UNIVERSIDAD JUÁREZ AUTÓNOMA DE TABASCO

DIVISIÓN ACADÉMICA DE CIENCIAS Y TECNOLOGÍAS DE LA INFORMACIÓN

GENERACIÓN DE RUTAS DE REPARTO A PARTIR DE METAHEURÍSTICA DE OPTIMIZACIÓN

TESIS PARA OBTENER EL GRADO DE:

MAESTRO EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA:

ROMEO VALENCIA GÓMEZ

BAJO LA DIRECCIÓN DE:

DRA. CRISTINA LÓPEZ RAMÍREZ

EN CODIRECCIÓN:

DR. RICARDO RAMOS AGUILAR

CUNDUACÁN, TABASCO, A: NOVIEMBRE 2025

UNIVERSIDAD JUÁREZ AUTÓNOMA DE TABASCO

DIVISIÓN ACADÉMICA DE CIENCIAS Y TECNOLOGÍAS DE LA INFORMACIÓN

GENERACIÓN DE RUTAS DE REPARTO A PARTIR DE METAHEURÍSTICA DE OPTIMIZACIÓN

TESIS PARA OBTENER EL GRADO DE:

MAESTRO EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA:

ROMEO VALENCIA GÓMEZ

BAJO LA DIRECCIÓN DE:

DRA. CRISTINA LÓPEZ RAMÍREZ

EN CODIRECCIÓN:

DR. RICARDO RAMOS AGUILAR

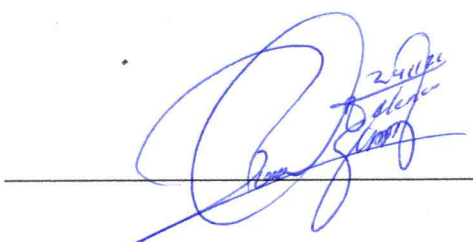
CUNDUACÁN, TABASCO, A: NOVIEMBRE 2025

Declaración de Autoría y Originalidad

En la Ciudad de Cunduacán el día veinticinco del mes de Noviembre del año 2025, el que suscribe **Romeo Valencia Gómez**, alumno del Programa de la **Maestría en Ciencias de la Computación** con número de matrícula **222H21005**, adscrito a la **División Académica de Ciencias y Tecnologías de la Información**, de la Universidad Juárez Autónoma de Tabasco, como autor de la Tesis presentada para la obtención de Grado de Maestría y titulada **Generación de Rutas de Reparto a Partir de Metaheurística de Optimización**, dirigida por la Dra. Cristina López Ramírez y el Dr. Ricardo Ramos Aguilar.

DECLARO QUE: La Tesis es una obra original que no infringe los derechos de propiedad intelectual ni los derechos de propiedad industrial u otros, de acuerdo con el ordenamiento jurídico vigente, en particular, la LEY FEDERAL DEL DERECHO DE AUTOR (Decreto por el que se reforman y adicionan diversas disposiciones de la Ley Federal del Derecho de Autor del 01 de Julio de 2020 regularizando, aclarando y armonizando las disposiciones legales vigentes sobre la materia), en particular, las disposiciones referidas al derecho de cita. Del mismo modo, asumo frente a la Universidad cualquier responsabilidad que pudiera derivarse de la autoría o falta de originalidad o contenido de la Tesis presentada de conformidad con el ordenamiento jurídico vigente.

Cunduacán, Tabasco a 25 de Noviembre de 2025.



Estudiante: Romeo Valencia Gómez

Cunduacán, Tabasco, a 22 de octubre de 2025
Oficio No. 1917/2025/DACYTI/D

Asunto: Autorización de impresión de Tesis

C. Romeo Valencia Gómez
Egresado de la Maestría en Ciencias de la Computación

En virtud de que cumple satisfactoriamente los requisitos establecidos en el Reglamento General de Estudios de Posgrado vigente en la Universidad, informo a Usted que se autoriza la impresión del trabajo recepcional ***"Generación de rutas de reparto a partir de metaheurística de optimización"***, para presentar examen y obtener el Grado de Maestro en Ciencias de la Computación.

Sin otro particular, aprovecho la ocasión para enviarle un afectuoso saludo.

Atentamente

Dr. Óscar Alberto González González
Director

UNIVERSIDAD JUÁREZ AUTÓNOMA DE TABASCO
DIVISIÓN ACADÉMICA DE CIENCIAS Y TECNOLOGÍAS DE LA INFORMACIÓN

C.c.p. Mtra. Yenny Lorena Dussán Rojas. – Encargada del despacho de la Coordinación de Posgrado.
Archivo.
Consecutivo.

DR.*OAGG/YLDR

Carretera Cunduacán-Jalpa Km. 1, Colonia Esmeralda, C.P. 86690.
Cunduacán, Tabasco, México.
Tel: (993) 358 1500 ext. 6727; (914) 336 0616; Fax: (914) 336 0870
E-mail: direccion.dacyti@ujat.mx

Carta de Cesión de Derechos

Villahermosa, Tabasco a 25 de Noviembre de 2025.

Por medio de la presente manifiesto haber colaborado como AUTOR en la producción, creación y/o realización de la obra denominada: **Generación de Rutas de Reparto a Partir de Metaheurística de Optimización.**

Con fundamento en el artículo 83 de la Ley Federal del Derecho de Autor y toda vez que, la creación y/o realización de la obra antes mencionada se realizó bajo la comisión de la Universidad Juárez Autónoma de Tabasco; entendemos y aceptamos el alcance del artículo en mención de que tenemos el derecho al reconocimiento como autores de la obra, y a la Universidad Juárez Autónoma de Tabasco mantendrá en un 100 % la titularidad de los derechos patrimoniales por un período de 20 años sobre la obra en la que colaboramos, por lo anterior, cedemos el derecho patrimonial exclusivo en favor de la Universidad.

COLABORADOR

Estudiante: Romeo Valencia Gómez

TESTIGOS

Dra. Cristina López Ramírez

Dr. Ricardo Ramos Aguilar

Agradecimientos

A Dios, por su guía, fortaleza y fidelidad en cada etapa de este trabajo, y por concederme la sabiduría y perseverancia necesarias para concluir este proyecto.

A mi esposa, por su amor, paciencia y apoyo incondicional. Gracias por acompañarme en este camino, por comprender mis ausencias y por animarme cuando el cansancio parecía vencer.

A mi madre, por su cariño, y por impulsarme siempre a seguir preparándome. Sus palabras y apoyo han sido fundamentales en mi vida.

A mis hijos, quienes también son mi motivación. Gracias por su comprensión, por inspirarme a ser mejor cada día y por recordarme la importancia de seguir creciendo para ellos.

¡Solo a Dios la Gloria!

Índice general

Índice de Figuras IV

Índice de Tablas V

Resumen VI

Abstract VII

1. Generalidades 1

1.1. Introducción . . . 1

1.2. Planteamiento del problema . . . 2

1.2.1. Definición del problema . . . 2

1.2.2. Delimitación de la investigación . . . 3

1.2.2.1. Delimitación conceptual y metodológica . . . 3

1.2.2.2. Alcance práctico de la implementación del modelo . . . 4

1.3. Pregunta de investigación e hipótesis . . . 5

1.4. Objetivo general . . . 5

1.5. Objetivos específicos . . . 5

1.6. Justificación . . . 6

1.7. Metodología utilizada . . . 7

2. Marco teórico 10

2.1. Conceptos y teorías fundamentales de la investigación . . . 10

2.2. Literatura relacionada . . . 12

2.3. Estudios similares con AG y ACO . . . 14

2.4. Limitaciones de las metaheurísticas . . . 14

2.5. Marco tecnológico 15
2.5.1. Lenguaje de programación 15
2.5.2. Entorno de desarrollo 16
2.5.3. Librerías y paquetes 16
2.5.4. Servicios de mapas y cálculo de distancias 16
2.5.5. Plataforma de ejecución 17

3. Modelado del problema y conjunto de datos 18

3.1. Modelo matemático 18
3.2. Estructura del dominio del problema 20
3.3. Descripción estadística del conjunto de datos 21
3.4. Justificación del tamaño del problema 22
3.5. Visualización de la red de distribución 22
3.6. Calidad de los datos y limitaciones 23
3.6.1. Limitaciones de los datos y de las APIs de distancia/tiempo 24
3.6.2. Efecto de cambios urbanos en el modelado 27
3.7. Estructura de la matriz de distancias 27

4. Implementación de algoritmos y diseño experimental 29

4.1. Implementación del Algoritmo Genético (AG) 29
4.1.1. Pseudocódigo del AG aplicado al TSP: 29
4.1.2. Parámetros utilizados en AG 30
4.1.3. Justificación de parámetros del AG 30
4.2. Implementación del Algoritmo de Colonia de Hormigas (ACO) 31
4.2.1. Pseudocódigo del ACO para TSP 31
4.2.2. Parámetros utilizados 32
4.2.2.1. Justificación de parámetros en ACO 32
4.3. Comparación de parámetros AG vs. ACO 33
4.4. Diseño experimental 33
4.4.1. Complejidad computacional esperada 34
4.5. Criterios de evaluación 36

5. Resultados y análisis **37**

5.1. Resultados experimentales 37

5.1.1. Convergencia por iteraciones (mediana de 25 corridas) 38

5.1.2. Convergencia por iteraciones (comparativa) 42

5.2. Análisis estadístico 44

5.3. Comparación de estabilidad 44

5.4. Visualización comparativa 45

5.5. Discusión de resultados 46

5.5.1. Validación externa y comparación con pruebas de referencia 48

5.6. Comparación general entre AG y ACO 50

6. Conclusiones y trabajos futuros **51**

6.1. Conclusión general 51

6.2. Aportes de la investigación 52

6.3. Limitaciones del estudio 52

6.4. Trabajos futuros 53

6.5. Recomendaciones prácticas para empresas locales 53

6.5.1. Antes de implementar: datos mínimos y preparación 53

6.5.2. Elección de algoritmo según contexto 54

6.5.3. Configuraciones recomendadas (guía rápida) 54

6.5.4. Integración tecnológica (bajo costo) 54

6.5.5. Métricas y tablero (KPIs) 55

6.5.6. Mantenimiento y robustez 55

6.5.7. Calcular Costo–beneficio 55

6.5.8. Checklist de adopción 55

6.5.9. Reflexión final sobre aplicabilidad y escalabilidad 56

Anexo **57**

Bibliografía **58**

Índice de figuras

1.1. Diagrama del proceso metodológico 8

3.1. Diagrama conceptual del dominio 21
3.2. Visualización geográfica de la red de distribución 23

4.1. Complejidad esperada (escala lineal) con $P{=}A{=}200$ y $G{=}I{=}3000$. 35
4.2. Crecimiento en escala log: GA (aprox. lineal en n) vs. ACO (aprox. cuadrático). . . . 35
4.3. Aumento relativo del costo al pasar de $n{=}10$ a $n{=}60$: GA $\approx 6\times$; ACO $\approx 36\times$. 36

5.1. AG, $n{=}10$. Mejora 300$\rightarrow$3000: P=20 +9.80 %, P=200 +1.90 %. 39
5.2. AG, $n{=}60$. Mejora 300$\rightarrow$3000: P=20 +17.93 %, P=200 +23.07 %. 39
5.3. ACO, $n{=}10$. Mejora 300$\rightarrow$3000: P=20 +1.27 %, P=200 +0.00 %. 40
5.4. ACO, $n{=}60$. Mejora 300$\rightarrow$3000: P=20 +3.71 %, P=200 +2.03 %. 40
5.5. Convergencia comparativa por iteraciones (n=10) 42
5.6. Convergencia comparativa por iteraciones (n=60). Mejora 300$\rightarrow$3000: AG(P=20) +17.93 %, AG(P=200) +23.07 %, ACO(P=20) +3.71 %, ACO(P=200) +2.03 %. 43
5.7. Comparación de estabilidad entre AG y ACO (Desviación estándar) 45
5.8. Comparación del rendimiento promedio entre AG y ACO 46
5.9. Evolución del costo promedio en AG y ACO para la instancia TSPLIB-gr17. 49

Índice de tablas

2.1. Comparativa de estudios con AG y ACO en escenarios urbanos/VRP 14

3.1. Resumen estadístico del conjunto de datos 22
3.2. Riesgos derivados de datos y APIs de ruteo, su efecto en los resultados y mitigaciones recomendadas. 27

4.1. Parámetros utilizados en los experimentos 30
4.2. Parámetros del AG: valores usados, criterio y respaldo bibliográfico. 31
4.3. Parámetros utilizados en el algoritmo de colonia de hormigas 32
4.4. Comparación de parámetros entre el Algoritmo Genético (AG) y la Colonia de Hormigas (ACO) 33
4.5. Configuraciones experimentales para AG y ACO 33

5.1. Resultados comparativos entre AG y ACO 38
5.2. Resultados de la prueba Mann-Whitney U entre AG y ACO 44
5.3. Resultados de AG y ACO en la instancia TSPLIB-gr17 49
5.4. Comparación cualitativa entre AG y ACO 50

6.1. Guía de configuración por tamaño y esfuerzo computacional. 54

Resumen

El presente trabajo muestra la implementación de un modelo de optimización para la generación de rutas de reparto urbanas, mediante la aplicación de técnicas metaheurísticas: el Algoritmo Genético (AG) y el Algoritmo de Colonia de Hormigas (ACO). Para ello, se abordó el Problema del Agente Viajero (TSP) como aproximación inicial al Problema de Ruteo de Vehículos (VRP), utilizando datos reales obtenidos de establecimientos comerciales en Villahermosa, Tabasco, a partir del Directorio Estadístico Nacional de Unidades Económicas (DENUE) del INEGI. Las distancias entre puntos se calcularon a través de APIs de Google Maps y MapQuest, generando matrices simétricas de entrada.

Ambos algoritmos fueron implementados en Python y evaluados bajo ocho configuraciones experimentales, variando el número de nodos, población e iteraciones. Se utilizó como métrica principal la distancia total recorrida. La evaluación estadística se realizó mediante la prueba no paramétrica de Mann–Whitney U, que permitió comparar la calidad y estabilidad de las soluciones entre algoritmos. Los resultados obtenidos mostraron que el ACO superó consistentemente al AG en calidad promedio de las rutas, desviación estándar y escalabilidad.

Como conclusión general, se valida la hipótesis de investigación: el uso de metaheurísticas, en particular el algoritmo ACO, permite mejorar la planeación operativa de una empresa de reparto, reduciendo significativamente la distancia recorrida. Se recomienda extender este modelo a escenarios más complejos como el VRP con múltiples vehículos, restricciones logísticas y condiciones dinámicas del entorno.

Palabras clave: ruteo de vehículos, TSP, algoritmos genéticos, colonia de hormigas, optimización combinatoria, metaheurísticas, logística urbana.

Abstract

This paper presents the implementation of an optimization model for generating urban delivery routes, using metaheuristic techniques: the Genetic Algorithm (GA) and the Ant Colony Algorithm (ACO). To this end, the Traveling Salesperson Problem (TSP) was addressed as an initial approximation to the Vehicle Routing Problem (VRP), using real data obtained from commercial establishments in Villahermosa, Tabasco, from the National Statistical Directory of Economic Units (DENUE) of the INEGI (National Institute of Statistics and Geography). Distances between points were calculated using Google Maps and MapQuest APIs, generating symmetric input matrices.

Both algorithms were implemented in Python and evaluated under eight experimental setups, varying the number of nodes, population, and iterations. The total distance traveled was used as the primary metric. Statistical evaluation was performed using the nonparametric Mann–Whitney U test, which allowed comparing the quality and stability of solutions between algorithms. The results showed that the ACO consistently outperformed the GA in average route quality, standard deviation, and scalability.

In conclusion, the research hypothesis is validated: the use of metaheuristics, particularly the ACO algorithm, allows for improved operational planning of a delivery company, significantly reducing the distance traveled. It is recommended that this model be extended to more complex scenarios such as VRP with multiple vehicles, logistical constraints, and dynamic environmental conditions.

Keywords: vehicle routing, TSP, genetic algorithms, ant colony, combinatorial optimization, metaheuristics, urban logistics.

Capítulo 1

Generalidades

1.1. Introducción

En los últimos años, el crecimiento acelerado del comercio electrónico ha transformado profundamente los modelos de distribución y entrega de productos (OECD, 2020). Esta transformación ha impulsado una mayor demanda de eficiencia en las operaciones logísticas, especialmente en zonas urbanas, donde factores como la optimización del uso vehicular, el cumplimiento de ventanas horarias y la satisfacción del cliente adquieren un papel estratégico (Montoya-Torres et al., 2015; UNCTAD, 2021).

En Tabasco, las actividades terciarias —que agrupan al comercio al por menor, al por mayor, así como al transporte y almacenamiento— experimentaron un crecimiento del 2.9 % durante el año 2023, destacando el aumento del 4.7 % en el comercio mayorista y del 2.2 % en el minorista, lo que refleja una tendencia favorable para el desarrollo del comercio electrónico en la región (INEGI, 2024).

Ante estos desafíos, la planeación óptima de rutas se vuelve crucial para las empresas de paquetería y logística. El Problema de Ruteo de Vehículos (VRP) ha sido ampliamente estudiado en este contexto, aunque su complejidad —al ser un problema NP-difícil— ha motivado el uso de técnicas aproximadas para encontrar soluciones eficientes cuando el número de ubicaciones es elevado (Lawler et al., 1985).

Dentro de estas técnicas, destacan las metaheurísticas, como el Algoritmo Genético (AG) (Holland, 1975) y el Algoritmo de Colonia de Hormigas (ACO) (Dorigo y Di Caro, 1999), los cuales han

demostrado su efectividad en problemas de optimización combinatoria. Su aplicación al Problema del Agente Viajero (TSP) —una versión simplificada del VRP— permite explorar su desempeño en escenarios controlados y sirve como base metodológica para problemas logísticos más complejos.

En este trabajo se plantea la implementación de un modelo computacional de optimización basado en AG y ACO para la generación de rutas de reparto urbanas en Villahermosa, Tabasco, tomando como punto de partida el TSP, con miras a una futura aplicación en instancias más completas del VRP.

1.2. Planteamiento del problema

1.2.1. Definición del problema

La aceleración del comercio electrónico ha generado una presión creciente sobre los sistemas de distribución urbana, especialmente en términos de puntualidad, cobertura y eficiencia operativa (OECD, 2020; UNCTAD, 2021). Este fenómeno es particularmente desafiante para empresas logísticas que deben enfrentar altos volúmenes de entrega bajo condiciones cambiantes y recursos limitados.

En Tabasco, esta tendencia se refleja en el crecimiento sostenido del comercio formal, donde las actividades terciarias crecieron un 2.9 % en 2023. Este aumento se debe en gran parte al dinamismo del comercio mayorista y minorista (INEGI, 2024), sectores directamente influenciados por el auge del comercio electrónico. A su vez, a nivel nacional, el comercio digital ya representa el 6.4 % del PIB, superando en crecimiento a la economía general con un alza anual del 8.5 % (INEGI, 2025).

Uno de los retos centrales consiste en diseñar rutas de reparto que minimicen el tiempo y la distancia recorrida, maximizando al mismo tiempo la eficiencia del servicio. Las consecuencias de no lograrlo incluyen costos elevados, insatisfacción del cliente y pérdida de competitividad (Toth y Vigo, 2014).

Resolver este tipo de problemas implica enfrentarse a modelos como el Problema de Ruteo de Vehículos (VRP) y el Problema del Agente Viajero (TSP), que presentan una elevada complejidad

computacional por su naturaleza NP-dura (Lawler et al., 1985). Dado que los métodos exactos son ineficientes en instancias grandes, se recurre a enfoques aproximados como las metaheurísticas.

En este contexto, surge la necesidad de implementar y validar algoritmos como AG y ACO en situaciones reales y localizadas. Pese a sus ventajas teóricas, aún existe escasez de estudios que exploren su efectividad en entornos específicos como el estado de Tabasco, donde los desafíos logísticos presentan características propias.

1.2.2. Delimitación de la investigación

1.2.2.1. Delimitación conceptual y metodológica

Este trabajo plantea la implementación de un modelo computacional de optimización para la generación de rutas de reparto eficientes en la ciudad de Villahermosa, Tabasco. Por razones metodológicas y de viabilidad computacional, se ha optado por abordar el Problema del Agente Viajero (TSP) como una aproximación válida al Problema de Ruteo de Vehículos (VRP). El TSP permite evaluar el desempeño de las metaheurísticas seleccionadas (AG y ACO) en escenarios bien definidos, antes de considerar una extensión más compleja hacia múltiples vehículos y restricciones adicionales.

La investigación tiene una orientación monoobjetivo, considerando únicamente la minimización de la distancia total recorrida, dejando fuera otras variables como capacidad de carga, ventanas de tiempo o consumo energético. Los datos utilizados provienen del Directorio Estadístico Nacional de Unidades Económicas (DENUE), específicamente de los establecimientos ubicados en la ciudad de Villahermosa. Para la obtención de las distancias entre puntos, se emplearon los servicios de Google Maps Platform y MapQuest Developer, lo que garantiza una representación fiel del entorno urbano.

Si bien este enfoque implica una simplificación del problema logístico real, permite sentar las bases teóricas, metodológicas y experimentales para una futura extensión en la implementación del modelo al VRP completo. Además, se espera que los resultados de esta tesis sirvan como antecedente para investigaciones futuras orientadas a la optimización de rutas de reparto en contextos regionales.

1.2.2.2. Alcance práctico de la implementación del modelo

En cuanto a su alcance práctico, la implementación del modelo trasciende el ámbito académico y puede ser adoptado por diferentes actores económicos en Tabasco. En particular:

- Empresas de paquetería y mensajería local, incluyendo negocios de comercio electrónico, refaccionarias, farmacias y supermercados con reparto a domicilio, que buscan optimizar sus tiempos y costos de entrega.
- Cooperativas de transporte de mercancías ligeras, como repartidores en motocicletas o motocarros, que requieren planificar rutas eficientes en zonas con condiciones de tráfico variables.
- Instituciones educativas y de investigación, interesadas en aplicar y validar modelos de optimización en contextos reales, ya sea con fines académicos o de transferencia tecnológica.

La implementación del modelo se realizó considerando el uso de herramientas computacionales de libre acceso y de amplia disponibilidad, de modo que pueda implementarse aun en organizaciones con recursos limitados. En particular, puede ejecutarse utilizando:

- Lenguajes de programación como Python, que dispone de librerías para grafos, algoritmos evolutivos y metaheurísticas.
- Sistemas gestores de bases de datos como MySQL o PostgreSQL, que permiten administrar de manera eficiente la información sobre direcciones, distancias y tiempos de viaje.
- Plataformas de desarrollo web como Django o Flask, que facilitan la creación de interfaces gráficas para que usuarios sin formación técnica puedan generar rutas de manera automatizada.
- APIs de mapas, tales como Google Maps u OpenStreetMap, que brindan la posibilidad de obtener coordenadas geográficas y distancias ajustadas a las condiciones de tránsito en tiempo real.

Este trabajo no sólo plantea una propuesta metodológica, sino que también establece una herramienta flexible y adaptable para mejorar la eficiencia logística en el contexto local, contri-

buyendo a la reducción de costos operativos y al fortalecimiento de la competitividad frente al crecimiento del comercio electrónico en la región.

1.3. Pregunta de investigación e hipótesis

Pregunta de investigación

En el contexto actual de crecimiento acelerado del comercio electrónico y de las crecientes exigencias logísticas, surge la necesidad de implementar modelos computacionales que permitan optimizar las rutas de distribución de productos, reduciendo costos y mejorando la calidad del servicio. En este marco, la presente investigación se plantea la siguiente pregunta:

¿La implementación de un modelo computacional de ruteo de vehículos, basado en metaheurísticas de optimización, permite reducir la distancia total recorrida y mejorar la planeación en una empresa de reparto?

Hipótesis de investigación

La implementación de un modelo computacional de ruteo de vehículos, basado en metaheurísticas de optimización como el Algoritmo Genético (AG) y el Algoritmo de Colonia de Hormigas (ACO), permitirá mejorar la planeación operativa de una empresa de reparto mediante la reducción de la distancia total recorrida por su flota.

1.4. Objetivo general

Implementar un modelo computacional de ruteo de vehículos, utilizando técnicas metaheurísticas de optimización —específicamente el Algoritmo Genético (AG) y el Algoritmo de Colonia de Hormigas (ACO)—, con el fin de mejorar la planeación operativa en una empresa de reparto mediante la reducción de la distancia total recorrida por su flota.

1.5. Objetivos específicos

Para alcanzar el objetivo general, se establecen los siguientes objetivos específicos:

1. Identificar las variables, restricciones y características del entorno urbano que afectan la generación de rutas de reparto, mediante revisión documental y análisis de datos del contexto local (Villahermosa, Tabasco).

2. Implementar el modelo de ruteo de vehículo de optimización basado en el Algoritmo Genético (AG) y el Algoritmo de Colonia de Hormigas (ACO), programado en lenguaje Python, para la solución del TSP como aproximación inicial al VRP.

3. Ejecutar pruebas experimentales y comparar el rendimiento de ambas metaheurísticas, midiendo la distancia total recorrida en diferentes escenarios con distinto número de nodos, y analizando estadísticamente los resultados para validar la efectividad del modelo.

4. Analizar los resultados obtenidos para determinar cuál de las técnicas proporciona mejores soluciones y establecer recomendaciones para su aplicación práctica en sistemas logísticos reales.

1.6. Justificación

En el contexto actual, las empresas de logística y distribución enfrentan una creciente presión por mejorar sus tiempos de entrega, optimizar el uso de recursos y responder con flexibilidad a la demanda de servicios. Esta situación ha sido intensificada por el auge del comercio electrónico, particularmente tras la pandemia por COVID-19, la cual generó una transformación significativa en los hábitos de consumo y en las cadenas de suministro (OECD, 2020). Ante este panorama, la planeación eficiente de rutas de reparto se ha convertido en un factor estratégico para la competitividad de las empresas, especialmente en zonas urbanas.

El Problema de Ruteo de Vehículos (VRP) y su submodelo simplificado, el Problema del Agente Viajero (TSP), son representaciones matemáticas ampliamente utilizadas para modelar la distribución de bienes. Sin embargo, su resolución exacta es computacionalmente costosa, ya que pertenecen a la clase de problemas NP-duros, cuya complejidad aumenta exponencialmente con el tamaño del problema(Laporte, 2009). Por esta razón, es indispensable recurrir a técnicas de optimización aproximadas, entre las que destacan las metaheurísticas, debido a su capacidad

para explorar eficientemente grandes espacios de búsqueda y encontrar soluciones de calidad en tiempos razonables(Talbi, 2009).

Esta investigación justifica su relevancia en tres niveles:

1. Académico: Contribuye al campo de la inteligencia artificial y la optimización combinatoria, mediante la implementación y evaluación comparativa de dos metaheurísticas consolidadas —el Algoritmo Genético (AG) y el Algoritmo de Colonia de Hormigas (ACO)—, aplicadas en un entorno real. Esto permite validar y extender conocimientos previos documentados por autores como Dorigo y Stützle (2004) y Goldberg (1989).

2. Práctico: La propuesta representa una solución funcional y replicable para el diseño de rutas de reparto eficientes. Al utilizar datos reales de la ciudad de Villahermosa, Tabasco, se garantiza su aplicabilidad en contextos logísticos locales. Esto puede beneficiar a pequeñas y medianas empresas que carecen de sistemas avanzados de optimización, brindándoles herramientas accesibles para mejorar sus operaciones.

3. Social: Optimizar las rutas de reparto no solo tiene un impacto económico en términos de ahorro de costos y combustible, sino que también contribuye a la reducción de emisiones contaminantes, descongestionamiento vial y mejora de la calidad del servicio al cliente, lo cual repercute directamente en la calidad de vida urbana (Gendreau y Potvin, 2010; Montoya-Torres et al., 2015).

Finalmente, es importante resaltar que, aunque existen múltiples estudios sobre el VRP y el TSP, la mayoría se desarrollan en contextos abstractos o internacionales. Esta tesis representa uno de los pocos esfuerzos dirigidos a resolver un problema real de optimización logística en el estado de Tabasco, y se espera que siente un precedente para futuras investigaciones en la región.

1.7. Metodología utilizada

La presente investigación se enmarca dentro de un enfoque cuantitativo y aplicado, orientado a la implementación y validación del modelo computacional para la generación de rutas de reparto

utilizando metaheurísticas de optimización. La metodología adoptada sigue un esquema estructurado que permite responder a la pregunta de investigación y alcanzar los objetivos planteados, garantizando la replicabilidad y validez de los resultados obtenidos.

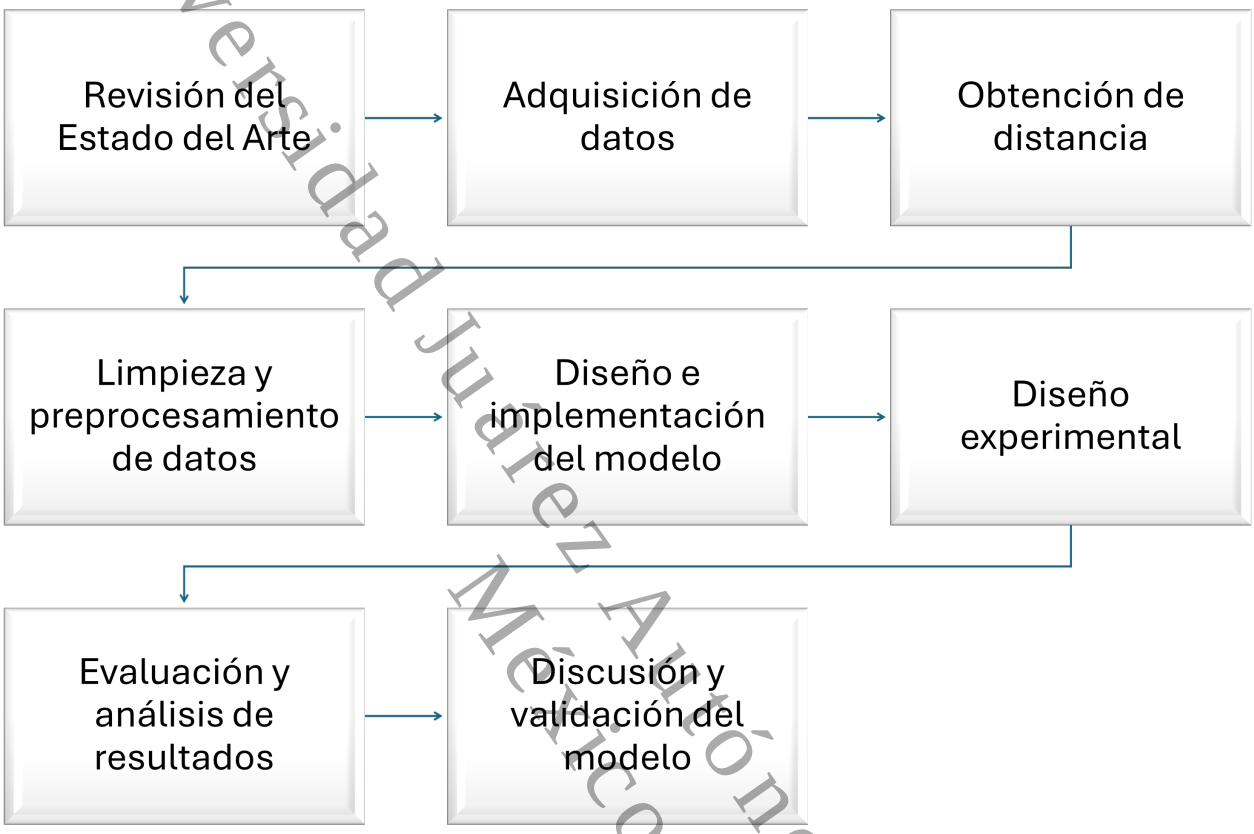


Figura 1.1. Diagrama del proceso metodológico

Etapas metodológicas:

1. Revisión del estado del arte. Se realizó una revisión sistemática de la literatura científica relacionada con el Problema del Agente Viajero (TSP), el Problema de Ruteo de Vehículos (VRP) y el uso de metaheurísticas como el Algoritmo Genético (AG) y el Algoritmo de Colonia de Hormigas (ACO). Esta etapa permitió identificar las mejores prácticas, operadores más utilizados y parámetros comunes en aplicaciones similares (Dorigo y Stützle, 2004; Talbi, 2009; Toth y Vigo, 2014).

2. Adquisición de datos. Se utilizaron datos reales provenientes del Directorio Estadístico Nacional de Unidades Económicas (DENUE) del INEGI, filtrando los registros correspondientes a la ciudad de Villahermosa, Tabasco. De esta base de datos se seleccionaron puntos de

entrega representativos, obteniendo sus coordenadas geográficas (latitud y longitud) necesarias para la generación de la matriz de distancias.

3. Obtención de distancias. Se emplearon los servicios web de Google Maps Platform y MapQuest Developer para calcular la distancia entre pares de puntos seleccionados. Estas herramientas garantizan la precisión de los datos, reflejando las condiciones reales del entorno urbano, tales como vialidades y sentidos de circulación (Developers, 2023; MapQuest, 2023)

4. Limpieza y preprocesamiento de datos. Se eliminaron registros duplicados, incompletos o con errores de geolocalización, y se estandarizó el formato de los datos para su uso en los algoritmos. Posteriormente, se construyó la matriz de distancias que sirvió como entrada para los modelos de optimización.

5. Diseño e implementación del modelo. Se codificaron dos algoritmos metaheurísticos: el Algoritmo Genético (AG) y el Algoritmo de Colonia de Hormigas (ACO), utilizando el lenguaje de programación Python. Se definieron las estructuras de datos, operadores, parámetros y condiciones de parada para cada algoritmo, con base en la literatura especializada(Developers, 2023; Dorigo y Stützle, 2004)

6. Diseño experimental. Se estructuraron ocho experimentos, combinando diferentes tamaños de población (20 y 200), número de iteraciones (300 y 3000) y cantidad de nodos (10 y 60). Cada experimento se ejecutó 25 veces de forma independiente para asegurar la validez estadística y minimizar el sesgo aleatorio.

7. Evaluación y análisis de resultados Los resultados de cada ejecución se evaluaron en términos de la distancia total recorrida, aplicando estadística descriptiva (media, desviación estándar, mínimo y máximo). Para la comparación entre algoritmos se utilizó la prueba no paramétrica Mann-Whitney U, al tratarse de datos que no siguen distribución normal.

8. Discusión y validación de la implementación del modelo Finalmente, se analizaron los resultados con base en los objetivos específicos, determinando el comportamiento de cada algoritmo y proponiendo líneas de mejora y extensión para futuras investigaciones, particularmente hacia el modelo completo del VRP.

Capítulo 2

Marco teórico

2.1. Conceptos y teorías fundamentales de la investigación

En esta sección se presentan los conceptos y fundamentos teóricos necesarios para comprender el enfoque metodológico de la presente investigación. Dado que el problema abordado corresponde a un contexto de optimización combinatoria en el área de logística, se abordan temas relacionados con la computación evolutiva, la metaheurística, el problema del agente viajero (TSP), y el problema de ruteo de vehículos (VRP).

Computación evolutiva. La computación evolutiva es una rama de la inteligencia artificial que se basa en los principios de la evolución biológica para resolver problemas complejos. Esta área engloba una familia de algoritmos estocásticos conocidos como algoritmos evolutivos, los cuales simulan procesos naturales como la selección, reproducción y mutación para explorar espacios de búsqueda (Bäck et al., 2000). En general, su objetivo es optimizar funciones mediante poblaciones de soluciones que evolucionan a lo largo de generaciones.

Los algoritmos evolutivos son una familia de metaheurísticas inspiradas en el proceso de selección natural, los cuales han demostrado ser altamente efectivos para resolver problemas de optimización complejos. En particular, los algoritmos genéticos utilizan operadores como selección, cruce y mutación para explorar el espacio de soluciones y mejorar iterativamente los resultados. Según Coello (2019), algoritmos se caracterizan por su flexibilidad, capacidad para trabajar con espacios de búsqueda complejos y robustez frente a funciones objetivo no lineales o con múltiples óptimos. Su aplicación en problemas combinatorios como el TSP es ampliamente reconocida en

la literatura.

Metaheurística. Una metaheurística es un procedimiento general de optimización que combina estrategias heurísticas de exploración y explotación para encontrar soluciones aproximadas de alta calidad a problemas complejos. Estas técnicas son especialmente útiles para resolver problemas NP-difíciles, en los que las soluciones exactas no son factibles en tiempos razonables (Blum y Roli, 2003).

A diferencia de las heurísticas tradicionales, las metaheurísticas están diseñadas para ser adaptables a diferentes dominios y no dependen estrictamente de la estructura del problema. Entre las más conocidas se encuentran los algoritmos genéticos, colonia de hormigas, recocido simulado y búsqueda tabú(Talbi, 2009).

Optimización combinatoria. La optimización combinatoria se enfoca en encontrar la mejor solución posible dentro de un conjunto finito (aunque frecuentemente muy grande) de alternativas. Estos problemas están presentes en diversas áreas como la logística, la ingeniería, las telecomunicaciones y la administración de recurso(Papadimitriou y Steiglitz, 1998). En general, los problemas de optimización combinatoria se definen por una función objetivo a minimizar o maximizar, y un conjunto de restricciones que deben cumplirse.

Problema del Agente Viajero (TSP, por sus siglas en inglés: Traveling Salesman Problem). Es uno de los problemas más estudiados dentro de la optimización combinatoria. Se plantea de la siguiente manera: dado un conjunto de ciudades y las distancias entre ellas, ¿cuál es el recorrido más corto que permite visitar cada ciudad exactamente una vez y regresar al punto de partida? (Dantzig et al., 1954).

El TSP es un problema NP-difícil, lo que significa que no se conoce un algoritmo polinomial que garantice encontrar la solución óptima en todos los casos. Por esta razón, se utilizan métodos aproximados como heurísticas y metaheurísticas para obtener soluciones eficientes en un tiempo razonable(Lawler et al., 1985). En el contexto de esta tesis, el TSP se utiliza como un modelo base para abordar el problema de ruteo de vehículos.

Problema de Ruteo de Vehículos(VRP, por sus siglas en inglés: Vehicle Routing Problem). Es una extensión del TSP que contempla una flota de vehículos que deben atender a un conjunto de clientes desde uno o más depósitos. Cada vehículo tiene un límite de capacidad y debe visitar uno o varios clientes de forma que se minimice el costo total (generalmente la distancia recorrida)

y se cumplan las restricciones logísticas(Toth y Vigo, 2014).

Existen múltiples variantes del VRP, incluyendo:

- VRP con capacidad limitada (CVRP)
- VRP con ventanas de tiempo (VRPTW)
- VRP con múltiples depósitos (MDVRP)

En la presente investigación se aborda el TSP como un caso simplificado del VRP, con el objetivo de validar el uso de metaheurísticas antes de implementar modelos más complejos con múltiples vehículos y restricciones.

2.2. Literatura relacionada

La generación de rutas de reparto eficientes es un problema de alta relevancia en la investigación operativa y la inteligencia artificial aplicada. A lo largo de los años, múltiples estudios han abordado el Problema del Agente Viajero (TSP) y el Problema de Ruteo de Vehículos (VRP), desarrollando tanto métodos exactos como técnicas heurísticas y metaheurísticas. Esta revisión presenta los enfoques más destacados en la literatura y justifica la elección metodológica de la presente investigación para instancias grandes búsqueda dispersa.

El TSP ha sido ampliamente estudiado desde mediados del siglo XX. (Dantzig et al., 1954) propusieron una formulación inicial mediante programación lineal entera. Posteriormente, (Lawler et al., 1985) recopilaron múltiples estrategias exactas para su solución, aunque todas ellas presentan limitaciones computacionales para instancias de gran tamaño. Estas restricciones dieron lugar al desarrollo de técnicas heurísticas, entre ellas el algoritmo de vecino más cercano, el algoritmo 2-opt y las búsquedas locales, que ofrecen soluciones rápidas aunque no siempre óptimas(Reinelt, 1994).

En cuanto al VRP, su complejidad aumenta al considerar múltiples vehículos, restricciones de capacidad, ventanas de tiempo y múltiples depósitos. (Toth y Vigo, 2014) clasifican los métodos de solución en tres grandes grupos: métodos exactos (como ramificación y acotamiento), heurísticas clásicas y metaheurísticas. En particular, destacan que las metaheurísticas permiten abordar instancias realistas con cientos de nodos y restricciones, siendo más versátiles y escalables.

En el contexto de las metaheurísticas, el Algoritmo Genético (AG) ha sido aplicado con éxito al TSP y al VRP, gracias a su capacidad para combinar soluciones mediante operadores de cruce y mutación. (Holland, 1975) sentó las bases teóricas del algoritmo, y (D. E. Goldberg, 1989) propuso mejoras significativas en su implementación. En la actualidad, se utilizan variantes del AG adaptadas al TSP, como el cruce ordenado (order crossover) o el cruce parcialmente mapeado (PMX, por sus siglas en Inglés), que respetan la estructura de permutación del problema (Potvin, 1996).

Por otro lado, el Algoritmo de Colonia de Hormigas (ACO) fue introducido por (Dorigo y Di Caro, 1999) como una técnica inspirada en el comportamiento de las hormigas al encontrar caminos mínimos. Su aplicación al TSP ha demostrado ser altamente efectiva, gracias al uso de rastros de feromonas que guían la exploración de soluciones. Estudios comparativos han demostrado que, en instancias con mayor número de nodos, ACO tiende a superar al AG en términos de calidad de solución y estabilidad de los resultados (Stützle y Hoos, 2000b).

A nivel nacional, los trabajos sobre optimización de rutas mediante metaheurísticas en contextos locales son escasos. En México, algunos estudios han aplicado estas técnicas en redes eléctricas, planificación agrícola o logística de transporte público, pero existen muy pocos antecedentes dirigidos a la optimización de rutas de reparto urbano con datos reales de localidades como Villahermosa, Tabasco. Esta tesis busca precisamente cubrir ese vacío, proponiendo una solución funcional basada en datos locales y con potencial de aplicación práctica.

2.3. Estudios similares con AG y ACO

Tabla 2.1. Comparativa de estudios con AG y ACO en escenarios urbanos/VRP

Estudio (año)	Contexto / Instancias	Algoritmos	Hallazgos clave
Aldoraibi et al., 2024	Ruteo urbano real con prioridades; ciudades (LA, NY, Chicago)	ACO, GA, PSO	*ACO* obtuvo las rutas más cortas en LA y Chicago; *GA* logró la mejor distancia en NY; compara tiempo, costo y robustez con datos reales.
Ran et al., 2023	"Green city logistics" (planeación de rutas urbanas)	GA mejorado (GA+SA) vs ACO/ABC/PSO	El GA mejorado (criterio de Metropolis + recocido simulado) supera a ACO/ABC/PSO en calidad de solución y estabilidad; converge más rápido y evita óptimos locales.
Ky Phuc y Phuong Thao, 2021 *Logistics* (2021)	MPMDVRP con ventanas de tiempo y *flota heterogénea* (alto realismo operativo)	ACO (enfoque aplicado)	ACO maneja múltiples restricciones reales (TW, capacidades, duración del conductor) y escala a instancias grandes, con enfoque de costo total.
Tan et al., 2024 *World Electric Vehicle Journal* (2024)	Logística urbana con *tipos de carga diversos* (MT-CVRP); caso Beijing	ACO mejorado (Lévy-EGACO)	Ahorro de costos del 1.8–2.5 % frente a algoritmos comparados; mejora búsqueda (Lévy flights) y actualización de feromonas en malla urbana compleja.
Chen et al., 2025 *Electronics* (2025)	VRPTW (benchmark de Solomon)	Híbrido GA+ACO (IGA-ACO)	Igualó BKS en clase C y redujo vehículos 24.45 % (clase R); ligera mejora en RC (0.19 %); fuerte balance entre uso de vehículos y distancia total.

2.4. Limitaciones de las metaheurísticas

Aunque las metaheurísticas (AG y ACO) han mostrado buen desempeño en ruteo urbano, presentan limitaciones que deben reconocerse al interpretar resultados y transferir soluciones a otros contextos Blum y Roli, 2003; Dorigo y Stützle, 2004; Talbi, 2009.

- Dependencia de parámetros. Su rendimiento es sensible a hiperparámetros (tamaño de población, tasas de cruce/mutación; α, β, ρ en ACO). El ajuste inadecuado degrada calidad y estabilidad; el tuning exhaustivo puede ser costoso en cómputo.
- Riesgo de sobreajuste. Optimizar parámetros sobre un conjunto reducido de instancias o una métrica única puede producir configuraciones que no generalizan a nuevos escenarios, demandas o topologías urbanas.
- Variabilidad estocástica. Los resultados fluctúan entre corridas; se requieren múltiples eje-

cuciones, estadísticas de dispersión y pruebas no paramétricas para comparar algoritmos con rigor.

- Falta de garantías de optimización. No aseguran soluciones óptimas globales ni cotas de error; su eficacia depende del equilibrio exploración–explotación y de operadores/vecindarios elegidos.
- Escalabilidad y restricciones complejas. El desempeño puede deteriorarse al crecer el tamaño del problema o al incorporar ventanas de tiempo, flotas heterogéneas y penalizaciones, que exigen diseño cuidadoso de funciones de costo.

Buenas prácticas para mitigar: (i) separar instancias en conjuntos de entrenamiento/validación/-prueba; (ii) usar búsqueda de hiperparámetros sistemática (p. ej., búsqueda aleatoria o bayesiana) y reportar el costo computacional; (iii) hibridar con búsqueda local (2-opt/3-opt) para mejorar intensificación; (iv) ejecutar múltiples corridas con semillas distintas y reportar medias, desviaciones y pruebas de significancia; (v) evaluar robustez ante cambios de demanda/distancias para reducir el sobreajuste (Blum & Roli, 2003; Eiben & Smith, 2003).

2.5. Marco tecnológico

El desarrollo del sistema propuesto para la generación de rutas de reparto se llevó a cabo mediante herramientas tecnológicas actuales que permiten la implementación eficiente de algoritmos de optimización y el manejo de datos geográficos. En este apartado se describen los lenguajes de programación, bibliotecas, entornos de desarrollo y servicios de terceros utilizados, así como su justificación técnica.

2.5.1. Lenguaje de programación

El sistema fue desarrollado en el lenguaje Python 3.11.5, una plataforma de programación de alto nivel, multiparadigma y con una amplia comunidad en el ámbito científico. Python es ampliamente utilizado en proyectos de inteligencia artificial, análisis de datos y optimización, debido a su simplicidad sintáctica y la disponibilidad de bibliotecas especializadas(Van Rossum y Drake Jr.,

2009) . En particular, su versatilidad lo hace ideal para implementar algoritmos metaheurísticos y realizar análisis estadísticos.

2.5.2. Entorno de desarrollo

Se utilizó Visual Studio Code (VS Code) versión 1.84.2 como entorno de desarrollo integrado (IDE). Esta herramienta permite una edición de código fluida, depuración avanzada, integración con sistemas de control de versiones y soporte para múltiples extensiones orientadas a la productividad en Python.

2.5.3. Librerías y paquetes

Para la implementación de los algoritmos y el manejo de datos se emplearon las siguientes bibliotecas:

- NumPy: para operaciones numéricas y manejo eficiente de matrices y vectores(Harris et al., 2020).
- Pandas: para la manipulación y análisis de estructuras tabulares, útil en el procesamiento de los datos del DENUE.
- Matplotlib: para la visualización de resultados, gráficas de dispersión, boxplots y análisis comparativo entre algoritmos.
- Requests y JSON: para el consumo de APIs externas (MapQuest, Google Maps) y el manejo de respuestas en formato JSON.

2.5.4. Servicios de mapas y cálculo de distancias

Para la obtención de la matriz de distancias reales entre ubicaciones, se utilizaron servicios de terceros que ofrecen interfaces de programación (APIs):

- Google Maps Platform: mediante su API de direcciones y distancia, se calcularon rutas reales considerando la red vial urbana actual. Este servicio ofrece datos altamente confiables, aunque con límites de uso y costos asociados(Developers, 2023).

- MapQuest Developer: como alternativa gratuita para obtener distancias, particularmente útil durante las pruebas preliminares. Este servicio también considera restricciones viales y condiciones de tráfico.

Ambas plataformas proporcionan coordenadas geográficas, rutas estimadas y tiempos de viaje, lo que permitió construir una matriz de distancias precisa como entrada para los algoritmos de optimización.

2.5.5. Plataforma de ejecución

Los algoritmos fueron ejecutados en un entorno local con las siguientes características de hardware y sistema operativo:

- Sistema operativo: Windows 11 Home Single Language
- Procesador: AMD Ryzen 5 5500U con gráficos Radeon
- Memoria RAM: 8 GB
- Almacenamiento: SSD de 237 GB

Este entorno fue suficiente para realizar todas las combinaciones experimentales definidas en el diseño, cada una con 25 ejecuciones independientes, sin requerir procesamiento distribuido ni acceso a infraestructura de cómputo en la nube.

Capítulo 3

Modelado del problema y conjunto de datos

Es importante garantizar la calidad del modelado y la pertinencia del conjunto de datos debido a que son aspectos fundamentales para poder validar correctamente los resultados en un estudio de optimización logística. En esta tesis, se ha utilizado el modelo matemático clásico del Problema del Agente Viajero (TSP), utilizando datos reales obtenidos del Directorio Estadístico Nacional de Unidades Económica (DENUE), del Instituto Nacional de Estadística y Geografía (INEGI).

El objetivo es simular de forma realista un escenario de reparto urbano en la ciudad de Villahermosa, Tabasco, con el fin de evaluar y validar el comportamiento de dos algoritmos metaheurísticos de optimización. En este capítulo se describen los elementos que componen la formulación matemática del problema, el diagrama conceptual del dominio, y los procedimientos utilizados para adquirir, transformar y preparar los datos de entrada.

3.1. Modelo matemático

El Problema del Agente Viajero (TSP) se representa formalmente como un problema de optimización combinatoria en el que se busca minimizar la distancia total recorrida por un agente que debe visitar un conjunto de ciudades una sola vez y regresar al punto de partida.

Sea un conjunto de n ciudades numeradas del 1 al n, y sea d_{ij} la distancia entre la ciudad i y la ciudad j. La variable de decisión x_{ij} toma el valor de 1 si el trayecto va de la ciudad i a la

ciudad j, y 0 en caso contrario.

Función objetivo:

$$\text{Minimizar} \quad Z = \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} d_{ij} \cdot x_{ij} \tag{3.1}$$

Donde:

d_{ij} es la distancia o costo de ir de la ciudad i a la ciudad j.

x_{ij} es una variable binaria

$$x_{ij} = \begin{cases} 1, & \text{si se viaja directamente de la ciudad } i \text{ a la ciudad } j \\ 0, & \text{en caso contrario} \end{cases}$$

Z es el valor total de la distancia recorrida, que queremos minimizar.

Sujeta a las restricciones:

1. Cada ciudad debe ser visitada exactamente una vez:

$$\sum_{j=1, j \neq i}^{n} x_{ij} = 1 \quad \forall i \in \{1, 2, \ldots, n\} \tag{3.2}$$

2. Cada ciudad debe ser llegada exactamente una vez:

$$\sum_{i=1, i \neq j}^{n} x_{ij} = 1 \quad \forall j \in \{1, 2, \ldots, n\} \tag{3.3}$$

3. Eliminación de subcircuitos (restricciones de subtour):

$$u_i - u_j + n \cdot x_{ij} \leq n - 1 \quad \forall\, i \neq j,\ i, j \in \{2, 3, \ldots, n\} \tag{3.4}$$

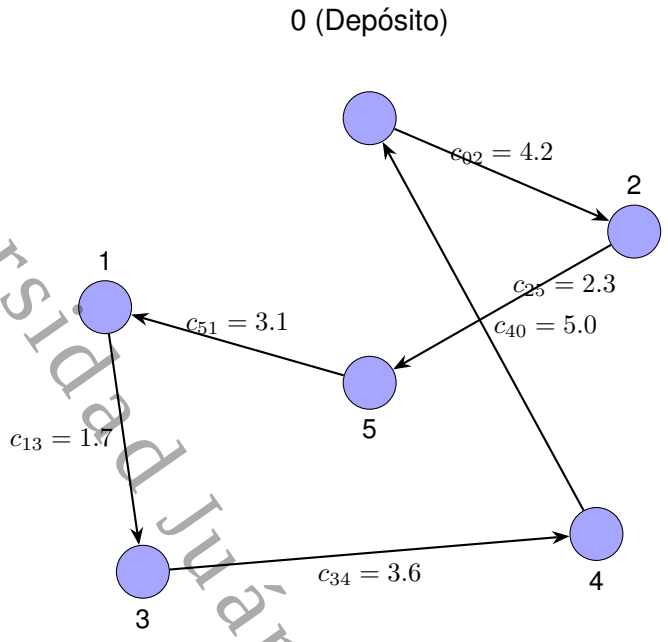
Donde u_i y u_j son variables auxiliares que impiden la formación de ciclos más pequeños que el circuito completo (Miller et al., 1960).

3.2. Estructura del dominio del problema

El problema abordado se desarrolla dentro de un dominio específico: la distribución urbana de productos en la ciudad de Villahermosa. En este contexto, cada punto de entrega representa una unidad económica activa, registrada en el DENUE. La estructura del dominio considera las siguientes entidades clave:

- Nodo: punto geográfico correspondiente a una ubicación económica (cliente).
- Distancia: costo asociado al desplazamiento entre dos nodos.
- Ruta: secuencia ordenada de nodos a ser visitados.
- Algoritmo: técnica de optimización utilizada para generar dicha ruta.

Estos elementos interactúan en un sistema cuyo objetivo es minimizar el recorrido total. La matriz de distancias generada constituye el insumo fundamental para los algoritmos de optimización y se basa en distancias reales extraídas desde servicios web de mapeo, como Google Maps y MapQuest.



Conceptos clave (Cap. 3)
V: conjunto de nodos, $V = \{0, \ldots, n\}$; 0 es depósito.
A: conjunto de arcos (i, j).
c_{ij}: distancia del arco (i, j).
r: ruta como secuencia que inicia y termina en 0.
Distancia total: $\sum_{(i,j) \in r} c_{ij}$

Figura 3.1. Diagrama conceptual del dominio

.

3.3. Descripción estadística del conjunto de datos

El conjunto de datos extraído del DENUE incluyó 60 ubicaciones seleccionadas dentro de la ciudad de Villahermosa, correspondientes a distintos giros comerciales. Se procesaron las coordenadas geográficas (latitud y longitud) y se construyó una matriz de distancias reales utilizando la API de Google Maps y MapQuest.

A continuación, se presentan algunos estadísticos descriptivos sobre las distancias medidas (en kilómetros) entre pares de ubicaciones:

Tabla 3.1. Resumen estadístico del conjunto de datos

Estadístico	Valor aproximado
Número de nodos	60
Número de combinaciones	3,540 (sin repetición)
Distancia mínima	0.50 km
Distancia máxima	12.75 km
Distancia promedio	4.95 km
Desviación estándar	2.85 km

3.4. Justificación del tamaño del problema

Para los experimentos, se seleccionaron dos escenarios: uno con 10 nodos y otro con 60 nodos. El caso de 10 nodos representa un entorno reducido, útil para observar el comportamiento básico de los algoritmos con bajo costo computacional. El escenario de 60 nodos, en cambio, se aproxima a condiciones reales de operación en empresas de reparto urbano, donde un solo vehículo debe cubrir múltiples puntos de entrega distribuidos en una ciudad mediana como Villahermosa.

Estos dos escenarios permiten comparar rendimiento y escalabilidad de los algoritmos metaheurísticos en diferentes niveles de complejidad.

3.5. Visualización de la red de distribución

A continuación, se muestra una representación gráfica de los 60 nodos seleccionados sobre un mapa de Villahermosa, Tabasco.

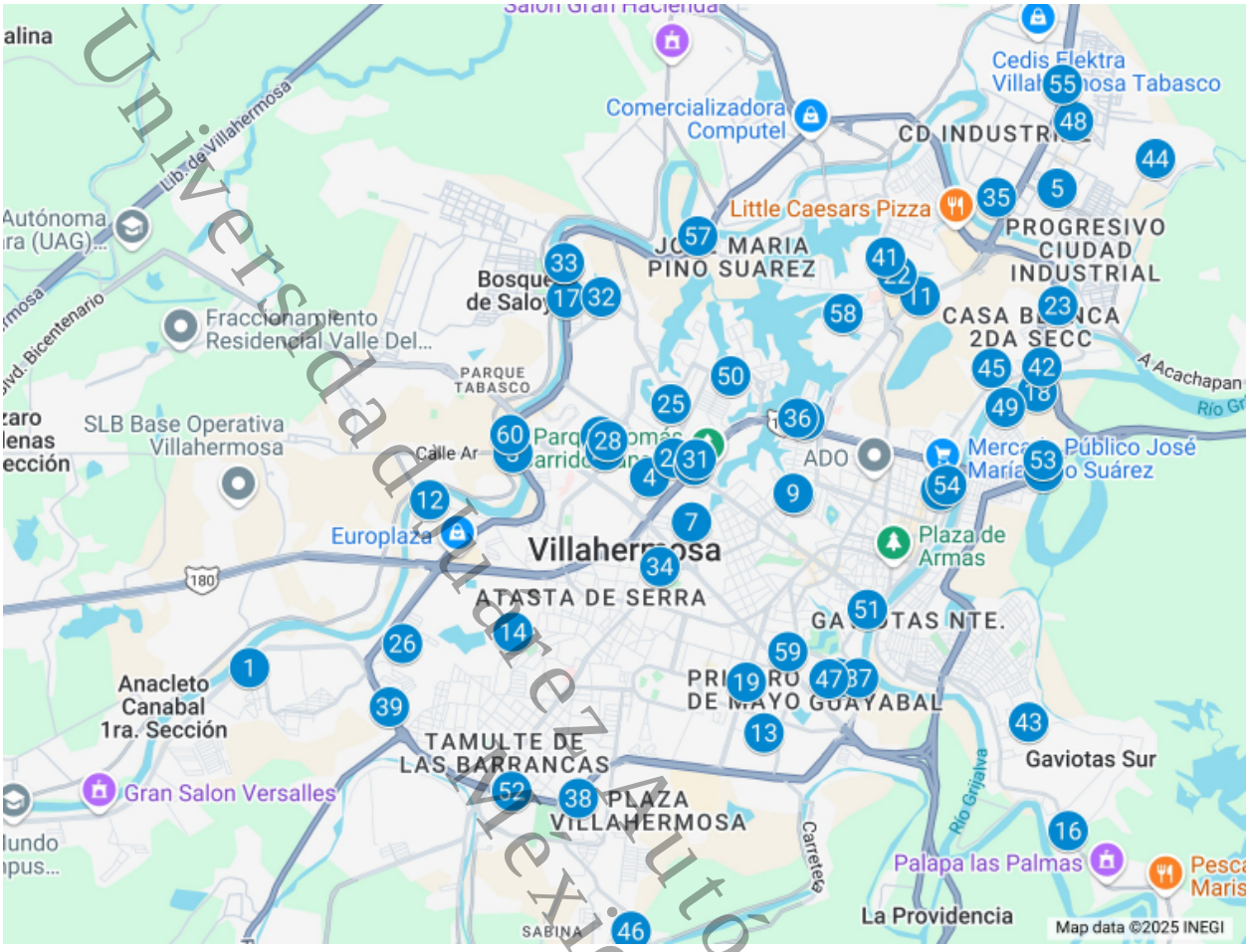


Figura 3.2. Visualización geográfica de la red de distribución

3.6. Calidad de los datos y limitaciones

Aunque el DENUE es una fuente oficial y confiable, existen algunas consideraciones que deben tomarse en cuenta:

- Actualización: La base de datos puede tener desfases temporales respecto a negocios que ya cerraron o cambiaron de domicilio.
- Homogeneidad: Algunas ubicaciones están muy concentradas, lo que puede afectar la distribución espacial de los nodos.
- Dependencia de servicios externos: Las APIs de Google Maps y MapQuest tienen límites de uso, tiempos de respuesta variables y no siempre consideran el tráfico en tiempo real.

Pese a estas limitaciones, los datos utilizados proporcionan un entorno suficientemente realista para la validación de las metaheurísticas.

3.6.1. Limitaciones de los datos y de las APIs de distancia/tiempo

El cálculo de costos c_{ij} (distancias o tiempos) se basa en servicios externos (p. ej., Google Maps, MapQuest) y en fuentes de direcciones/POIs (p. ej., DENUE/INEGI). Esta dependencia introduce varias limitaciones que afectan la *reproducibilidad*, la *robustez* y la *transferencia* de los resultados.

Dependencia de la fuente y de versiones del mapa. Cada proveedor usa un motor de ruteo, cartografía y reglas de tránsito distintas (sentidos, giros, límites de velocidad). Cambios en la versión del mapa (nuevas calles, cierres, rediseño vial) pueden alterar c_{ij} sin modificar el algoritmo de optimización, generando soluciones distintas frente a la misma instancia lógica.

Sesgo temporal y condiciones operativas. Las APIs pueden devolver valores *estáticos* (distancia geodésica o tiempo promedio) o *dinámicos* condicionados por la hora y el tráfico. La estacionalidad (p. ej., temporada de lluvias e inundaciones en zonas urbanas de Tabasco), eventos puntuales (obras, bloqueos, festividades) o cambios de la hora de consulta impactan las matrices de costo.

Geocodificación y precisión espacial. Errores o ambigüedad en geocodificación (direcciones incompletas, nombres duplicados) desplazan nodos y distorsionan c_{ij}. En entornos densos, pequeños errores de ubicación (decenas de metros) pueden reordenar rutas óptimas cuando hay múltiples alternativas similares.

Parámetros de consulta y configuración. Opciones como `avoid=tolls/highways`, `mode=driving`, `departure_time=now` vs. tiempos por defecto cambian las respuestas. Si el *tuning* de metaheurísticas se hizo con una configuración y luego se consulta con otra, la comparación entre algoritmos se sesga.

Cuotas, muestreo y latencia. Límites de peticiones (rate limiting) pueden forzar muestreos parciales o escalonados en el tiempo, introduciendo variabilidad entre días u horas. Además, respuestas con caché del proveedor pueden no reflejar condiciones reales en el momento de la consulta.

Métrica objetivo y caso de uso. Optimizar por *distancia* ignora variaciones de *tiempo* (tráfico, semáforos), y viceversa. Si el objetivo operativo es puntualidad, usar sólo distancia puede subestimar cuellos de botella urbanos.

Efecto de cambios urbanos en el modelo

Si las condiciones urbanas cambian, la matriz $C = \{c_{ij}\}$ cambia y, con ella, la frontera de soluciones. Esto afecta:

- Calidad absoluta: la distancia/tiempo total cambia en magnitud.
- Ranking de heurísticas: AG y ACO pueden intercambiar posiciones si sus sesgos de búsqueda reaccionan distinto a la nueva topología/costos.
- Estabilidad: la dispersión entre corridas tiende a aumentar si la red se vuelve más heterogénea.

Buenas prácticas para mitigar

1. Versionado y trazabilidad de datos. Registrar proveedor, fecha/hora de consulta, parámetros (`mode`, `avoid`, `departure_time`), y el *hash* de cada matriz C usada en experimentos.
2. Instantáneas (snapshots) y caché interno. Persistir las matrices C empleadas en los análisis para garantizar *reproducibilidad exacta* de resultados.
3. Validación cruzada multi-fuente. Comparar métricas con al menos dos proveedores (p. ej., Google/MapQuest u OSRM-OpenStreetMap) y reportar la sensibilidad de la solución a la fuente de datos.
4. Escenarios y análisis de sensibilidad. Evaluar soluciones bajo varios escenarios s (picos de tráfico, lluvias, cierres) perturbando c_{ij} (p. ej., $\pm 5\,\%$, $\pm 10\,\%$) o usando `departure_time` distintos; reportar medias, percentiles y degradación relativa.
5. Datos operativos reales. Cuando sea posible, contrastar con trazas GPS/telemetría para recalibrar costos y verificar desviaciones sistemáticas (sobre/subestimación de tiempos).

6. Selección de métrica acorde al objetivo. Si la prioridad es puntualidad, optimizar por *tiempo* (o costo compuesto) y no sólo por distancia; considerar funciones objetivo multi-criterio.
7. Reentrenamiento periódico. Actualizar C y re-ejecutar la optimización tras cambios relevantes (nueva vialidad, obras prolongadas, temporada de lluvias).
8. Robustez en metaheurísticas. Incluir búsqueda local (2-opt/3-opt) e iterar sobre múltiples semillas; usar pruebas de significancia entre algoritmos en cada escenario para evitar conclusiones espurias.
9. Operación offline cuando aplique. Para estabilidad, mantener un flujo de procesamiento alternativo con ruteo offline (p. ej., OSRM+OSM) que permita evaluar escenarios sin depender de cuotas y latencia de terceros.

En síntesis, las APIs de mapas aportan practicidad y realismo, pero su naturaleza *dinámica* exige documentar versiones, controlar parámetros y evaluar sensibilidad. Sólo así las conclusiones sobre el desempeño de AG y ACO se mantienen válidas cuando el entorno urbano cambia.

3.6.2. Efecto de cambios urbanos en el modelado

Tabla 3.2. Riesgos derivados de datos y APIs de ruteo, su efecto en los resultados y mitigaciones recomendadas.

Riesgo (datos/APIs)	Efecto en resultados	Mitigación recomendada
Proveedor / versión del mapa	Variación de c_{ij} por cambios en sentidos, límites o calles nuevas; pérdida de reproducibilidad.	Versionar proveedor, fecha/hora y parámetros; conservar *snapshots* de matrices C; validación cruzada (p. ej., Google vs. OSRM/OSM).
Variabilidad temporal (tráfico, clima, eventos)	Diferencias entre consultas (hora pico vs. valle, lluvias/inundaciones); cambio de ranking entre soluciones.	Escenarios con distintas `departure_time`; análisis de sensibilidad (± 5–$10\,\%$ en c_{ij}); re-ejecuciones periódicas.
Cambios urbanos (obras, cierres, bloqueos)	Perturbaciones locales elevan c_{ij}, rutas dejan de ser óptimas; mayor dispersión entre corridas.	Monitoreo operativo; actualización de C tras cambios relevantes; reoptimización con nuevos datos.
Geocodificación imprecisa	Desplazamiento de nodos; rutas subóptimas por ubicaciones mal resueltas o duplicadas.	Validación manual/muestreo; normalización de direcciones; uso de coordenadas verificadas (GPS/telemetría).
Parámetros de consulta inconsistentes	Diferencias por `mode`, `avoid`, `traffic_model`, etc.; comparaciones sesgadas entre algoritmos.	Estandarizar parámetros; registrar configuración; repetir consultas con la misma *pipeline*.
Cuotas, caché y latencia de APIs	Muestreo parcial o respuestas inconsistentes en el tiempo; matrices C no replicables.	Caché interno y *batching*; respetar *rate limits*; registrar ID de respuesta y *timestamp*.
Métrica objetivo no alineada (distancia vs. tiempo)	Ruta óptima en distancia no minimiza tiempo/impuntualidad; decisiones subóptimas operativas.	Definir métrica según objetivo (tiempo o multi-criterio); reportar trade-offs distancia/tiempo.
Cobertura/topología incompleta	Calles privadas, restricciones de giro o pesos no modelados $\rightarrow$ soluciones inviables.	Verificación de viabilidad con reglas locales; penalizaciones/filtrado de arcos; uso de datos operativos.
Outliers, ruido o fallos de API	c_{ij} anómalos alteran la solución; inestabilidad entre corridas.	Detección y recorte de outliers; imputación robusta; repetición de consultas fallidas.
Sobreajuste de hiperparámetros a una sola matriz C	Configuración que rinde bien en un escenario pero no generaliza a otros.	Partición en *train/valid/test* de instancias; búsqueda de hiperparámetros (aleatoria/bayesiana); evaluar robustez multi-escenario.

3.7. Estructura de la matriz de distancias

La matriz de distancias utilizada es almacenada en formato CSV y cargada como estructura de tipo `numpy.ndarray` en Python para ser utilizada por los algoritmos de optimización. Se accede

a ella mediante índices, lo que facilita su integración con los operadores de cruce, mutación y evaluación de costo total.

Capítulo 4

Implementación de algoritmos y diseño experimental

Este capítulo presenta el desarrollo computacional de los algoritmos metaheurísticos utilizados en la investigación: Algoritmo Genético (AG) y Algoritmo de Colonia de Hormigas (ACO). También se describe el diseño experimental que permitió comparar el rendimiento de ambas técnicas en distintos escenarios, utilizando datos reales del entorno urbano de Villahermosa, Tabasco. La implementación se realizó en el lenguaje Python y se evaluó mediante pruebas sistemáticas sobre diferentes tamaños de problema.

4.1. Implementación del Algoritmo Genético (AG)

El Algoritmo Genético (AG) es una técnica inspirada en la evolución biológica y basada en mecanismos como la selección natural, cruce genético y mutación (Holland, 1975). Su aplicación al TSP implica trabajar con representaciones de soluciones como permutaciones de ciudades.

4.1.1. Pseudocódigo del AG aplicado al TSP:

1. Inicializar una población de soluciones aleatorias (permutaciones de nodos).
2. Evaluar cada solución (distancia total recorrida).

3. Repetir durante N generaciones:
   a. Seleccionar padres mediante torneo binario.
   b. Aplicar cruce PMX para generar descendientes.
   c. Aplicar mutación por inversión con baja probabilidad.
   d. Evaluar nuevos individuos.
   e. Reemplazar la población anterior (generacional).
4. Retornar la mejor solución hallada.

4.1.2. Parámetros utilizados en AG

Tabla 4.1. Parámetros utilizados en los experimentos

Parámetro	**Valor**
Codificación	Permutación
Inicialización	Aleatoria
Cruce	PMX (Partially Mapped Crossover)
Mutación	Inversión
Selección	Torneo binario
Población	20 y 200 individuos
Generaciones	300 y 3000
Probabilidad de cruce	0.8
Probabilidad de mutación	0.01

4.1.3. Justificación de parámetros del AG

En problemas de ruteo con representación por permutación (p. ej., TSP/VRP), adoptamos operadores y probabilidades clásicas con amplio respaldo en la literatura. Usamos cruce PMX con probabilidad alta ($p_c = 0.8$) para favorecer recombinación de *building blocks*, mutación por inversión con $p_m = 0.01 \approx 1/n$ para mantener diversidad sin destruir tours prometedores, y selección por torneo con $k = 3$ que ofrece una presión moderada y estable (Blickle y Thiele, 1996; Eiben y Smith, 2003; D. E. Goldberg, 1989; Michalewicz, 1996). Para el tamaño de población y el número de generaciones, seguimos guías que recomiendan poblaciones en “low hundreds” y presupuestos de evaluación suficientes para mezclar y madurar soluciones ($N = 200$, $G = 3000$) (Eiben y Smith, 2003; D. Goldberg et al., 1992; Katoch et al., 2020). En todo caso, estos parámetros pueden refinarse con *tuning* automático (p. ej., *irace*) (López-Ibáñez et al., 2016).

Tabla 4.2. Parámetros del AG: valores usados, criterio y respaldo bibliográfico.

Parámetro	Símbolo	Valor usado	Criterio/Justificación	Referencia(s)
Cruce (PMX)	p_c	0.8	Cruces específicos de permutación (PMX/OX) preservan factibilidad y adyacencias; prob. de cruce alta (~ 0.6-0.9) favorece recombinación eficaz.	(D. E. Goldberg, 1989; Michalewicz, 1996)
Mutación (inversión)	p_m	0.01	Regla clásica $p_m \approx 1/n$; con $n{=}60 \Rightarrow \approx 0.0167$. Valor ligeramente conservador para no destruir *building blocks* cuando p_c es alto.	(Eiben y Smith, 2003; Michalewicz, 1996)
Selección por torneo	k	3	Torneo con $k = 2$–4 genera presión moderada; robusto ante escalado de aptitud y evita convergencia prematura.	(Blickle y Thiele, 1996; Eiben y Smith, 2003)
Tamaño de población	N	200	"Low hundreds" recomendado; poblaciones mayores ayudan a discriminar bajo ruido y mantienen diversidad.	(Eiben y Smith, 2003; D. Goldberg et al., 1992; Katoch et al., 2020)
Generaciones	G	3000	Presupuesto de evaluación $N \times G$ suficiente (600k); puede complementarse con criterio de convergencia.	(Eiben y Smith, 2003; Katoch et al., 2020)
Ajuste automático		(opcional)	Recomendada validación con *irace* sobre instancias locales para minimizar sesgo manual.	(López-Ibáñez et al., 2016)

4.2. Implementación del Algoritmo de Colonia de Hormigas (ACO)

El ACO se basa en el comportamiento de las hormigas reales, las cuales dejan rastros de feromona que influyen en el camino que otras hormigas eligen (Dorigo y Di Caro, 1999). Para el TSP, cada hormiga construye una solución iterativamente, guiada por la cantidad de feromona y la visibilidad (inversa de la distancia).

4.2.1. Pseudocódigo del ACO para TSP

1. Inicializar feromonas uniformemente en todos los arcos.

2. Repetir durante N iteraciones:

a. Para cada hormiga:

   i. Construir una ruta probabilística guiada por feromonas y distancia.

   ii. Evaluar la ruta (distancia total).

b. Evaporar parte de la feromona de todos los arcos.

c. Reforzar feromonas en los arcos de la mejor ruta global.

3. Retornar la mejor solución encontrada.

4.2.2. Parámetros utilizados

Tabla 4.3. Parámetros utilizados en el algoritmo de colonia de hormigas

Parámetro	**Valor**
Número de hormigas	Igual al número de nodos
Feromonas iniciales	0.1
Evaporación (ρ)	0.5
Cantidad depositada (Q)	1
Importancia de feromona (α)	1
Importancia de visibilidad (β)	2
Estrategia de actualización	Global

4.2.2.1. Justificación de parámetros en ACO

En el ACO para TSP/VRP, los parámetros α, β y ρ regulan el equilibrio entre explotación del rastro de feromonas y el uso de información heurística $(\eta_{ij} = 1/d_{ij})$. Siguiendo la literatura clásica e implementaciones de referencia, utilizamos $\alpha = 1$, $\beta = 2$ y $\rho = 0.5$.

- β (peso heurístico): $\beta = 2$ es un valor consolidado en ACS tras optimización preliminar y se usa como base en TSP (Dorigo y Gambardella, 1997).
- α (peso de feromona): $\alpha = 1$ es el valor empleado por MAX–MIN Ant System y su distribución `acotsp`, ampliamente replicada en TSP (Stützle y Hoos, 2000a).
- ρ (evaporación): valores más altos fomentan exploración y evitan estancamiento; adoptamos $\rho = 0.5$, coherente con `acotsp`, adecuado cuando los costos pueden variar por condiciones urbanas (Dorigo y Stützle, 2004).

4.3. Comparación de parámetros AG vs. ACO

Tabla 4.4. Comparación de parámetros entre el Algoritmo Genético (AG) y la Colonia de Hormigas (ACO)

Parámetro	AG	ACO
Representación	Permutación	Construcción paso a paso
Inicialización	Estocástica (población aleatoria)	Feromonas iguales
Cruce	PMX	No aplica
Mutación	Inversión	No aplica
Selección de padres	Torneo binario	No aplica
Selección de sobrevivientes	Generacional	No aplica
Tamaño de población	20 y 200	Igual al número de nodos
Iteraciones	300 y 3000	300 y 3000
Parámetros específicos	$P_c = 0.8,\ P_m = 0.01$	$\alpha = 1,\ \beta = 2,\ \rho = 0.5$

4.4. Diseño experimental

Se diseñó un experimento factorial que incluye los siguientes factores:

- Tamaño del problema: 10 y 60 nodos
- Tamaño de la población: 20 y 200
- Número de iteraciones: 300 y 3000

Cada combinación fue ejecutada 25 veces para obtener resultados estadísticamente representativos.

Tabla 4.5. Configuraciones experimentales para AG y ACO

Código	Algoritmo(s)	Nodos	Población	Iteraciones
E1	AG y ACO	10	20	300
E2	AG y ACO	60	20	300
E3	AG y ACO	10	20	3000
E4	AG y ACO	60	20	3000
E5	AG y ACO	10	200	300
E6	AG y ACO	60	200	300
E7	AG y ACO	10	200	3000
E8	AG y ACO	60	200	3000

4.4.1. Complejidad computacional esperada

Los métodos exactos para TSP/VRP (p. ej., Bellman–Held–Karp, *branch-and-bound*) crecen de forma exponencial en n y se vuelven impracticables para instancias reales (Blum y Roli, 2003; Eiben y Smith, 2003). En contraste, las metaheurísticas usadas aquí:

- AG con representación por permutación y operadores PMX+inversión tiene costo por generación proporcional a $O(P\,n)$ (evaluación de la población y operadores $O(n)$), por lo que el costo total es $O(P\,G\,n)$.
- ACO construye A recorridos por iteración; cada hormiga realiza n elecciones con normalización de probabilidades sobre $O(n)$ candidatos, lo que da $O(A\,n^2)$ por iteración y $O(A\,I\,n^2)$ en total (Dorigo y Stützle, 2004).

Estas complejidades son *polinomiales*, por lo que las soluciones son típicamente *subóptimas* (sin garantía de optimalidad global), pero *escalables* a tamaños de interés práctico. En términos operativos: el AG crece aproximadamente lineal con n (a recurso computacional fijo $P{\times}G$), mientras que ACO crece de forma cuadrática; aun así, ACO mostró mejor *estabilidad* y calidad de solución en nuestros escenarios, lo que justifica su mayor costo.

Evidencia de crecimiento con n (10 $\rightarrow$ 60). Bajo la configuración fija de esta tesis ($P{=}A{=}200$, $G{=}I{=}3000$), los modelos de costo predicen:

$$\frac{T_{\mathsf{GA}}(60)}{T_{\mathsf{GA}}(10)} \approx \frac{60}{10} = 6\times, \qquad \frac{T_{\mathsf{ACO}}(60)}{T_{\mathsf{ACO}}(10)} \approx \frac{60^2}{10^2} = 36\times. \tag{4.1}$$

Donde $TGA(n)$ y $TACO(n)$ representan el tiempo de ejecución del algoritmo genético (GA) y del algoritmo de colonia de hormigas (ACO), respectivamente, para una instancia de tamaño n.

Esto explica el incremento pronunciado del tiempo al pasar de instancias pequeñas ($n{=}10$) a escenarios realistas ($n{=}60$), especialmente en configuraciones con poblaciones grandes ($P{=}200$) y números altos de iteraciones ($G{=}3000$). La Figura 4.1 y la Figura 4.2 muestran la tendencia esperada, y la Figura 4.3 destaca el factor de aumento relativo.

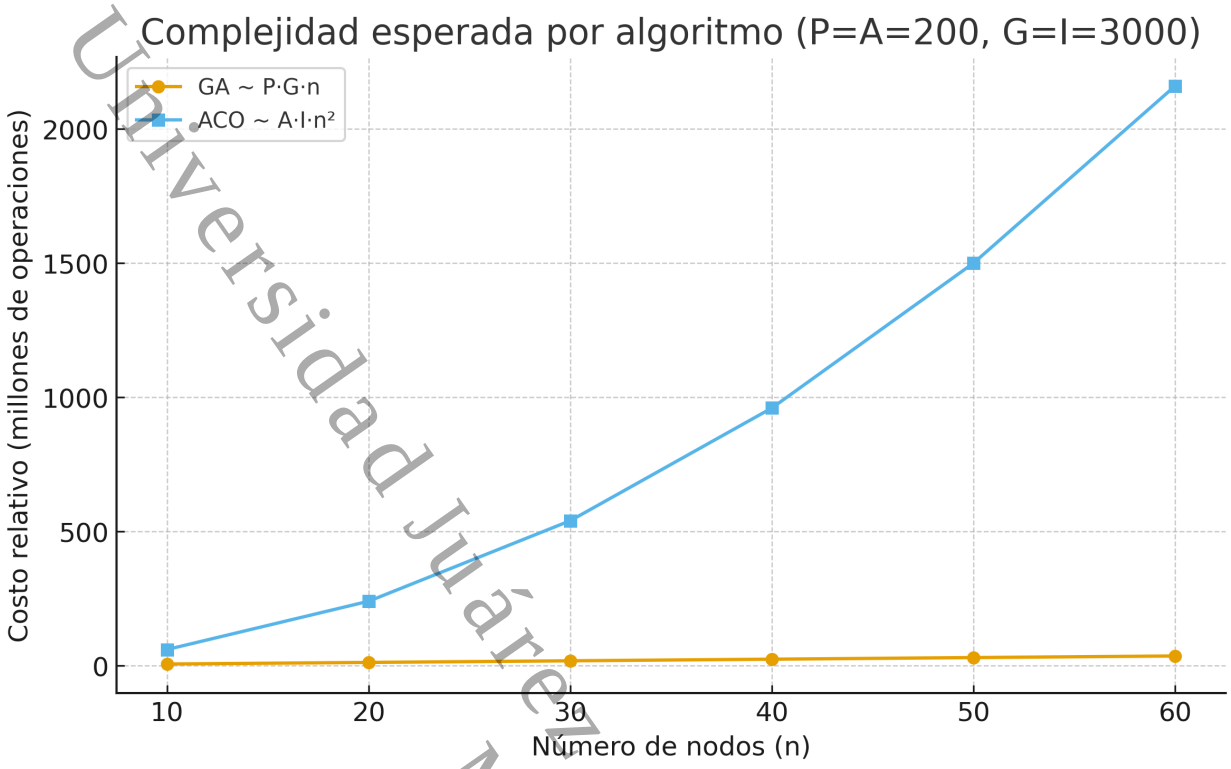


Figura 4.1. Complejidad esperada (escala lineal) con $P{=}A{=}200$ y $G{=}I{=}3000$.

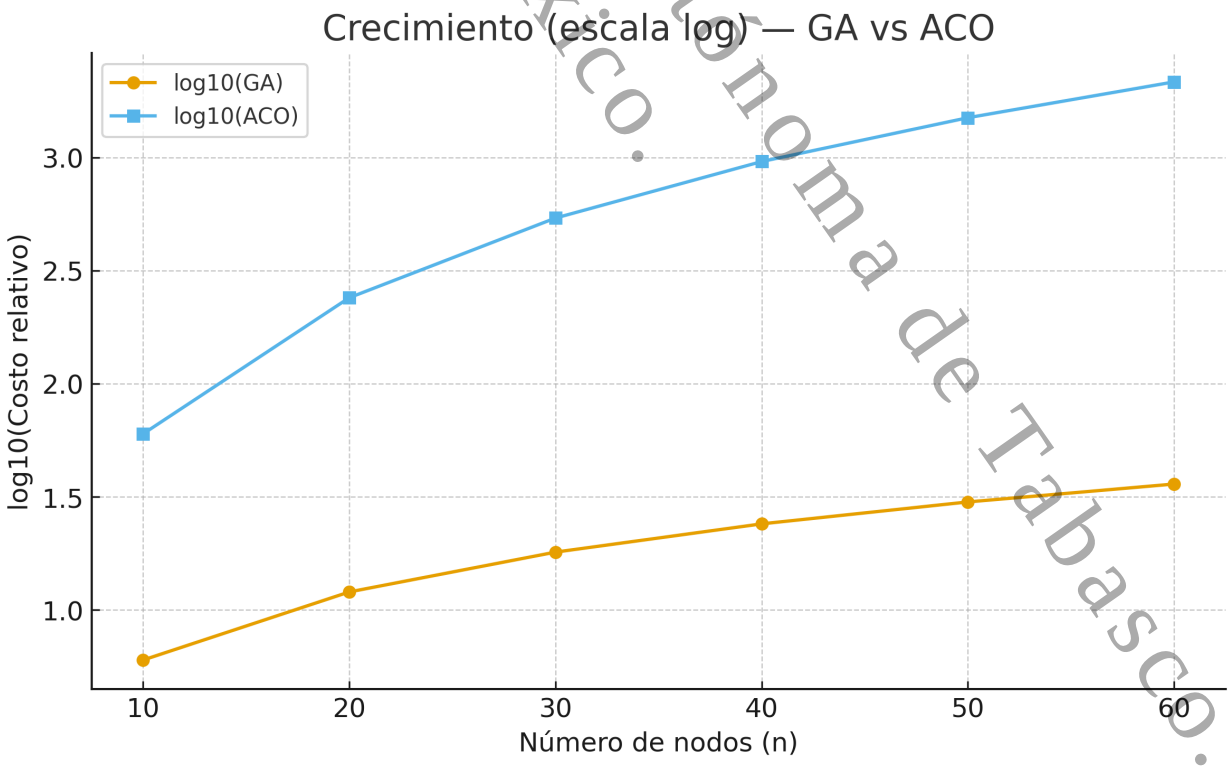


Figura 4.2. Crecimiento en escala log: GA (aprox. lineal en n) vs. ACO (aprox. cuadrático).

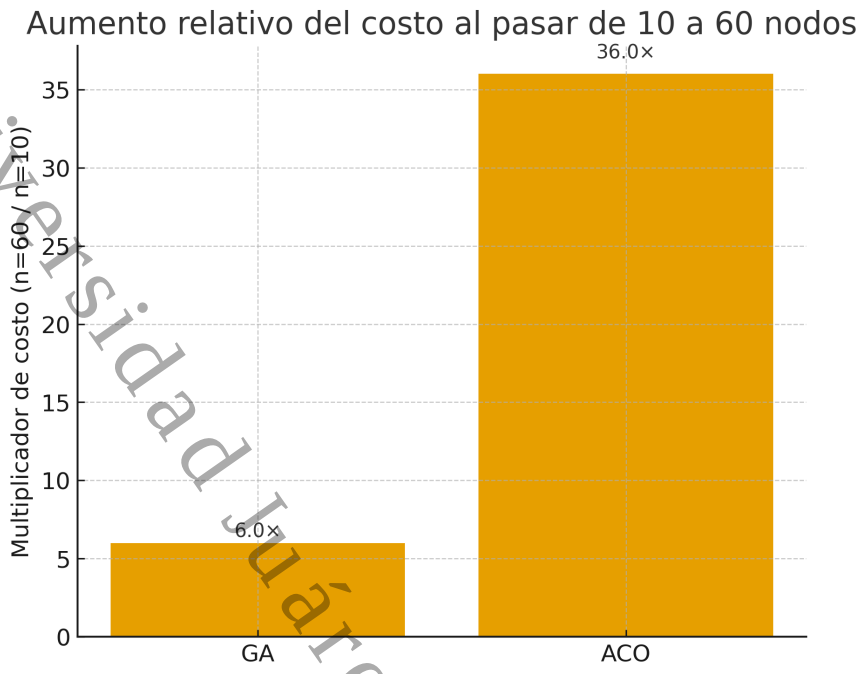


Figura 4.3. Aumento relativo del costo al pasar de $n{=}10$ a $n{=}60$: GA $\approx 6\times$; ACO $\approx 36\times$.

4.5. Criterios de evaluación

El rendimiento de cada algoritmo fue evaluado con base en:

- Distancia total recorrida (valor de la función objetivo del TSP)
- Mejor solución, promedio, peor solución y desviación estándar
- Comparación entre AG y ACO mediante la prueba estadística Mann–Whitney U, al no asumir distribución normal de los datos (Gibbons y Chakraborti, 2011)

Este enfoque permite evaluar la calidad y estabilidad de las soluciones obtenidas por cada algoritmo en distintos escenarios.

Capítulo 5

Resultados y análisis

Este capítulo presenta los resultados obtenidos a partir de la implementación y ejecución de los algoritmos Algoritmo Genético (AG) y Algoritmo de Colonia de Hormigas (ACO) en diferentes configuraciones experimentales, utilizando datos reales de ubicaciones en la ciudad de Villahermosa, Tabasco.

Se realizaron 8 configuraciones de prueba, cada una evaluada con 25 ejecuciones independientes, para analizar el comportamiento de ambos algoritmos en términos de calidad de la solución (distancia recorrida), estabilidad (desviación estándar) y rendimiento relativo.

5.1. Resultados experimentales

A continuación se presentan en la Tabla 5.1 los resultados promedio de cada configuración. Se incluye el valor mínimo, promedio, desviación estándar y máximo de la distancia recorrida por las rutas generadas por cada algoritmo.

Tabla 5.1. Resultados comparativos entre AG y ACO

Exp.	Alg.	Nodos	Pob.	Iter.	Mejor	Promedio	Desv. Est.	Peor
E1	AG	10	20	300	22.13	25.07	1.12	26.95
	ACO				**22.13**	**23.15**	**0.45**	**24.01**
E2	AG	60	20	300	185.25	197.58	8.87	214.43
	ACO				**67.05**	**70.67**	**1.65**	**72.86**
E3	AG	10	20	3000	22.13	23.17	0.79	24.89
	ACO				**22.13**	**22.80**	**0.46**	**23.70**
E4	AG	60	20	3000	143.56	161.64	9.89	185.57
	ACO				**64.75**	**68.40**	**1.85**	**72.21**
E5	AG	10	200	300	22.13	22.79	0.53	24.00
	ACO				**22.13**	**22.38**	**0.39**	**23.23**
E6	AG	60	200	300	129.57	145.74	6.91	160.77
	ACO				**65.41**	**68.28**	**1.66**	**71.48**
E7	AG	10	200	3000	22.13	22.54	0.47	23.87
	ACO				**22.13**	**22.34**	**0.31**	**23.06**
E8	AG	60	200	3000	102.38	111.23	6.18	125.32
	ACO				**64.89**	**66.94**	**1.38**	**70.73**

5.1.1. Convergencia por iteraciones (mediana de 25 corridas)

Con los experimentos E1-E8, en Figura 5.1, Figura 5.2, Figura 5.3 y Figura 5.4 se comparan las medianas finales al incrementar el numero de iteraciones de 300 a 3000, para poblaciones de 20 y 200 individuos.

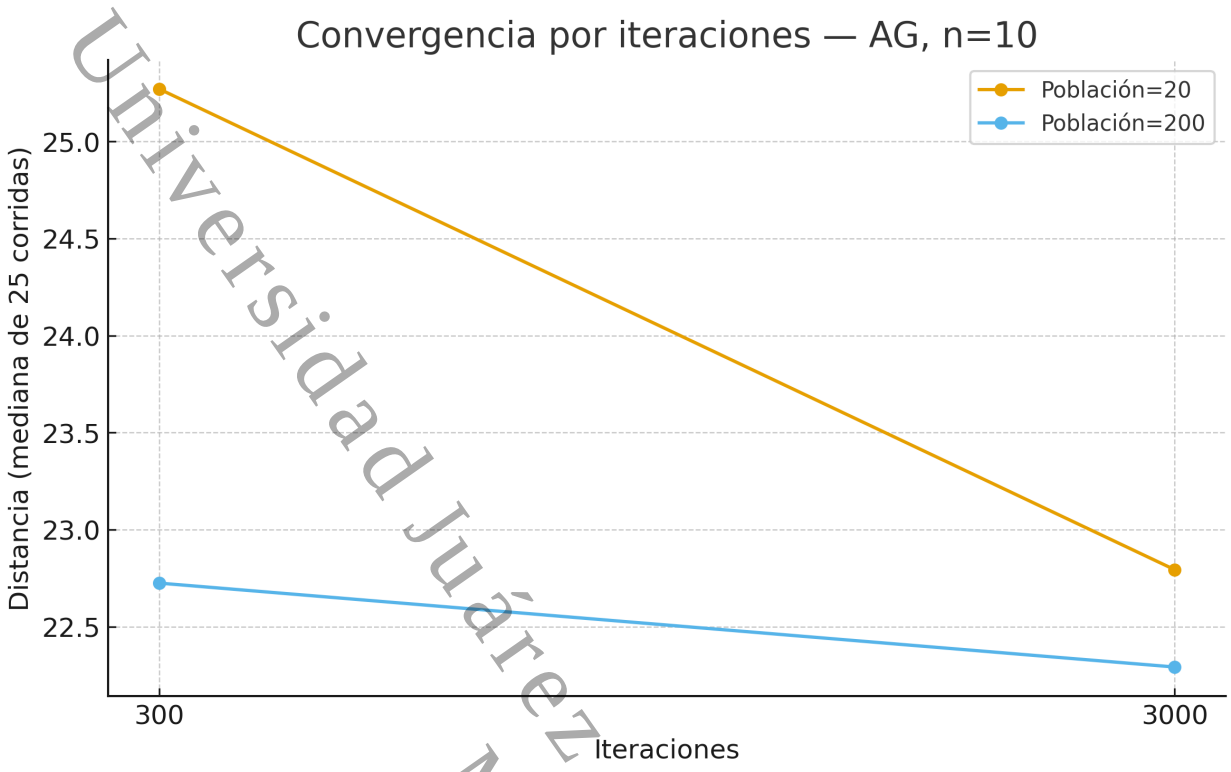


Figura 5.1. AG, n=10. Mejora 300→3000: P=20 +9.80 %, P=200 +1.90 %.

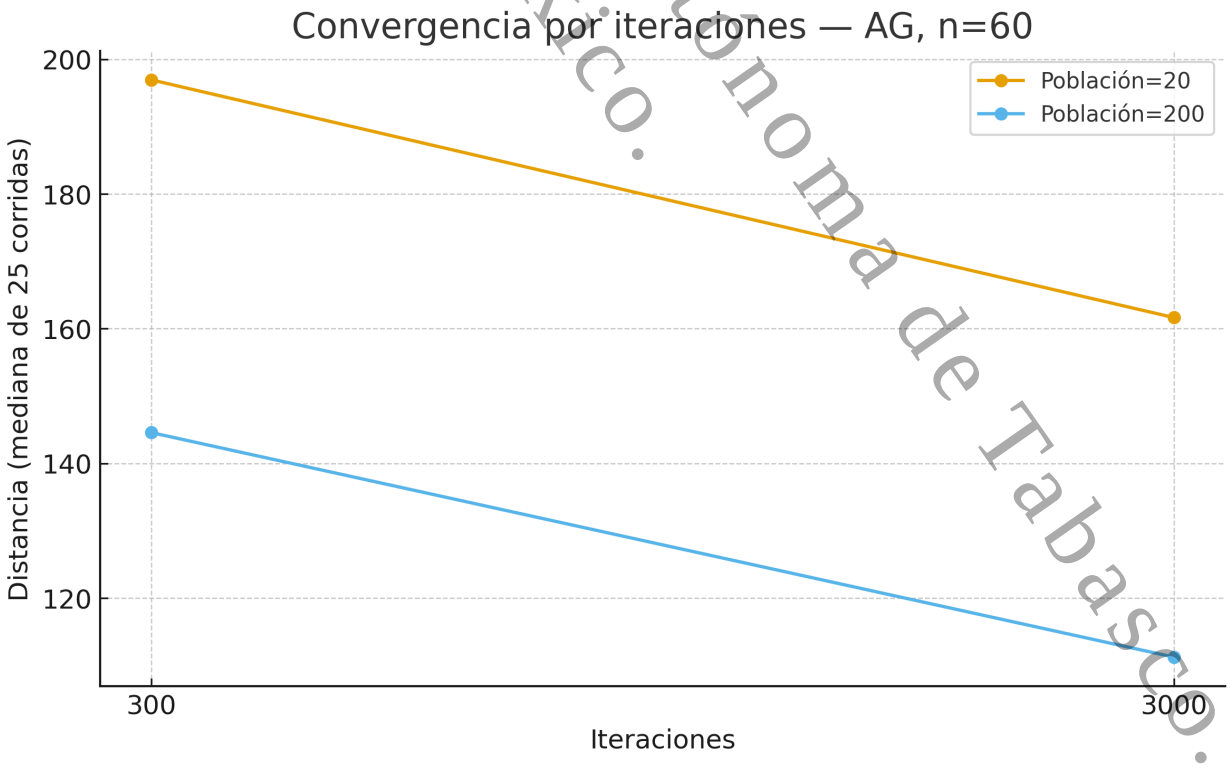


Figura 5.2. AG, n=60. Mejora 300→3000: P=20 +17.93 %, P=200 +23.07 %.

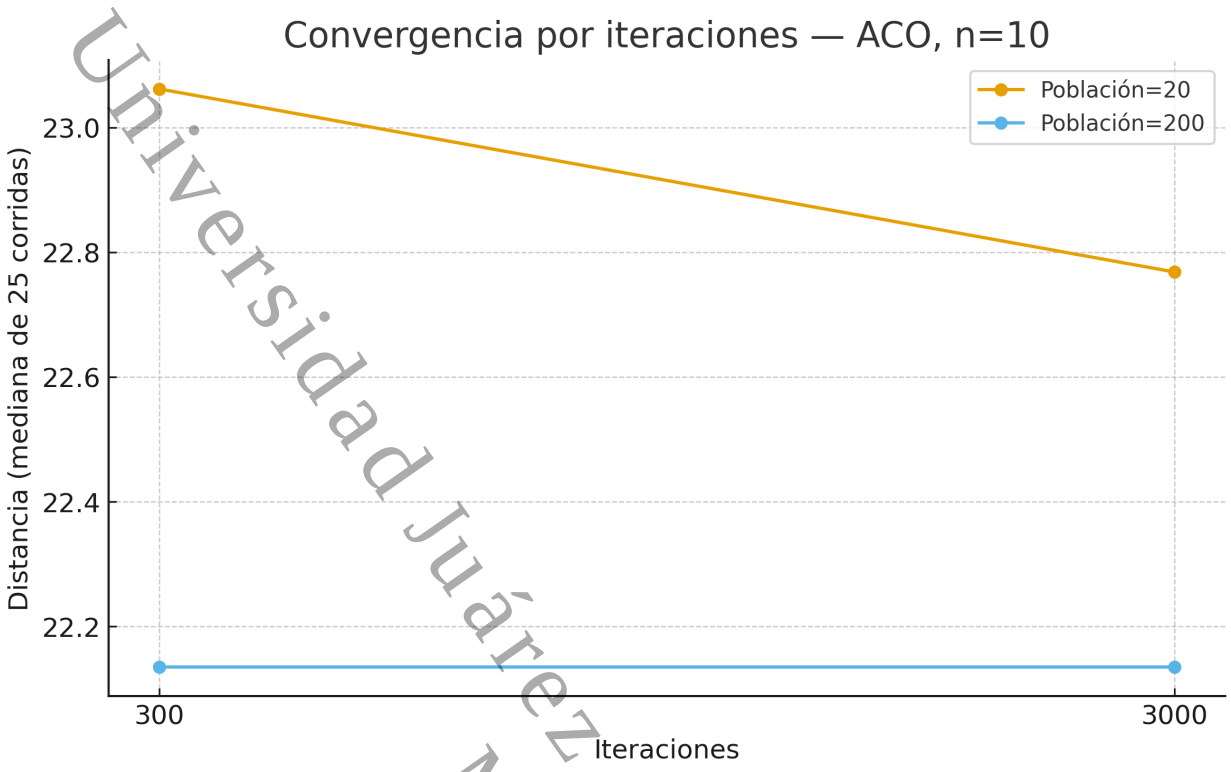


Figura 5.3. ACO, n=10. Mejora 300→3000: P=20 +1.27 %, P=200 +0.00 %.

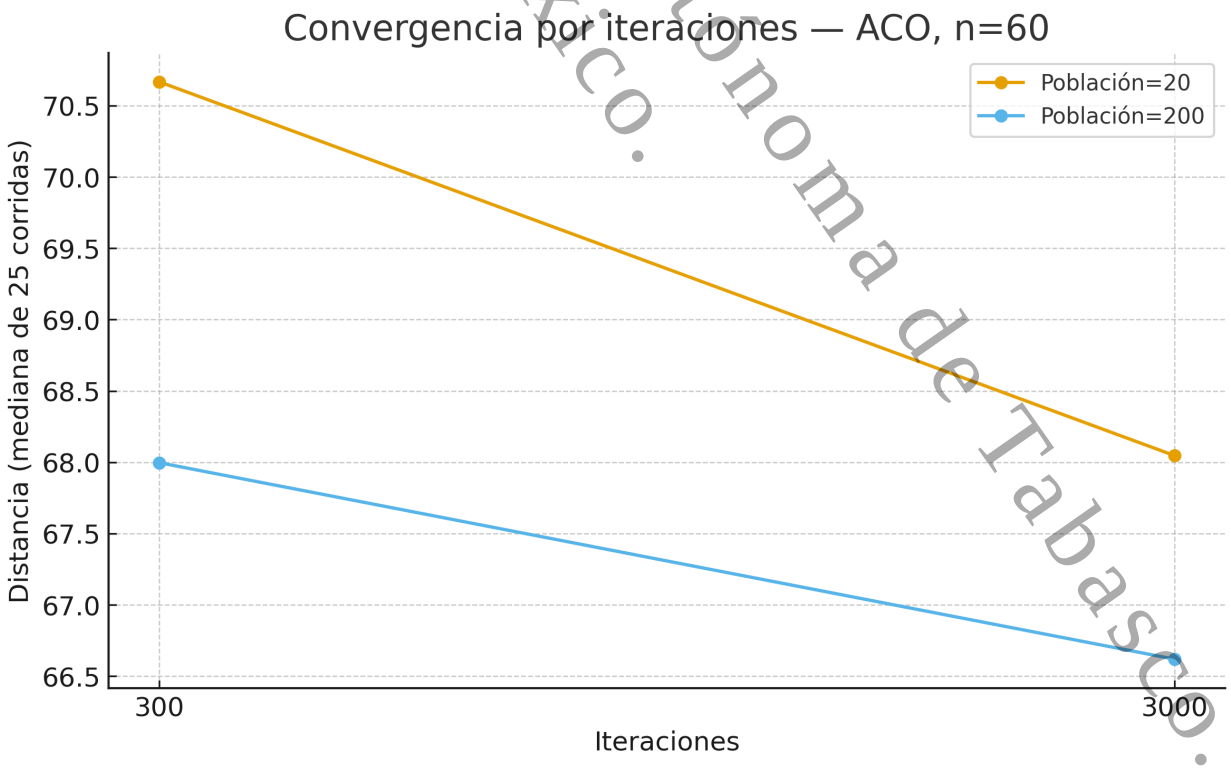


Figura 5.4. ACO, n=60. Mejora 300→3000: P=20 +3.71 %, P=200 +2.03 %.

Lectura e interpretación de las tendencias de convergencia

AG, n=10 (Fig. 5.1). Cada línea muestra la *mediana* (25 corridas) de la mejor distancia para $P \in \{20, 200\}$ al pasar de 300 a 3000 iteraciones. Se observa una mejora clara con P=20 (+9.80 %) y marginal con P=200 (+1.90 %). Esto sugiere que, en instancias pequeñas, el AG ya explora suficientemente el espacio con P=200 y 300 iteraciones, por lo que el aumento del costo computacional produce rendimientos decrecientes. La distancia más baja implica mejor calidad de solución.

AG, n=60 (Fig. 5.2). Para el problema grande, el AG se beneficia notablemente del costo computacional: con P=20 mejora +17.93 % y con P=200 +23.07 %. La ganancia adicional con población grande indica que la diversidad (más individuos) y el mayor número de iteraciones favorecen la exploración y refinamiento del tour en escenarios urbanos más complejos.

ACO, n=10 (Fig. 5.3). El ACO muestra poca sensibilidad al incremento de iteraciones en el caso pequeño: +1.27 % con P=20 y +0.00 % con P=200. Esto es consistente con una convergencia rápida del rastro de feromonas en espacios reducidos: con 300 iteraciones ya se obtienen soluciones competitivas y el salto a 3000 aporta poca ganancia adicional.

ACO, n=60 (Fig. 5.4). En el escenario grande, ACO mantiene medianas bajas (mejores) y mejora de forma moderada al aumentar iteraciones: +3.71 % (P=20) y +2.03 % (P=200). La estabilidad del método (curvas más “planas”) sugiere menor variabilidad entre corridas y una intensificación del rastro que, si bien asegura calidad, ofrece retornos decrecientes frente a un mayor costo computacional.

Implicación práctica común. En instancias pequeñas, ambos algoritmos muestran rendimientos decrecientes al pasar de 300 a 3000 iteraciones (especialmente ACO y AG con P=200). En instancias grandes, el costo computacional adicional favorece especialmente al AG (mejoras superiores al 17 %–23 %), mientras que ACO mantiene un desempeño estable con ganancias moderadas. La elección del par {algoritmo, P} debe balancearse con el tiempo de cómputo disponible (ver Sec. 4.4.1).

5.1.2. Convergencia por iteraciones (comparativa)

Con los experimentos E1–E8 (Población $\in \{20, 200\}$; Iteraciones $\in \{300, 3000\}$), en la Figura 5.5 y Figura 5.6 se comparan AG y ACO en un mismo eje para $n = 10$ y $n = 60$.

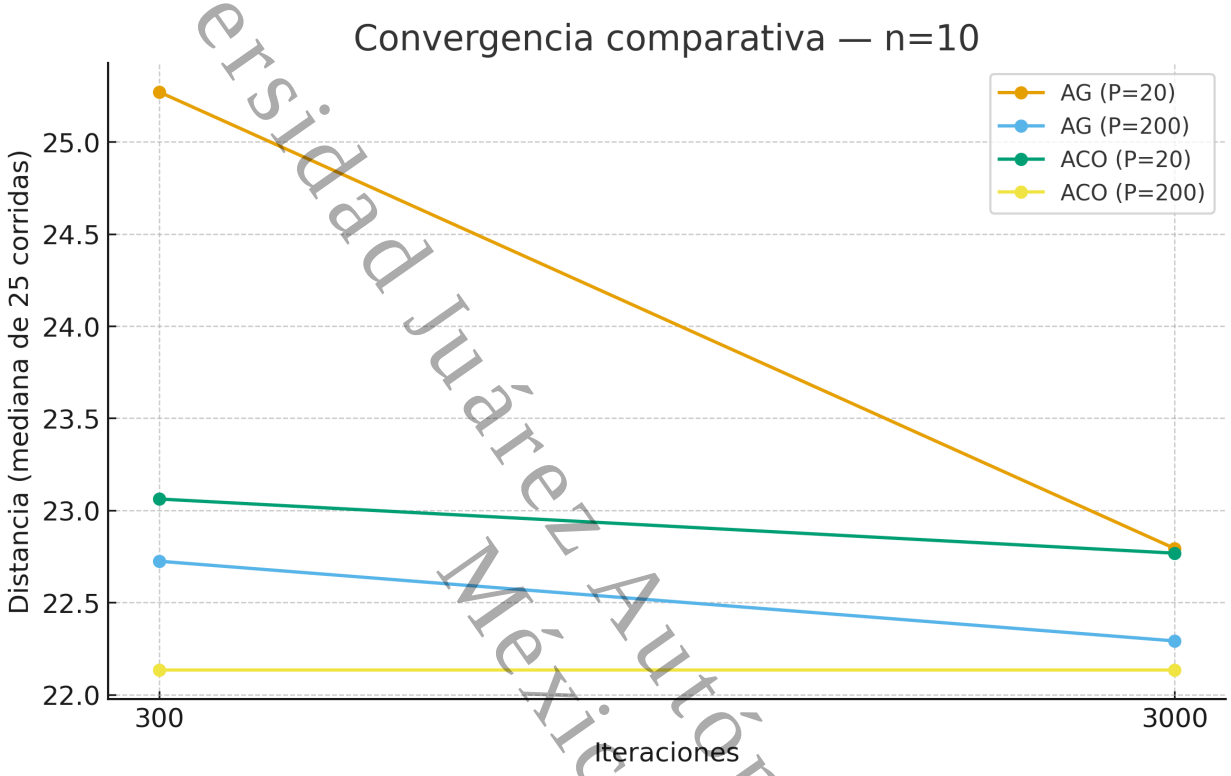


Figura 5.5. Convergencia comparativa por iteraciones (n=10)
. Mejora 300→3000: AG(P=20) +9.80 %, AG(P=200) +1.90 %, ACO(P=20) +1.27 %, ACO(P=200) +0.00 %.

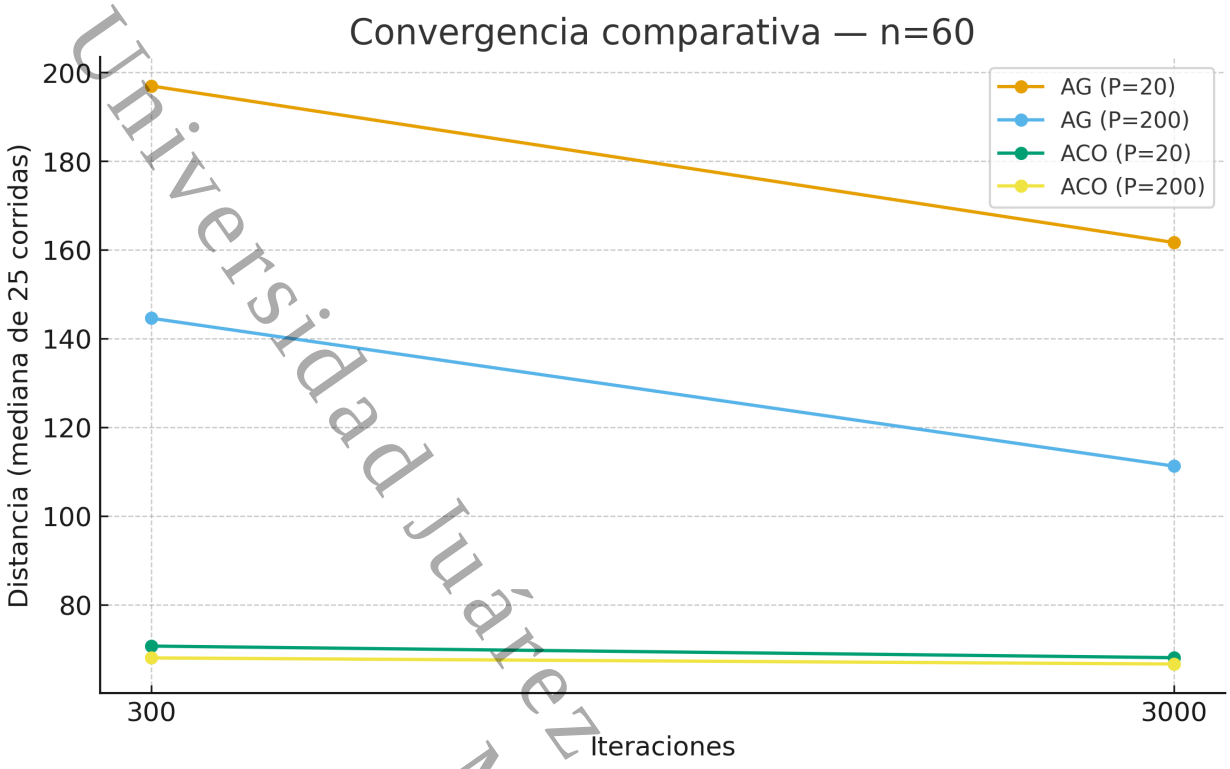


Figura 5.6. Convergencia comparativa por iteraciones (n=60). Mejora 300→3000: AG(P=20) +17.93 %, AG(P=200) +23.07 %, ACO(P=20) +3.71 %, ACO(P=200) +2.03 %.

Las Figuras 5.5 y 5.6 muestran la *mediana* (sobre 25 corridas independientes) de la mejor distancia alcanzada por configuración al aumentar el número de iteraciones de 300 a 3000; se reportan dos líneas por algoritmo (poblaciones $P{=}20$ y $P{=}200$). Menor distancia implica mejor calidad de solución. La mejora porcentual indicada en los pies de figura se calcula como $\big(\text{mediana}_{300} - \text{mediana}_{3000}\big)/\text{mediana}_{300} \times 100\,\%$.

Resultados para $n{=}10$. Las curvas confirman que el aumento de iteraciones aún aporta ganancia, pero con *rendimientos decrecientes*: ACO mejora poco (1.27 % con $P{=}20$; 0.00 % con $P{=}200$), señal de rápida estabilización; el AG mejora 9.80 % ($P{=}20$) y 1.90 % ($P{=}200$). En instancias pequeñas, ACO es poco sensible al mayor presupuesto porque ya produce tours competitivos con 300 iteraciones.

Resultados para $n{=}60$. En el escenario urbano (más difícil), el efecto del presupuesto es más marcado. El AG se beneficia claramente del salto 300→3000 (17.93 % con $P{=}20$; 23.07 % con $P{=}200$), mientras que ACO mejora moderadamente (3.71 % y 2.03 %, respectivamente). Esto sugiere que, para n grande, *más iteraciones y mayor población* favorecen especialmente al AG,

en tanto que ACO mantiene su comportamiento más estable.

Implicaciones prácticas. (i) Si el tiempo de cómputo es limitado (presupuestos cercanos a 300 iteraciones), ACO ofrece buena calidad con baja sensibilidad al aumento de presupuesto. (ii) Si se dispone de más cómputo y el problema es grande (n=60), el AG se beneficia más al crecer iteraciones y población, aunque con mayor costo computacional (ver Sec. 4.4.1). (iii) Para decidir el presupuesto operativo, el par {tamaño del problema, algoritmo} y el objetivo (calidad vs. tiempo) deben equilibrarse; las curvas proporcionan una guía empírica para ese compromiso.

5.2. Análisis estadístico

Para cada experimento, se aplicó la prueba Mann–Whitney U con un nivel de significancia de $\alpha = 0.05$ (95 % de confianza) con el fin de determinar si existen diferencias estadísticamente significativas entre AG y ACO (Mann y Whitney, 1947). Esta prueba es adecuada para datos no paramétricos y distribuciones desconocidas.

Tabla 5.2. Resultados de la prueba Mann-Whitney U entre AG y ACO

Experimento	Valor p (Mann-Whitney U)	Diferencia significativa
E1	1.97e-07	Sí
E2	1.42e-09	Sí
E3	0.158	No significativa
E4	1.42e-09	Sí
E5	0.0015	Sí
E6	1.42e-09	Sí
E7	0.0699	No significativa
E8	1.42e-09	Sí

Estos resultados indican que ACO supera consistentemente a AG en escenarios más complejos (mayor número de nodos), mientras que en instancias pequeñas el desempeño es comparable.

5.3. Comparación de estabilidad

En cuanto a la estabilidad de los resultados obtenidos por ambos algoritmos, se observa que el ACO presenta una menor desviación estándar en todos los escenarios evaluados, lo que indica

una mayor consistencia entre ejecuciones. Por ejemplo, en el escenario con 60 nodos y 3000 iteraciones (E8), el algoritmo genético (AG) registró una desviación estándar de 6.18, mientras que ACO obtuvo 1.38. Asimismo, para el caso de 10 nodos y 3000 iteraciones (E7), AG presentó una desviación estándar de 0.47, mientras que ACO mostró un valor ligeramente inferior de 0.31. Estos resultados permiten concluir que ACO no solo obtiene mejores soluciones promedio, sino que sus resultados son más estables y confiables en diferentes configuraciones experimentales.

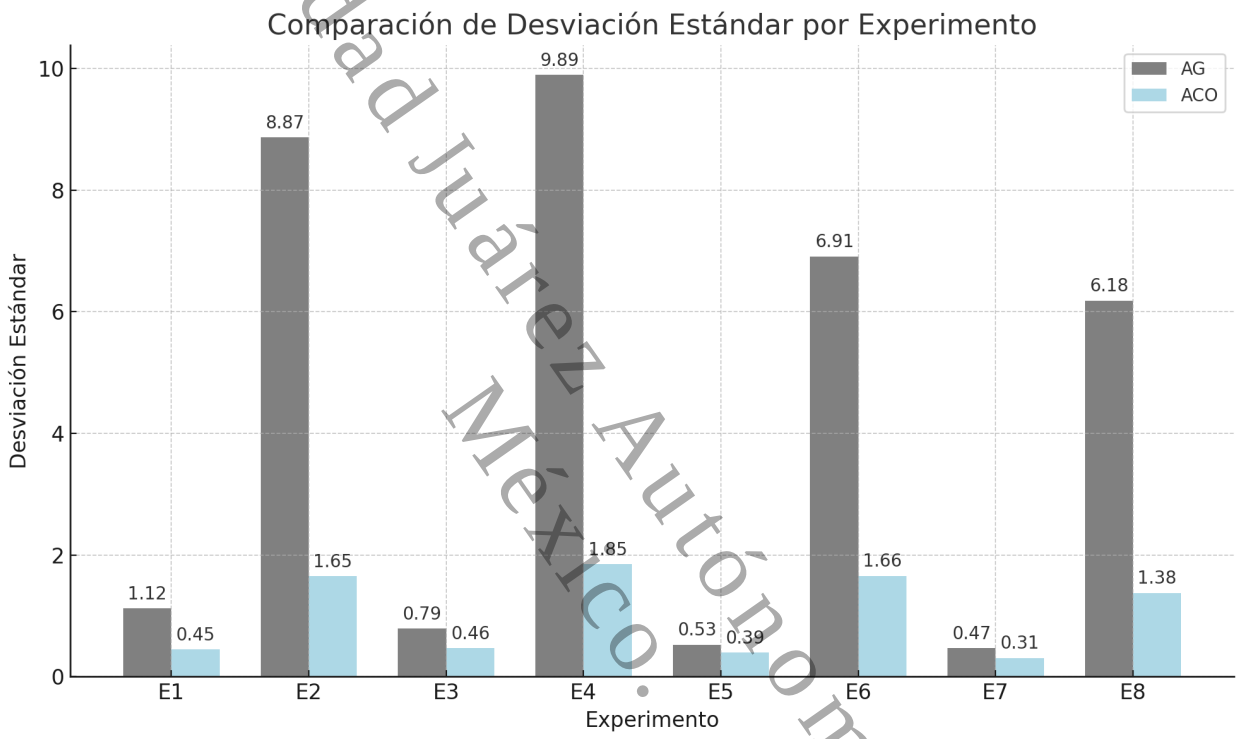


Figura 5.7. Comparación de estabilidad entre AG y ACO (Desviación estándar)

5.4. Visualización comparativa

A continuación se muestra la comparación del rendimiento promedio de los algoritmos AG y ACO en cada experimento. Se observa que ACO presenta consistentemente valores promedio más bajos, lo que indica un mejor desempeño general frente a AG.

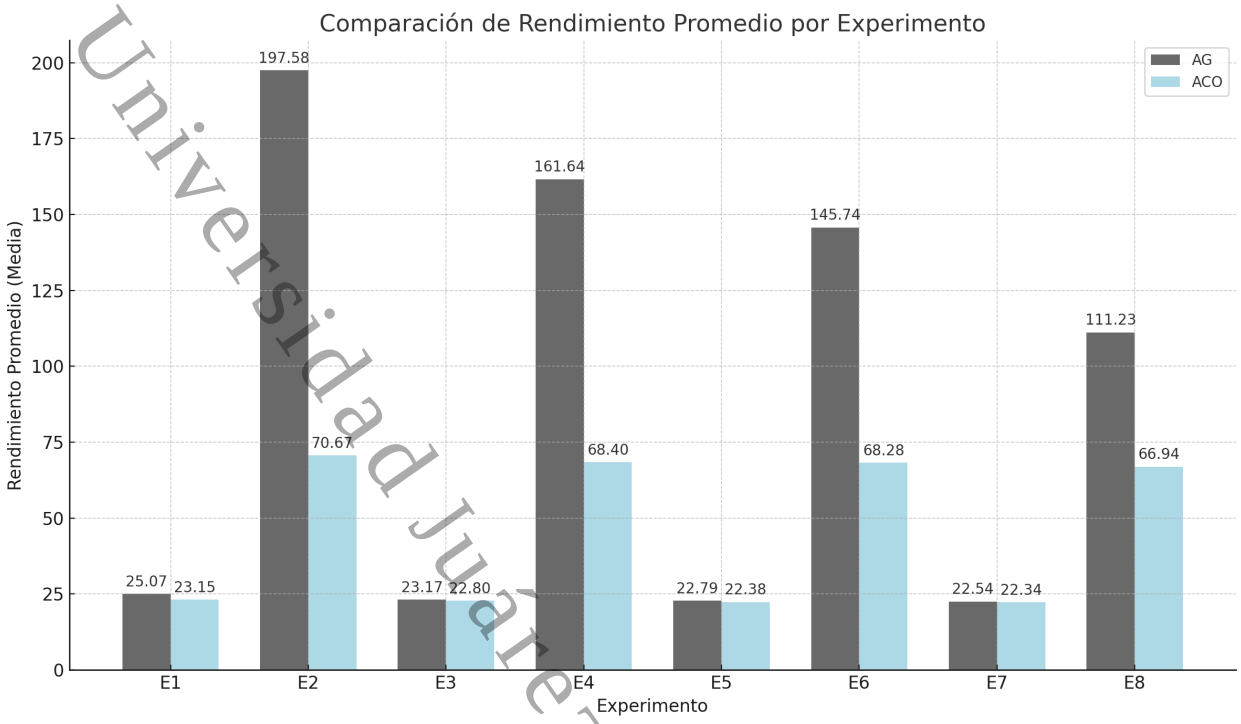


Figura 5.8. Comparación del rendimiento promedio entre AG y ACO

5.5. Discusión de resultados

Los resultados de este capítulo muestran que ACO alcanza, en promedio, *menores distancias* y *menor dispersión* entre corridas que AG (véase Figs. 5.5–5.6 y la tabla comparativa). A continuación discutimos las causas desde dos ángulos: (i) la estructura del dominio de ruteo urbano y (ii) la codificación y operadores utilizados por cada método.

(i) Ajuste al dominio (ruteo urbano con distancias métricas). En TSP/VRP con distancias métricas (triangularidad y penalización por desvíos), la heurística local $\eta_{ij} = 1/d_{ij}$ es altamente informativa: preferir aristas cortas suele construir buenas rutas por adyacencia de clientes. ACO explota esta regularidad desde la *construcción* misma del tour (regla de probabilidad ponderada por feromona y visibilidad) y refuerza *aristas* que aparecen en soluciones de alta calidad. Esta "memoria por aristas" encaja naturalmente con el dominio y acelera la intensificación sobre subestructuras útiles (*clusters*, cadenas cercanas), lo que se refleja en convergencia más estable y rápida (Blum y Roli, 2003; Dorigo y Stützle, 2004).

(ii) Codificación y operadores (efecto en AG). El AG implementado usa representación por

permutación con cruce *PMX* y mutación por *inversión*. PMX preserva posiciones relativas, pero no está orientado a *preservar aristas* de alta calidad; en TSP esto es crítico, pues la estructura del óptimo descansa en un conjunto pequeño de aristas buenas. En ausencia de operadores *edge-aware* (p. ej., *Edge Recombination, Edge Assembly Crossover*) o de una búsqueda local explícita (2–opt/3–opt) acoplada al AG, el algoritmo tiende a mezclar rutas rompiendo adyacencias útiles, requiriendo *más población e iteraciones* para recuperar esas aristas por mutación/selección. Esto explica que, en nuestros experimentos, el AG mejore más cuando se incrementa el presupuesto (300→3000), pero aún así quede por detrás de ACO en calidad media (Blum y Roli, 2003).

(iii) Dinámica exploración–explotación. Con $\beta \approx 2$ la visibilidad de ACO guía fuertemente la elección local; simultáneamente, una evaporación moderada (ρ en torno a 0.5) evita estancamiento reforzando de forma adaptativa las mejores aristas globales. Esta combinación favorece trayectorias de convergencia *rápidas y estables*, particularmente en grafos con estructura geográfica clara (ruteo urbano). En cambio, el AG con torneo $k{=}3$, p_c alto y p_m bajo puede sufrir *presión de selección* elevada y pérdida de diversidad si no se acompaña de operadores que preserven aristas o de búsqueda local sistemática, lo que retrasa la intensificación efectiva.

(iv) Sensibilidad a parámetros y presupuesto. Las gráficas de convergencia por iteraciones muestran que ACO es menos sensible al incremento de presupuesto: con 300 iteraciones ya alcanza valores competitivos (mejoras marginales al pasar a 3000), mientras que AG capitaliza mucho más el aumento de iteraciones y población, en especial con $n{=}60$. Esto sugiere que la configuración utilizada en ACO está mejor alineada con la estructura del problema, mientras que el AG requeriría *operadores más informados* o *hibridación* para cerrar la brecha.

Cómo verificar esta explicación (robustez).

- Ablaciones en AG: sustituir PMX por operadores orientados a aristas (p. ej., *ERX/EAX*) y acoplar 2–opt/3–opt en cada generación; comparar contra la configuración actual (Larrañaga et al., 1999).
- Tuning automático: ajustar (p_c, p_m, k, N) y (α, β, ρ) con *irace* en instancias locales.
- Escenarios de datos: repetir en grafos menos “métricos” (costos con asimetrías/ruido) para ver si la ventaja de ACO disminuye cuando η_{ij} pierde poder predictivo.

En conjunto, estos elementos explican por qué, con los parámetros y operadores aquí empleados, ACO supera a AG en nuestras instancias; y también señalan rutas concretas para reducir la brecha cuando el AG se hibrida con operadores y búsqueda local específicos del dominio.

Los resultados demuestran que:

- En instancias pequeñas (10 nodos), ambos algoritmos son efectivos, aunque ACO sigue ofreciendo ventajas marginales.
- En instancias medianas (60 nodos), ACO supera ampliamente a AG en calidad de solución, tiempo y estabilidad.
- AG es más sensible a los parámetros de población e iteración, mientras que ACO mantiene su rendimiento con menos ajustes.

Estas observaciones coinciden con lo reportado en estudios previos (Dorigo y Stützle, 2004; Stützle y Hoos, 2000b), y validan el uso de ACO como técnica robusta para la optimización de rutas en contextos reales.

Este hallazgo refuerza la solidez del ACO como metaheurística aplicable no solo a problemas discretos como el TSP, sino también a contextos más generales (Hernández-Ocaña et al., 2022).

5.5.1. Validación externa y comparación con pruebas de referencia

Con el propósito de realizar una validación externa de los algoritmos propuestos, se seleccionó la instancia `gr17` de la biblioteca TSPLIB, cuyo valor óptimo conocido es de 2085 unidades de distancia. Se ejecutaron tanto el Algoritmo Genético (AG) como el Algoritmo de Colonia de Hormigas (ACO), bajo las cuatro configuraciones experimentales definidas en la Sección 4.4, correspondientes a combinaciones de tamaño de población ($P \in \{20, 200\}$) y número de generaciones ($Iter \in \{300, 3000\}$). Para cada configuración se realizaron 25 corridas independientes, dando un total de 100 ejecuciones por algoritmo.

Los resultados obtenidos se resumen en el Cuadro 5.3. En él se presentan, para cada experimento, el mejor valor encontrado, el promedio, la desviación estándar, el peor valor y el número de corridas realizadas.

Tabla 5.3. Resultados de AG y ACO en la instancia TSPLIB-gr17

Experimento	Mejor	Media	Desv. Est.	Peor	Corridas
AG_Exp1	2342	3032.40	259.81	3521	25
AG_Exp2	2208	2384.88	125.70	2617	25
AG_Exp3	2223	2354.16	72.21	2504	25
AG_Exp4	2085	2114.36	37.48	2208	25
ACO_Exp1	2085	2104.52	22.30	2149	25
ACO_Exp2	2085	2098.12	15.68	2149	25
ACO_Exp3	2085	2085.36	1.80	2094	25
ACO_Exp4	2085	2085.84	4.20	2106	25

De la tabla se observa que ambos algoritmos alcanzaron el valor óptimo de la instancia en al menos una corrida, sin embargo, el ACO presentó una mayor estabilidad, reflejada en desviaciones estándar considerablemente menores, especialmente en los experimentos con mayor número de generaciones y hormigas (ACO_Exp3 y ACO_Exp4). En contraste, los resultados de AG mostraron mayor dispersión, siendo más sensibles a la configuración de parámetros: únicamente en el experimento AG_Exp4 (con mayor población e iteraciones) se alcanzó el valor óptimo.

En conclusión, para la instancia TSPLIB-gr17, el ACO se mostró más robusto y consistente que el AG, manteniendo resultados cercanos o iguales al óptimo en todas sus ejecuciones, mientras que el AG requirió configuraciones más exigentes para acercarse al valor óptimo.

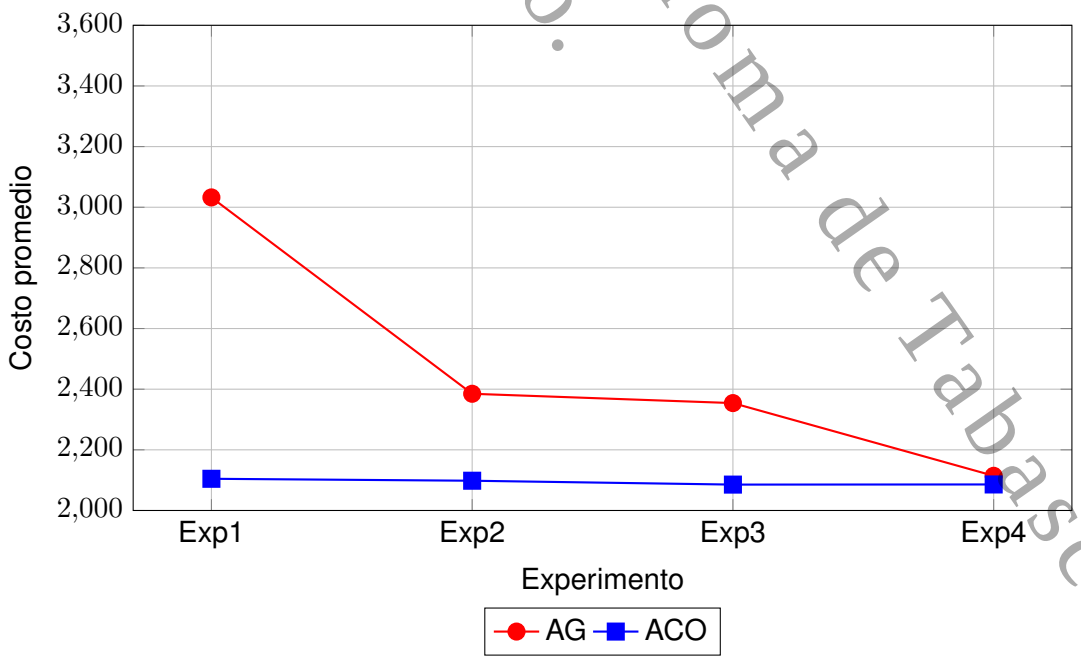


Figura 5.9. Evolución del costo promedio en AG y ACO para la instancia TSPLIB-gr17.

5.6. Comparación general entre AG y ACO

Tabla 5.4. Comparación cualitativa entre AG y ACO

Criterio	AG	ACO	Mejor
Calidad promedio de solución	Buena en problemas pequeños	Excelente en todos los escenarios	ACO
Estabilidad	Menor (mayor desviación estándar)	Mayor (menor desviación estándar)	ACO
Sensibilidad a parámetros	Alta (requiere ajuste fino)	Baja (se adapta bien con parámetros estándar)	ACO
Tiempo de ejecución estimado*	Mayor en escenarios grandes (más iteraciones)	Menor o similar, según configuración	ACO
Escalabilidad	Disminuye su rendimiento con más nodos	Mantiene calidad y estabilidad al escalar	ACO
Facilidad de implementación	Media (requiere operadores de cruce/mutación)	Alta (estructura más directa y controlada)	ACO

* Tiempo medido empíricamente en pruebas no cronometradas con 60 nodos y 3000 iteraciones.

Interpretación final:

El algoritmo de colonia de hormigas supera consistentemente al algoritmo genético en todos los aspectos relevantes del problema tratado: calidad de solución, estabilidad, escalabilidad y robustez frente a configuraciones. Aunque ambos algoritmos pueden resolver instancias pequeñas del TSP de manera eficaz, ACO ofrece un rendimiento superior y más confiable para aplicaciones prácticas en escenarios urbanos reales.

Capítulo 6

Conclusiones y trabajos futuros

6.1. Conclusión general

El objetivo de esta investigación fue diseñar e implementar un modelo computacional para la generación de rutas de reparto utilizando técnicas de optimización metaheurística, específicamente el Algoritmo Genético (AG) y el Algoritmo de Colonia de Hormigas (ACO). Para ello, se empleó una instancia del Problema del Agente Viajero (TSP) como una aproximación simplificada y controlada al Problema de Ruteo de Vehículos (VRP).

Los resultados obtenidos a partir de ocho configuraciones experimentales y veinticinco ejecuciones independientes por cada algoritmo permiten establecer que:

- ACO superó consistentemente a AG en términos de calidad de solución, especialmente en escenarios más complejos (60 nodos y 3000 iteraciones).
- La estabilidad de ACO fue superior, mostrando menor desviación estándar en todos los experimentos.
- AG tuvo un rendimiento aceptable únicamente en problemas pequeños, pero mostró mayor sensibilidad a parámetros como el tamaño de la población y el número de generaciones.
- Ambos algoritmos generaron soluciones viables, aunque ACO demostró ser más robusto, eficiente y escalable frente a escenarios de mayor complejidad.

En relación con la hipótesis de investigación, los resultados permiten validarla, ya que se comprobó que la implementación de un modelo computacional basado en metaheurísticas, y en particular mediante AG y ACO, posibilita mejorar la planeación operativa de una empresa de reparto a través de la reducción de la distancia total recorrida por su flota.

6.2. Aportes de la investigación

Este trabajo presenta varias contribuciones relevantes:

- La implementación de dos metaheurísticas clásicas en un entorno urbano real, con datos obtenidos del DENUE e integrados con servicios de mapas como Google Maps y MapQuest.
- Un diseño experimental riguroso, con múltiples escenarios y análisis estadístico formal (prueba de Mann–Whitney U).
- La validación del ACO como técnica eficiente y estable para el TSP, abriendo paso a su uso en escenarios de mayor complejidad logística.
- Un enfoque aplicado al contexto local (Villahermosa, Tabasco), que puede servir como referencia para empresas de reparto de la región.

6.3. Limitaciones del estudio

A pesar de los resultados positivos, esta tesis presenta algunas limitaciones:

- El problema resuelto fue un TSP clásico con un solo vehículo, sin considerar capacidades, ventanas de tiempo ni múltiples depósitos.
- La evaluación no incluyó tiempos de ejecución detallados, aunque se observó un desempeño aceptable en entorno local.
- La solución no se integró a un sistema de información geográfica (SIG) o interfaz gráfica interactiva.

Estas limitaciones no restan validez al modelo desarrollado, pero sí indican oportunidades de mejora.

6.4. Trabajos futuros

A partir de este estudio, se plantean las siguientes líneas de investigación y desarrollo:

1. Extender el modelo al VRP completo, incorporando múltiples vehículos, capacidades de carga, restricciones horarias y otras condiciones logísticas reales.
2. Incorporar otros enfoques metaheurísticos o híbridos, como recocido simulado (SA), búsqueda tabú o hibridaciones ACO+AG, para analizar su impacto en el rendimiento.
3. Desarrollar una interfaz de usuario o sistema web, donde empresas pequeñas puedan ingresar sus puntos de entrega y recibir rutas optimizadas de forma automática.
4. Incluir métricas de eficiencia energética o emisiones de CO_2, con el fin de alinear la optimización logística con objetivos de sostenibilidad.
5. Aplicar el modelo en otras ciudades o regiones, comparando cómo varía el rendimiento según la densidad urbana y la conectividad vial.

La presente investigación demuestra que las metaheurísticas son herramientas poderosas para resolver problemas logísticos reales. Su correcta aplicación, validación experimental y adaptación al contexto pueden traducirse en beneficios tangibles para empresas, ciudades y usuarios finales. Se espera que este trabajo sirva como base para futuros desarrollos en el área de optimización aplicada a la logística urbana en México y América Latina.

6.5. Recomendaciones prácticas para empresas locales

6.5.1. Antes de implementar: datos mínimos y preparación

Para que el ruteo entregue beneficios medibles, verifique que cuenta con:

- Direcciones geocodificadas (lat, lon) confiables.
- Matriz de costos (tiempo/distancia) consistente con la red vial local; idealmente precomputada y con *caché*.

- Capacidades por vehículo, tiempos de servicio y, si aplica,ventanas de tiempo.
- SLA (tiempos máximos/prioridades) para traducir en penalizaciones o restricciones.
- Histórico base (km, retrasos, % a tiempo) para comparar *antes/después*.

6.5.2. Elección de algoritmo según contexto

- Instancias pequeñas ($n \leq 30$) o poco esfuerzo computacional: ACO ofrece buena calidad con baja dispersión.
- Instancias medianas ($30 < n \leq 60$): ACO mantiene estabilidad; con mayor *esfuerzo computacional* (iteraciones/población), AG mejora sustantivamente.

6.5.3. Configuraciones recomendadas (guía rápida)

Tabla 6.1. Guía de configuración por tamaño y esfuerzo computacional.

Tamaño	Esfuerzo	Recomendado	Motivo
$n \leq 30$	bajo (plan rápido)	ACO (P=20, it=300)	Calidad estable con poco cómputo
$30 < n \leq 60$	medio	ACO (P=200, it=3000)	Mejora calidad manteniendo estabilidad
$30 < n \leq 60$	alto	AG (P=200, it=3000)	Ganancia considerable con más búsqueda
$n > 60$ (si aplica)	medio/alto	ACO (P=200, it=3000)	Robustez en urbano grande

Notas. Ajustes finos (*tuning*) pueden mejorar resultados locales; si el tiempo de cómputo es crítico, priorice configuraciones “bajas” con *caché* de matrices.

6.5.4. Integración tecnológica (bajo costo)

- Stack: Python 3.x; NumPy/Pandas; motor de ruteo (OSRM/GraphHopper o equivalente); DB ligera (SQLite/PostgreSQL).
- Matrices de costos: construir/actualizar fuera de hora pico; *caché* por cuadrantes urbanos; degradar a distancia euclidiana si falla la API.
- Interfaz: CSV simples para pedidos/vehículos; tablero con KPIs (Sec. 6.5.5) para seguimiento.

6.5.5. Métricas y tablero (KPIs)

- KM totales por día/ruta (objetivo: ↓).
- % de entregas a tiempo y retraso promedio (min).
- Uso de capacidad (volumen/peso por vehículo).
- Tiempo total de ruta y paradas/hora.
- Variabilidad interdiaria (estabilidad del plan).

6.5.6. Mantenimiento y robustez

- Actualización de datos: refrescar geocodificación y *caché* semanalmente o cuando cambie la red vial.
- Tuning periódico: revisar parámetros mensualmente o ante cambios de demanda.
- Plan de contingencia: si falla la API o sube la latencia, usar *caché*, reducir iteraciones y replanear con heurísticas rápidas.

6.5.7. Calcular Costo–beneficio

1. Estime el baseline (3–4 semanas): km, horas, % a tiempo.
2. Haga un piloto de 2 semanas con el ruteo propuesto (misma demanda).
3. Calcule Δkm, Δtiempo, Δ % a tiempo; valore ahorro de combustible/horas vs. costo de cómputo/persona.
4. Si $\Delta\text{km} \geq 5$–10 % y/o mejora en % a tiempo, escale gradualmente.

6.5.8. Checklist de adopción

1. Datos listos (clientes, vehículos, tiempos, direcciones validadas).
2. Matriz de costos vigente y con *caché*.

3. KPIs definidos y tablero básico.

4. Elección de algoritmo y esfuerzo según la Tabla 6.1.

5. Piloto con comparación *antes/después* y decisión de escalamiento.

6.5.9. Reflexión final sobre aplicabilidad y escalabilidad

Si bien los experimentos y validaciones se realizaron con datos urbanos de Villahermosa, la aplicabilidad del modelo no se limita a esta ciudad. La lógica de las metaheurísticas (AG y ACO) descansa en principios generales de exploración y explotación de soluciones que son *independientes del contexto geográfico*. Por tanto, el modelo es transferible a otras ciudades de Tabasco e incluso a entornos metropolitanos más grandes, siempre que se cuente con: (i) datos de localización confiables, (ii) matrices de costos actualizadas y (iii) capacidad de procesamiento acorde al tamaño del problema.

En cuanto a la escalabilidad, los resultados muestran que el costo computacional crece con el número de nodos (n), pero también que la calidad de las soluciones mejora de manera consistente al aumentar el *esfuerzo computacional*. Esto implica que en escenarios de mayor escala (p. ej., empresas de reparto en ciudades como Mérida, Puebla o CDMX), el modelo sigue siendo útil, aunque requerirá ajustar los parámetros de población/iteraciones o incorporar técnicas híbridas para mantener tiempos de respuesta aceptables.

Finalmente, el caso de Villahermosa sirve como *laboratorio de referencia*, pero la aportación real del trabajo es metodológica: un esquema de ruteo flexible, adaptable a diversos tamaños de operación y escalable a diferentes regiones del país, siempre que se integren fuentes de datos confiables y se ajusten los parámetros de ejecución a la realidad operativa de cada empresa.

Alojamiento de la Tesis en el Repositorio Institucional	
Título de la tesis:	Generación de Rutas de Reparto a Partir de Metaheurística de Optimización
Autor:	Romeo Valencia Gómez
ORCID:	https://orcid.org/0009-0002-8222-9185
Resumen:	El presente trabajo muestra la implementación de un modelo de optimización para la generación de rutas de reparto urbanas, mediante la aplicación de técnicas metaheurísticas: el Algoritmo Genético (AG) y el Algoritmo de Colonia de Hormigas (ACO). Para ello, se abordó el Problema del Agente Viajero (TSP) como aproximación inicial al Problema de Ruteo de Vehículos (VRP), utilizando datos reales obtenidos de establecimientos comerciales en Villahermosa, Tabasco, a partir del Directorio Estadístico Nacional de Unidades Económicas (DENUE) del INEGI. Las distancias entre puntos se calcularon a través de APIs de Google Maps y MapQuest, generando matrices simétricas de entrada. Ambos algoritmos fueron implementados en Python y evaluados bajo ocho configuraciones experimentales, variando el número de nodos, población e iteraciones. Se utilizó como métrica principal la distancia total recorrida. La evaluación estadística se realizó mediante la prueba no paramétrica de Mann–Whitney U, que permitió comparar la calidad y estabilidad de las soluciones entre algoritmos. Los resultados obtenidos mostraron que el ACO superó consistentemente al AG en calidad promedio de las rutas, desviación estándar y escalabilidad. Como conclusión general, se valida la hipótesis de investigación: el uso de metaheurísticas, en particular el algoritmo ACO, permite mejorar la planeación operativa de una empresa de reparto, reduciendo significativamente la distancia recorrida. Se recomienda extender este modelo a escenarios más complejos como el VRP con múltiples vehículos, restricciones logísticas y condiciones dinámicas del entorno.
Palabras clave:	ruteo de vehículos, TSP, algoritmos genéticos, colonia de hormigas, optimización combinatoria, metaheurísticas, logística urbana.
Referencias citadas:	En la siguiente página se muestran las referencias.

Bibliografía

Aldoraibi, R. I., Alanazi, F., Alaskar, H., & Alanazi, A. (2024). Optimising Delivery Routes Under Real-World Constraints: A Comparative Study of Ant Colony, Particle Swarm and Genetic Algorithms. *International Journal of Advanced Computer Science and Applications*, *15*(10). https://doi.org/10.14569/IJACSA.2024.0151081

Bäck, T., Fogel, D. B., & Michalewicz, Z. (2000). *Evolutionary Computation 1: Basic Algorithms and Operators*. Institute of Physics Publishing.

Blickle, T., & Thiele, L. (1996). A Comparison of Selection Schemes Used in Evolutionary Algorithms. *Evolutionary Computation*, *4*(4), 361-394. https://doi.org/10.1162/evco.1996.4.4.361

Blum, C., & Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys*, *35*(3), 268-308. https://doi.org/10.1145/937503.937505

Chen, G., Gao, J., & Chen, D. (2025). Research on Vehicle Routing Problem with Time Windows Based on Improved Genetic Algorithm and Ant Colony Algorithm. *Electronics*, *14*(4), 647. https://doi.org/10.3390/electronics14040647

Coello, 2. C. A. C. (Ed.). (2019). *Computación evolutiva* (2.ª ed.). Academia Mexicana de Computación, A.C.

Dantzig, G. B., Fulkerson, D. R., & Johnson, S. M. (1954). Solution of a large-scale traveling-salesman problem. *Journal of the Operations Research Society of America*, *2*(4), 393-410. https://doi.org/10.1287/opre.2.4.393

Developers, G. (2023). Google Maps Platform documentation [Retrieved from https://developers.google.com/maps].

Dorigo, M., & Di Caro, G. (1999). The Ant Colony Optimization Meta-Heuristic. En *New Ideas in Optimization* (pp. 11-32). McGraw-Hill.

Dorigo, M., & Gambardella, L. M. (1997). Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. *IEEE Transactions on Evolutionary Computation*, *1*(1), 53-66.

Dorigo, M., & Stützle, T. (2004). *Ant Colony Optimization*. MIT Press.

Eiben, A., & Smith, J. (2003). *Introduction to Evolutionary Computing*. Springer.

Gendreau, M., & Potvin, J.-Y. (2010). *Handbook of Metaheuristics* (2nd). Springer.

Gibbons, J. D., & Chakraborti, S. (2011). *Nonparametric Statistical Inference* (5th). CRC Press.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley.

Goldberg, D., Deb, K., & Clark, J. (1992). Genetic Algorithms, Noise, and the Sizing of Populations. *Complex Systems*, *6*, 333-362.

Harris, C. R., Millman, K. J., van der Walt, S. J., & et al. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357-362. https://doi.org/10.1038/s41586-020-2649-2

Hernández-Ocaña, B., Hernández-Torruco, J., Chávez-Bosquez, O., & Montané-Jiménez, L. G. (2022). Análisis Comparativo de los Algoritmos Basados en Abejas y Hormigas en el Problema de la Esfera. *Revista Politécnica*, *50*(2), 55-62.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press.

INEGI. (2024). Producto Interno Bruto por Entidad Federativa (PIBE) 2023: Tabasco [Consultado en agosto de 2025]. https://www.inegi.org.mx/contenidos/saladeprensa/boletines/2024/PIBEF/PIBEF2023_Tab.pdf

INEGI. (2025). Valor Agregado Bruto del Comercio Electrónico 2023 (VABCOEL) [Consultado en agosto de 2025]. https://www.inegi.org.mx/contenidos/saladeprensa/boletines/2025/vabcoel/vabcoel2025_02.pdf

Katoch, S., Chauhan, S., & Kumar, V. (2020). A Review on Genetic Algorithm: Past, Present, and Future. *Complex & Intelligent Systems*, *6*, 263-289.

Ky Phuc, P. N., & Phuong Thao, N. L. (2021). Ant Colony Optimization for Multiple Pickup and Multiple Delivery Vehicle Routing Problem with Time Window and Heterogeneous Fleets. *Logistics*, *5*(2), 28. https://doi.org/10.3390/logistics5020028

Laporte, G. (2009). Fifty years of vehicle routing. *Transportation Science*, *43*(4), 408-416. https://doi.org/10.1287/trsc.1090.0301

Larrañaga, P., Kuijpers, C. M. H., Murga, R. H., Inza, I., & Dizdarevic, S. (1999). Genetic Algorithms for the Travelling Salesman Problem: A Review of Representations and Operators. *Artificial Intelligence Review*, *13*(2), 129-170. https://doi.org/10.1023/A:1006529012972

Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A. H. G., & Shmoys, D. B. (1985). *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. Wiley.

López-Ibáñez, M., Dubois-Lacoste, J., Stützle, T., & Birattari, M. (2016). The irace package: Iterated Racing for Automatic Algorithm Configuration. *Operations Research Perspectives*, *3*, 43-58.

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

MapQuest. (2023). MapQuest Developer Documentation [Retrieved from https://developer.mapquest.com].

Michalewicz, Z. (1996). *Genetic Algorithms + Data Structures = Evolution Programs* (3rd). Springer.

Miller, C. E., Tucker, A. W., & Zemlin, R. A. (1960). Integer Programming Formulation of Traveling Salesman Problems. *Journal of the ACM*, *7*(4), 326-329.

Montoya-Torres, J. R., López Franco, J., Nieto Isaza, S., Felizzola Jiménez, H., & Herazo-Padilla, N. (2015). Modelo de ruteo para la distribución urbana de mercancías: revisión sistemática de literatura. *DYNA*, *82*(191), 206-215. https://doi.org/10.15446/dyna.v82n191.46158

OECD. (2020). E-commerce in the time of COVID-19 [Recuperado de https://www.oecd.org/coronavirus/policy-responses/e-commerce-in-the-time-of-covid-19-3a2b78e8/].

Papadimitriou, C. H., & Steiglitz, K. (1998). *Combinatorial Optimization: Algorithms and Complexity*. Dover Publications.

Potvin, J.-Y. (1996). Genetic algorithms for the traveling salesman problem. *Annals of Operations Research*, *63*(3), 337-370.

Ran, L., Ran, S., & Meng, C. (2023). Green city logistics path planning and design based on genetic algorithm. *PeerJ Computer Science*, *9*, e1347. https://doi.org/10.7717/peerj-cs.1347

Reinelt, G. (1994). *The Traveling Salesman: Computational Solutions for TSP Applications*. Springer.

Stützle, T., & Hoos, H. H. (2000a). MAX–MIN Ant System. *Future Generation Computer Systems*, *16*(8), 889-914.

Stützle, T., & Hoos, H. H. (2000b). MAX–MIN Ant System. *Future Generation Computer Systems*, *16*(8), 889-914.

Talbi, E.-G. (2009). *Metaheuristics: From Design to Implementation*. Wiley.

Tan, L., Zhu, K., & Yi, J. (2024). Vehicle Route Planning of Diverse Cargo Types in Urban Logistics Based on Enhanced Ant Colony Optimization. *World Electric Vehicle Journal*, *15*(9), 405. https://doi.org/10.3390/wevj15090405

Toth, P., & Vigo, D. (2014). *Vehicle Routing: Problems, Methods, and Applications* (2nd). SIAM.

UNCTAD. (2021). COVID-19 and E-commerce: A Global Review. *United Nations Conference on TradeDevelopment*.

Van Rossum, G., & Drake Jr., F. L. (2009). *The Python Language Reference Manual*. Network Theory Ltd.

Anexos

Código ilustrativo de las metaheurísticas

Algoritmo Genético (AG): fragmento de configuración y bucle

```
1
2  def principal(nPoblacion, nGeneraciones, nCiudades): #Modulo que ejecuta el algoritmo con
       metodos ya definidos
3      global distancias
4      global Pc #Probabilidad de Cruce
5      global Pm #Probablidad de Mutacion
6      global Tt #Tamano del Torneo
7
8      pSeleccion=0.6
9      pCruce=0.8
10     pMutacion=0.01
11
12     pConvergencia=nGeneraciones
13     TamTorneo=3
14
15     n_seleccion=4
16     n_cruce=1
17     n_mutacion=4
18
19     Lc=nCiudades-1 # Longitud de Cromosomas de cada individuo #Maximo de direcciones: 69
20     N=nPoblacion #Poblacion de cada generacion
21     Ps=pSeleccion #Probabilidad de seleccion
22     Pc=pCruce #Probabilidad de Cruce
23     Pm=pMutacion #Probabilidad de Mutacion
24     Ng=nGeneraciones #Numero de generaciones
25     convergencia=pConvergencia #Numero de Generaciones sin cambio como condicion de paro
26     Tt=TamTorneo #Tamano del Torneo
27     distancias=carga_matriz_distancia(Lc)
28
29     ciudades=genera_ciudades(Lc)
30     ranking=[]
31     datos=[]
32     poblacioninicial=genera_poblacion(ciudades,N)
33     print("Espere un momento, construyendo las generaciones...")
34     inicioTime=0
35     FinTime=0
36     Contador=0
37     inicioTime=time.time()
```

```
    poblacion=poblacioninicial
    mejor=poblacioninicial[0]

    for generacion_id in range(Ng):
        padres_seleccionados=seleccion(poblacion,Ps,n_seleccion)
        nuevos_hijos=cruza(padres_seleccionados,N,n_cruce)
        hijos_mutados=mutate(nuevos_hijos,n_mutacion)
        poblacion=padres_seleccionados+hijos_mutados

        mejor_ruta=sorted(poblacion, key=lambda ruta: ruta.distance)[0]
        mejor_ruta.set_generacion(generacion_id)
        mejor_ruta.set_operadores(n_seleccion,n_cruce,n_mutacion)
        ranking.append(mejor_ruta)

        if mejor.distance<=mejor_ruta.distance:
            Contador+=1
        else:
            Contador=0
            mejor=mejor_ruta
        datos.append(mejor.distance)

        if Contador==convergencia:
            break

    return sorted(ranking, key=lambda ruta: ruta.distance)[0]

def main():
     #Configuracion del Experimento
    #NoExp="_Exp1"; nPoblacion=20; nGeneraciones=300; nCiudades=10;
    #NoExp="_Exp2"; nPoblacion=20; nGeneraciones=300; nCiudades=60;
    #NoExp="_Exp3"; nPoblacion=20; nGeneraciones=3000; nCiudades=10
    #NoExp="_Exp4"; nPoblacion=20; nGeneraciones=3000; nCiudades=60
```

Colonia de Hormigas (ACO): clase y configuración de experimento

```
class AntColony(object):

    def __init__(self, distances, n_ants, n_best, n_iterations, decay, alpha=1, beta=1):
        self.distances = distances
        self.pheromone = np.ones(self.distances.shape) / len(distances)
        self.all_inds = range(len(distances))
        self.n_ants = n_ants
        self.n_best = n_best
        self.n_iterations = n_iterations
```

```
        self.decay = decay
        self.alpha = alpha
        self.beta = beta

    def run(self):
        distance_logs=[]
        shortest_path = None
        all_time_shortest_path = ("placeholder", [np.inf,0])
        for i in range(self.n_iterations):
            all_paths = self.gen_all_paths()
            self.spread_pheronome(all_paths, self.n_best, shortest_path=shortest_path)
            shortest_path = min(all_paths, key=lambda x: x[1])
            if shortest_path[1][0] < all_time_shortest_path[1][0]:
                all_time_shortest_path = shortest_path
            distance_logs.append(all_time_shortest_path[1][0])
        return all_time_shortest_path,distance_logs

    def spread_pheronome(self, all_paths, n_best, shortest_path):
        sorted_paths = sorted(all_paths, key=lambda x: x[1][0])
        for path, dist in sorted_paths[:n_best]:
            for move in path:
                self.pheromone[move] += 1.0 / self.distances[move]

    def gen_path_dist(self, path):
        total_dist = 0
        distancias_ind=[]

        for ele in path:
            total_dist += self.distances[ele]
            distancias_ind.append(self.distances[ele])

        return total_dist, distancias_ind

    def gen_all_paths(self):
        all_paths = []
        for i in range(self.n_ants):
            path = self.gen_path(0)
            all_paths.append((path, self.gen_path_dist(path)))
        return all_paths

    def gen_path(self, start):
        path = []
        visited = set()
        visited.add(start)
```

```
        prev = start
        for i in range(len(self.distances) - 1):
            move = self.pick_move(self.pheromone[prev], self.distances[prev], visited)
            path.append((prev, move))
            prev = move
            visited.add(move)
        path.append((prev, start))
        return path

    def pick_move(self, pheromone, dist, visited):
        pheromone = np.copy(pheromone)
        pheromone[list(visited)] = 0

        row = (pheromone ** self.alpha) * (( 1.0 / dist) ** self.beta)

        norm_row = row / row.sum()
        move = np_choice(self.all_inds, 1, p=norm_row)[0]
        return move

def carga_matriz_distancia(num_Direcciones):
    matriz=pd.DataFrame(pd.read_csv("TSP_Data/Matriz_Distancias70.csv"))
    return matriz.iloc[:num_Direcciones,:num_Direcciones]


def principal(n_ants, n_iteration, tour_size):

    n_best=10
    decay=0.5
    alpha=1
    beta=2
```

```
    print("Dejemos que las hormigas hagan su trabajo...")
    ant_colony = AntColony(distances, n_ants, n_best, n_iteration, decay, alpha, beta)
    shortest_path,log = ant_colony.run()

    return sorted(log)[0], shortest_path


def main():
    #Configuracion del Experimento
    #NoExp="_Exp1"; nPoblacion=20; nGeneraciones=300; nCiudades=10;
    #NoExp="_Exp2"; nPoblacion=20; nGeneraciones=300; nCiudades=60;
    #NoExp="_Exp3"; nPoblacion=20; nGeneraciones=3000; nCiudades=10
    #NoExp="_Exp4"; nPoblacion=20; nGeneraciones=3000; nCiudades=60
    #NoExp="_Exp5"; nPoblacion=200; nGeneraciones=300; nCiudades=10
```

```
    #NoExp="_Exp6"; nPoblacion=200; nGeneraciones=300; nCiudades=60
    #NoExp="_Exp7"; nPoblacion=200; nGeneraciones=3000; nCiudades=10
    NoExp="_Exp8"; nPoblacion=200; nGeneraciones=3000; nCiudades=60
    resultados=[]
```

```
    archivo="Resultados\Resultados_ACO"+NoExp
    toExportar.to_csv(archivo+".csv",encoding="utf8")

if __name__ == "__main__":
  main()
```

Configuración de referencia usada en los experimentos

- **GA** (`codigo/V2_agenteviajero_Pruebas_sanitizado.py`): torneo determinista ($T = 3$), cruce PMX, mutación por inversión; $p_{sel} = 0.6$, $p_c = 0.8$, $p_m = 0.01$; criterio de convergencia por generaciones; instancia $n = 60$.
- **codigo/ACO** (`codigo/ACO.py`): $n\_best = 10$, *decay* $= 0.5$, $\alpha = 1$, $\beta = 2$; diagonal $= \infty$; *Exp8* con $nPoblacion = 200$, $nGeneraciones = 3000$, $nCiudades = 60$ y 25 corridas por experimento.