UNIVERSIDAD AUTÓNOMA DE CHIHUAHUA

FACULTAD DE INGENIERÍA

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

OPTIMIZACIÓN DE RUTAS DE TRANSPORTE CON RESTRICCIÓN DE CAPACIDAD UTILIZANDO ALGORITMOS BIO-INSPIRADOS

POR:

ELIANDIS MATOS MOREIRA

TESIS PRESENTADA COMO REQUISITO PARA OBTENER EL GRADO DE

MAESTRO EN INGENIERÍA EN COMPUTACIÓN

CHIHUAHUA, CHIH., MÉXICO **AGOSTO DE 2019**

Optimización de rutas de transporte con restricción de capacidad utilizando algoritmos bio-inspirados. Tesis presentada por Eliandis Matos Moreira como requisito parcial para obtener el grado de Maestro en Ingeniería en Computación, ha sido aprobada y aceptada por:

M.I. Javier González Cantú
Director de la Facultad de Ingeniería

Dr. Alejandro Villalobos Aragón
Secretario de Investigación y Posgrado

M.S.I. Karina Rocío Requena Yáñez
Coordinador(a) Académico

Dr. Luis Fernando Gaxiola Orduño
Director(a) de Tesis

Agosto 2019

Fecha

Comité:
Dr. Luis Fernando Gaxiola Orduño
Dra. Vania Carolina Álvarez Olivas
Dr. Luis Carlos González Gurrola
Dr. Alain Manzo Martínez

Eliandis Matos Moreira
Villa del Sirio 5620, Colonia
Villa del Sol, CP: 31124
Chihuahua, Chih.

UNIVERSIDAD AUTÓNOMA DE
CHIHUAHUA

22 de agosto de 2019

ING. ELIANDIS MATOS MOREIRA
Presente

En atención a su solicitud relativa al trabajo de tesis para obtener el grado de Maestro en Ingeniería en Computación, nos es grato transcribirle el tema aprobado por esta Dirección, propuesto y dirigido por el director **Dr. Luis Fernando Gaxiola Orduño** para que lo desarrolle como tesis, con el título: **OPTIMIZACIÓN DE RUTAS DE TRANSPORTE CON RESTRICCIÓN DE CAPACIDAD UTILIZANDO ALGORITMOS BIO-INSPIRADOS.**

FACULTAD DE INGENIERÍA
Circuito No.1, Campus Universitario 2
Chihuahua, Chih., México. C.P. 31125
Tel. (614) 442-95-00
www.fing.uach.mx

UNIVERSIDAD AUTÓNOMA DE
CHIHUAHUA

Capítulo 4. Experimentación, resultados

4.1 Experimentación con GA
4.2 Experimentación con MOGA
4.3 Experimentación con PSO
4.4 Experimentación con ACO
4.5 Resumen de resultados de algoritmos con las configuraciones básicas
4.6 Propuesta de función de cruce para GA y MOGA
4.7 Comparaciones estadísticas entre los resultados de GA y MOGA con y sin la propuesta de cruce
4.8 Resumen de resultados de GA y MOGA con propuesta de cruce
4.9 Propuesta de algoritmo híbrido ACO-MOGA
4.10 Comparaciones con otros trabajos

Capítulo 5. Discusión y aportaciones

5.1 Discusión
5.2 Discusión de la hipótesis
5.3 Aportaciones
5.4 Trabajo futuro

Conclusiones
Referencias

Solicitamos a Usted tomar nota de que el título del trabajo se imprima en lugar visible de los ejemplares de las tesis.

A T E N T A M E N T E
"Naturam subiecit aliis"

EL DIRECTOR

FACULTAD DE INGENIERÍA U.A.CH. DIRECCIÓN

M.I. JAVIER GONZÁLEZ CANTÚ

EL SECRETARIO DE INVESTIGACIÓN Y POSGRADO

DR. ALEJANDRO VILLALOBOS ARAGÓN

FACULTAD DE INGENIERÍA
Circuito No.1, Campus Universitario 2
Chihuahua, Chih., México. C.P. 31125
Tel. (614) 442-95-00
www.fing.uach.mx

Dedicatoria

A mi adorado hijo y mi amada esposa que con su cariño y apoyo hicieron realidad este sueño.

Agradecimientos

Un gran agradecimiento para M.S.I. Karina Rocío Requena Yáñez, quien, como coordinadora de la maestría, me brindó la oportunidad de cursar este posgrado dando un importante cambio en mi vida y que, a la vez, me aconsejó durante todo el camino ayudando a resolver todas mis inquietudes.

Un sincero agradecimiento a mi tutor el Dr. Luis Fernando Gaxiola Orduño, quien se hizo responsable de mi plan de estudios y mi trabajo de tesis. Gracias por su ayuda, tanto en conocimientos y materias como por su comprensión.

Al Consejo de la Ciencia y Tecnología (CONACYT) de México por su gran apoyo económico, sin este no creo haber podido solventar mi estancia en este maravilloso país.

A la Secretaría de Investigación y Posgrado de la Facultad de Ingeniería de la Universidad Autónoma de Chihuahua, por aceptarnos a mí y a mis compañeros extranjeros y permitirnos formar parte de este proyecto.

Resumen

En este trabajo se proponen soluciones al para el problema de ruteo de vehículos abierto con restricción de capacidad y balance entre rutas (OCVRPRB por sus siglas en inglés) a través de algoritmos bio-inspirados. Estos algoritmos son metaheurísticas que se basan en conceptos biológicos. Su flexibilidad y facilidad los han vuelto una opción recurrente para resolver problemas (sobre todo de optimización) de alta complejidad. Para OCVRPRB se diseñó un formato de solución (individuo o cromosoma dependiendo del algoritmo) de tal manera que su adaptación a varios algoritmos sea más sencilla además de la posibilidad de combinar algunos de ellos. Se utilizaron finalmente un algoritmo genético (GA), un algoritmo genético multiobjetivo (MOGA), un algoritmo de optimización de enjambre de partículas (PSO), una variante multiobjetivo de optimización de enjambre de partículas (MOPSO) y un algoritmo de optimización de colonias de hormigas (ACO). Luego de los experimentos; los algoritmos genéticos realizaron un buen desempeño destacándose un poco más GA el multiobjetivo (MOGA). El algoritmo de colonia de hormigas resultó ser muy conveniente para este tipo de problema por su velocidad de convergencia y buenas soluciones. Por último, se implementó una co-hibridación entre ACO y MOGA que alcanzó los mejores resultados para el problema planteado.

Índice de Contenido

Índice de Tablas

Índice de Figuras

Índice de Ilustraciones

OPTIMIZACIÓN DE RUTAS DE TRANSPORTE CON RESTRICCIÓN DE CAPACIDAD UTILIZANDO ALGORITMOS BIO-INSPIRADOS

Capítulo 1. Introducción

El área de la computación es un área rica en oportunidades para realizar investigación básica y aplicada de forma simultánea. Un buen ejemplo es el diseño de algoritmos para problemas computacionalmente complejos, lo cual permite desde un punto de vista matemáticamente riguroso caracterizar problemas con base en sus requerimientos de solución. Por otro lado, podemos a través del estudio de estos problemas proponer mejores formas de solucionarlos en la práctica.

Problemas computacionalmente complejos se encuentran presentes en muchos escenarios de la vida real, tanto en el sector industrial como en el sector académico. Algunos ejemplos representativos son: la generación de horarios de clase en la universidad (timetabling problem) [1], la planeación de rutas de transporte (vehicle routing problem) [2], [3], entre muchos otros problemas de impacto económico, ecológico y social.

La complejidad computacional formalmente caracteriza estos problemas con base en el tiempo que un procedimiento tardaría en obtener una solución. Los problemas computacionalmente complejos son aquellos para los cuales no se conoce (y probablemente no exista) un algoritmo que los pueda resolver de forma perfecta. La primera característica de una solución obtenida de forma eficiente es su optimalidad. La segunda es el tiempo que el algoritmo tardaría en encontrar la solución óptima.

Un método exacto para este tipo de problemas consiste en realizar una búsqueda exhaustiva de todas las posibles soluciones al problema, esto que aparentemente suena simple, tiene como consecuencia que el tiempo de procesamiento para resolver problemas de tamaño razonable tome décadas o incluso siglos de cómputo [8]. La intratabilidad de estos problemas nos impide diseñar un método exacto para obtener la solución óptima, pero existen

opciones que generalmente proporcionan soluciones de buena calidad mediante la implementación de estrategias informadas.

Una de estas estrategias son los métodos de búsqueda mediante algoritmos bio-inspirados, los cuales han demostrado su eficiencia para abordar problemas complejos [4]. En teoría, los algoritmos bio-inspirados garantizan una solución óptima. Aunado a lo anterior, se puede mejorar sustancialmente el desempeño de estos métodos bio-inspirados al combinarlos y generar un híbrido entre dos métodos bio-inspirados o hibridarlos con métodos inteligentes como lógica difusa, es decir, se pueden proponer nuevos y mejores métodos bio-inspirados para solucionar problemas complejos.

Este trabajo se va a enfocar en encontrar soluciones al problema de enrutamiento de vehículos para el transporte de personal existente en la industria maquiladora y por tanto se define lo siguiente como objetivo general:

Diseñar algoritmos bio-inspirados para el problema integrado de planeación de rutas y balance de distancias en el transporte de trabajadores de la industria maquiladora.

Para cumplir con lo anterior, se plantearon los siguientes objetivos específicos:

- Identificar, analizar y caracterizar el problema VRP en sus distintas variantes en la literatura.
- Analizar los métodos y soluciones actuales en las para los problemas de rutas de vehículos y balance de rutas.
- Implementar al menos dos estrategias de solución.

La hipótesis de este trabajo se basa en que el diseño de algoritmos bio-inspirados para los problemas de rutas de vehículos resultará en una solución automatizada capaz de optimizar el número de autobuses, la distancia total y el balance entre las distancias a recorrer en cada ruta.

La justificación para este estudio es que existen en la ciudad de Chihuahua empresas que utilizan docenas de autobuses para prestar este servicio de transporte de personal a miles

de trabajadores. La planeación de las rutas de estos autobuses se realiza de forma manual. Aunado a lo anterior, dada la naturaleza de la industria maquiladora, la cual presenta una alta rotación de personal, se tienen que estar diseñando nuevas rutas periódicamente. El diseño de una heurística computacional aliviaría en gran medida esta tarea, además de los ahorros en tiempo (de traslado a los mismos trabajadores), económicos a la empresa y ecológicos (al reducir el monto de gases contaminantes por los vehículos).

En el sector industrial surge esta necesidad, pero las instituciones académicas también se podrían ver beneficiadas por este desarrollo tecnológico, ya que prestan servicios similares que pueden ser caracterizados como problemas de ruta de vehículos escolares.

1.2 ESTADO DEL ARTE

Actualmente hay mucha literatura sobre el problema de rutas de vehículos (VRP por sus siglas en inglés), dentro de las más significativas para este trabajo se encuentran las que se mencionan a continuación.

Según Baker y Ayechew [5], los mejores resultados conocidos para los VRP de referencia se han obtenido utilizando la búsqueda tabú o el recocido simulado. Los algoritmos genéticos (GA por sus siglas en inglés) se han aplicado ampliamente a varios problemas de optimización combinatoria, incluidos ciertos tipos de problemas de enrutamiento de vehículos, especialmente cuando se incluyen ventanas de tiempo. Sin embargo, según describen, hasta ese momento no parecían haber tenido un gran impacto en el VRP. Sus resultados computacionales se dan para un GA puro y obtienen resultados adicionales usando un híbrido de este GA con los métodos de búsqueda de vecindario, lo que demuestra que este enfoque es competitivo con la búsqueda tabú. El GA sencillo que proponen según sus resultados estadísticos funciona bien, aunque no equivale a la búsqueda tabú en términos de calidad de la solución. Sin embargo, la incorporación de tipos simples de búsqueda de vecindario en el GA produce una mejoría significativa, dando distancias que son solo 0.5% superiores a los mejores resultados conocidos hasta ese momento en promedio, con tiempos

de solución que no son excesivos. Llegan a la conclusión de que GA es un enfoque eficaz para resolver el VRP básico. Aunque el GA puro que desarrollaron funciona razonablemente bien, deciden que es mejor verlo más como un medio para diversificar la exploración del espacio de soluciones, junto con la búsqueda de vecinos. Por tanto, en ese sentido, es un fuerte competidor con otras heurísticas modernas para el VRP.

En un trabajo en el 2010, Ghoseiri y S. F. Ghannadpour [6], utilizan un algoritmo genético para resolver el problema multiobjetivo de enrutamiento de vehículos con ventanas de tiempo (VRPTW por sus siglas en inglés). Utilizan una interpretación directa del VRPTW como un problema multiobjetivo en el que se minimiza tanto el tamaño total de la flota requerida como la distancia de desplazamiento total, mientras que las limitaciones de la capacidad y las ventanas de tiempo están aseguradas. Según las conclusiones de los autores, la solución genera resultados competitivos.

Por otra parte, en el trabajo de Geetha en el año 2010 [7], se incorpora al algoritmo de optimización de enjambre de partículas (PSO por sus siglas en inglés) los operadores de cruce y mutación de GA. En su enfoque generan las partículas iniciales utilizando un algoritmo codicioso y la actualizan utilizando inercia variable. El cruce y la mutación del operador de GA lo incorporan al PSO para una mejor exploración de partículas. Dentro de sus resultados resaltan una mejora en el tiempo polinomial para resolver el VRP. Su enfoque demuestra las similitudes entre PSO y GA además de la posibilidad de unificar ambas estrategias para resolver problemas computacionalmente complejos como el VRP.

En G. Vaira [3] se proponen operadores del algoritmo genético, que implican el enfoque de destrucción y reconstrucción del uso de grandes barrios de búsqueda (large neighborhood search: LNS) en operadores de cruce y mutación, donde la heurística de inserción aleatoria en operadores genéticos se utiliza como método de reconstrucción. A diferencia de los enfoques de cruce tradicionales, utilizan nuevos operadores de cruce que se basan en la búsqueda de partes comunes en las soluciones principales para la generación de

la descendencia. Su propuesta produce soluciones en poco tiempo y con iguales o mejores resultados que otros algoritmos genéticos. Además, los operadores propuestos no están diseñados para un cierto problema específico y pueden aplicarse a diferentes problemas, por lo que hace a esta solución muy atractiva como punto de partida o comparación para esta investigación.

En el año 2015, Zuñiga Aguirre [8], desarrolla un modelo para el Problema de Ruteo de Vehículos con restricciones de capacidad y distancia (OVRPCD Open Vehicle Routing Problem Capacitated with Distance constraint) para el transporte de personal con balanceo. El método que utilizaron fue Programación Lineal Entera Mixta (MILP) o Metaheurística como un Procedimiento de Búsqueda Voraz Aleatorio y Adaptativo (GRASP). Este estudio reporta buenas soluciones en la mayoría de los casos en un tiempo de ejecución reducido.

En otro estudio de 2016, de Jairo G Carballo [9], se enfoca en realizar una comparación de las soluciones que genera un algoritmo genético cuando se utilizan diferentes funciones objetivos para el problema de rutas de vehículos con rutas balanceadas (VRPRB por sus siglas en inglés: Vehicle Routing Problem with Route Balancing). En sus resultados destacan que en más del 95% de las ocasiones, en las diferentes funciones objetivos (FO) obtuvieron puntajes muy distintos sin importar significativamente los operadores de cruce o mutación.

En el trabajo de Yahui Liu y Buyang Cao en el 2018 [10] definen a los VRP con restricciones de capacidad homogénea, capacidad heterogénea, tipos de vehículos por clientes (puntos de demanda) y ventanas de tiempo como modo completo de vehículos (FVM-VRP por su siglas en inglés). Su solución se basa en un algoritmo de optimización de colonias de hormigas (ACO por siglas en inglés) para resolver problemas de FVM-VRP. Su resultado es un ACO mejorado capaz de lidiar con las múltiples restricciones antes mencionadas; las soluciones de su algoritmo, afirman los autores, son satisfactorias y que pueden ser prometedoras para aplicaciones reales.

Por otro lado, hay investigaciones relacionadas con el uso de algoritmos bio-inspirados para otros tipos de problemas, multiobjetivo o simples, y de igual manera sus resultados son alentadores.

Saravanan, Asokan y Sachidanandam [11] hacen un estudio en 2002 sobre un enfoque de algoritmo genético multiobjetivo (GA) para la optimización de las operaciones de rectificado superficial. Su documento describe un procedimiento de optimización basado en GA para optimizar las condiciones de trituración, velocidad de la rueda, velocidad de la pieza de trabajo, profundidad del vendaje y plomo del vendaje, utilizando un modelo de función multiobjetivo con un enfoque ponderado para el rectificado superficial. Su procedimiento evalúa las condiciones de trituración óptimas sometidas a restricciones tales como daño térmico, parámetros de desgaste de la rueda y rigidez de la máquina herramienta. Su enfoque es multiobjetivo, pero poseen una sola función objetivo que es el resultado de un modelo matemático que mezcla las características antes mencionadas. Sus resultados con GA superan la técnica de programación cuadrática (QP) con la que se compararon, además analizaron la misma técnica solo con la función de minimizar el costo y de igual manera aseguran obtener buenas soluciones.

Capítulo 2. Fundamentación Teórica

En este capítulo se mencionan las variantes de VRP con mayor relación al objetivo de este trabajo. Asimismo, se describen las características de los algoritmos bio-inspirados que serán posteriormente implementados y se analizan soluciones a problemas similares.

2.1 Problema de Ruteo de Vehículos (VRP)

El problema de ruteo vehicular es una parte de la investigación de operaciones en donde se manejan restricciones que tratan de reflejar con mayor realismo la operación de los vehículos y que consecuentemente aumentan dificultad a la búsqueda de una solución factible. Es un problema combinatorio NP-completo (No-determinístico polinomial), este tipo de problemas no se pueden resolver en un tiempo polinomial; donde el tiempo polinomial es cuando el tiempo de ejecución de un algoritmo (mediante el cual se obtiene una solución al problema) es menor que un cierto valor calculado a partir del número de variables implicadas (generalmente variables de entrada), usando una fórmula polinómica [2].

El VRP se define en una red no dirigida completa $G = (V; E)$ con un conjunto de nodos $V = \{0; 1; ...; n\}$ y un conjunto de aristas E. El nodo 0 es un depósito con m vehículos idénticos de capacidad W; m se puede fijar a priori o se deja como una variable de decisión. Cada nodo $i > 0$ representa un cliente con una demanda no negativa qi y cada arista $(i; j)$ tiene un costo de viaje no negativo $cij = cji$. El VRP consiste en determinar un conjunto de m viajes de vehículo de costo total mínimo, de modo que cada viaje comienza y termina en el depósito, cada cliente es visitado exactamente una vez [12]. Existen varios tipos de restricciones que identifican las diversas variantes del VRP, algunas de ellas son las siguientes:

- CVRP. Problema de ruteo de vehículos con restricción de capacidad
- DVRP. Problema de ruteo de vehículos con restricción de distancia
- VRPCD. Problema de ruteo de vehículos con restricción de capacidad y distancia

- VRPTW. Problema de ruteo de vehículos con ventanas de tiempo
- VRPPD. Problema de ruteo de vehículos con recogidas y entregas
- OVRPCD. Problema abierto de ruteo de vehículos con restricción de capacidad y distancia

2.1.1 Problema de Ruteo de Vehículos con Restricción de Capacidad (CVRP)

En su aplicación a situaciones reales, el VRP se ve sujeto a múltiples restricciones que son inherentes a las diversas variantes del problema. La adición de restricción de capacidad a los vehículos constituye la primera y más básica de las variantes del VRP, el Problema de Ruteo de Vehículos con Restricción de Capacidad. Éste puede ser modelado como un grafo en el que los vértices consisten en n clientes más un depósito único. Cada cliente debe servirse con una cantidad qi de bienes $(i = 1, 2, ..., n)$ del depósito. Para entregar (o recolectar) estos bienes, se dispone de una flota de vehículos, cada uno de los cuales tiene asociada una máxima cantidad de bienes Q que puede transportar. Una solución de CVRP consiste en una colección de rutas donde cada cliente es visitado una y sólo una vez y la demanda total para cada vehículo es menor o igual a Q. De igual manera que en el VRP, en el CVRP se busca minimizar la distancia total recorrida por el conjunto de rutas [9].

2.1.2 Problema de Ruteo de Vehículos Abierto con Restricciones de Capacidad y Distancia (OVRPCD)

Para esta variante del VRP, se integran las restricciones de capacidad del CVRP y restricciones de distancia del DVRP lo que determina la variante del VRPCD, pero además se considera que las rutas son abiertas, es decir, a diferencia de las variantes anteriores donde el depósito es el punto inicial y final, aquí se inicia o se finaliza en el depósito, pero se puede finalizar o iniciar en cualquier nodo [8].

2.1.3 Problema de Ruteo de Vehículos con Balanceo de Rutas (VRPRB)

Constituye una variante multi objetivo del CVRP en la cual, además de buscar minimizar la distancia total recorrida por los vehículos con la restricción de capacidad máxima, busca un balance entre las rutas, esto puede ser en términos de la carga total del vehículo o la longitud recorrida por cada ruta, siendo la distancia quizá el criterio más comúnmente utilizado. Se busca introducir cierto grado de equidad a la solución sin descuidar el aspecto principal del problema [9].

2.1.4 Balanceo de Rutas

Resulta inapropiado optimizar rutas de transporte si esto implica generar cargas diferentes de trabajo para los conductores. Por lo que es necesario que el balance esté presente al generar rutas para vehículos en un problema real. En el trabajo de Jairo Lozano de 2016 [9] refieren además que no se ha encontrado en la literatura del VRPRB una comparación cuantitativa de la calidad de las soluciones generadas por un AG al seleccionar diferentes funciones objetivo (FO) para medir el balanceo entre rutas.

Existen diversas funciones para calcular el balanceo de rutas, pero este trabajo se va a enfocar en las siguientes tres [9]:

- Standard Deviation (Std_Dev). Minimiza la desviación estándar de las longitudes de las rutas.

$$min\, f(x) = \sqrt{\frac{1}{|T|}\sum_{t\epsilon T}\left(l_t - \bar{l}\right)^2} \qquad (2.1)$$

- Mean Absolute Deviation (MAD). Minimiza el promedio de las diferencias entre la longitud de cada ruta respecto al promedio de las longitudes de las rutas.

$$min\, f(x) = \frac{1}{|T|}\sum_{t\epsilon T}|l_t - \bar{l}| \qquad (2.2)$$

- Gini Coefficient (Gini). Minimiza el coeficiente de Gini, uno de los índices más utilizados en estudios de equidad en general.

$$min\, f(x) = \frac{1}{2|T|^2 \bar{l}} \sum_{t \epsilon T} \sum_{t' \epsilon T} |l_t - l_{t'}| \qquad (2.3)$$

2.2 DESCRIPCIÓN DE MÉTODOS DE SOLUCIÓN

En la mayor parte de la literatura estudiada se identifican los siguientes métodos para resolver el problema de ruteo de vehículos [1], [5], [13], [14]:

- Procedimientos Exactos
- Heurísticas
- Metaheurísticas

2.2.1 Procedimientos Exactos

Como su nombre lo indica, este enfoque propone calcular todas las soluciones posibles hasta que se alcance el mejor resultado (ejemplos: Branch & Bound, Branch & Cut) [13]. Dada la complejidad de los problemas, sólo las instancias con pocos clientes pueden ser resueltas consistentemente por métodos exactos. En este tipo de metodologías, suele resolverse alguna relajación del problema usando por ejemplo Branch & Bound. También se han propuesto algoritmos basados en Programación Dinámica que aceleran los cálculos mediante una relajación del espacio de estados. Por otro lado, algunos problemas, se pueden resolver usando métodos de generación de columnas [15].

2.2.2 Heurísticas

Las heurísticas, son procedimientos simples que realizan una exploración limitada del espacio de búsqueda y dan soluciones de calidad aceptable (no necesariamente óptimas) en tiempos de cálculo moderados. Son menos sofisticadas algorítmicamente que los métodos exactos, por lo que es más fácil programarlas y a la vez son más fáciles de comprender [15].

2.2.3 Metaheurísticas

Para obtener mejores resultados que con los métodos heurísticos, es necesario recurrir a técnicas que realicen una mejor exploración del espacio de soluciones. En los diferentes trabajos de investigación se hace evidente que, por su complejidad, la manera más eficiente de solucionar esta clase de problemas, en términos de tiempo y costo, es el uso de metaheurísticas. Una metaheurística es un conjunto de conceptos que se pueden utilizar para definir los métodos heurísticos que se pueden aplicar a una amplia serie de diferentes problemas. En otras palabras, una metaheurística puede ser vista como un marco algorítmico general que se puede aplicar a diferentes problemas de optimización con relativamente pocas modificaciones para que sean adaptados a un problema específico [13]. A continuación, una breve lista de estos algoritmos:

- Ant Algorithms (Colonia de hormigas)
- Constraint Programming (Programación de restricciones)
- Deterministic Annealing (Recocido determinístico)
- Genetic Algorithms (Algoritmo Genético)
- Simulated Annealing (Recocido simulado)
- Particle Swarm Optimization (optimización de enjambre de partículas)
- Taboo Search (Búsqueda Tabú)

2.3 PROBLEMA MULTIOBJETIVO

En este trabajo se identificaron tres objetivos, reducir el número de autobuses a utilizar, balancear las rutas y que el recorrido (distancia total a recorrer) en general sea el más corto. Minimizar los objetivos generalmente es un conflicto, ya que la mejora en un aspecto puede comprometer al valor de otro, por lo que, encontrar una única solución no es factible, en vez de eso se buscan un conjunto de soluciones que represente el mejor balance entre todos los objetivos. Este tipo de situaciones en la literatura se representan como problemas

multiobjetivo, donde el conjunto de soluciones óptimas en un espacio de decisión X se denota en general como conjunto óptimo de Pareto [16].

Para comparar las soluciones será necesario utilizar alguna métrica que permita la comparación de problemas de esta índole. En estos tipos de casos en particular lo más recomendado es el hipervolumen ya que se conoce que permite detectar entre varios conjuntos de soluciones si son significativamente diferentes [16].

Los algoritmos bio-inspirados son metaheurísticas que se basan en conceptos biológicos. Su flexibilidad y facilidad los han vuelto una opción recurrente para resolver problemas (sobre todo de optimización) de alta complejidad. Algunos tienen una estructura similar y esa característica resulta atractiva para el desarrollo de un modelo del problema de tal manera que no sufran muchas modificaciones para experimentar con estos algoritmos [17]. En la Tabla 2.1 se detalla una pequeña comparativa entre algunos de estos algoritmos.

Tabla 2.1: Comparación de algoritmos bio-inspirados [18], [19]

Nombre	Operadores	P. de control	Ventajas	Aplicaciones
Algoritmo Genético (GA).	Cruzamiento, mutación, selección.	Tamaño de la población, número máximo de generaciones, probabilidad de sobrecruzamiento, probabilidad de mutación, tamaño del cromosoma.	Flexible, puede ser utilizado en problemas difíciles y poco conocidos, bajo coste computacional.	Ingeniería de control y procesamiento de señales, robótica, reconocimiento de patrones, optimización de diseños, reconocimiento de idiomas o sistemas de clasificación.
Programación Genética (GP).	Cruzamiento, mutación, selección.	Tamaño de la población, número máximo de generaciones, probabilidad de sobrecruzamiento, probabilidad de mutación.	Permite crear y optimizar programas enteros, aplicación con tamaño de solución variable.	Reconocimiento de patrones, clasificación, detección de cáncer.

Optimización de Enjambre de Partículas (PSO).	Inicializador, actualizador y evaluador.	Número de partículas, dimensión de partículas, rango de las partículas, velocidad máxima, factores de aprendizaje, número máximo de interacciones.	Eficaz, fácil de implementar, capacidad de memoria, alta variabilidad.	Optimización de funciones continuas no lineales, sistemas de energía eléctrica, procesamiento de imágenes.
Optimización de Colonia de Hormigas (ACO).	Actualizador de feromonas, medidor de feromonas, evaporación de feromonas.	Número de hormigas, tasa de evaporación de feromona, número de iteraciones, refuerzo de feromonas.	Muy eficaz.	Optimización, industria, telecomunicaciones, transportes.
Búsqueda tabú.	Búsqueda local a través de estructura de vecindad.	Desperdicio general y desperdicio individual. Depende en gran medida de sus parámetros iniciales.	Eficaz.	Se ha aplicado con éxito a la resolución de grandes combinatorias.

2.4 ALGORITMO GENÉTICO (GA)

El algoritmo genético (GA por sus siglas en inglés) es una técnica de optimización y búsqueda como resultado de una abstracción de la teoría de evolución de Darwin, basada en los principios de la genética y selección natural. Un GA permite que una población compuesta por muchos individuos evolucione, según las reglas de selección especificadas, a un estado que maximice o minimice la función objetivo en dependencia del contexto. El resultado final es que solo los individuos con mejores valores sobreviven. Un algoritmo genético se puede dividir en varias sub-partes: representación, evaluación de la función de aptitud o función objetivo (FO), inicialización, selección, recombinación (cruzamiento y mutación), terminación [3].

2.4.1 Selección de padres

La selección de padres se usa para identificar cromosomas que se usarán en la reproducción y sobrevivirán en la próxima generación. Existen diferentes técnicas para la selección, sin embargo, generalmente se simula un proceso de selección natural, donde los individuos "más fuertes" (mejor FO) se utilizan en la reproducción, de esta forma se garantizan hijos con buenas características [3].

2.4.2 Recombinación o cruce

Los operadores de cruce son una parte importante del algoritmo genético ya que simula la reproducción entre dos individuos donde los descendientes heredan características de los progenitores [3].

2.4.3 Mutación

La mutación es el proceso de crear un nuevo individuo al cambiar algunas características de una solución (cromosoma). Este procedimiento tiene como principal objetivo aumentar la posibilidad de un óptimo global y no estancarse en un óptimo local. Tanto el operador de cruce como la mutación se aplican mediante una probabilidad predefinida [3].

2.5 ALGORITMO GENÉTICO MULTIOBJETIVO (MOGA)

La mayoría de los problemas de optimización en aplicaciones del mundo real, implican la optimización de más de un objetivo. Los objetivos en la mayoría de los problemas suelen ser contradictorios, es decir, maximizar el rendimiento, minimizar los costes, maximizar la fiabilidad, etc. En este caso, una solución extrema no satisfaría ambas funciones objetivas y la solución óptima de un objetivo no sería necesariamente la mejor solución para otro(s) objetivo(s). Por lo tanto, las diferentes soluciones producirán compensaciones entre diferentes objetivos y se requiere un conjunto de soluciones para representar las soluciones óptimas de todos los objetivos. Básicamente, un problema de optimización multiobjetivo

tiene más de una función objetivo, en problemas de ingeniería generalmente dos objetivos [20].

2.5.1 Algoritmo genético multiobjetivo: NSGA-II

Existen varios estudios que demuestran que existen operadores adicionales que se deben incluir en un algoritmo evolutivo EA (Evolutive Algorithm por sus siglas en inglés). Dentro de estos operadores se encuentran asignar una aptitud física a los miembros de la población en función de las soluciones no dominadas (Rango 1) y preservar la diversidad entre las soluciones del mismo frente no dominado. Particularmente, el interés ha sido introducir el elitismo para mejorar las propiedades de convergencia de un MOEA (Multi-Objective Evolutive Algorithm) [21].

Según Deb, Pratap, Agarwal y Meyarivan [21] la propuesta NSGA-II, reemplazan el enfoque de la función de compartir con un enfoque de comparación abarrotada que elimina las dificultades anteriores en cierta medida. Este enfoque no requiere ningún parámetro definido por el usuario para mantener la diversidad entre los miembros de la población y posee una mejor complejidad computacional O(MN2). Este proceso se basa en que, para cada función objetivo, se asigna un valor de distancia infinita a las soluciones de límites (soluciones con valores de función más pequeños y más grandes). A todas las demás soluciones intermedias se les asigna un valor de distancia igual a la diferencia normalizada absoluta en los valores de función de dos soluciones adyacentes. Luego realizan un ordenamiento en función de las distancias y se irán agregando a la próxima generación en ese orden. De esta manera aseguran diversidad en las poblaciones para disminuir la posibilidad de quedar atrapados en óptimos locales.

2.6 Optimización de Enjambre de Partículas (PSO)

Está inspirado en patrones de comportamiento social de organismos que viven e interactúan dentro de grandes grupos. En particular, incorpora comportamientos de observados en bandadas de aves, bancos de peces o enjambres de abejas, e incluso el

comportamiento social humano, del que ha surgido el paradigma de la "inteligencia de enjambre". Podría implementarse y aplicarse fácilmente para resolver diversos problemas de optimización de funciones. Su principal fortaleza es su rápida convergencia, que se compara favorablemente con muchos otros algoritmos de optimización global [22].

Para la imitación de lo que ocurre en la naturaleza, se establece un enjambre de "**partículas**" (posibles soluciones) que serían puntos en la nube de soluciones, capaces de hacer cálculos sencillos. Cada partícula tiene dos parámetros, la posición que es la solución de la nube en la que se encuentra actualmente, y la velocidad a la que se mueve por los ejes de la nube de soluciones (determinando así también la dirección de la partícula). Además, cada partícula es capaz de "recordar" la posición con mejor aptitud en la que se ha encontrado (***pbest***) y la posición con mejor fitness de todas las partículas vecinas (***gbest***). En cada iteración la nueva posición y velocidad de la partícula viene determinada por sus valores anteriores, pero también está influenciada con una intensidad aleatoria por ***pbest*** y ***gbest***. De esta forma se guía a la partícula hacia una zona con mejores soluciones [18].

Su ejecución se basa en los siguientes pasos [18]:

1. Iniciación, se coloca cada partícula del enjambre en una posición aleatoria en la nube de soluciones.
2. Evaluación del fitness de las partículas.
3. Comparación del valor del fitness de cada partícula con su ***pbest*** (mejor encontrado hasta el momento) correspondiente.
4. En el caso de ser mayor, se sustituye el ***pbest*** por la nueva posición.
5. Identificación de la partícula con mejor fitness sustituyendo ***gbest*** (mejor solución encontrada de forma global por sus vecinas***)*** por el valor de su posición.
6. Modificación de los parámetros de cada partícula y cálculo de su nueva posición.
7. Repetición de los pasos hasta que se llegue al número de ciclos indicado o a un fitness aceptable.

El movimiento de una partícula de enjambre consta de dos componentes principales: un componente estocástico y un componente determinista. En la Ecuación 2.4 se muestra cómo se calcula el valor de la velocidad $\boldsymbol{v_i^{t+1}}$. Sean $\boldsymbol{x_i}$ y $\boldsymbol{v_i}$ el vector de posición y la velocidad de la partícula $\boldsymbol{i}$, respectivamente; $\boldsymbol{g^*}$ es el actual mejor valor global y $\boldsymbol{x_i^*}$ su mejor posición en la historia. El nuevo vector de velocidad está determinado por:

$$v_i^{t+1} = v_i^t + \alpha\varepsilon_1[g^* - x_i^t] + \beta\varepsilon_2[x_i^* - x_i^t] \tag{2.4}$$

donde $\boldsymbol{\varepsilon_1}$ y $\boldsymbol{\varepsilon_2}$ son dos vectores aleatorios, y cada entrada toma los valores entre 0 y 1. Los parámetros $\boldsymbol{\alpha}$ y $\boldsymbol{\beta}$ son los parámetros de aprendizaje o las constantes de aceleración, que normalmente se pueden tomar como $\alpha \approx \beta \approx 2$. La nueva posición $\boldsymbol{x_i^{t+1}}$ se calcula como se muestra en la Ecuación 2.5 [22].

$$x_i^{t+1} = x_i^t + v_i^{t+1} \tag{2.5}$$

2.7 Optimización de Enjambre de Partículas Multiobjetivo (MOPSO)

El PSO por la alta velocidad de convergencia parece particularmente adecuado para la optimización multiobjetivo. El MOPSO permite que el algoritmo PSO sea capaz de lidiar con los problemas de optimización multiobjetivo. La analogía de PSO con algoritmos evolutivos hace que usar un esquema de clasificación de Pareto sea una forma más sencilla de ampliar el enfoque para manejar los problemas de optimización multiobjetivo. El registro histórico de las mejores soluciones encontradas por una partícula (individuo) podría usarse para almacenar soluciones no dominadas generadas en el pasado (esto sería similar a la noción de elitismo utilizado en la optimización multiobjetivo evolutiva). El uso de mecanismos de atracción global combinados con un archivo histórico de vectores no

dominados encontrados previamente motivaría la convergencia hacia soluciones no dominadas globalmente [23].

2.7.1 Operador de mutación en algoritmo de optimización de enjambre de partículas multiobjetivo

En el trabajo de Carlos Coello [23], se utiliza el concepto de mutación en PSO. Se hace referencia a la velocidad de convergencia de PSO y de cómo dicha rapidez puede ser perjudicial en el contexto de la optimización multiobjetivo ya que el algoritmo puede quedar estancado en un óptimo local. Esto motivó el desarrollo de un operador de mutación que intenta explorar con todas las partículas al comienzo de la búsqueda, disminuyendo con respecto al número de iteraciones, el número de partículas que son afectadas por el operador de mutación. El operador de mutación propuesto se aplica no solo a las partículas del enjambre, sino también al rango de cada variable de diseño del problema a resolver. Con este comportamiento se intenta cubrir el rango completo de cada variable de diseño al comienzo de la búsqueda y luego reducir el rango cubierto con el tiempo, utilizando una función no lineal.

El uso de operadores de mutación en PSO no es nuevo. Sin embargo, el enfoque mencionado anteriormente no solo es agregar capacidades exploratorias a PSO, sino que también garantizar que se explore el rango completo de cada variable de decisión. Según los autores, este tipo de operador de mutación es novedoso, al menos en el contexto de los enfoques de PSO utilizados para la optimización multiobjetivo [23].

El enfoque mencionado anteriormente es similar al proceso de mutación de los GA, esto puede afectar considerablemente la velocidad de convergencia del MOPSO, pero garantiza una mejor exploración aumentando las posibilidades de encontrar mejores soluciones.

2.8 OPTIMIZACIÓN DE COLONIA DE HORMIGAS (ACO)

El algoritmo de optimización de colonia de hormigas (ACO por sus siglas en inglés) es un método inspirado por el comportamiento de búsqueda de comida que realizan las colonias de algunas especies de hormigas, por una forma de comunicación conocida como estigmergia (forma de comunicación no simbólica, local, que se produce al modificar el ambiente.) [18].

2.8.1 Fase de Construcción

Inicialmente, cada hormiga se coloca en un punto elegido al azar. En cada paso de la construcción, la hormiga *k* aplica una regla de elección de acción probabilística. En particular, la probabilidad con la que la hormiga *k*, actualmente en el punto *i*, elige ir al punto *j* en la iteración del algoritmo es [24]:

$$p_{ij}^{k} = \frac{[\tau_{ij}(t)]^{\alpha} * [n_{ij}]^{\beta}}{\sum_{l \in N_i^k} [\tau_{il}(t)]^{\alpha} * [n_{il}]^{\beta}} \quad if\ j \in N_i^k \qquad (2.6)$$

donde $\boldsymbol{n_{ij} = 1/d_{ij}}$ es un valor heurístico disponible a priori, α y β son dos parámetros que determinan la influencia relativa del camino de la feromona y la información heurística, y $\boldsymbol{N_i^k}$ es el vecindario factible de la hormiga *k*, es decir, el conjunto de puntos que la hormiga *k* no ha visitado aún. Si $\alpha = 0$, es más probable que se seleccionen las ciudades más cercanas. Si $\beta = 0$, solo funciona la amplificación de feromonas: este método conducirá a la rápida aparición de una situación de estancamiento con la correspondiente generación de recorridos que, en general, son fuertemente óptimos locales [24].

2.8.2 Actualización de feromonas

Después de que todas las hormigas han construido sus recorridos, se actualizan los rastros de las feromonas. El proceso se realiza reduciendo primero la fuerza de la feromona en todos los arcos por un factor constante (tasa de evaporación) y luego permitiendo que cada hormiga agregue feromona en los arcos que ha visitado en su recorrido (Ecuación 2.7).

$$\tau_{ij}(t+1) = (1-\rho) * \tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}(t) \qquad (2.7)$$

donde $0 < \rho \leq 1$ es la tasa de evaporación de las feromonas. El parámetro ρ se usa para evitar la acumulación ilimitada de los rastros de la feromona y permite que el algoritmo "olvide" las decisiones erróneas tomadas previamente. Si las hormigas no eligen una arista, la potencia de su feromona asociada disminuye la cantidad de feromonas que k coloca en las aristas o arcos que ha visitado es $\boldsymbol{\Delta\tau_{ij}^{k}(t)}$ y se define de la siguiente manera:

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} 1/L^{k}(t) & if\ arc(i,j)\ is\ used\ by\ k \\ 0 & otherwise \end{cases} \qquad (2.8)$$

donde $\boldsymbol{L^{k}(t)}$ es la duración del recorrido de la hormiga k actual. Según la Ecuación 2.8, cuanto mejor es el recorrido de la hormiga, más aroma recibe la feromona. En general, los arcos más utilizados por las hormigas y que están contenidos en recorridos más cortos recibirán más feromonas y, por lo tanto, también es aumenta la probabilidad de que se elijan en futuras iteraciones [24].

2.8.3 Reforzando mejores soluciones

Cuando se han construido las soluciones, a menudo se requieren algunas acciones opcionales. Estas son llamadas acciones de demonio (*daemon actions* en inglés), y se pueden usar para implementar acciones específicas del problema, que no pueden ser realizadas por hormigas individuales. El mayor uso consiste en la aplicación de la búsqueda local a las soluciones construidas: las soluciones optimizadas localmente se usan para decidir qué actualización se realizará [25].

2.8.4 Enfoque híbrido de ACO y GA.

En el año 2001, S. Shtovba, E. Geraldo Nepomuceno, y M. Reimann [26], proponen un nuevo enfoque híbrido para resolver el VRP. El enfoque se basa en la combinación de un

algoritmo ACO con un GA. Los resultados preliminares muestran que el enfoque híbrido no produjo beneficios adicionales con respecto a la calidad de la solución, pero sus autores aclaran que pudo deberse al bajo rendimiento del GA que implementaron. Si bien su enfoque híbrido no superó al algoritmo ACO independiente, los resultados mostraron que, dada una buena solución, el GA puede, por medio de la mutación, mejorar esta solución y, por lo tanto, respaldar a las hormigas para encontrar buenas soluciones más rápidamente.

El estudio anteriormente mencionado es un ejemplo de la posibilidad de aunar fuerzas de más de un algoritmo bio-inspirado con el fin de mejorar la solución final. Sus resultados quizás no fueron los esperados, pero brinda un enfoque que puede ser utilizado en este trabajo.

Hasta este punto se han mencionado conceptos relevantes para la solución de la problemática de este trabajo. Para una representación gráfica se utiliza la teoría de grafos donde se especifican detalles importantes como la conexiones entre los puntos de demanda.

2.9 TEORÍA DE GRAFOS

Un grafo es un conjunto, no vacío, de objetos llamados vértices (o nodos) y una selección de pares de vértices, son denominadas aristas. Pueden ser dirigidos donde la orientación de la arista es en un solo sentido o no dirigidos cuando la conexión es en ambos sentidos. Por lo general se representan como $G = (V, E)$ donde V es el conjunto de vértices (en este caso puntos de demanda) y E es el conjunto de aristas que unen los vértices (líneas o conexiones entre los puntos) [27].

Capítulo 3. Metodología

En este capítulo se describe la metodología utilizada y los métodos de solución propuestos. Varios algoritmos bio-inspirados fueron analizados para este trabajo, entre todos se seleccionaron los que por su diseño sea más común el trabajo de las soluciones como es el caso de GA, PSO y ACO, mencionados en el capítulo anterior.

3.1 Método Propuesto

El enfoque de este trabajo es diseñar algoritmos bio-inspirados para el problema integrado de planeación y balanceo de rutas de transporte de trabajadores de la industria maquiladora. Fue necesario diseñar la estructura del individuo como un conjunto de vehículos que a su vez tengan incluidos las rutas por la que debe pasar y en qué orden. Además, se establecen como restricciones, que durante el recorrido de cada vehículo no exceda su capacidad, partiendo del hecho de que en cada punto no exista una demanda superior a la capacidad máxima de los vehículos y solo pasarán por un punto una sola vez.

Por la naturaleza de este trabajo se utiliza una metodología experimental, de tal manera que el desarrollo va a estar orientado a experimentos sobre la parametrización de los algoritmos seleccionados, así como el estudio de posibles mejoras en las funciones objetivo y la estructura de la solución.

Se cuenta con una base de datos (Tabla 3.1) con 364 puntos incluyendo la ubicación del depósito. Esta base de datos fue a partir de los datos proporcionado por la empresa maquiladora JABIL Circuit de Chihuahua, S. de R.L. de C.V (JABIL) en el 2015 para el trabajo de Eduardo Zúñiga Aguirre [8]. La base de datos se puede consultar en la liga https://github.com/eliandis/vrp_greedy/blob/master/db/tests.csv.

Tabla 3.1: Características de los datos.

Cantidad de puntos (N)	Demandas (D)	Demanda Total (TD)	Capacidad Máxima (Q_0)
364	1 - 8	643	40

El diseño del individuo (posible solución) es de suma importancia para este problema, primeramente, se planteó la idea de valores binarios, pero, el orden del paso de cada vehículo

por los puntos de demandas debe permitir que no sea necesariamente en el orden en que aparecen inicialmente en las bases de datos ejemplo: depósito, punto 9, punto 2, punto 4, etc.

Según lo anterior, la representación del cromosoma quedaría como se muestra en la Figura 3.1.

Figura 3.1: Diseño del cromosoma.

Los primeros genes de cada bloque (con borde más claro en la Figura 3.1) serían el comienzo de cada vehículo con la posibilidad de que se active o no dependiendo de si su valor es 1 (activo) o 0 (inactivo), los otros valores de los recuadros siguientes, representarían los puntos (paradas del autobús) por donde debe pasar. Por ejemplo, las rutas de este cromosoma quedarían de la siguiente manera:

- Vehículo 1: (1: Activo) Depósito – Punto 123– Punto 25– Punto 3– Punto 320, etc.
- Vehículo 2: (0: Inactivo) Depósito – Punto 310– Punto 40– Punto 57– Punto 21, etc.
- Vehículo 3: (1: Activo) Depósito – Punto 241– Punto 2– Punto 10– Punto 17, etc.
- Vehículo 4: (1: Activo) Depósito – Punto 128– Punto 5– Punto 305– Punto 19, etc.

Esta representación es similar a la encontrada en el trabajo de Geetha [7]. La diferencia radica en que, por la inclusión de balanceo de rutas, se hace necesario delimitar donde comienza cada vehículo. Esto surgió ante el caso en que cierto vehículo quede con espacio como para que se agregue otra ruta y no se controle correctamente el balance si la demanda del próximo punto en la representación (posible inicio de otra ruta) puede ser incluida aún en la capacidad del vehículo.

En los algoritmos de un solo objetivo, se utiliza esta representación con una función objetivo que fusiona en una ecuación matemática la cantidad de vehículos a utilizar con la distancia total a recorrer más el balance de las rutas:

$$FO = \left(\sum_{i=1}^{v} on(Vi) * 10^{R2}\right) + 10^{R1} * \left(\sum_{i=1}^{v} on(Vi) * dist(Vi)\right) + balance(V) \qquad (3.1)$$

Donde:

- *FO* es la función objetivo
- *v* es la cantidad de vehículos
- *Vi* es vehículo en la posición i
- *R1* es la cantidad de dígitos de la distancia total
- *R2* es *R1* más la cantidad de dígitos de *v*
- *on()* es la función que determina si está activo el vehículo
- *dist()* es la función que calcula la distancia a recorrer por un vehículo
- *balance()* es la función que calcula el balance entre todas las rutas de una solución o individuo

El proceso de evaluación de cada individuo comienza por una lista de puntos visitados que se inicializa vacía, luego unas variables para contar los vehículos a utilizar (*vehiclesCount*), la demanda total recogida en todos los puntos (*demands*), así como la distancia total a recorrer entre todos los vehículos que conformarían las rutas del cromosoma (*totalDist*). Se fracciona el individuo en los bloques de vehículos con sus respectivas rutas; se analiza cada bloque (vehículo) preguntando primeramente por el primer valor que indica si se activa o no el vehículo (1 o 0). En caso de estar activo, se incrementa la variable de la cantidad de vehículos, se recorre cada punto que lo conforma. Si el punto seleccionado aún no está visitado y la demanda más la acumulada hasta en el momento es menor o igual a la capacidad del vehículo, se adiciona a la lista de puntos visitados y se calcula la distancia entre el anterior y el actual (en caso de ser el primero, la distancia es respecto al depósito), se almacena además la demanda existente en dicha locación.

Al terminar el análisis de todos los vehículos, se verifica que la demanda recogida satisface la demanda total, si cumple con dicha restricción se procede a la evaluación de la fórmula presentada en la Figura 3.1, donde la $\sum_{i=1}^{v} a(Vi)) * 10^{R2}$ se reduce a la multiplicación de la cantidad de vehículos acumulada por 10^{R2}. Luego $\sum_{i=1}^{v} dist(Vi) * a(Vi)$, es la distancia acumulada entre los vehículos que se activaron con los puntos que se seleccionaron para visitar. En el multiobjetivo, la única variación en el proceso de verificación es que la cantidad de vehículos y la distancia total acumuladas forman por separado un objetivo en sí. Para los casos donde no se cumple con la demanda total (no factibles), los valores tienden a infinito para descartar al individuo en el proceso de cruce y de reinserción para la próxima generación (Ilustración 3.1).

Notación utilizada en el proceso de evaluación del individuo o solución:

- *solActual*: Solución o individuo a analizar
- *N*: Total de puntos a recorrer incluyendo el depósito
- *M*: Máxima cantidad de puntos en un vehículo
- K_0: Máximo número de vehículos a utilizar
- Q_0: Capacidad máxima de los vehículos
- *D*: Vector de demandas de cada punto
- *TD*: Demanda total (suma de elementos de *D*)
- *VT*: Vector de distancias de *solActual*
- *MDist*: Matrix de distancias entre todos los puntos
- *K*: Cantidad de vehículos real utilizada por *solActual*
- *Q*: Demanda total recogida por la *solActual*
- *T*: Distancia total recorrida por *solActual*
- *R1*: Cantidad de dígitos de la suma de todas las distancias posibles iniciales para la cantidad máxima de puntos.
- *R2*: Cantidad de dígitos de la suma de K_0 más *R1*

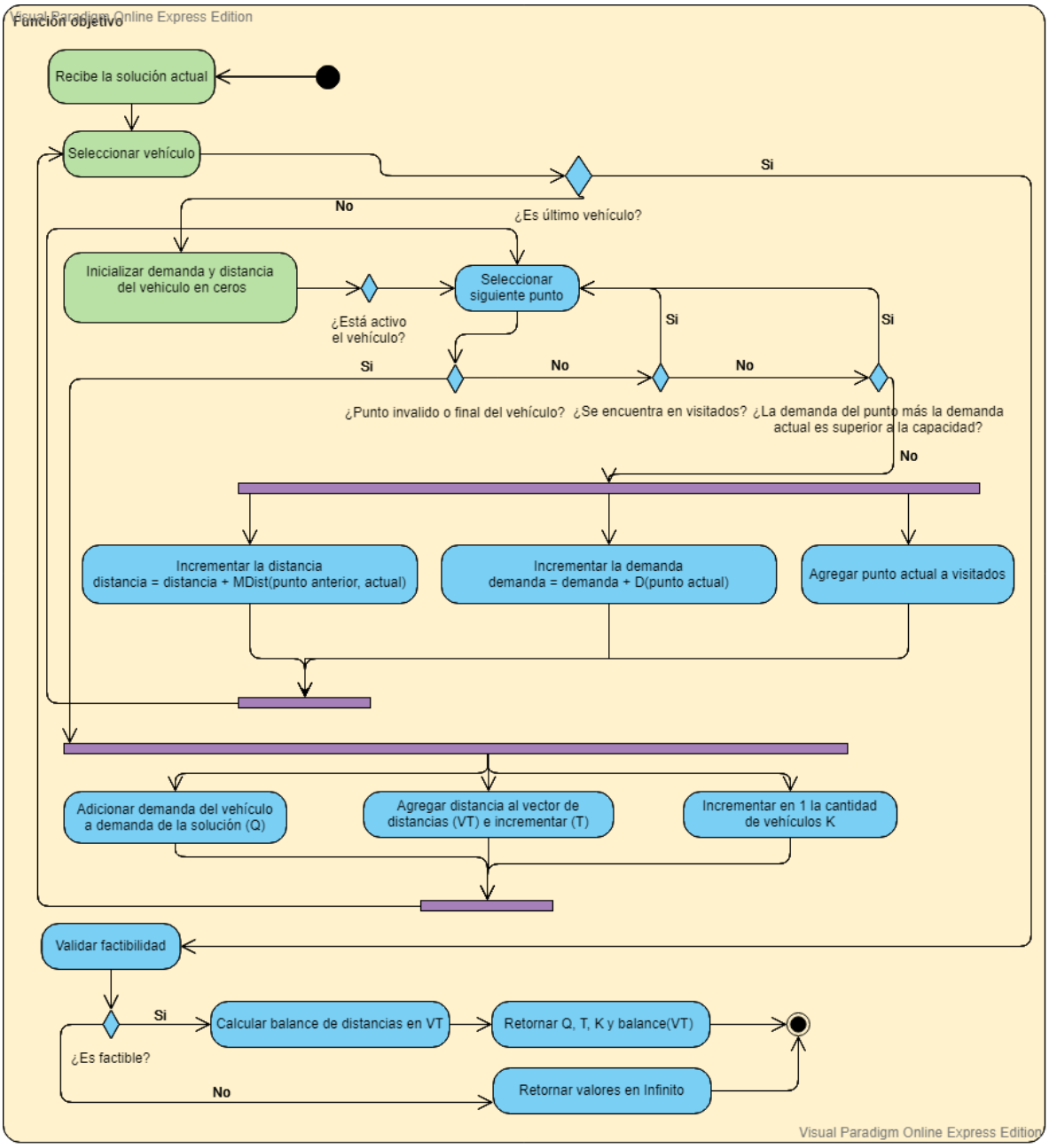

Ilustración 3.1 Diagrama de actividades de la Función Objetivo (elaboración propia).

3.2 INICIALIZACIÓN DE LOS ALGORITMOS

Debido a que los problemas reales pueden contener una gran cantidad de puntos, se hace necesario reducir, de ser posible, el tamaño de los individuos con el objetivo de disminuir el costo computacional. Para esto se decide que la cantidad máxima de puntos por

los que puede pasar un vehículo va a estar condicionada por las demandas que existen en cada punto. Por tanto, se procede a hacer un ordenamiento de las demandas de manera ascendente. Luego se van acumulando las demandas de tal forma que la suma no exceda la capacidad máxima de los vehículos. La cantidad de demandas acumuladas será la cantidad máxima de puntos en cada vehículo. Para dar un poco más de espacio de búsqueda al algoritmo este resultado se triplica, aun así, es muy inferior y no se sacrifica espacio de exploración para los algoritmos y se reduce el tamaño del individuo al mismo tiempo.

De esta manera, si el tamaño del individuo era $N^* K0\ (364 * 34) = 12376$, ahora el resultado sería $M^* K0 * 3(42 * 34 * 3) = 4284$. Al reducir esta longitud del individuo en 8092 puntos aseguramos un mejor rendimiento en cuanto a tiempo de ejecución (Ilustración 2.1).

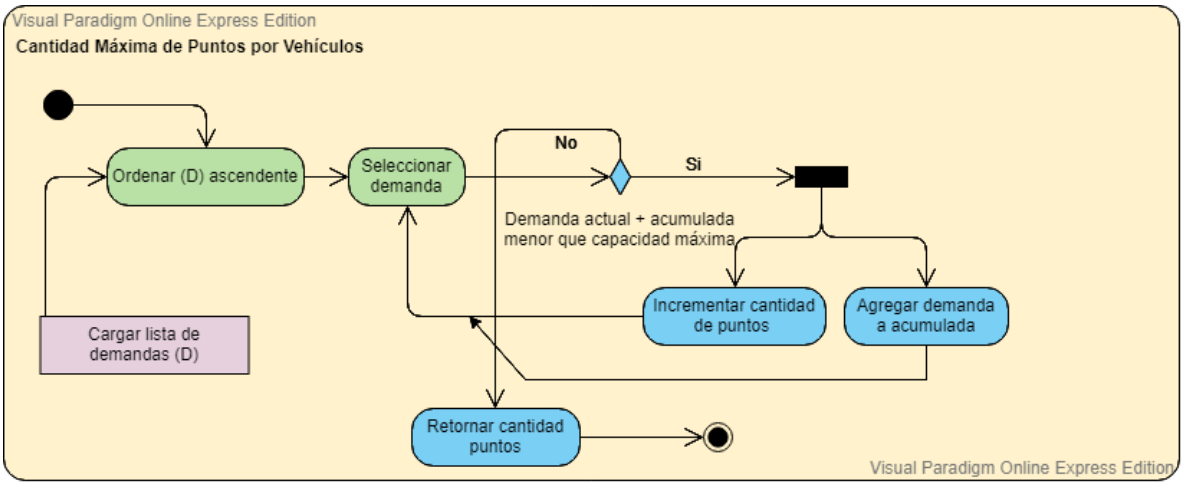

Ilustración 3.2: Diagrama de actividades para calcular cantidad máxima de puntos por vehículos (elaboración propia).

Capítulo 4. Experimentación, Resultados

Lo experimentos estarán divididos en 4 partes, primero se describirán los experimentos realizados con el algoritmo genético, seguido del algoritmo genético multiobjetivo, luego el algoritmo de optimización de partículas, y por último la variante multiobjetivo para el PSO. Todos estos serán realizados bajo las mismas condiciones en cuanto a recursos de una computadora de escritorio con un procesador Core i7 con 16 GB de memoria RAM en el software Matlab.

4.1 EXPERIMENTACIÓN CON GA

En capítulos anteriores se describen que los objetos de este trabajo son minimizar la cantidad de vehículos (min K), minimizar la distancia total (min T) y minimizar el balance entre las rutas (min B). Particularmente el GA utiliza una sola función objetivo por lo que para su caso debemos utilizar la Ecuación 3.1. Sus parámetros se describen en la Tabla 4.1.

Tabla 4.1. Parámetros usados en GA.

Parámetros	Valor o Función
Individuos	300
Genes	4284 (42*34*3)
Límite de generaciones estancadas	3000
Límite de Generaciones	6000
Cruce	Selección por torneo ***selectiontournament***
Mutación	Cruce de dos puntos ***crossovertwopoint*** (0.7)
Función de Balance	Desviación estándar (***stdDevDistance***)

Estos parámetros son comúnmente usados para los algoritmos genéticos y son de los que están por defecto en Matlab. A continuación, en la Tabla 4.2 se muestran los resultados de GA.

Tabla 4.2. Resultados de GA.

Prueba	Tiempo (horas)	Cantidad de Vehículos	Distancias	Balance

1	4,10	18	1621400,00	3140,4
2	2.35	18	1487100,00	3029,8
3	4.23	17	1556000,00	4068
4	5,75	18	1545000,00	5035,1
5	3.25	18	1424300,00	6491,2
6	3,15	18	1665500,00	4839,6
7	3,45	18	1621400,00	3140,4

4.2 EXPERIMENTACIÓN CON MOGA

El algoritmo genético multiobjetivo brinda la posibilidad de separar cada uno de los objetivos. El resultado que comúnmente arrojan estos algoritmos está compuesto por varias soluciones que forman un frente de Pareto. El MOGA utiliza el enfoque del NSGA-II para asegurar la diversidad en las poblaciones y reducir la posibilidad de estancarse en óptimos locales. Sus parámetros se describen en la Tabla 4.3.

Tabla 4.3. Parámetros usados en MOGA.

Parámetros	Valor o Función
Individuos	300
Genes	4284 (42*34*3)
Límite de generaciones estancadas	3000
Límite de generaciones	6000
Selección	Selección por torneo ***selectiontournament***
Cruce	Cruce de dos puntos crossovertwopoint (0.7)
Mutación	Mutación uniforme ***mutationuniform*** (0.3)
Función de Balance	Desviación estándar (***stdDevDistance***)

Estos parámetros son comúnmente usados para los algoritmos genéticos multiobjetivo y son de los que están por defecto en Matlab para estos problemas multiobjetivo, En este caso en particular solo se cambia la función de mutación por defecto de ***mutationadapfeasible*** para que las condiciones fueran similares con el GA mencionado

en los resultados previos. La Tabla 4.4 muestra los resultados de los experimentos con MOGA.

Tabla 4.4. Resultados de MOGA.

Prueba	Tiempo (horas)	Cantidad de Vehículos	Distancias	Balance
1	3,15	20	2359100,00	60623,00
2	3,25	21	2323700,00	64132,00
3	4,10	22	2183400,00	58467,00
4	5,79	19	2187000,00	55897,00
5	5,10	20	2144200,00	55347,00
6	4,29	21	2130800,00	59739,00
7	3,50	22	2129300,00	62429,00

4.3 EXPERIMENTACIÓN CON PSO

El algoritmo de optimización de enjambre de partículas utiliza, al igual que el GA, la Ecuación 3.1 debido a que también se basa en un solo objetivo. Su configuración es muy sencilla ya que no necesita de muchos parámetros. Sus parámetros se describen en la Tabla 4.5.

Tabla 4.5. Parámetros usados en PSO.

Parámetros	PSO
Tamaño del enjambre	300
Tamaño de las partículas	4284 (42*34*3)
Iteraciones	10000
Tolerancia	10e-10
Función de Balance	Desviación estándar (***stdDevDistance***)

Estos parámetros son comúnmente usados para los algoritmos de enjambre de partículas. La Tabla 4.6 muestra los resultados de los experimentos con PSO.

Tabla 4.6. Resultados de PSO.

Prueba	Tiempo (horas)	Cantidad de Vehículos	Distancias	Balance
1	0.0076371	17	3301000,00	38167,00
2	0.0059378	17	3264700,00	39291,00
3	0.0097346	17	3189500,00	39245,00
4	0.0071542	17	3073400,00	37281,00

5	0.011774	17	3204300,00	35970,00
6	0.0095267	17	3302500,00	36431,00
7	0.01458	17	3404600,00	35127,00

4.4 EXPERIMENTACIÓN CON MOPSO

El algoritmo multiobjetivo de optimización de enjambre de partículas está basado en el trabajo de Carlos Coello [23]. Su configuración resulta también muy sencilla, pero incorpora otros parámetros que no posee el PSO. Sus parámetros se describen en la Tabla 4.7.

Tabla 4.7. Parámetros usados en MOPSO.

Parámetros	MOPSO
Tamaño del enjambre	300
Tamaño de las partículas	4284 (42*34*3)
Iteraciones	10000
Tolerancia	10e-10
Porcentaje de mutación uniforme	0.5
Función de Balance	Desviación estándar (***stdDevDistance***)

Estos parámetros son comúnmente usados para los algoritmos de enjambre de partículas. La Tabla 4.8 muestra los resultados de los experimentos con MOPSO.

Tabla 4.8. Resultados de MOPSO.

Prueba	Tiempo (horas)	Cantidad de Vehículos	Distancias	Balance
1	0,40578	17	3251900,00	35209,00
2	0,40578	17	3187800,00	35532,00
3	0,40578	17	3094900,00	36809,00
4	0,40578	17	3042000,00	38735,00
5	0,40578	17	2949700,00	42821,00
6	0,40578	18	2899800,00	49436,00
7	0,40578	17	3251900,00	35209,00

4.5 EXPERIMENTACIÓN CON ACO

El algoritmo optimización de colonia de hormigas implementado no requiere de muchos parámetros y resulta un algoritmo muy sencillo de implementar. Sus parámetros se describen en la Tabla 4.9.

Tabla 4.9. Parámetros usados en ACO.

Parámetros	ACO
Cantidad de Hormigas	27
Tamaño de las soluciones	4284 (42*34*3)
Iteraciones	6000
Alpha	3
Beta	4
Porcentaje de Evaporación	0.1
Función de Balance	Desviación estándar (***stdDevDistance***)

Los resultados obtenidos por ACO se muestran a continuación en la Tabla 4.10

Tabla 4.10. Resultados de ACO.

Prueba	Tiempo (horas)	Cantidad de Vehículos	Distancias	Balance
1	1.10	17	677090,00	10182,00
2	1.30	17	695640,00	11793,00
3	1.15	17	620670,00	11876,00
4	1.23	17	709560,00	11782,00
5	1.42	17	719230,00	10925,00
6	1.41	17	707500,00	11868,00
7	1.36	17	646560,00	12324,00

4.5 RESUMEN DE RESULTADOS DE ALGORITMOS CON LAS CONFIGURACIONES BÁSICAS

En las tablas anteriores se muestran como los algoritmos con pocas configuraciones pueden encontrar resultados factibles. Lamentablemente esta característica no resulta suficiente para este trabajo ya que se desea buscar además de factibilidad, resultados más eficientes.

Tabla 4.11. Comparación de promedios de resultados.

Algoritmo	Tiempo (horas)	Cantidad de Vehículos	Distancias	Balance
GA	3,3	18	1643450	3990

MOGA	4,168571	20,71429	2208214	59519,14
PSO	0,00947777	17	3248571	37358,86
MOPSO	0,40578	17,14286	3096857	39107,29
ACO	**1.28**	**17**	**682321,4**	**11535,71**

Como se muestra en la Tabla 4.11, el algoritmo ACO obtiene los mejores resultados en general respecto a los objetivos principales de este trabajo que son la cantidad de vehículos, la distancia total y el balance entre rutas. Durante las ejecuciones de los experimentos se pudo comprobar que las soluciones eran encontradas generalmente en las primeras 200 iteraciones. Los algoritmos genéticos obtuvieron un mejor desempeño que los de optimización de enjambre de partículas (PSO), pero sobrepasaron doblemente a los resultados del ACO. En este punto surgió la propuesta de implementar una función de cruce para los algoritmos genéticos que respete hasta donde sea posible la estructura de los cromosomas con la idea de explotar esta característica de este tipo de algoritmos.

4.6 PROPUESTA DE FUNCIÓN DE CRUCE PARA GA Y MOGA

Por la particularidad de este trabajo, se propone desarrollar otra función de cruce para los algoritmos genéticos. Como se explicó anteriormente en el diseño de los individuos, se necesita estar seguro en cada parte de la ejecución donde termina y empieza otro vehículo, por lo que, a la hora de realizar la combinación entre dos padres debemos considerar algunos casos:

- Los padres pueden ser muy similares o incluso iguales.
- Ordenar de manera elitista los puntos en cada vehículo de los padres.
- Obtener descendientes a partir de la selección de puntos entre rutas aleatorias de los padres.
- Activar o desactivar los vehículos de cada ruta en función de valor de los padres con una probabilidad de 50% para cada progenitor.

El objetivo principal de esta función de cruce es respetar las posiciones de cada vehículo, garantizar el intercambio solo entre rutas y no de forma aleatoria entre los genes del cromosoma. La posibilidad de insertar un procedimiento de selección según cierta actitud

(ejemplo: distancia respecto al anteriormente seleccionado) entre los valores de las rutas de los progenitores, respetando la capacidad de los vehículos.

El proceso es bastante sencillo, primeramente, se ajustan ambos padres lo que consiste en eliminar aquellos valores que son incorrectos (puntos repetidos y valores con puntos flotantes probablemente introducidos por el proceso de mutación) y dejando solamente los valores que realmente forman la solución. En el proceso de limpieza o corrección se introduce un procedimiento que intenta encontrar de forma elitista (probabilidad de 20%) un mejor orden para los puntos del vehículo que se está analizando; esto se realiza cambiando las posiciones de los puntos de recogida verificando si disminuye la distancia total de su ruta.

Como es bien conocido en la vida real, siempre existe una probabilidad de que un descendiente adquiera los mismos atributos de solo uno de sus progenitores, por esta razón, además de que posiblemente el proceso de limpieza mejorara la aptitud de los padres, se decidió que el hijo pueda heredar todo de sólo uno de sus padres con una probabilidad de un 10%. En caso de que no ocurriera lo anteriormente mencionado, el siguiente paso es seleccionar de forma aleatoria, preferentemente una permutación de las posiciones de los vehículos, vehículos de cada progenitor con los que se crearía uno nuevo para la solución descendiente.

Una vez seleccionados los vehículos (o rutas), se procede a hacer un análisis de uno en uno de los valores de ambos vehículos de tal forma que se seleccione el de mejor aptitud respecto al anterior seleccionado (en función de la distancia en este caso). Se termina esta parte del proceso cuando ya no se pueda asignar ningún otro punto al nuevo vehículo por exceder de su capacidad o llegar al final de los valores de los vehículos seleccionados.

En la Ilustración 4.1 podemos visualizar el proceso realizado por la propuesta de función de cruce.

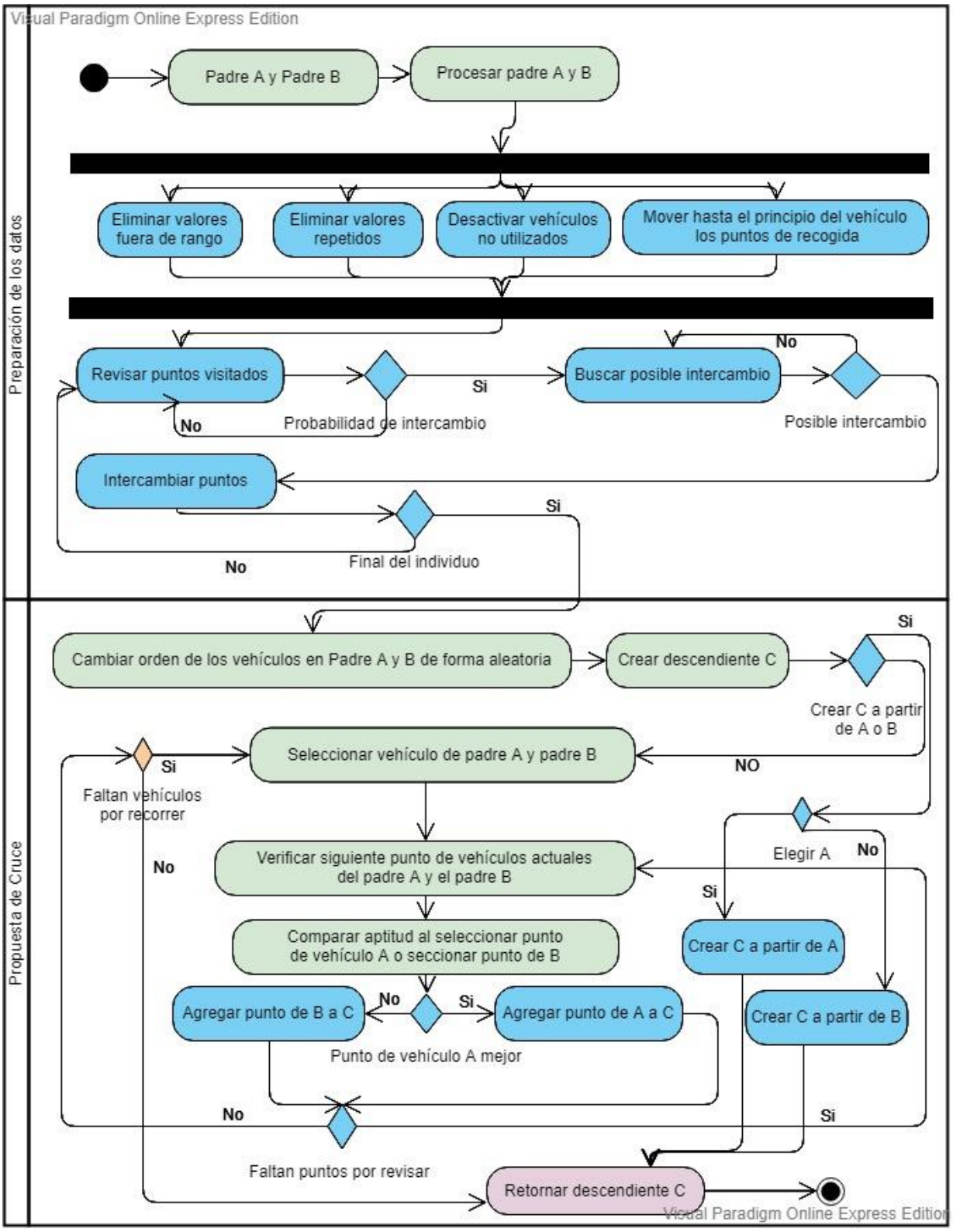

Ilustración 4.1 Diagrama de actividades realizadas para la función de cruce propuesta (elaboración propia).

4.6.1 Resultados de GA con la propuesta de función de cruce

Luego de realizar las modificaciones pertinentes al GA, se realizaron 30 pruebas con la nueva función de cruce y los resultados se muestran en la Tabla 4.12 a continuación:

Tabla 4.12. Resultados de GA con propuesta de función de cruce.

Prueba	Tiempo (horas)	Cantidad de Vehículos	Distancias	Balance
1	8.13	19	688550.00	7821.9
2	8.00	22	784420.00	7815.9
3	8.0919	19	712300.00	9025.3
4	7.9803	18	663610.00	7245.8
5	7.8355	22	708310.00	8254.7
6	7.8835	19	826710.00	8913.7
7	7.969	19	662240.00	6885.4

4.6.2 Resultados de MOGA con la propuesta de función de cruce

Luego de realizar las modificaciones pertinentes en la configuración de MOGA, se ejecutaron 30 pruebas con la propuesta de función de cruce y los resultados se muestran en la Tabla 4.13.

Tabla 4.13. Resultados de MOGA con propuesta de función de cruce.

Prueba	Tiempo (horas)	Cantidad de Vehículos	Distancias	Balance
1	8.38	17	475570.00	12185
2	8.34	18	434270.00	9153.9
3	8.41	17	481290.00	12086
4	8.51	17	483290.00	11438
5	8.39	17	488940.00	10798
6	8.40	18	501560.00	7821.3
7	8.47	22	539940.00	5102.6

4.7 COMPARACIONES ESTADÍSTICAS ENTRE LOS RESULTADOS DE GA Y MOGA CON Y SIN LA PROPUESTA DE CRUCE

Como se pudo observar en los resultados que aparecen en las tablas anteriores de las pruebas con los GA y los MOGA, a simple vista se podría decir que ambos algoritmos

mejoraron cuando se utilizó la propuesta de cruce. No obstante, se realizaron varias pruebas estadísticas para corroborar si las mejoras son estadísticamente significativas.

4.7.1 Comparaciones con la prueba de Friedman

En las estadísticas no paramétricas, el procedimiento más conocido para probar las diferencias entre más de dos muestras relacionadas es la prueba de Friedman. Esta prueba es un análogo no paramétrico del análisis paramétrico de varianza bidireccional. El objetivo de esta prueba es determinar si podemos concluir a partir de una muestra de resultados que existe una diferencia entre los efectos del tratamiento [28]. En este caso se aplicó para comparar los resultados los algoritmos en función de las distancias encontradas por cada uno de ellos en 20 pruebas.

En la Tabla 4.14 se muestran los valores de las distancias encontradas por los algoritmos GA y GA con la propuesta de función de cruce (a partir de ahora GA+).

Tabla 4.14. Resultados de las distancias de GA y GA+.

Algorithms	GA	GA +
Test1	3144800.00	688550.00
Test2	2806100.00	784420.00
Test3	2997000.00	712300.00
Test4	2932600.00	764340.00
Test5	3095600.00	708310.00
Test6	3042400.00	802810.00
Test7	2984600.00	859270.00
Test8	3128300.00	843790.00
Test9	3208200.00	846690.00
Test10	3067300.00	648500.00
Test11	3097800.00	903100.00
Test12	2913200.00	802240.00
Test13	2961500.00	764450.00
Test14	2942700.00	904130.00
Test15	3082300.00	922370.00
Test16	3030600.00	673000.00
Test17	2936400.00	663610.00

Test18	2994900.00	823830.00
Test19	3017400.00	819900.00
Test20	2869500.00	829140.00

Tabla 4.15. Resultados de los rangos de los valores de distancias de GA y GA+.

Algorithms	GA	GA +
Test1	2	1
Test2	2	1
Test3	2	1
Test4	2	1
Test5	2	1
Test6	2	1
Test7	2	1
Test8	2	1
Test9	2	1
Test10	2	1
Test11	2	1
Test12	2	1
Test13	2	1
Test14	2	1
Test15	2	1
Test16	2	1
Test17	2	1
Test18	2	1
Test19	2	1
Test20	2	1
Suma de rangos (R)	40	20

Tabla 4.16. Parámetros utilizados para la comparación GA y GA+ en la prueba Friedman.

Nivel de significancia (α)	Número de muestras (k)	Grados de libertad (v= k-1)	Cantidad de muestras (n)
0.05	2	1	20

El objetivo de esta prueba fue comprobar si existe diferencia entre los resultados de distancias entre GA y GA+, para ello se especificaron los parámetros (Tabla 4.16) y se calcularon los rangos de cada resultado (Tabla 4.15).

$$Fr = \frac{12}{nk(k+1)} \sum_{i=1}^{k} R_i^2 - 3n(k+1) \qquad (4.1)$$

Hipótesis:

H₀: $\tilde{\mu}_1 = \tilde{\mu}_2$ No existe diferencia entre GA y GA+ en cuanto a las distancias encontradas por cada algoritmo.

H₁: $\tilde{\mu}_1 \neq \tilde{\mu}_2$ Existe diferencia entre GA y GA+ en cuanto a las distancias encontradas por cada algoritmo.

Región crítica: $X^2 \geq 3.841$ (de la tabla A.5) [29].

Decisión: Como $Fr = 20$ (tiene una distribución Chi cuadrada) y $Fr \geq 3.841$, podemos decir que existe evidencia suficiente para rechazar H_0. Por lo tanto, se acepta la hipótesis alternativa y se puede afirmar que existe diferencia para GA y GA+ en cuanto a las distancias encontradas por ambos algoritmos.

Lo siguiente fue realizar el mismo análisis, pero en este caso entre resultados de MOGA y MOGA+ (MOGA con propuesta de cruce).

En la Tabla 4.17 se muestran los valores de las distancias encontradas por los algoritmos MOGA y MOGA+.

Tabla 4.17. Resultados de las distancias de MOGA y MOGA+.

Algorithms	MOGA	MOGA+
Test1	2015300.00	475570.00
Test2	2084900.00	525130.00
Test3	2327600.00	540030.00
Test4	2727900.00	425850.00
Test5	2459800.00	431010.00
Test6	2590900.00	442200.00
Test7	2475500.00	442000.00
Test8	2140800.00	429690.00
Test9	2143700.00	431640.00
Test10	2353200.00	474680.00
Test11	2570000.00	476910.00
Test12	2459900.00	449790.00

Test13	2540400.00	367290.00
Test14	2025500.00	379230.00
Test15	1942800.00	395720.00
Test16	2442900.00	407030.00
Test17	2000800.00	562050.00
Test18	2334000.00	370470.00
Test19	2280100.00	497760.00
Test20	2153500.00	475300.00

Tabla 4.18. Resultados de los rangos de los valores de distancias de MOGA y MOGA+.

Algorithms	MOGA	MOGA +
Test1	2	1
Test2	2	1
Test3	2	1
Test4	2	1
Test5	2	1
Test6	2	1
Test7	2	1
Test8	2	1
Test9	2	1
Test10	2	1
Test11	2	1
Test12	2	1
Test13	2	1
Test14	2	1
Test15	2	1
Test16	2	1
Test17	2	1
Test18	2	1
Test19	2	1
Test20	2	1
Suma de rangos (R)	40	20

El objetivo de esta prueba es el mismo de la anterior, comprobar si existe diferencia entre los resultados de distancias entre MOGA y MOGA+, para ello se especificaron los mismos parámetros de la Tabla 4.16 y se calcularon los rangos de cada resultado (Tabla 4.18).

Hipótesis:

H₀: $\tilde{\mu}_1 = \tilde{\mu}_2$ No existe diferencia entre MOGA y MOGA+ en cuanto a las distancias encontradas por cada algoritmo.

H₁: $\tilde{\mu}_1 \neq \tilde{\mu}_2$ Existe diferencia entre MOGA y MOGA+ en cuanto a las distancias encontradas por cada algoritmo.

Región crítica: $X^2 \geq 3.841$ (de la tabla A.5) [29].

Decisión: Como $Fr = 20$ (tiene una distribución Chi cuadrada) y $Fr \geq 3.841$, podemos decir que existe evidencia suficiente para rechazar ***H₀***. Por lo tanto, se acepta la hipótesis alternativa y se puede afirmar que existe diferencia para MOGA y MOGA+ en cuanto a los resultados distancias.

4.7.2 Comparaciones con la Suma de Rangos de Wilcoxon

El procedimiento no paramétrico por lo general es una alternativa adecuada para la prueba de la teoría normal cuando la suposición de normalidad no es válida. Cuando interesa probar la igualdad de las medias de dos distribuciones continuas que evidentemente no son normales, y las muestras son independientes, la prueba de la suma de rangos de Wilcoxon o la prueba de dos muestras de Wilcoxon es una alternativa apropiada a la prueba ***t*** de dos muestras [29]. En este caso se aplicó para comparar los resultados los algoritmos en función de las distancias encontradas por cada uno de ellos en 20 pruebas, similar a las pruebas de Friedman anteriores lo que en este caso la idea es demostrar no solo que no son iguales, sino, que una es menor que otra estadísticamente.

Tabla 4.19. Resultados de los rangos ordenados de los valores de distancias de GA y GA+.

Datos	Rangos	Datos	Rangos
648500	1	2806100.00	21
663610	2	2869500.00	22
673000	3	2913200.00	23
688550	4	2932600.00	24
708310	5	2936400.00	25

712300	6	2942700.00	26
764340	7	2961500.00	27
764450	8	2984600.00	28
784420	9	2994900.00	29
802240	10	2997000.00	30
802810	11	3017400.00	31
819900	12	3030600.00	32
823830	13	3042400.00	33
829140	14	3067300.00	34
843790	15	3082300.00	35
846690	16	3095600.00	36
859270	17	3097800.00	37
903100	18	3128300.00	38
904130	19	3144800.00	39
922370	20	3208200.00	40

En la Tabla 4.19 se muestran los valores ordenados por rangos de extraídos de la Tabla 4.14 y con los mismos parámetros de la Tabla 4.16. Los valores en negro pertenecen a los resultados de GA+ y los de letras rojas a los de GA.

Hipótesis:

H0: $\tilde{\mu}_1 = \tilde{\mu}_2$ No existe diferencia entre GA y GA+ en cuanto a las distancias encontradas por cada algoritmo.

H1: $\tilde{\mu}_1 > \tilde{\mu}_2$ Las distancias encontradas por GA+ son menores que las que obtiene GA.

Región crítica: $U \leq 138$ (de la tabla A.17) [29].

Decisión: Como $U2 = 0$, podemos decir que existe evidencia suficiente para rechazar ***H0***. Por lo tanto, se acepta la hipótesis alternativa y se puede afirmar que las distancias encontradas por GA+ son menores que las encontradas por GA.

El mismo procedimiento se realizó para las comparaciones entre las distancias encontradas por MOGA y MOGA+. Los valores de las distancias los podemos encontrar en

la Tabla 4.17. A continuación se muestra la Tabla 4.20 donde se reflejan los resultados de los rangos ordenados de MOGA (letras rojas) y MOGA+ (letras negras).

Tabla 4.20. Resultados de los rangos ordenados de los valores de distancias de MOGA y MOGA+.

Datos	Rangos	Datos	Rangos
367290.00	1	1942800.00	21
370470.00	2	2000800.00	22
379230.00	3	2015300.00	23
395720.00	4	2025500.00	24
407030.00	5	2084900.00	25
425850.00	6	2140800.00	26
429690.00	7	2143700.00	27
431010.00	8	2153500.00	28
431640.00	9	2280100.00	29
442000.00	10	2327600.00	30
442200.00	11	2334000.00	31
449790.00	12	2353200.00	32
474680.00	13	2442900.00	33
475300.00	14	2459800.00	34
475570.00	15	2459900.00	35
476910.00	16	2475500.00	36
497760.00	17	2540400.00	37
525130.00	18	2570000.00	38
540030.00	19	2590900.00	39
562050.00	20	2727900.00	40

Hipótesis:

H₀: $\tilde{\mu}_1 = \tilde{\mu}_2$ No existe diferencia entre MOGA y MOGA+ en cuanto a las distancias encontradas por cada algoritmo.

H₁: $\tilde{\mu}_1 > \tilde{\mu}_2$ Las distancias encontradas por MOGA+ son menores que las que obtiene MOGA.

Región crítica: $U \leq 138$ (de la tabla A.17) [29].

Decisión: Como $U2 = 0$, se llega a la conclusión de que existe evidencia suficiente para rechazar H_0. Por lo tanto, se acepta la hipótesis alternativa y se puede afirmar que las distancias encontradas por MOGA+ son menores que las encontradas por MOGA.

4.8 RESUMEN DE RESULTADOS DE GA Y MOGA CON PROPUESTA DE CRUCE

Como se pudo observar en las pruebas estadísticas anteriores, los resultados de los algoritmos genéticos con la propuesta de función de cruce resultan más competitivos que con el tradicional cruce en dos puntos (***crossovertwopoint***). Ver Tabla 4.21.

Tabla 4.21. Resumen de las pruebas estadísticas respecto a los resultados de distancias.

Algoritmos	Prueba de Friedman	Suma de Rangos de Wilcoxon
GA vs GA+	Diferentes distancias	GA+ menores distancias
MOGA vs MOGA+	Diferentes distancias	MOGA+ menores distancias

4.9 PROPUESTA DE ALGORITMO HÍBRIDO ACO-MOGA

Al analizar todos los resultados, se llegó a la conclusión de que tal vez una mezcla entre el algoritmo ACO y MOGA (los de mejores resultados obtenidos), pueda mejorar en gran medida las soluciones. El principal objetivo es explotar la rápida convergencia en soluciones lo suficientemente buenas de los ACO más la capacidad de exploración y el manejo de múltiples objetivos de los MOGA (con propuesta de función de cruce). Para esto se propone generar la población inicial de MOGA con los mejores de las iteraciones del ACO (sin repetir soluciones) según un porcentaje de la población para el MOGA. Los resultados de esta combinación ACO-MOGA se muestran en la Tabla 4.22.

Tabla 4.22. Resultados de ACO-MOGA con propuesta de función de cruce.

Prueba	Tiempo (horas)	Cantidad de Vehículos	Distancias	Balance
1	10,924	17	290630,00	7378,2
2	10,92	18	293890,00	7257
3	10,917	17	294130,00	7943,7
4	10,924	20	295340,00	8083,8
5	10,91	17	295860,00	7773,6
6	11,8	17	296640,00	6834,1
7	10,93	18	298940,00	5964,2

Al analizar estos resultados se pudo constatar que, para el problema de ruteo de vehículos abierto con restricción de capacidad y balance entre rutas (OCVRPRB), el algoritmo ACO-MOGA obtuvo resultados sobresalientes respecto a los demás algoritmos analizados anteriormente.

A continuación, se muestran ilustraciones de gráficas comparativas de los resultados de GA y MOGA con la propuesta de cruce implementada, los resultados de ACO y las soluciones encontradas por ACO-MOGA (MOGA+ o MOGA con función de cruce). En este punto no fue necesario mostrar más detalles de PSO y MOPSO porque sus soluciones no resultaron competitivas con la de los otros algoritmos.

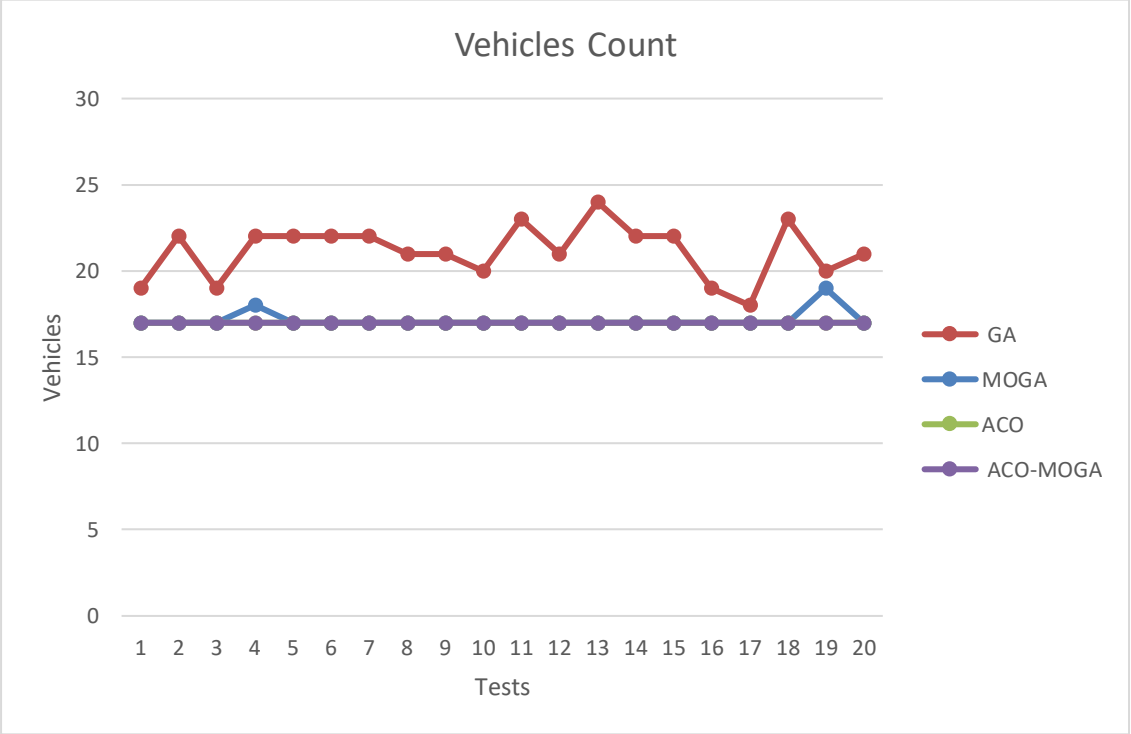

Ilustración 4.2 Cantidad de vehículos encontrados por los algoritmos GA, MOGA, ACO y ACO-MOGA

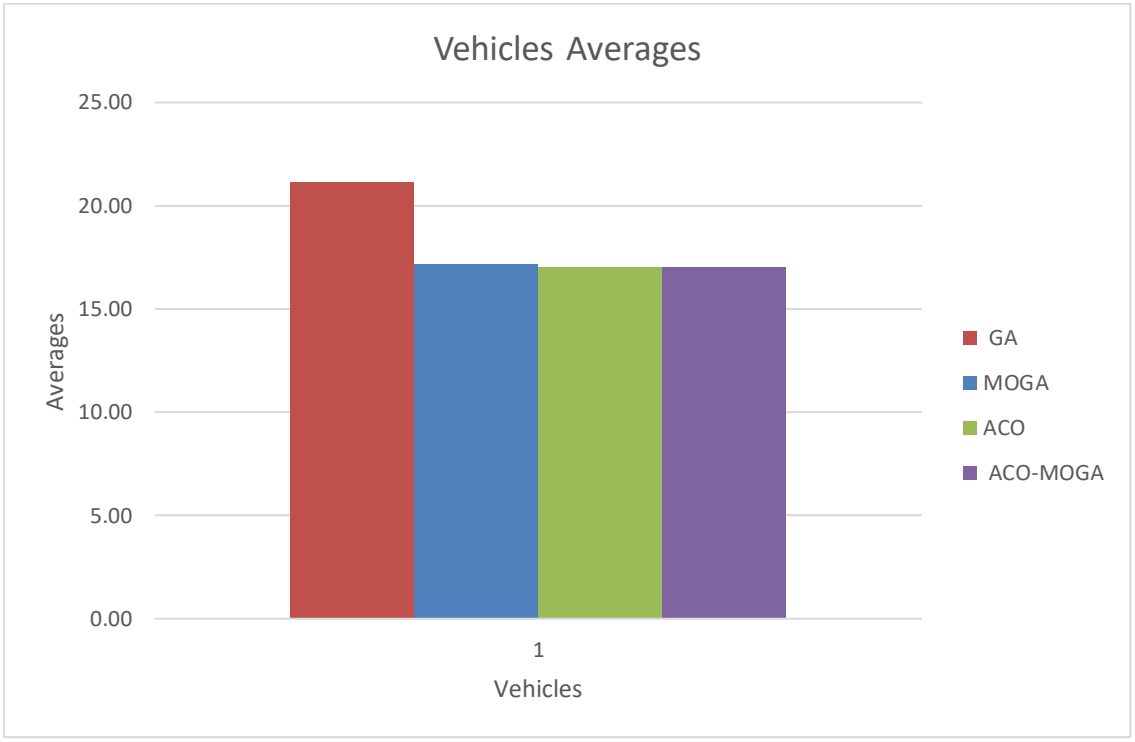

Ilustración 4.3 Promedio de vehículos encontrados por los algoritmos GA, MOGA, ACO y ACO-MOGA

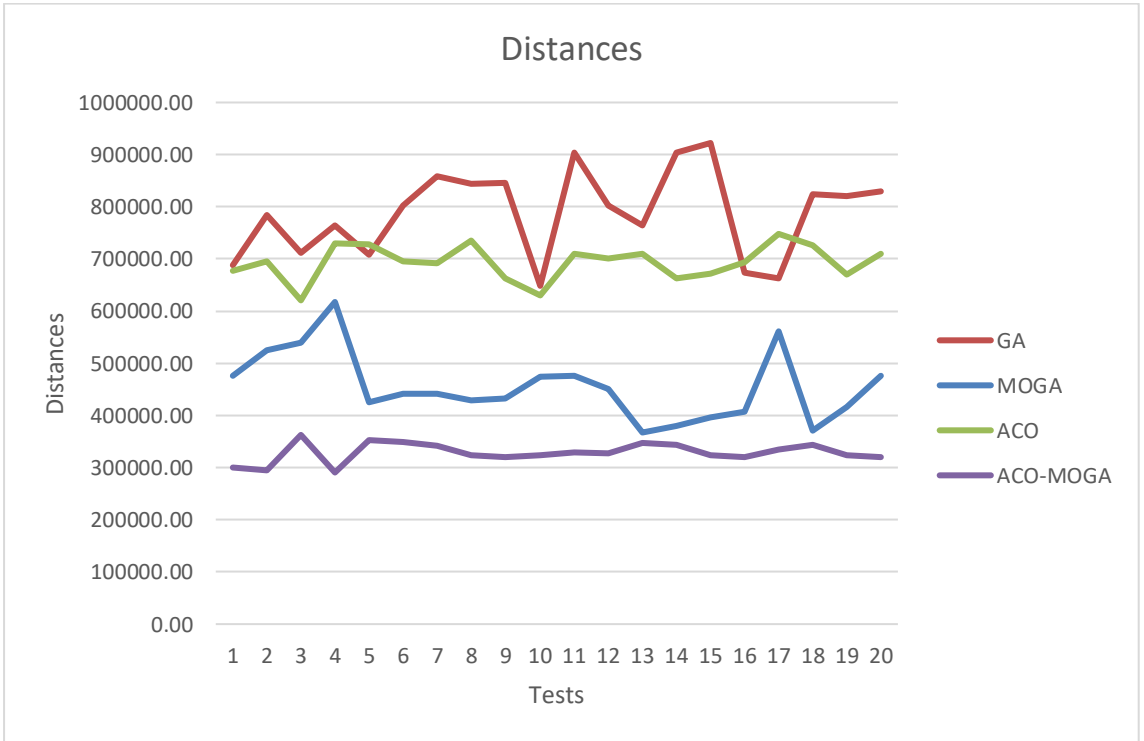

Ilustración 4.4 Distancias encontradas por los algoritmos GA, MOGA, ACO y ACO-MOGA

Ilustración 4.5 Promedio de distancias encontradas por los algoritmos GA, MOGA, ACO y ACO-MOGA

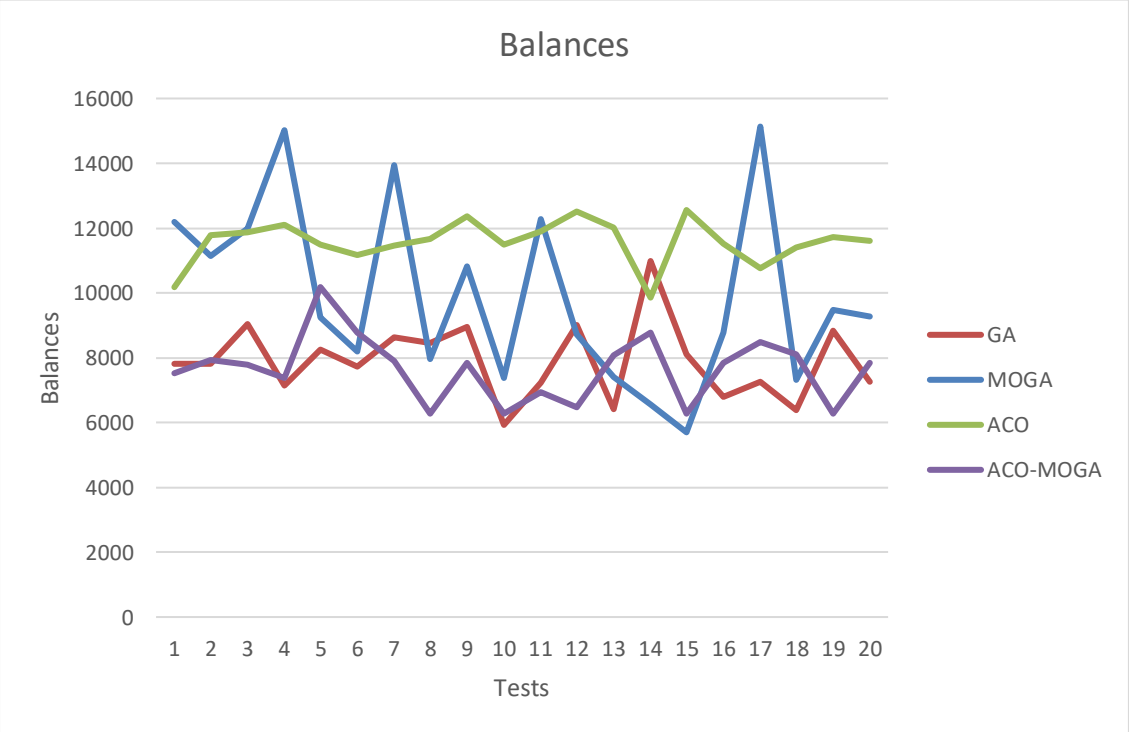

Ilustración 4.6 Balance (Standard Deviation) de distancias encontradas por los algoritmos GA, MOGA, ACO y ACO-MOGA

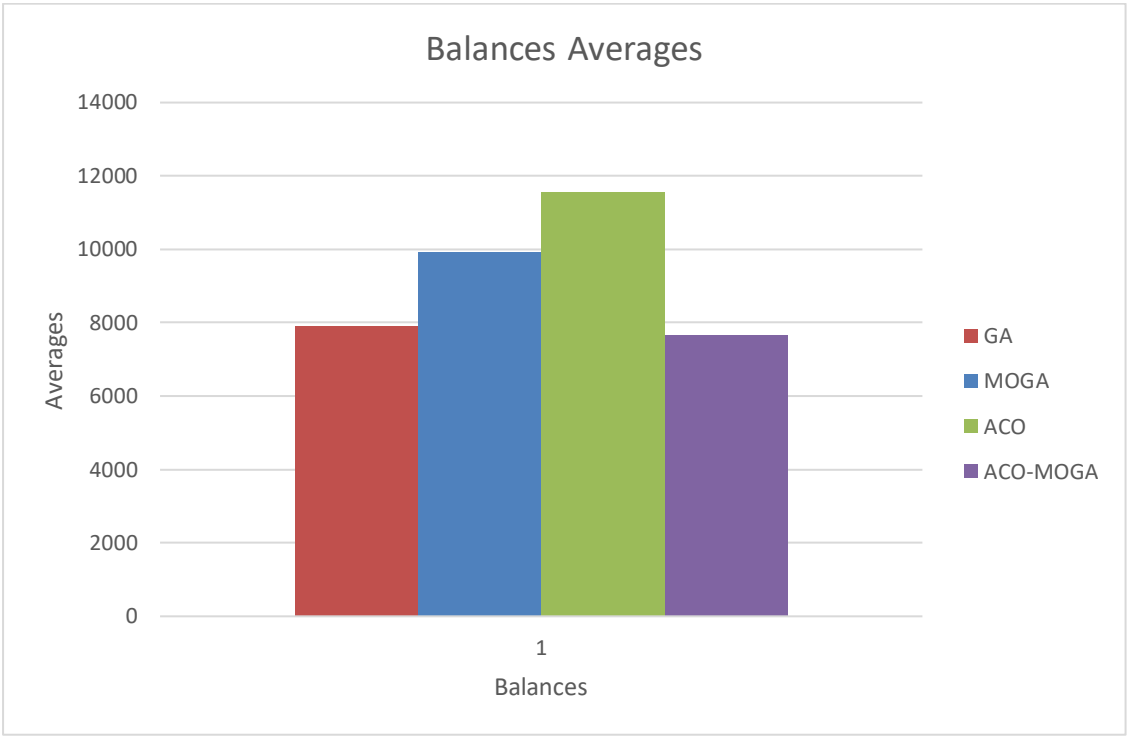

Ilustración 4.7 Promedio de balance (Standard Deviation) de distancias encontradas por los algoritmos GA, MOGA, ACO y ACO-MOGA

Al analizar la Ilustración 4.2 se observa que la cantidad de vehículos encontrados por cada algoritmo no son muy diferentes, hecho que podemos corroborar en la Ilustración 4.3 donde aparecen los promedios de los vehículos. Solo el GA obtuvo los peores resultados.

En la Ilustración 4.4 se nota como los resultados de ACO-MOGA se mantienen inferiores que los de los demás algoritmos. Al revisar los promedios de las distancias en la Ilustración 4.5, el algoritmo que más se aproxima a los resultados de ACO-MOGA es el MOGA.

En la Ilustración 4.6 se observa que los balances encontrados no son muy diferentes, pero al revisar los promedios de la Ilustración 4.7, el promedio de balance de rutas encontradas por ACO-MOGA es levemente inferior que los encontrados por GA, convirtiéndose de esta forma ACO-MOGA, en el algoritmo con mejores resultados en general.

4.10 COMPARACIONES CON OTROS TRABAJOS

En este apartado se describen las pruebas realizadas con el algoritmo ACO-MOGA, el cual fue el de mejores resultados, con las bases de datos que analizaron en el trabajo de "An Improved Simulated Annealing for the Capacitated Vehicle Routing Problem (CVRP)"[30] en julio 2018. Además, se realizó una comparación con la tesis de Eduardo Zúñiga [8] sobre la misma base de datos que se utilizó en este trabajo.

4.10.1 Comparación de Improved SA con ACO-MOGA

El algoritmo Simulated Annealing (SA) es un algoritmo meta-heurístico basado en la oportunidad para resolver problemas de optimización combinatoria adaptados de procesos de enfriamiento de metales o materiales en termodinámica [30]. En el trabajo de julio 2018 [30] proponen una mejora o modificación del SA el cual llamaron Improved SA para resolver el problema de CVRP y su objetivo principal fue evitar que el SA quede atrapado en los óptimos locales.

Para ejecutar las pruebas correspondientes para la comparación de Improved SA y ACO-MOGA se realizaron algunas modificaciones debido a que el problema y los objetivos no son los mismos. En este caso, todas las pruebas se ajustaron en función del CVRP y, por tanto, los vehículos deben regresar al depósito y no se calcula el balance entre rutas. La comparación se realizó con algunas de las bases de datos analizadas por el trabajo antes mencionado [30], estas bases de datos están disponibles en la siguiente liga http://neo.lcc.uma.es/vrp/vrp-instances/capacitated-vrp-instances/.

Tabla 4.23. Comparación de Resultados.

Improved SA			ACO-MOGA	
Base de datos	Distancias promedio	Mínima Distancia	Distancias promedio	Mínima Distancia
A-n32-k5	1111,2	1070	**821,5**	**816**
A-n80-k10	2674,8	2634	**2168,6**	**2075**
A-n69-k9	1787,6	1698	**1487,69**	**1353**
B-n31-k5	726,4	708	**689,5**	**672**
B-n50-k7	1135,6	1022	**800,4**	**777**
B-n78-k10	1921,2	1850	**1486**	**1426**

E-n51-k5	818,4	754	**627,5**	**582**
P-n101-k4	1375,2	1308	**908,67**	**872**

La tabla 4.23 muestra cómo el algoritmo ACO-MOGA obtiene mejores resultados en las pruebas con las distintas bases de datos tomadas de la literatura.

4.10.2 Comparación con datos reales de GRASP con ACO-MOGA

En el trabajo de 2015 [8] se particionó la base de datos proporcionada por JABIL de 363 puntos de recogida más el depósito en distintos tamaños, finalmente hicieron pruebas con todos los puntos. A continuación, se muestra la tabla comparativa (Tabla 4.24) sobre el mismo conjunto de datos, el cual ellos llamaron MAQ-N363.

Tabla 4.24. Comparación de Resultados GRASP y ACO-MOGA.

MAQ-N363			
GRASP		ACO-MOGA	
Rutas	Distancia total	Rutas	Distancia total
17	499780.4581	17	290630,00
19	603904.8984	18	286970,00
17	499780.4581	17	294130,00

Capítulo 5. Discusión y Aportaciones

En este capítulo se realiza un análisis de los objetivos del trabajo, se mencionan las principales aportaciones y lo que se considera apropiado para trabajos futuro.

5.1 DISCUSIÓN

Al inicio de este proyecto se plantearon varios objetivos que serían la guía para la elaboración del mismo. Luego de los resultados satisfactorios de las pruebas realizadas podemos indicar lo siguiente:

- Se identificaron un grupo de variantes del VRP como CVRP, OVRCD y VRPRB. De estas, según las restricciones nuestro el objetivo general, se decide utilizar como concepto "el problema de rutas abiertas de vehículos con restricción de capacidad y balance entre rutas" (OCVRPRB).
- En la literatura se encontraron varios métodos de solución para este tipo de problemas, en muchos casos, el enfoque utilizado implicaba metaheurísticas como es el caso de Simulated Annealing (SA), Genetics Algorithms (GA) y Ant Colonization Optimization (ACO). Además, soluciones matemáticas de optimización o factibilidad como el Mixed Integer Linear Programming (MILP) unido a una estrategia voraz como Greedy Random Adaptive Search Procedure (GRASP).
- Durante el desarrollo de este trabajo se implementaron varios algoritmos bio-inspirados como es el caso de GA, MOGA, PSO, MOPSO, ACO. De estos algoritmos resaltaron el MOGA y el ACO dando lugar a la creación del hibrido ACO-MOGA. El desempeño de algunos como el PSO y MOPSO no fueron los esperados, aunque cabe resaltar que encontraron solucione factibles en corto tiempo, pero no fueron lo suficientemente competitivas. Por otra parte, los algoritmos genéticos GA y MOGA, con las funciones tradicionales de cruce y mutación no obtuvieron los mejores resultados al menos para el diseño del

cromosoma propuesto; debido a esto surgió la propuesta de una nueva función de cruce con la cual ambos algoritmos mejoraron considerablemente sus resultados.

5.2 DISCUSIÓN DE LA HIPÓTESIS

Inicialmente se planteó que:

"El diseño de algoritmos bio-inspirados para los problemas de rutas de vehículos resultará en una solución automatizada capaz de optimizar el número de autobuses, la distancia total y el balance entre las distancias a recorrer en cada ruta".

Como se pudo observar en el capítulo anterior, los algoritmos implementados en este trabajo, constituyen por sí solos soluciones automatizadas para resolver el problema de OCVRPRB, no obstante, las soluciones que resaltaron por encima de las otras fueron las obtenidas por ACO-MOGA. Este algoritmo fue capaz de obtener resultados competitivos comparados con otros métodos de solución sobre otras bases de datos de VRP encontradas en la literatura (Tabla 4.23 y Tabla 4.24).

5.3 APORTACIONES

En este trabajo se logró implementar varios algoritmos bio-inspirados. Cada variante de estos demostró sus fortalezas y debilidades para resolver el problema de OCVRPRB.

Se desarrolló una nueva función de cruce para los algoritmos genéticos donde primeramente se ordenan, con una probabilidad dada, los vehículos de las soluciones padres; luego se seleccionan aleatoriamente vehículos entre los progenitores y se intercambian sus valores en dependencia de su aptitud.

El objetivo principal de esta función de cruce fue respetar las posiciones de cada vehículo, garantizar el intercambio solo entre rutas y no de forma aleatoria entre los genes del cromosoma. Además, la posibilidad de insertar un procedimiento de selección según cierta característica (ejemplo: distancia respecto al anteriormente seleccionado) entre los valores de las rutas de los progenitores, respetando la capacidad de los vehículos.

Se implementó un algoritmo híbrido entre colonia de hormigas ACO y el algoritmo genético multiobjetivo MOGA donde:

- Por su rápida convergencia y buenos resultados, se genera la población inicial con el ACO hasta un cierto porcentaje de la población total (ejemplo: 75%).
- Por la característica de los MOGA de no quedar atrapados fácilmente en óptimos locales y de tratarse de un problema multiobjetivo, se utiliza este para continuar la búsqueda de las soluciones.

El resultado final fue un algoritmo con mejores soluciones que las encontradas por separado por ACO y MOGA en ambos casos.

5.4 TRABAJO FUTURO

Este trabajo se realizó desde un principio con fines científicos, pero, a medida que se fueron ejecutando los experimentos con cada uno de los algoritmos, pudimos constatar la posibilidad de extrapolar las soluciones a sistemas en la nube, aplicaciones móviles, web o de escritorio, donde se puedan incluir la geolocalización utilizando los servicios de Google Mapas, además de la posibilidad de realizar los cálculos de distancias en tiempo real con dichos servicios. Por esto se plantean como trabajo futuro lo siguiente:

- Estudio de lenguajes de programación o frameworks de aplicaciones web que sean capaces de manipular en tiempo real mucha información y que puedan brindar la posibilidad de multihilos.
- Realizar las modificaciones necesarias en los algoritmos para que sean capaces de manejar volumen y peso en vez de solo cantidades enteras como capacidad de los vehículos.

Conclusiones

Durante el desarrollo de este trabajo se realizaron múltiples experimentos con varios algoritmos bio-inspirados, lo que resultó en un estudio muy completo para el problema de VRP en algunas de sus variantes como el CVRP, el VRPRB y el OCVRPRB. El desempeño de estos algoritmos en cuanto a tiempo de ejecución fue aceptable ya que obtuvieron soluciones factibles en tiempos considerables. La implementación de cada uno de ellos resultó ser muy sencilla, tal y como se menciona en la literatura. No obstante, dependiendo del problema, se pueden hacer modificaciones, sin alterar el comportamiento de los algoritmos, para mejorar sus resultados como fue el caso de la propuesta de cruce para los algoritmos genéticos. Dada las similitudes entre algunos fue posible crear una hibridación entre ACO y MOGA con el objetivo de explotar las fortalezas de cada uno de ellos en busca de mejores soluciones. El algoritmo hibrido ACO-MOGA resultó ser muy prometedor en la solución del problema de ruteo de vehículos e incluso en algunas de sus variantes dependiendo de las restricciones. El comportamiento en general de los algoritmos fue satisfactorio y deben ser tomados en consideración a la hora de resolver problemas similares de VRP.

Referencias

[1] D. de Werra, “The combinatorics of timetabling,” *Eur. J. Oper. Res.*, vol. 96, no. 3, pp. 504–513, 1997.

[2] F. Sánchez Hernández, “Aplicación del Modelo VRP (Vehicle Routing Problem) para La Optimización de una Red de Distribución,” p. 8180, 2015.

[3] G. Vaira, “Genetic Algorithm For Vehicle Routing Problem,” *Vilnius Univ. Technol. Sci. Informatics Eng.*, vol. 1, no. 1, pp. 2–94, 2014.

[4] G. Pólya and (Universidade De Princeton), *How to solve it 1*. 1977.

[5] B. M. Baker and M. A. Ayechew, “A genetic algorithm for the vehicle routing problem,” *Comput. Oper. Res.*, vol. 30, no. 5, pp. 787–800, 2003.

[6] K. Ghoseiri and S. F. Ghannadpour, “Multi-objective vehicle routing problem with time windows using goal programming and genetic algorithm,” *Appl. Soft Comput. J.*, vol. 10, no. 4, pp. 1096–1107, 2010.

[7] S. Geetha, “A Hybrid Particle Swarm Optimization with Genetic Operators for Vehicle Routing Problem,” vol. 1, no. 4, pp. 181–188, 2010.

[8] E. Zuñiga Aguirre, “MODELO MULTI-OBJETIVO DE MILP Y METAHEURÍSTICA GRASP PARA EL OVRPCD EN TRANSPORTE DE PERSONAL CON BALANCEO,” Universidad Autónoma de Chihuahua, 2015.

[9] J. G. L. Carballo, “Una Comparación Estadística de Funciones Objetivo para El Problema de Ruteo de Vehículos con Balanceo de Rutas,” 2016.

[10] Y. Liu and B. Cao, “Solving Full-Vehicle-Mode Vehicle Routing Problems Using ACO,” no. September, 2018.

[11] R. Saravanan, P. Asokan, and M. Sachidanandam, “A multi-objective genetic algorithm (GA) approach for optimization of surface grinding operations,” *Int. J. Mach. Tools Manuf.*, vol. 42, no. 12, pp. 1327–1334, 2002.

[12] C. Prins, “A simple and effective evolutionary algorithm for the vehicle routing problem,” *Comput. Oper. Res.*, vol. 31, no. 12, pp. 1985–2002, 2004.

[13] A. Mediorreal, “Laboratorios Veterland , Laboratorios Callbest y Cosméticos Marlioü París Trabajo de grado Presentado por : Andrés Felipe Mediorreal Carrillo Directora : María Paula Ramírez PONTIFICIA UNIVERSIDAD JAVERIANA DEPARTAMENTO DE INGENIERÍA INDUSTRIAL BOGOTÁ 201,” pp. 1–65, 2014.

[14] J. M. Daza, J. R. Montoya, and F. Narducci, “RESOLUCIÓN DEL PROBLEMA DE ENRUTAMIENTO DE VEHÍCULOS CON LIMITACIONES DE CAPACIDAD UTILIZANDO UN PROCEDIMIENTO METAHEURÍSTICO DE DOS FASES (SOLVING THE CAPACITATED VEHICLE ROUTING PROBLEM USING A TWOPHASE METAHEURISTIC PROCEDURE),” *Revista EIA, ISSN 1794-1237 Número 12*, Medellín (Colombia), pp. 23–38, 2009.

[15] M. A. Vásquez Morales, “Desarrollo de un framework para el problema de ruteo de

vehículos," 2007.

[16] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. Da Fonseca, "Performance assessment of multiobjective optimizers: An analysis and review," *IEEE Trans. Evol. Comput.*, vol. 7, no. 2, pp. 117–132, 2003.

[17] A. Villagra, D. Pandolfi, M. Lasso, and M. D. S. Pedro, "Algoritmos Evolutivos en la tarea de clasificación," *VIII Work. Investig. en Ciencias la Comput.*, 2006.

[18] I. R. Medina, "Revisión de los Algoritmos Bioinspirados," The University of Manchester, 2014.

[19] X. Lin, S. Member, S. Ke, Z. Li, H. Weng, and X. Han, "A Fault Diagnosis Method of Power Systems Based on Improved Objective Function and Genetic Algorithm-Tabu Search," vol. 25, no. 3, pp. 1268–1274, 2010.

[20] K. Amouzgar, "Multi-Objective Optimization using Genetic Algorithms Kaveh Amouzgar THESIS WORK 2012 Multi-Objective Optimization using Genetic Algorithms," *Thesis*, vol. 00, no. vx, 2012.

[21] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, 2002.

[22] Y. Xin-She, C. Zhihua, X. Renbin, G. Amir Hossein, and K. Mehmet, *Swarm Intelligence and Bio - Inspired Computation*. 2013.

[23] C. A. C. Coello, G. T. Pulido, and M. S. Lechuga, "Handling Multiple Objectives With Particle Swarm Optimization," *IEEE Trans. Evol. Comput.*, vol. 8, no. June 2004, 2004.

[24] M. Dorigo, "ACO Algorithms for the Traveling Salesman Problem," no. April 1999, 1999.

[25] M. Dorigo and K. Socha, "An Introduction to Ant Colony Optimization.," no. Mayo, 2006.

[26] S. Shtovba, E. Geraldo Nepomuceno, and M. Reimann, "A hybrid ACO-GA approach to solve Vehicle Routing Problems," 2001, no. January.

[27] J. G. Miranda, "Introducción a la teoría de grafos," pp. 111–135, 2005.

[28] S. García, A. Fernández, J. Luengo, and F. Herrera, *Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power*, vol. 180, no. 10. Elsevier Inc., 2010.

[29] R. E. Walpole, R. H. Myers, S. L. Myers, and K. Ye, *Probabilidad y estadística para ingeniería y ciencias*, 9th ed. 2012.

[30] M. Farhanna, M. Wayan Firdaus, and S. Purnomo Budi, "AN IMPROVED SIMULATED ANNEALING FOR THE CAPACITATED VEHICLE ROUTING PROBLEM (CVRP)," *J. Ilm. KURSOR*, vol. 9, no. 3, 2018.