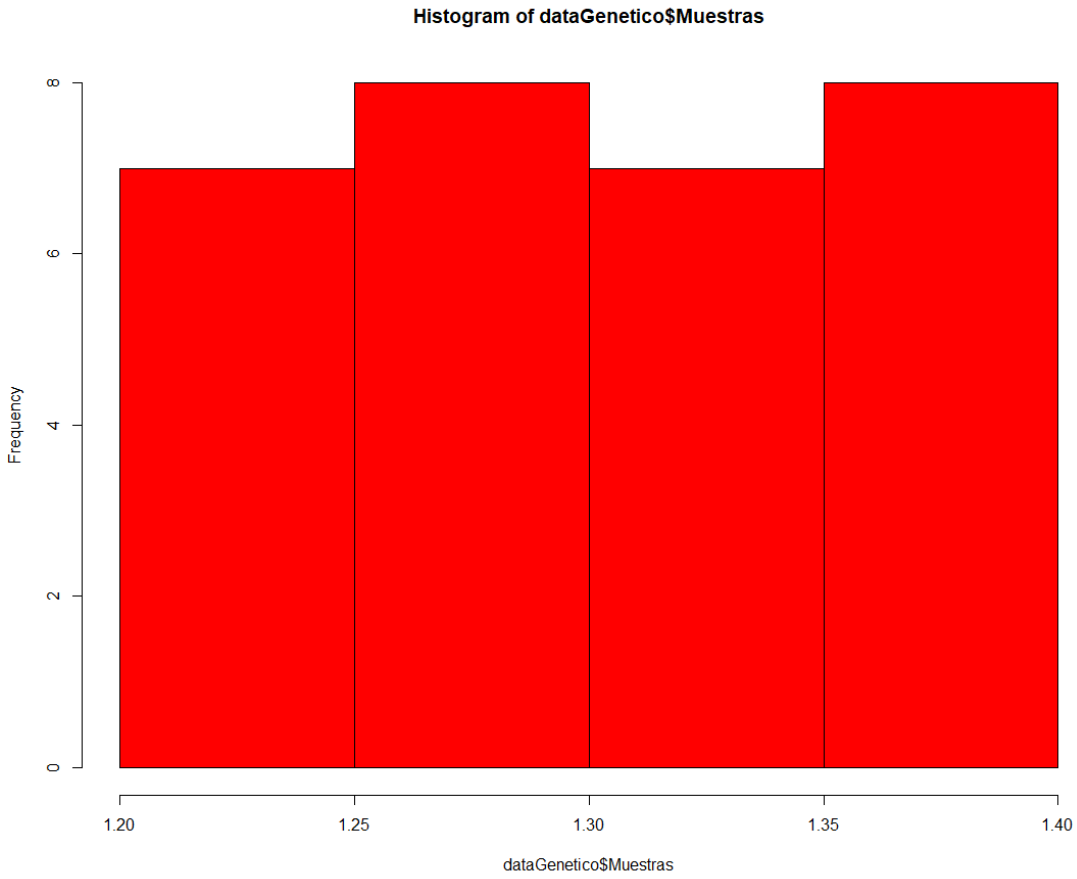


Figura 8.2. Histograma de los resultados del algoritmo genético

En el caso del algoritmo memético se encuentra una menor dispersión en los datos siendo el promedio y la mediana reducidos por los casos donde el fitness obtenido no fue tan bueno en el caso del algoritmo genético.

Sin embargo, hay un factor importante a considerar que es que las soluciones convergen para los mismos radios en con el número máximo de antenas distintos en el caso del algoritmo genético. Hay algunas soluciones obtenidas con 3 antenas que obtienen una cobertura muy alta pero que no alcanzan el mismo nivel de fitness por que el uso de 1 antena adicional y estas son las que reducen el fitness en ambos casos y aumentan la dispersión de las iteraciones previas de donde se selecciona el fitness máximo.

Debido a que el caso base del radio de 45 metros y 20 antenas máximas ya fue evaluado en la etapa de calibración, se usan los tiempos obtenidos en 30 resultados para estimar el tiempo de 10 resultados.

Las muestras obtenidas pasarán distintas pruebas con el objetivo de evaluar los datos:

8.2. Prueba de Shapiro-Wilk

Se realizará esta prueba con el objetivo de determinar si hay normalidad en las muestras.

Para ambos casos:

H0: Hay normalidad en las muestras.
H1: No hay normalidad en las muestras.

```
> shapiro.test(dataMemetico$Muestras)

        Shapiro-Wilk normality test

data:  dataMemetico$Muestras
W = 0.8845, p-value = 0.003583
```

Figura 8.3. Prueba de normalidad para el algoritmo memético

Con un nivel de significancia del 0.05, se rechaza la hipótesis nula de que hay normalidad en los datos en las muestras del algoritmo memético. Pues, el p-valor es menor al grado de significancia.

```
> shapiro.test(dataGenetico$Muestras)

        Shapiro-Wilk normality test

data:  dataGenetico$Muestras
W = 0.93683, p-value = 0.07473
```

Figura 8.4. Prueba de normalidad para el algoritmo genético

Con un nivel de significancia del 0.05, no se rechaza la hipótesis nula de que hay normalidad en los datos en las muestras del algoritmo genético. Pues, el p-valor es mayor al grado de significancia.

Debido a que no en ambas muestras se presenta normalidad, pero sería posible asumirla puesto que se tienen 30 muestras y el p-valor obtenido para las pruebas no paramétricas será aproximado porque hay muestras con valores iguales, se procede a realizar una prueba para verificar el requisito de homocedasticidad para descartar por completo el uso de prueba Z.

8.3. Prueba de F de Fisher

Se procede a realizar la prueba

H0: Las varianzas muestrales presentan homogeneidad.
H1: Las varianzas muestrales no presentan homogeneidad.

```
> var.test(dataMemetico$Muestras,dataGenetico$Muestras,alternative="two.sided")

        F test to compare two variances

data:  dataMemetico$Muestras and dataGenetico$Muestras
F = 0.90262, num df = 29, denom df = 29, p-value = 0.7845
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.4296142 1.8963958
sample estimates:
ratio of variances
         0.9026176
```

Figura 8.5. Prueba de homocedasticidad en las varianzas para ambas muestras

Con un nivel de significancia del 0.05, no se rechaza la hipótesis nula de que las varianzas muestrales presentan homogeneidad. Pues, el p-valor es mayor al grado de significancia.

Primero se procede a evaluar los resultados mediante una prueba no paramétrica.

8.4. Prueba de U Mann-Whitney

Se procede a plantear 3 pruebas, siendo la primera para comprobar si la alternativa memética es mejor, si es peor o si es igual en resultados con las siguientes hipótesis:

8.4.1. Prueba no paramétrica de victoria del algoritmo memético

H0: La mediana del algoritmo memético no supera a la mediana del algoritmo genético.
H1: La mediana del algoritmo memético supera a la mediana del algoritmo genético.

```
> wilcox.test(dataMemetico$Muestras,dataGenetico$Muestras,alternative = "greater")

        Wilcoxon rank sum test with continuity correction

data:  dataMemetico$Muestras and dataGenetico$Muestras
W = 532, p-value = 0.1138
alternative hypothesis: true location shift is greater than 0
```

Figura 8.6. Prueba de U Mann-Whitney para verificar si la mediana memética es superior

Con un nivel de significancia del 0.05, no se rechaza la hipótesis nula de que la mediana del algoritmo memético no supera a la mediana del algoritmo genético. Pues, el p-valor es mayor al grado de significancia.

8.4.2. Prueba no paramétrica de derrota del algoritmo memético

H0: La mediana del algoritmo memético no es superada por la mediana del algoritmo genético.
H1: La mediana del algoritmo memético es superada por la mediana del algoritmo genético.

```
> wilcox.test(dataMemetico$Muestras,dataGenetico$Muestras,alternative = "less")

        Wilcoxon rank sum test with continuity correction

data:  dataMemetico$Muestras and dataGenetico$Muestras
W = 532, p-value = 0.889
alternative hypothesis: true location shift is less than 0
```

Figura 8.7. Prueba de U Mann-Whitney para verificar si la mediana memética es inferior

Con un nivel de significancia del 0.05, no se rechaza la hipótesis nula de que la mediana del algoritmo memético no es superada por la mediana del algoritmo genético. Pues, el p-valor es mayor al grado de significancia.

8.4.3. Prueba no paramétrica de empate del algoritmo memético

H0: La mediana del algoritmo memético es igual a la mediana del algoritmo genético.
H1: La mediana del algoritmo memético es diferente a la mediana del algoritmo genético.

```
> wilcox.test(dataMemetico$Muestras,dataGenetico$Muestras,alternative = "two.sided")

        Wilcoxon rank sum test with continuity correction

data:  dataMemetico$Muestras and dataGenetico$Muestras
W = 532, p-value = 0.2276
alternative hypothesis: true location shift is not equal to 0
```

Figura 8.8. Prueba de U Mann-Whitney para verificar si la mediana memética es igual

Con un nivel de significancia del 0.05, no se rechaza la hipótesis nula de que la mediana del algoritmo memético es igual a la mediana del algoritmo genético. Pues, el p-valor es mayor al grado de significancia.

8.5. Prueba Z

Se procede a plantear 3 pruebas, siendo la primera para comprobar si la alternativa memética es mejor, si es peor o si es igual en resultados con las siguientes hipótesis:

8.5.1. Prueba paramétrica de victoria del algoritmo memético

H0: La media del algoritmo memético no supera a la media del algoritmo genético.
H1: La media del algoritmo memético supera a la media del algoritmo genético.

```
> z.test(x=dataMemetico$Muestras, y=dataGenetico$Muestras, mu=0, sigma.x=sd(dataMemetico$Muestras), sigma.y=sd(dataGenetico$Muestras),alternative = "greater")

        Two-sample z-Test

data:  dataMemetico$Muestras and dataGenetico$Muestras
z = 0.81715, p-value = 0.2069
alternative hypothesis: true difference in means is greater than 0
95 percent confidence interval:
 -0.01143463          NA
sample estimates:
mean of x mean of y
 1.313148  1.301859
```

Figura 8.9. Prueba Z para verificar si la media memética es superior

Con un nivel de significancia del 0.05, no se rechaza la hipótesis nula de que la media del algoritmo memético no supera a la mediana del algoritmo genético. Pues, el p-valor es mayor al grado de significancia.

8.5.2. Prueba paramétrica de derrota del algoritmo memético

H0: La media del algoritmo memético no es superada por la media del algoritmo genético.
H1: La media del algoritmo memético es superada por la media del algoritmo genético.

```
> z.test(x=dataMemetico$Muestras, y=dataGenetico$Muestras, mu=0, sigma.x=sd(dataMemetico$Muestras), sigma.y=sd(dataGenetico$Muestras),alternative = "less")

        Two-sample z-Test

data:  dataMemetico$Muestras and dataGenetico$Muestras
z = 0.81715, p-value = 0.7931
alternative hypothesis: true difference in means is less than 0
95 percent confidence interval:
        NA 0.0340122
sample estimates:
mean of x mean of y
 1.313148  1.301859
```

Figura 8.10. Prueba Z para verificar si la media memética es inferior

Con un nivel de significancia del 0.05, no se rechaza la hipótesis nula de que la media del algoritmo memético no es superada por la mediana del algoritmo genético. Pues, el p-valor es mayor al grado de significancia.

8.5.3. Prueba paramétrica de empate del algoritmo memético

H0: La media del algoritmo memético es igual a la media del algoritmo genético.
H1: La media del algoritmo memético es diferente a la media del algoritmo genético.

```
> z.test(x=dataMemetico$Muestras, y=dataGenetico$Muestras, mu=0, sigma.x=sd(dataMemetico$Muestras), sigma.y=sd(dataGenetico$Muestras),alternative = "two.sided")

        Two-sample z-Test

data:  dataMemetico$Muestras and dataGenetico$Muestras
z = 0.81715, p-value = 0.4138
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.01578784  0.03836540
sample estimates:
mean of x mean of y
 1.313148  1.301859
```

Figura 8.11. Prueba Z para verificar si la media memética es igual

Con un nivel de significancia del 0.05, no se rechaza la hipótesis nula de que la media del algoritmo memético es igual a la mediana del algoritmo genético. Pues, el p-valor es mayor al grado de significancia.

8.6. Discusión

Teniendo en consideración los resultados de las pruebas, no se tiene suficiente evidencia para asumir que los resultados del algoritmo memético.

Sin embargo, para las pruebas no paramétricas, el caso más cercano a ser aceptado es aquel donde el algoritmo memético tiene mejor valor que el algoritmo genético debido a que su p-valor es de 0.1138. Siendo seguido del p-valor de 0.2276 para determinar si la hipótesis de que ambas medianas son distintas se puede aceptar. Si se aumentase el p-valor hasta 0.2276 se podría argumentar que el algoritmo memético tiene mejor mediana pero no se contaría con una precisión correcta ya que por cada 100 experimentos habría un 88.62% de veces en las que el algoritmo memético tendría mejor mediana. Lo cual no es suficiente para afirmar que el memético virtualmente siempre tendrá mejor mediana. Puesto que por lo general en estas pruebas se considera un p-valor menor al 5% para aceptar la hipótesis alternativa y además el p-valor es aproximado por existir muestras con el mismo valor.

Por otro lado, en las pruebas paramétricas asumiendo normalidad por tratarse de 30 valores, también se cumple la misma tendencia, pero con p-valores más altos. Lo cual reduce la posibilidad de victorias del algoritmo memético.

Se concluye que el desempeño de ambos algoritmos tiende a ser similar.

Con esta etapa se concluye el resultado esperado R7 del objetivo específico O3.

Referencias:

About. (2022). RStudio. https://www.rstudio.com/about/

Aho, A. V., Hopcroft, J. E., & Ullman, J. D. (1974). *The design and analysis of computer algorithms*. Addison-Wesley Pub. Co.

Average Warehouse Sizes & Space Planning Tips. (2022, agosto 15). Warehouse1. https://www.wh1.com/warehouse-square-footage-tips/#:~:text=In%20the%20past%2C%20most%20warehouses,50%2C001%20square%20feet%20or%20larger

Benmezal, L., Benhamou, B., & Boughaci, D. (2017). Some Neighbourhood Approaches for the Antenna Positioning Problem. *2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI)*, 1001–1007. https://doi.org/10.1109/ICTAI.2017.00154

Bhuwania, A., Subba, P., & Roy, U. K. (2016). Positioning WiFi access points using Particle Swarm Optimization. *2016 Second International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*, 112–115. https://doi.org/10.1109/ICRCICN.2016.7813641

Chakravarty, S., & Shekhawat, A. (1992). Parallel and serial heuristics for the minimum set cover problem. *The Journal of Supercomputing*, *5*(4), 331–345. https://doi.org/10.1007/BF00127952

Chen, Z.-G., Lin, Y., Gong, Y.-J., Zhan, Z.-H., & Zhang, J. (2021). Maximizing Lifetime of Range-Adjustable Wireless Sensor Networks: A Neighborhood-Based Estimation of Distribution Algorithm. *IEEE Transactions on Cybernetics*, *51*(11), 5433–5444. https://doi.org/10.1109/TCYB.2020.2977858

Chow, A. S., & Bucknall, T. (2012). Conclusion: User needs and library technology. En *Library Technology and User Services* (pp. 131–134). Elsevier.

https://doi.org/10.1016/B978-1-84334-638-8.50014-X

Dean, A., Draguljić, D., & Voss, D. (2017). *Design and Analysis of Experiments* (2nd ed. 2017). Springer International Publishing : Imprint: Springer. https://doi.org/10.1007/978-3-319-52250-0

Gao, J., Tsao, H.-S. J., & Wu, Y. (2003). *Testing and quality assurance for component-based software*. Artech House.

Grochla, K., & Polys, K. (2020). Heuristic Algorithm for Gateway Location Selection in Large Scale LoRa Networks. *2020 International Wireless Communications and Mobile Computing (IWCMC)*, 777–782. https://doi.org/10.1109/IWCMC48107.2020.9148435

Grossman, T., & Wool, A. (1997). Computational experience with approximation algorithms for the set covering problem. *European Journal of Operational Research*, *101*(1), 81–92. https://doi.org/10.1016/S0377-2217(96)00161-0

Gupta, R., Muttoo, S. K., & Pal, S. K. (2019). Meta-Heuristic Algorithms to Improve Fuzzy C-Means and K-Means Clustering for Location Allocation of Telecenters Under E-Governance in Developing Nations. *INTERNATIONAL JOURNAL of FUZZY LOGIC and INTELLIGENT SYSTEMS*, *19*(4), 290–298. https://doi.org/10.5391/IJFIS.2019.19.4.290

Hajipour, V., Fattahi, P., Bagheri, H., & Babaei Morad, S. (2022). Dynamic maximal covering location problem for fire stations under uncertainty: Soft-computing approaches. *International Journal of System Assurance Engineering and Management*, *13*(1), 90–112. https://doi.org/10.1007/s13198-021-01109-8

Harris, C. M. (Ed.). (2006). *Dictionary of architecture & construction* (4th ed). McGraw-Hill.

Haupt, R. L. (2000). Optimum population size and mutation rate for a simple real genetic algorithm that optimizes array factors. *IEEE Antennas and Propagation Society International Symposium. Transmitting Waves of Progress to the Next Millennium.*

2000 Digest. Held in conjunction with: USNC/URSI National Radio Science Meeting (Cat. No.00CH37118), *2*, 1034–1037. https://doi.org/10.1109/APS.2000.875398

How Far Will Your Wi-Fi Signal Reach? (2022, abril 28). EPB. https://epb.com/get-connected/gig-internet/how-far-will-your-wi-fi-signal-reach/#:~:text=Many%20people%20are%20hindering%20their,about%2015%20meters%20of%20reach

Huang, C., Lu, J., & Sun, L.-Q. (2022). Location Optimization of VTS Radar Stations Considering Environmental Occlusion and Radar Attenuation. *ISPRS International Journal of Geo-Information*, *11*(3), 183. https://doi.org/10.3390/ijgi11030183

Jain, R., Kasturi, R., & Schunck, B. G. (1995). *Machine vision*. McGraw-Hill.

Kamenetsky, M., & Unbehaun, M. (2002). Coverage planning for outdoor wireless LAN systems. *2002 International Zurich Seminar on Broadband Communications Access - Transmission - Networking (Cat. No.02TH8599)*, 49-1-49–6. https://doi.org/10.1109/IZSBC.2002.991793

Mann, H. B., & Whitney, D. R. (1947). On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other. *The Annals of Mathematical Statistics*, *18*(1), 50–60. https://doi.org/10.1214/aoms/1177730491

Morshed, K. M., Karmokar, D. K., & Numan-Al-Mobin, A. Md. (2009). Numerical analysis of impedance matched Inverted-L antennas for Wi-Fi operations. *2009 12th International Conference on Computers and Information Technology*, 691–696. https://doi.org/10.1109/ICCIT.2009.5407323

Moscato, P. (1989). On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. *Caltech Concurrent Computation Program, C3P, Technical Report, 826*.

Moscato, P., & Cotta, C. (2003). A Gentle Introduction to Memetic Algorithms. En F. Glover

& G. A. Kochenberger (Eds.), *Handbook of Metaheuristics* (Vol. 57, pp. 105–144). Kluwer Academic Publishers. https://doi.org/10.1007/0-306-48056-5_5

Paseo por el lenguaje C#. (2022). Microsoft. https://docs.microsoft.com/es-es/dotnet/csharp/tour-of-csharp/

Picincu, A. (2019). *The Importance of Warehousing in a Logistics System*. Chron. https://smallbusiness.chron.com/importance-warehousing-logistics-system-74825.html

Renfrew, C. (1972). *The emergence of civilisation: The Cyclades and the Aegean in the third millennium B.C.* Methuen.

Rowell, D. (2022). *Antenna Solutions for Warehouse Wi-Fi Deployments*. Packet6. https://packet6.com/antenna-solutions-warehouse-wifi-deployments/

Rudolf F, G. (1999). *Modern Dictionary of Electronics* (7th Edition). Elsevier. https://doi.org/10.1016/C2009-0-26363-X

Sahoo, J., & Sahoo, B. (2020). Solving Target Coverage Problem in Wireless Sensor Networks using Greedy Approach. *2020 International Conference on Computer Science, Engineering and Applications (ICCSEA)*, 1–4. https://doi.org/10.1109/ICCSEA49143.2020.9132907

Sait, S. M., & Youssef, H. (1999). *Iterative computer algorithms with applications in engineering: Solving combinatorial optimization problems*. IEEE Computer Society.

Shapiro, S. S., & Wilk, M. B. (1965). An analysis of variance test for normality (complete samples). *Biometrika*, *52*(3–4), 591–611. https://doi.org/10.1093/biomet/52.3-4.591

Singh, R. (2003). Wi-fi. *The Computer Bulletin*, *45*(6), 28–28. https://doi.org/10.1093/combul/45.6.28

Sprinthall, R. C. (2014). *Basic statistical analysis*.

Taherdangkoo, M., Paziresh, M., Yazdi, M., & Bagheri, M. (2013). An efficient algorithm for

function optimization: Modified stem cells algorithm. *Open Engineering*, *3*(1), 36–50. https://doi.org/10.2478/s13531-012-0047-8

Ting, C.-K., & Liao, C.-C. (2010). A memetic algorithm for extending wireless sensor network lifetime. *Information Sciences*, *180*(24), 4818–4833. https://doi.org/10.1016/j.ins.2010.08.021

Verner, J. M., Brereton, O. P., Kitchenham, B. A., Turner, M., & Niazi, M. (2012). Systematic literature reviews in global software development: A tertiary study. *16th International Conference on Evaluation & Assessment in Software Engineering (EASE 2012)*, 2–11. https://doi.org/10.1049/ic.2012.0001

WHAT IS SCRUM? (2022). Scrum.org. https://www.scrum.org/resources/what-is-scrum

Why Is Warehouse WiFi So Important? (2020). ClarusWMS. https://claruswms.co.uk/why-is-warehouse-wifi-so-important/#:~:text=Warehouse%20WiFi%20solutions%20can%20enable,as%20refrigerators%2C%20freezers%20and%20chillers

Zhi, Z., Wu, J., Meng, X., Yao, M., Hu, Q., & Tang, Z. (2019). AP Deployment Optimization in Non-Uniform Service Areas: A Genetic Algorithm Approach. *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*, 1–5. https://doi.org/10.1109/VTCFall.2019.8891308

Anexos:

A. Criterios de Exclusión

https://docs.google.com/spreadsheets/d/1x2Mfb1aJMkzdxnXC75zGuWH6NxQvOwZw/edit?usp=sharing&ouid=108268562402702132966&rtpof=true&sd=true

(En caso no se pueda acceder al anterior)

https://docs.google.com/spreadsheets/d/168yOunwCUNQSzVdofSPj-DiPWPIAp09t/edit?usp=sharing&ouid=108268562402702132966&rtpof=true&sd=true

B. Entregable 1.1 (Las fuentes usadas se encuentran en la sección de referencias)

B.1. Título de tesis

"IMPLEMENTACIÓN DE UN ALGORITMO MEMÉTICO PARA LA DISTRIBUCIÓN DE ANTENAS WIFI EN ALMACENES DE GRANDES DIMENSIONES."

B.2. Asesor

El asesor de este proyecto es el Mg. Rony Cueva Moscoso.

B.3. Plan de trabajo

En este proyecto de tesis, el plan de trabajo aplicado será colaborativo y comunicativo, las reuniones serán semanales para absolver dudas y levantar observaciones sobre los entregables, se promoverá una relación asesor-asesorado horizontal y basada en el respeto mutuo, además se utilizará herramientas como Zoom, Gmail, WhatsApp y Google Drive. Además, se persistirá en la investigación por parte del asesorado y el apoyo técnico por parte del asesor garantizará cumplir con los objetivos del curso.

A continuación, se tiene el plan de trabajo detallado que se utilizará, se muestra en la tabla B.1:

Tabla B.1 Plan de trabajo

SEM.	Entregable	Fecha entrega entregable v1	Fecha reunión con asesor	Fecha entrega entregable v2	Fecha revisión final	Fecha entregable
1	EP1.1	Jueves	Miércoles	Viernes	Sábado	Lunes
2	EP1.2	Jueves	Miércoles	Viernes	Sábado	Lunes
3	EP1.3	Jueves	Miércoles	Viernes	Sábado	Lunes
4	EP1.4	Jueves	Miércoles	Viernes	Sábado	Lunes
5	EP1.5	Jueves	Miércoles	Viernes	Sábado	Lunes
6	E1	Domingo	Lunes	Martes	Miércoles	Viernes
7	EP2.1	Jueves	Miércoles	Viernes	Sábado	Lunes
8	E2	Domingo	Lunes	Martes	Miércoles	Viernes
9						
10						
11	E3	Domingo	Lunes	Martes	Miércoles	Viernes

12	**E3, video mejorado y EP4.1**	**Jueves**	**Miércoles**	**Viernes**	**Sábado**	**Lunes**
13	**E4**	**Miércoles**	**Jueves**	**Viernes**	**Sábado**	**Domingo**
14						
15						

B.4. Cronograma de reuniones

Las reuniones asesor-asesorado se realizarán de la manera que se muestra en la Tabla B.2:

Tabla B.2 Cronograma de reuniones

Día	Hora	Motivo
miércoles	11am	Recepción de dudas y revisión de observaciones en base al avance del entregable que se presentará los días **miércoles** de la siguiente semana.
Viernes o Sábado	4pm o 5pm	Reunión opcional corta para recepción de dudas con respecto a la versión corregida que se presentará los días **viernes o sábado.**

B.5. Área del proyecto

El área de desarrollo de este proyecto es Ciencias de la Computación (Computer Science), según se especifica en la Computing Curricula ACM-IEEE 2020.

B.6. Descripción del proyecto

En la actualidad es necesario establecer la forma en que se puede transmitir información en un área. Adicionalmente, gracias a los recientes avances en tecnologías de comunicación y miniaturización de hardware (y a la masificación de estos), se facilitó el surgimiento de dispositivos inalámbricos que envían y reciben información (Ting & Liao, 2010).

Entre las principales tecnologías tenemos los puntos de acceso que tienen antenas integradas, las cuales permiten una propagación de 360 grados de la señal wifi. Sin embargo, estas tienen la desventaja de que no pueden cubrir óptimamente en zonas cercanas al suelo y tienen un rango limitado de metros que impide poner una sola antena para cubrir toda el área para el caso de un almacén de grandes dimensiones. Por tanto, se dispone también de antenas direccionales cuya cobertura puede ser de 80, 65, 55

grados entre otras medidas con el fin de cubrir el área con mayor eficacia (Rowell, 2022).

Sin embargo, estos dispositivos son caros y su instalación también y debido al área del proyecto el enfoque será en la búsqueda de una forma de cómo ubicarlos de manera adecuada y es por esto que es necesaria una solución que nos permita optimizar la cobertura.

Si se llega a aplicar una solución para la optimización del posicionamiento y número de antenas, se puede reducir su compra y/o construcciones, además de otros costos adicionales que puedan surgir.

Se realizará la búsqueda del arreglo óptimo de antenas mediante un algoritmo memético aplicado al problema de cobertura de conjuntos mínima (Minimum Set Cover) teniendo como universo el área del almacén con F = {S1 … Sk }, siendo F una colección de i conjuntos, siendo cada conjunto el área cubierta por una antena "i" buscando el menor número de conjuntos S que cubran el universo (Chakravarty & Shekhawat, 1992) y se comparará su desempeño respecto al uso de otros algoritmos.

Entre los algoritmos metaheurísticos se eligió el memético debido a su practicidad en problemas de optimización y que hace uso de toda la información disponible respecto al problema (Moscato & Cotta, 2003). También se usará .NET para implementar el algoritmo de optimización de cobertura usando R y diversas librerías de .NET para el desarrollo de las diversas pruebas estadísticas y de hipótesis.

C. Plan de Tesis

C.1. Justificación

En la actualidad las compañías buscan tener acceso rápido y preciso a diversos tipos de información necesario para atender las demandas del mercado para lo cual es necesaria una solución que agilice las operaciones y una manera de controlar los productos que manejan en conjunto con el uso de un almacén.

La presencia de un almacén permite tener un control del inventario pudiendo distribuir productos más eficientemente e incluso saber cuándo ya no se cuenta con stock, de modo que se pueden tomar decisiones operativas al respecto (Picincu, 2019). Sin embargo, el uso de almacenes tiene que ser complementado con una efectiva comunicación entre los trabajadores.

Por tanto, el uso de redes inalámbricas (Wi-Fi) en almacenes tiene el potencial de permitir la rápida automatización de procesos intensivos tales como la recepción, desembarco, almacenamiento, conteo de órdenes, recogidas y empaquetamientos. Asimismo, pueden permitir un buen rendimiento al laburar en ambientes difíciles de almacenamiento a bajas temperaturas como refrigeradoras y congeladoras (*Why Is Warehouse WiFi So Important?*, 2020). Pero desgraciadamente, los emisores de señal Wi-Fi suelen ser puestos de maneras que se consideran adecuadas solamente a criterio del personal instalador y esto ocasiona que se coloquen más de lo necesario e incluso es posible que se deje un área sin cobertura debido a que es posible que se agrupen los emisores en zonas muy juntas y la señal estaría concentrada en una porción del terreno (Bhuwania et al., 2016).

La adecuada planificación del posicionamiento de antenas emisoras permitiría la comunicación continua de los trabajadores del almacén. En adición, lograría evitar gastar recursos innecesarios que son adicionales a las propias antenas (como instalaciones y compras) y abarcar la mayor cantidad de área posible reduciendo la atenuación entre paredes (debido a que las ondas de radio no atraviesan todos materiales de la misma manera) para evitar zonas sin conexión (Ting & Liao, 2010).

Por lo general, se suelen usar algoritmos populares que ya son muy antiguos para problemas de este tipo, principalmente el algoritmo genético.

Este algoritmo tiene diversos problemas siendo el más notable el tener cierta tendencia a converger alrededor de un óptimo local (Taherdangkoo et al., 2013), siendo este término la denominación de una solución en la que los resultados obtenidos solo se quedan cerca de un resultado que aparenta ser el mejor de todo un conjunto de soluciones pero que no necesariamente lo es.

Con todo lo anterior mencionado, en la búsqueda de evitar disponer de antenas innecesarias, además de otros recursos desperdiciados, se plantea la implementación de un algoritmo de optimización para buscar la optimización del orden en las antenas Wi-Fi en un almacén de grandes dimensiones.

Se escoge un algoritmo memético debido a que es práctico en problemas de

optimización al utilizar toda la información disponible (Moscato & Cotta, 2003) y además permite una alta adaptabilidad y búsqueda local individual en los elementos de la población (Moscato, 1989).

C.2. Viabilidad

Debido a que en diversos cursos a lo largo de la carrera de Ingeniería Informática y en el ambiente laboral del alumno tesista se trabajó bastante en el entorno de .NET, es posible implementar el algoritmo memético (y el algoritmo con el que se comparará) tomando como referencia la documentación mostrada en el estado del arte, así como la información referente a clases, funciones y bibliotecas que sean necesarias. Es importante mencionar que se cuenta con el equipo informático necesario, también con las computadoras de la universidad en caso de emergencia, y el software adicional necesario es de acceso público.

De manera similar, se eligió la metodología Scrum debido a que el alumno tesista ha elaborado múltiples proyectos bajo estas condiciones y ha mostrado ser de gran ayuda para organizar de manera puntual el proyecto correspondiente.

Asimismo, el asesor proveerá de pautas y apoyo en el desarrollo y planeamiento del algoritmo, así como en la adaptación del algoritmo de comparación, debido a que cuenta con bastante experiencia en el tema.

En conclusión, debido a que la información necesaria y las herramientas de trabajo están disponibles, se cuenta con el apoyo del asesor y la metodología de trabajo está definida, es posible determinar que el proyecto es viable.

C.3. Alcance

El proyecto de tesis pertenece al tema de algoritmos de optimización (que a su vez es perteneciente al área de ciencias de la computación) que permitirá determinar un conjunto de posiciones óptimas con el número de antenas Wi-Fi adecuadas para poder mejorar el uso de recursos sin perder señal.

Por lo cual se procederá a definir las restricciones, variables y parámetros que permitirán buscar una configuración adecuada. Teniendo estos elementos definidos se prepara la formulación del algoritmo memético. Una vez que lo anterior se tenga definido, se podrá establecerse la función objetivo con las restricciones adecuadas para poder cuantificar el grado de optimización alcanzado. Será posible usar para verificar la confiabilidad del algoritmo apenas se establezcan las pruebas de caja negra.

Se deja planteado el flujo y las estructuras de datos usados por el proyecto.

El algoritmo memético será implementado mediante 2 módulos siendo el primero aquel que realizará la combinación de la población y el segundo que permitirá la búsqueda local en la población.

Posteriormente, se realizará una adaptación del algoritmo genético mencionado en la problemática y el estado del arte. Para esto se planteará el pseudocódigo de la solución primero para luego poder realizar la revisión de la estructura planteada con el apoyo del

especialista en ciencias de la computación. Una vez se tenga un planteamiento válido y aprobado, se procederá al desarrollo del código correspondiente.

Finalmente se implementará la experimentación numérica para comparar los resultados de las ejecuciones de ambos algoritmos de modo que se pueda verificar la distribución de los resultados de ambos algoritmos desarrollados/adaptados. Primero se usará la prueba de Shapiro-Wilk para evaluar la normalidad y ver si es que es más conveniente continuar con la prueba Z o la prueba de U Mann-Whitney en el planteamiento de la hipótesis y su verificación.

C.4. Limitaciones

Es posible que se tomen consideraciones respecto a la complejidad de la infraestructura del almacén debido a que no todos los almacenes poseen la misma. Asimismo, la atenuación de la señal inalámbrica en las paredes del almacén planteado y el cambio energético en la intensidad de las antenas son aspectos que no serán considerados ya que se encuentran más relacionados al área de Ingeniería Electrónica.

Estas limitantes están orientadas a ser consideradas al momento de plantear las variables, parámetros, restricciones y la función objetivo que son elementos fundamentales del proyecto.

Por otro lado, la capacidad del hardware de procesamiento denotada por el tiempo de reloj, los procesadores e hilos del CPU, así como la memoria disponible en las distintas unidades de almacenamiento del equipo en el cual se ejecutará el proyecto serán elementos que restringirán los parámetros de ejecución del proyecto. Algunos de estos parámetros son el número de iteraciones y el nivel de búsqueda local de la población.

C.5. Riesgos

A continuación, se procede a listar los riesgos con su respectiva descripción, los síntomas de su eventual ocurrencia, las probabilidades de ocurrencia, el impacto generado y el nivel de severidad, una acción para mitigar el riesgo por cada uno y las acciones de contingencia en la tabla C.1.

Las probabilidades de ocurrencia y el impacto se miden del 1 al 5 en función de su intensidad. La severidad se calcula multiplicando las probabilidades de ocurrencia y el impacto, un nivel cercano al 25 implica una severidad extrema mientras que un nivel de 1 denota una severidad muy baja.

Tabla C.1 Tabla de riesgos

Descripción	Síntomas	P	I	S	Mitigación	Contingencia
Falla de la computadora donde se realiza el desarrollo.	Apagados repentinos, tiempos de respuesta alargados.	1	4	4	Trabajar con las máquinas de la universidad.	Llevar el equipo a servicio técnico.
El especialista del área de ciencias de la computación no cuenta con la disponibilidad.	El especialista continuamente no asiste a las reuniones.	1	3	3	Corroborar con trabajos similares.	Buscar a otro especialista.
Bibliotecas o módulos adicionales no accesibles (data no disponible).	En diversas ocasiones las funciones no están disponibles fácilmente.	2	4	8	Buscar módulos alternativos o similares y adaptarlos.	Implementar lo básico de estos módulos para tener las funciones necesarias.
El planteamiento de las estructuras para el algoritmo no es adecuado.	Los resultados obtenidos no son realmente buenos.	3	5	15	Realizar pruebas adicionales	Replantear las estructuras y cambiar la implementación.

C.6. Estructura de descomposición del trabajo

A continuación, se procede a presentar la estructura de descomposición del trabajo en la Figura C.1

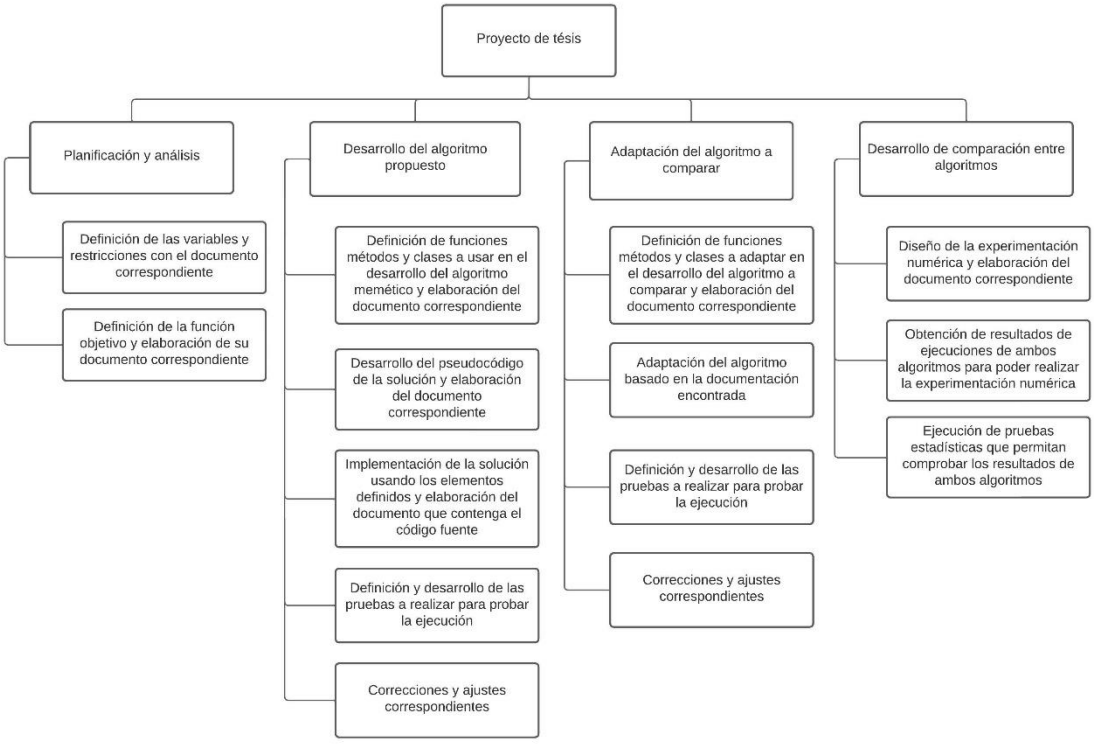


Figura C.1. Estructura de descomposición del trabajo

C.7. Lista de tareas

Se presentan las tareas a realizar durante el proyecto en la Tabla C.2.

Tabla C.2 Tabla de lista de tareas

Índice	Objetivo	Tarea	Duración estimada en días	Esfuerzo asociado (horas/ persona)	Costo total
1	-	Entrega del entregable 4 del curso de Tesis 1.	2	8	240
2	Objetivo 1	Definición de los parámetros, variables y restricciones.	4	14	420
3		Elaboración del documento que contendrá su declaración.	1	4	120
4		Reunión con asesores y corrección de errores.	1	2	160
5		Validación de los parámetros, variables y Restricciones con especialista.	1	2	160
6		Definición de la función objetivo.	1	5	150
7		Elaboración del documento que contendrá los detalles de su formulación.	1	6	180
8		Reunión con asesores y corrección de errores.	1	2	160
9		Validación de la función objetivo con especialista.	1	2	160
10	Objetivo 2	Definición de las funciones, métodos y clases a implementar en el algoritmo memético.	7	21	630
11		Elaboración del documento que contendrá el detalle de las funciones, métodos y clases.	1	3	90
12		Elaboración del documento que contendrá el pseudocódigo de la solución.	1	8	240
13		Reunión con asesores y corrección de errores.	2	6	480
14		Validación de las funciones, métodos, clases y resultados de las pruebas con especialista.	1	2	160
15		Codificación del algoritmo memético en el código fuente.	5	25	750
16		Elaboración de las pruebas de funcionamiento.	1	6	180

17		Reunión con asesores y corrección de errores.	1	5	400
18		Validación del código fuente y el resultado de las pruebas con el especialista.	1	2	160
19	Objetivo 3	Elaboración de funciones y métodos adaptados para el algoritmo a comparar adaptado en el código fuente.	6	24	720
20		Elaboración de las pruebas de funcionamiento.	2	8	240
21		Reunión con asesores y corrección de errores.	1	3	240
22		Validación del código fuente del algoritmo adaptado y el resultado de las pruebas con el especialista.	1	2	160
23		Desarrollo de pruebas adicionales que permitan comprobar configuraciones adicionales y posibles errores.	1	5	150
24		Elaboración que contendrá el detalle y resultado de estas pruebas.	1	2	60
25		Reunión con asesores y corrección de errores.	2	4	320
26		Validación de los resultados y configuraciones propuestos con el especialista.	1	2	160
27		Diseño de la experimentación numérica en su documento correspondiente.	5	20	600
28		Reunión con asesores y corrección de errores.	2	4	320
29		Validación de la experimentación numérica propuesta con el especialista.	1	3	240
30		Obtención de los resultados de la experimentación numérica.	2	8	240
31		Reunión con el asesor para la revisión y ajustes que se deben considerar.	2	4	320
32		Validación de los resultados de la experimentación.	1	1	80

C.8. Cronograma del proyecto

Se procederá a listar el cronograma del proyecto en la tabla C.3.

Tabla C.3. Cronograma del proyecto

Índice	Tarea	Fecha inicio	Fecha fin
1	Entrega del entregable 4 del	16/06/2022	18/06/2022

	curso de Tesis 1.		
2	Definición de los parámetros, variables y restricciones.	15/08/2022	19/08/2022
3	Elaboración del documento que contendrá su declaración.	19/08/2022	20/08/2022
4	Reunión con asesores y corrección de errores.	20/08/2022	21/08/2022
5	Validación de los parámetros, variables y Restricciones con especialista.	21/08/2022	22/08/2022
6	Definición de la función objetivo.	22/08/2022	23/08/2022
7	Elaboración del documento que contendrá los detalles de su formulación.	23/08/2022	24/08/2022
8	Reunión con asesores y corrección de errores.	24/08/2022	25/08/2022
9	Validación de la función objetivo con especialista.	25/08/2022	26/08/2022
10	Definición de las funciones, métodos y clases a implementar en el algoritmo memético.	26/08/2022	2/09/2022
11	Elaboración del documento que contendrá el detalle de las funciones, métodos y clases.	2/09/2022	3/09/2022
12	Elaboración del documento que contendrá el pseudocódigo de la solución.	3/09/2022	4/09/2022
13	Reunión con asesores y corrección de errores.	4/09/2022	6/09/2022
14	Validación de las funciones, métodos, clases y resultados de las pruebas con especialista.	6/09/2022	7/09/2022
15	Codificación del algoritmo memético en el código fuente.	7/09/2022	12/09/2022
16	Elaboración de las pruebas de funcionamiento.	12/09/2022	13/09/2022
17	Reunión con asesores y corrección de errores.	13/09/2022	14/09/2022
18	Validación del código fuente y el resultado de las pruebas con el especialista.	14/09/2022	15/09/2022
19	Elaboración de funciones y métodos adaptados para el algoritmo a comparar adaptado en el código fuente.	15/09/2022	21/09/2022

20	Elaboración de las pruebas de funcionamiento.	21/09/2022	23/09/2022
21	Reunión con asesores y corrección de errores.	23/09/2022	24/09/2022
22	Validación del código fuente del algoritmo adaptado y el resultado de las pruebas con el especialista.	24/09/2022	25/09/2022
23	Desarrollo de pruebas adicionales que permitan comprobar configuraciones adicionales y posibles errores.	25/09/2022	26/09/2022
24	Elaboración que contendrá el detalle y resultado de estas pruebas.	26/09/2022	27/09/2022
25	Reunión con asesores y corrección de errores.	27/09/2022	29/09/2022
26	Validación de los resultados y configuraciones propuestos con el especialista.	29/09/2022	30/09/2022
27	Diseño de la experimentación numérica en su documento correspondiente.	30/09/2022	5/10/2022
28	Reunión con asesores y corrección de errores.	5/10/2022	7/10/2022
29	Validación de la experimentación numérica propuesta con el especialista.	7/10/2022	8/10/2022
30	Obtención de los resultados de la experimentación numérica.	8/10/2022	10/10/2022
31	Reunión con el asesor para la revisión y ajustes que se deben considerar.	10/10/2022	12/10/2022
32	Validación de los resultados de la experimentación.	12/10/2022	13/10/2022

C.9. Recursos

Los recursos con los que se cuenta para el desarrollo de este proyecto se detallan a continuación en la tabla C.4.

Tabla C.4 Tabla de recursos

Categoría	Elemento	Detalle
Personas involucradas y necesidades de capacitación	Alumno Tesista	Responsable del proyecto.
	Asesor	Provee asesoramiento para el desarrollo de la presente tesis.
	Especialista en el área de ciencias de la computación	Permitirá la validación de la información a presentar. Es necesario resaltar que el asesor es esta misma persona.
Equipamiento requerido	Computadora	Permitirá la elaboración de todo documento entregable y proyecto ejecutable requerido.
Herramientas requeridas	Visual Studio 2022	Permitirá el desarrollo de la implementación.
	R Studio	Permitirá el desarrollo de la experimentación numérica.
	Notepad ++	Permitirá realizar apuntes rápidos.
	Herramientas de Office (Word, Excel, etc.)	Permitirá el desarrollo y la lectura de los documentos que se manejarán de manera local.
	Herramientas de Google (Docs, Drive, Sheets)	Permitirá la gestión de documentos de manera digital.
	Herramientas de comunicación (Zoom, Gmail, WhatsApp)	Permitirá la comunicación entre las personas involucradas.

C.10. Costeo

Se muestra el análisis del costeo involucrado en este proyecto mediante la Tabla C.5.

Tabla C.5 Costeo estimado

Índice	Categoría	Elemento	Cantidad	Valor unitario en soles	Monto parcial en soles	Monto total en soles
1	Personas	Alumno tesista (horas)	213	30	6390	6390
2		Asesor / Especialista (horas)	46	50	2300	8690
3	Materiales y servicios	Internet (meses)	4	99	396	9086
4		Plan de datos móvil (meses)	4	30	120	9206
5		Computadora (horas)	213	1	213	9419

D. Validaciones

D.1. Parámetros, Variables y Restricciones

JUSTIFICACIÓN DE VALIDACIÓN

Mediante la presente autorización, el Ing. CUEVA MOSCOSO RONY, especialista en algoritmia, da conformidad de que el estudiante del curso de Proyecto de Tesis 2 RAFAEL JAIR BURGOS CHUQUI ha presentado el documento correspondiente a los parámetros variables y restricciones. La información es adecuada para alcanzar los objetivos esperados.

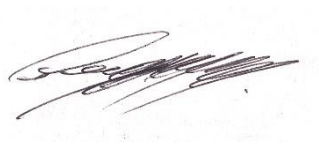
Mg. Rony Cueva Moscoso

DNI: 09942265

D.2. Función Objetivo

JUSTIFICACIÓN DE VALIDACIÓN

Mediante la presente autorización, el Ing. CUEVA MOSCOSO RONY, especialista en algoritmia, da conformidad de que el estudiante del curso de Proyecto de Tesis 2 RAFAEL JAIR BURGOS CHUQUI ha presentado el documento correspondiente a la Función Objetivo. La información es adecuada para alcanzar los objetivos esperados.



Mg. Rony Cueva Moscoso

DNI: 09942265

D.3. Estructuras de datos

JUSTIFICACIÓN DE VALIDACIÓN

Mediante la presente autorización, el Ing. CUEVA MOSCOSO RONY, especialista en algoritmia, da conformidad de que el estudiante del curso de Proyecto de Tesis 2 RAFAEL JAIR BURGOS CHUQUI ha presentado el documento correspondiente a las estructuras de datos. La información es adecuada para alcanzar los objetivos esperados.

Mg. Rony Cueva Moscoso

DNI: 09942265

D.4. Pseudocódigo

JUSTIFICACIÓN DE VALIDACIÓN

Mediante la presente autorización, el Ing. CUEVA MOSCOSO RONY, especialista en algoritmia, da conformidad de que el estudiante del curso de Proyecto de Tesis 2 RAFAEL JAIR BURGOS CHUQUI ha presentado el documento correspondiente al pseudocódigo. La información es adecuada para alcanzar los objetivos esperados.

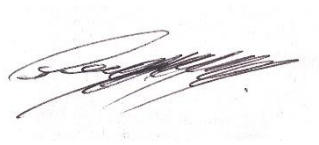
Mg. Rony Cueva Moscoso

DNI: 09942265

D.5. Código fuente

JUSTIFICACIÓN DE VALIDACIÓN

Mediante la presente autorización, el Ing. CUEVA MOSCOSO RONY, especialista en algoritmia, da conformidad de que el estudiante del curso de Proyecto de Tesis 2 RAFAEL JAIR BURGOS CHUQUI ha presentado la solución correspondiente y se ha verificado su correcta implementación. La información es adecuada para alcanzar los objetivos esperados.



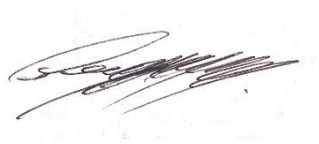
Mg. Rony Cueva Moscoso

DNI: 09942265

D.6. Pruebas y calibración

JUSTIFICACIÓN DE VALIDACIÓN

Mediante la presente autorización, el Ing. CUEVA MOSCOSO RONY, especialista en algoritmia, da conformidad de que el estudiante del curso de Proyecto de Tesis 2 RAFAEL JAIR BURGOS CHUQUI presentó el planteamiento para la calibración de variables para obtener resultados útiles a comparar en la sección de experimentación numérica. La información es adecuada para alcanzar los objetivos esperados.



Mg. Rony Cueva Moscoso

DNI: 09942265

D.7. Experimentación Numérica

JUSTIFICACIÓN DE VALIDACIÓN

Mediante la presente autorización, el Ing. CUEVA MOSCOSO RONY, especialista en algoritmia, da conformidad de que el estudiante del curso de Proyecto de Tesis 2 RAFAEL JAIR BURGOS CHUQUI presentó el planteamiento para la experimentación numérica así da su aprobación respecto a las conclusiones planteadas producto de la experimentación. La información es adecuada para alcanzar los objetivos esperados.

Mg. Rony Cueva Moscoso

DNI: 09942265

E. Métodos de soporte para los algoritmos

Los siguientes métodos pertenecen a la funcionalidad de los algoritmos y se presenta su pseudocódigo en las figuras correspondientes.

E.1. Métodos auxiliares generales

E.1.1 Inicializar Parámetros

Este método permite inicializar los diversos parámetros internos que dependen de los parámetros de entrada y que serán usados durante todo el resto de la ejecución del algoritmo, este método es usado para todos los algoritmos. Consiste en los siguientes pasos:

```
InicializarParametros()
BEGIN

    largoCelda <= largoAlmacen / totalCeldasHorizontal
    anchoCelda <= anchoAlmacen / totalCeldasVertical

    largoPixel <= largoCelda / dividirCeldaXPixel
    anchoPixel <= anchoCelda / dividirCeldaXPixel

    aproximadoCeldasPorRadioX <= Aproximar(radioAntena / largoCelda)
    IF (aproximadoCeldasPorRadioX == 0) aproximadoCeldasPorRadioX = 1;
    aproximadoCeldasPorRadioY <= Aproximar(radioAntena / anchoCelda)
    IF (aproximadoCeldasPorRadioY == 0) aproximadoCeldasPorRadioY = 1;

    areaTotalAlmacen <= largoAlmacen * anchoAlmacen

    mejoresSoluciones.inicializar()

    filasCandidatas.inicializar(filasDeseadas)


    acumuladorCandidata = 0;
    acumuladorMercaderia = 0;
    FOR(y = 0 TO y < totalCeldasVertical)
    {
        IF(acumuladorCandidata < filasCandidatasContiguas)
        {
            filasCandidatas.Add(y);
            acumuladorCandidata++;
            IF(acumuladorCandidata == filasCandidatasContiguas) acumuladorMercaderia = 0;
        }
        ELSE
        {
            acumuladorMercaderia++;
            IF(acumuladorMercaderia == filasMercaderiaContiguas) acumuladorCandidata = 0;
        }
    }


    movimientosCeldasX = Aproximar(porcentajeMovimientosX * totalCeldasHorizontal)
    movimientosCeldasY = Aproximar(porcentajeMovimientosY * filasCandidatas.Count())

    centroXCeldaDerechaLimite = (largoCelda / 2) + (totalCeldasHorizontal - 1)* largoCelda
    centroXCeldaIzquierdaLimite = (largoCelda / 2)

    centroYCeldaSuperiorLimite = (anchoCelda / 2) + (anchoCelda * filasCandidatas.Maximo())
    centroYCeldaInferiorLimite = (anchoCelda / 2) + (anchoCelda * filasCandidatas.Minimo())

END
```

Figura E.1.1. Inicializar Parámetros

- Se realiza el cálculo de las dimensiones de la celda para permitir el cálculo de coordenadas con sus medidas.

- Se realiza el cálculo de las dimensiones de los pixeles que conformarán cada celda con

la finalidad del cálculo del área.

- Se calcula un aproximado del número de celdas que conforman una antena horizontalmente y verticalmente (porque el número de celdas por lado no necesariamente es el mismo). En caso no alcancen a ocupar al menos una celda, se utiliza una celda.

- Se determina el área total del almacén usando las dimensiones de este.

- Se inicializa el número de filas candidatas con las filas deseadas.

- Se añaden las filas candidatas donde pueden ir antenas seguidas de las filas de mercadería que no pueden tener antenas según los parámetros de entrada, cada vez que se alcance el total de filas de un tipo se cambia al otro tipo. Se realiza hasta llegar al límite de celdas verticales.

- Se determina el número aproximado de movimientos en celdas que puede realizar una antena en el periodo de evolución. En el caso de las filas hay que considerar respecto al total de filas candidatas mientras que con las celdas horizontales se considera respecto al total general.

- Se determinan las coordenadas con orientación hacia la derecha e izquierda máximas y mínimas respectivamente en el eje X que puede tener una antena, en este caso es necesario tener en consideración la totalidad de columnas de celdas.

- Se determinan las coordenadas con orientación superior e inferior máximas y mínimas respectivamente en el eje Y que puede tener una antena, en este caso es necesario tener en consideración las filas candidatas de celdas únicamente.

E.1.2 Generación de Población Inicial

Con este método es posible generar una población inicial que permita la evaluación de distintas configuraciones de almacenes. Este método es usado para todos los algoritmos. Consiste en los siguientes pasos:

```
GenerarPoblacionInicial()
BEGIN
    FOR(i = 0 TO i = totalPoblacion - 1){
        individuo.inicializar()
        totalAntenas <- Aleatorio(0,MaximoAntenas)
        antenasEscogidas.inicializar()
        WHILE(antenasEscogidas.cantidad() < totalAntenas){
            posiciones <- ElegirPosicionesAleatorias(X,Y,filasCandidatas)
            WHILE(antenasEscogidas.contiene(posiciones)){
                posiciones <- ElegirPosicionesAleatorias(X,Y,filasCandidatas)
            }
        }
        FOR(j = 0 TO j <- limiteVertical - 1){
            FOR(k = 0 TO k = limiteHorizontal - 1){
                individuo.celda[j][k].inicializar()
                IF(antenasEscogidas.contiene(j,k)){
                    antena.inicializar(j,k)
                    individuo.celda[j][k].asignarAntena()
                    individuo.antenas.agregar(antena)
                }
            }
        }
        individuo.calcularFitness()
        poblacionInicial.agregar(individuo)
    }
    RETURN poblacionInicial
END
```

Figura E.1.2. Generación de Población Inicial

- La generación de población inicial comienza con la creación de la estructura que contendrá a la población.

- Se procede a elegir la cantidad de antenas de manera aleatoria entre 1 y el máximo de antenas para luego buscar posiciones aleatorias que pertenezcan al centro de una celda (dentro de las filas candidatas) que no hayan sido previamente elegidas.

- Una vez se tenga esta colección, se procede a explorar cada celda, así como asignarle sus atributos.

- En caso esta posición pertenezca a las posiciones elegidas para colocar antenas, se crea la antena con sus atributos y se añade la posición de esta a la celda actual.

- Se procede de este modo hasta explorar todas las celdas y se calcula el fitness (que a su vez calcula el área).

- Se añade el individuo a la población y se finaliza el método tenga tantos individuos como el número de población parametrizado.

E.1.3 Calcular Antenas Cercanas

Método que permite el cálculo del número de antenas cercanas a la posición determinada por un cuadrado de un lado que mide un diámetro de señal de antena.

Su desarrollo se presenta paso a paso:

```
calcularAntenasCercanas(centroX, centroY, antenasExteriores)
{
    limiteNorte <- centroY + aproximadoCeldasPorRadioY * anchoCelda

    limiteSur <- centroY - aproximadoCeldasPorRadioY * anchoCelda

    limiteEste <- centroX + aproximadoCeldasPorRadioX * largoCelda

    limiteOeste <- centroX - aproximadoCeldasPorRadioX * largoCelda

    antenasCercanas <- antenasExteriores.SacarAntenasConPosicionesDentroDeLosLimites
                       (limiteNorte,limiteSur,limiteEste,limiteOeste)

    RETORNAR antenasCercanas
}
```

Figura E.1.3 Calcular Antenas Cercanas

- Recibir las coordenadas de la posición y una colección de antenas

- Determinar los límites que determinan el área dentro de la cual se evaluará la presencia de antenas.

- Evaluar cada antena para verificar si su posición se encuentra dentro de los límites establecidos.

- Retornar el número de antenas que cumplen esta característica.

E.1.4 Crear Copia Almacén

Método que permite la copia de un almacén a otro completamente nuevo, perteneciente al objeto Almacén. Su desarrollo se detalla:

```
crearCopia()
{
    copia <- CrearAlmacen(indiceAlmacenes)

    FOR(antena IN EsteAlmacen.antenas)
    {
        antena <- crearAntena(antena.x,antena.y,antena.celdaX,antena.celdaY)
        copia.antenas.Add(antena)
    }
    celdasTotal.incializar()
    FOR(j = 0 TO j < totalCeldasVertical)
    {
        celdasFila.incializar()
        FOR(i = 0 TO i < totalCeldasHorizontal)
        {
            miniCelda.inicializar()
            IF(EsteAlmacen.celdas[j][i].antena < 0)
            {

                miniCelda <- new Celda(i, j, largoCelda, anchoCelda)
                miniCelda.antenaMasCercanaReciente <- EsteAlmacen.celdas[j][i].antenaMasCercanaReciente
            }
            ELSE
            {
                miniCelda <- new Celda(i, j, EsteAlmacen.celdas[j][i].antena, largoCelda, anchoCelda)
            }

            celdasFila.Add(miniCelda)
        }
        celdasTotal.Add(celdasFila)

    }
    copia.celdas <- celdasTotal
    copia.AreaCubierta <- EsteAlmacen.AreaCubierta
    copia.Fitness <- EsteAlmacen.Fitness
    RETURN copia
}
```

Figura E.1.4. Crear Copia Almacén

- Se Inicializa una copia que tendrá el mismo identificador.
- Se explora todas las antenas que pertenecen al almacén y se añaden a la copia.
- Se inicializan las celdas.
- Se procede a explorar horizontal y verticalmente las celdas.
- Se verifica si la celda original contiene la posición de un almacén.
- De ser el caso, se añade la posición.
- Se copian los parámetros restantes del almacén que no son estructuras.

E.1.5 Crear Reemplazo Almacén

Método que permite el reemplazo de un almacén en una posición por otro que pertenece a una posición diferente. Su desarrollo se detalla:

```
reemplazo(pos1, pos2)
{
    auxiliar = poblacion[pos2].crearCopia(totalCeldasHorizontal, totalCeldasVertical,
        largoCelda, anchoCelda)
    poblacion.Remover(pos1)
    poblacion.Insertar(pos1, aux)
}
```

Figura E.1.5. Crear Reemplazo Almacén

- Se crea una copia del almacén que será insertado haciendo uso del método explicado en el

Anexo E.1.4.

- Se elimina el almacén cuya posición será reemplazada.

- Se inserta la copia del almacén que ocupará su lugar.

E.1.6 Calcular Fitness

Método que permite determinar el fitness (o grado de cumplimiento con la función objetivo) a partir del almacén que le pertenece. Se detalla su funcionamiento:

```
calcularFitness(pesoArea, pesoAntenas)
{
    IF(AreaCubierta <= 0)
    {
        EsteAlmacen.AreaCubierta <- calcularArea();
    }
    EsteAlmacen.Fitness <- (pesoArea*(AreaCubierta)/(largoAlmacen * anchoAlmacen)) +
        (pesoAntenas / (antenas.numeroAntenas()))

    auxFitness <- Fitness;

    RETORNAR auxFitness;
}
```

Figura E.1.6. Calcular Fitness

- En caso no se tenga el área cubierta se procede a llamar a la sucesión de recursiones que se visualizan en la Figura 5.1.

- Se determina el valor del fitness en base al porcentaje del área cubierta, el número de antenas que contiene y los pesos correspondientes a ambos parámetros.

- Se crea una copia del valor para retornarlo a la operación que llamó al método.

E.2. Métodos auxiliares del Cálculo de Áreas

E.2.1. Calculo de Antena Mas Cercana que Cubre

Método que permite determinar una antena de la población que esté cercana a la celda o pixel (puede ser reutilizado para ambos solo requiriendo las coordenadas de su centro y sus dimensiones).

Se procede a mostrar su comportamiento:

```
antenaMasCercanaQueCubre(centroPX, centroPY, longX, longY, indexPreferencial)
{
    index <- 0
    totalAntenas <- antenas.totalAntenas()
    if (indexPreferencial != -1 AND indexPreferencial < totalAntenas)
    {
        a <- antenas[indexPreferencial].x
        b <- antenas[indexPreferencial].y
        posX <- 0, posY <- 0
        puntoMasLejano(ref posX, ref posY, indexPreferencial, centroPX, centroPY, longX, longY)
        distanciaX <- Math.Abs(posX - a)
        distanciaY <- Math.Abs(posY - b)
        distanciaReal <- RaizCuadrada(Potencia(distanciaX, 2) + Potencia(distanciaY, 2))
        IF(distanciaReal <= radioAntena)
            RETORNAR indexPreferencial;
    }
    FOR(index = 0 TO index < totalAntenas)
    {
        IF((indexPreferencial != -1) AND (indexPreferencial == index)) CONTINUAR
        antenas[index].x
        antenas[index].y
        posX <- 0
        posY <- 0
        puntoMasLejano(ref posX, ref posY, index, centroPX, centroPY, longX, longY)
        distanciaX <- Math.Abs(posX - a)
        distanciaY <- Math.Abs(posY - b)
        distanciaReal <- RaizCuadrada(Potencia(distanciaX, 2) + Potencia(distanciaY, 2))
        IF(distanciaReal <= radioAntena)
        {
            SALIR;
        }
    }
    IF(index == totalAntenas) index <- -1
    RETORNAR index;
}
```

Figura E.2.1. Planteamiento del cálculo de una antena que cubra a la celda o pixel

- Se verifica si se tiene un índice preferencial para explorar una antena primero.

- Si tiene el índice adecuado se procede a la exploración de la antena que le corresponde.

- Se realiza una búsqueda del punto más lejano del objeto respecto al centro de la antena y se sobrescribe sus coordenadas en las variables inicializadas en cero que se envían mediante un método que se detalla en el Anexo E.2.2.

- Se obtiene la distancia entre los puntos por el teorema de Pitágoras (usando las distancias entre ejes como catetos).

- Se verifica si la distancia es menor o igual a la antena (en cuyo caso se retorna la posición de la antena)

- En caso contrario se realiza el mismo procedimiento explorando todas las demás antenas excepto la antena ya explorada.

- Se retorna la posición de la antena que cumpla estas características (o -1 para denotar que no se encontró ninguna).

E.2.2. Calculo de Punto Más Lejano

Este método permite la determinación de la posición mas lejana perteneciente a una celda o pixel respecto al centro de una antena.

Se detalla su funcionamiento:

```
puntoMasLejano(ref posX, ref posY, antena, centroPX, centroPY, longX, longY)
{
    IF(antenas[antena].y == centroPY)
    {
        posY <- centroPY + (longY / 2);
    }
    ELSE IF(antenas[antena].y > centroPY)
    {
        posY <- centroPY - (longY / 2);
    }
    else
    {
        posY <- centroPY + (longY / 2);
    }

    IF(antenas[antena].x == centroPX)
    {
        posX <- centroPX + (longX / 2);
    }
    ELSE IF(antenas[antena].x > centroPX)
    {
        posX <- centroPX - (longX / 2);
    }
    else
    {
        posX <- centroPX + (longX / 2);
    }
}
```

Figura E.2.2. Determinación del punto más lejano entre un centro de antena y una pixel o celda

- Se recibe las dimensiones del objeto y las coordenadas de su centro.

- Se verifica si su coordenada central Y se encuentra igual que la coordenada Y de la posición de la antena. En cuyo caso simplemente se elige la coordenada Y de uno de los vértices superiores (es igual si se escogiese el inferior).

- En caso contrario se evalúa si la coordenada central Y se encuentra debajo de la coordenada Y de la posición de la antena. De ser este el caso se escoge la coordenada Y inferior y en caso contrario, la coordenada Y del vértice superior.

- Se verifica si su coordenada central X se encuentra igual que la coordenada X de la posición de la antena. En cuyo caso simplemente se elige la coordenada X de uno de los vértices más cercanos al lado derecho (es igual si se escogiese el izquierdo).

- En caso contrario se evalúa si la coordenada central X se encuentra a la izquierda de la coordenada X de la posición de la antena. De ser este el caso se escoge la coordenada X izquierda y en caso contrario, la coordenada Y del vértice derecho.

F. Código fuente de la solución

El proyecto de la solución se encuentra adjunto en el siguiente link:

https://drive.google.com/file/d/19tKJOwKaz58nKBhh0O-UEF-Pq5rmH3Kh/view?usp=sharing

G. Pruebas de Funcionamiento

Se procederá a verificar el funcionamiento de los algoritmos. Se tiene en consideración que los algoritmos deben ser probados con data orientada a la realidad, por lo cual se tomó en consideración un almacén de 10000 metros cuadrados debido a que un almacén de estas dimensiones es considerado grande (*Average Warehouse Sizes & Space Planning Tips*, 2022), así como una configuración de 45 metros de radio de antena (asumiendo señales de 2.4Ghz) (*How Far Will Your Wi-Fi Signal Reach?*, 2022). Como el objetivo es asegurar la funcionalidad, se realizarán pocas iteraciones y una población de 10 almacenes para visualizar los resultados de manera relativamente rápida.

Tabla G.1. Parámetros Generales

Parámetro	Valor
Largo Almacén	100
Ancho Almacén	100
Total Celdas Horizontal	33
Total Celdas Vertical	34
Radio Antena	45
Peso del área	1
Peso del número de antenas	100
Máximo de antenas	25
Máximo de iteraciones	10
Total Población	10
Porcentaje Mínimo Área	0.7

Tabla G.2. Parámetros exclusivos del algoritmo memético adaptado

Parámetro	Descripción
Iteraciones de la Evolución	5
Probabilidad de Mutación Memética	0.5
Porcentaje Movimientos X	0.3
Porcentaje Movimientos Y	0.3

Tabla G.3. Parámetros exclusivos del algoritmo genético adaptado

Parámetro	Descripción
Probabilidad de Mutación	0.50
Probabilidad de Cruce	0.80

Tabla G.4. Parámetros exclusivos del cálculo del área

Parámetro	Descripción
Pixel por celda	50

Tabla G.5. Resultados Algoritmo Memético

Parámetro	Ejecución 1	Ejecución 2	Ejecución 3
Fitness	50.74	50.78	25.93
Área cubierta	7451.90	7811.75	9365
Porcentaje del área:	%74.51	%78.12	%93.66
Antenas utilizadas	2	2	4

Tabla G.6. Resultados Algoritmo Genético

Parámetro	Ejecución 1	Ejecución 1	Ejecución 1
Fitness	20.98	50.75	50.85
Área cubierta	9876.81	7501.06	8500.82
Porcentaje del área:	%98.76	%75.01	%85.00
Antenas utilizadas	5	2	2

En las 6 ejecuciones ambos algoritmos no finalizaron sus ejecuciones debido a algún error.

Calculo del área:

Adicionalmente se utiliza un módulo de pruebas para evaluar el comportamiento del cálculo del área, se nuestra el pseudocodigo en la Figura G.1.

```
Prueba()
BEGIN
    individuo.inicializar()
    totalAntenas <- 1
    antenasEscogidas.inicializar()
    FOR(j = 0 TO j <- limiteVertical - 1){
        FOR(k = 0 TO k = limiteHorizontal - 1){
            individuo.celda[j][k].inicializar()
            IF(antenasEscogidas.contiene(j,k)){
                antena.inicializar(j,k)
                individuo.celda[j][k].asignarAntena()
                individuo.antenas.agregar(antena)
            }
        }
    }
    individuo.calcularFitness()
END
```

Figura G.1. Módulo de Prueba

El comportamiento de este módulo es igual a la generación de población inicial, pero se coloca una sola antena y en el centro del almacén con el fin de determinar el área calculada.

Tabla G.7. Relación de Áreas

Muestra/Parámetro	Área Real	Área cubierta	Radio	Pixeles por lado de celda
1	78.5398163397448309615 66084581988	76.07843137254903	5	25
2		77.3119429590018		50
3		77.71360665478312		75
4	6361.72512351933130788 6852851141	6339.750445632797	45	25
5		6351.008912655969		50
6		6354.530402059811		75
7	1963.49540849362077403 91521145497	1951.3868092691616	25	25
8		1957.508021390374		50

9		1959.5309962368783		75

Se puede observar que el área calculada nunca supera al área real y el número de pixeles por lado de celda determina la precisión de la aproximación.

Cada muestra tuvo el mismo valor en 10 ejecuciones.

H. Conjuntos de datos

H.1 Data usada en la calibración

Se presentan los datos a partir de los cuales se obtuvo la información necesaria para realizar la calibración de variables.

https://docs.google.com/spreadsheets/d/13O7DWaWWnGNaWZeZUAUpT73kVaWn9HVE/edit?usp=sharing&ouid=108268562402702132966&rtpof=true&sd=true

H.2 Muestras recolectadas para la experimentación

Se presentan las muestras a partir de las cuales se pudo realizar el proceso de experimentación numérica y se ubican en la Tabla G.7.

Tabla H.1. Muestras recolectadas para la experimentación

ALGORITMO MEMÉTICO			ALGORITMO GENÉTICO		
Mayor Fitness	Fitness Promedio	Desv. Estandar	Mayor Fitness	Fitness Promedio	Desv. Estandar
1.39487914438502	1.39051436720142	0.00462359957657	1.36604349376114	1.33304327985739	0.03047199873474
1.35634300000000	1.34155870000000	0.01739680812391	1.35033868092691	1.31282267379679	0.02522514127225
1.32435793226381	1.31599575757575	0.00791568901001	1.30945169340463	1.26595532976827	0.02585995598479
1.29108163992869	1.28761693404634	0.00287214872998	1.27037754010695	1.24362834224599	0.02432941021448
1.26204848484848	1.25878716577540	0.00369982764781	1.23672941176470	1.20024762923350	0.01869391590180
1.24999999999999	1.23777771836007	0.01034272026493	1.21748556149732	1.18252577540106	0.02285581757324
1.39487914400000	1.38201490190000	0.01785521036372	1.38207344028520	1.34655732620321	0.01892955994120
1.35634295900178	1.35266713012477	0.00341746457861	1.34440035700000	1.32434256690000	0.01723449450809
1.32435793226381	1.31915087344028	0.00713427841345	1.30909590017825	1.28580039215686	0.02185313898052
1.29140677361853	1.28901037433154	0.00413309765261	1.28958074866310	1.24838591800356	0.02071980566514
1.26204848484848	1.25825657754010	0.00361393081753	1.24338217468805	1.21903383244206	0.01468505395199
1.24999999999999	1.24444595365418	0.00660319642248	1.24689732620320	1.19258516934046	0.02521185269240
1.39487914438502	1.39242253119429	0.00334896375617	1.39060677361853	1.36462677361853	0.01334006228194
1.35634295900178	1.35266149732620	0.00287528775029	1.34638716577540	1.33117272727272	0.01505048279242

1.32435793226381	1.32036178253119	0.00386297354300	1.30868948306595	1.29773094474153	0.00996375196833
1.29140677361853	1.28950898395721	0.00329111727716	1.28375650623885	1.25968588235294	0.01322060438515
1.26164313725490	1.26009522281639	0.00151303966112	1.25048663101604	1.23373928698752	0.01260482683632
1.24999999999999	1.24656420677361	0.00521591420175	1.23137219251336	1.19965483065953	0.01428208559884
1.39487914438502	1.39157568627450	0.00378387697893	1.38683351158645	1.37188634581105	0.01048022345507
1.35634295900178	1.35553786096256	0.00061592090853	1.35182495543671	1.33934716577540	0.01138030028917
1.32435793226381	1.32163821746880	0.00228333993412	1.31536541889483	1.30869051693404	0.01185239899714
1.29140677361853	1.29102039215685	0.00055730749783	1.29108092691622	1.27479803921568	0.00888167346463
1.26204848484848	1.25911654188948	0.00338291522685	1.25250695187165	1.24140303030303	0.00672439195706
1.24999999999999	1.24819497326203	0.00468264089889	1.23534438502673	1.21524042780748	0.01403523515380
1.39487914438502	1.39433632798573	0.00078428484795	1.39487914438502	1.38636249554367	0.00555687907138
1.35634295900178	1.35571757575757	0.00058890374882	1.35415401069518	1.34583511586452	0.00511863624217
1.32435793226381	1.32097390374331	0.00911547823510	1.32080213903743	1.31530488413547	0.00359926996649
1.29140677361853	1.28719379679144	0.01306924395429	1.28722281639928	1.28018631016042	0.00621784441647
1.26204848484848	1.26113126559714	0.00150299155324	1.25431978609625	1.24734427807486	0.00458342639419
1.24999999999999	1.24974873440285	0.00038983144598	1.23429340463458	1.22659811051693	0.00626972886791

Hubo 3 tiempos de ejecución que no se pudo recuperar debido a que a las personas que ejecutaron estos procesos en sus computadoras no pudieron recuperar la información debido a que cometieron un error. Considerando la falta de tiempo y por recomendación del asesor se omitieron estos tiempos.

Se presentan las muestras con mayor detalle en el siguiente enlace:

https://docs.google.com/spreadsheets/d/1yTb2420fDYyD7FLcyPrJKjpELbJ-AALX/edit?usp=sharing&ouid=108268562402702132966&rtpof=true&sd=true