TECNOLÓGICO NACIONAL DE MÉXICO

NOMBRE DE LA MATERIA: Tópicos de Inteligencia Artificial

Proyecto de investigación: Optimización de rutas de distribución mediante Recocido Simulado en Culiacán

ALUMNOS: García Sánchez Sergio Jesús
Corrales Palazuelos Darío

CARRERA: Ingeniería en Sistemas Computacionales

NOMBRE DEL MAESTRO: Zuriel Dathan Mora Félix

Optimización de rutas de distribución mediante Recocido Simulado en Culiacán, Sinaloa

1. Introducción

La distribución eficiente de productos comestibles en cadenas de restaurantes o tiendas de conveniencia es un desafío logístico que involucra múltiples variables: costos de combustible, distancias, tiempos de entrega y disponibilidad de centros de distribución. En la ciudad de Culiacán, Sinaloa, la dispersión de las sucursales y los patrones de tráfico hacen que la planificación de rutas sea una tarea compleja.

Para abordar este problema, se propone una solución basada en algoritmos heurísticos, específicamente el método de Recocido Simulado. Este método permite encontrar configuraciones cercanas al óptimo global en problemas combinatorios donde los métodos exactos resultan computacionalmente costosos o inviables. El objetivo de este proyecto es optimizar las rutas de transporte entre centros de distribución y tiendas, reduciendo el costo total asociado al consumo de combustible y distancias recorridas.

2. Objetivo General

Diseñar e implementar una solución computacional basada en el algoritmo heurístico de Recocido Simulado para optimizar las rutas de distribución de productos comestibles desde los centros de distribución hacia las sucursales de una cadena de restaurantes en Culiacán, Sinaloa, mejorando la eficiencia logística y reduciendo los costos operativos.

3. Objetivos Específicos

- Analizar el problema logístico de distribución de productos comestibles en la zona de estudio.
- Modelar el problema como un caso de optimización combinatoria.
- Seleccionar y justificar el uso del algoritmo heurístico Recocido Simulado.
- Desarrollar una implementación computacional en Python.
- Simular escenarios de distribución con datos reales o ficticios.
- Evaluar el desempeño de la solución mediante métricas de costo y eficiencia.
- Documentar el proceso completo del proyecto y las dificultades enfrentadas.

4. Justificación

En la logística moderna, los costos de transporte representan una fracción significativa del gasto operativo. Implementar un método heurístico como el Recocido Simulado permite obtener soluciones eficientes en menor tiempo, aprovechando su capacidad para escapar de óptimos locales. Este enfoque ofrece una alternativa práctica frente a los algoritmos exactos que requieren más recursos computacionales.

Además, durante el desarrollo del proyecto se identificó una dificultad inicial: el mapa geoespacial proporcionado por el docente tenía errores en las coordenadas, lo que impedía visualizar correctamente las ubicaciones. Para resolverlo, se implementó un script en Python utilizando la librería Folium, generando un mapa interactivo con los centros de distribución en color rojo y las tiendas en azul.

5. Alcance

El proyecto se enfoca en la optimización de rutas entre centros de distribución y tiendas localizadas en Culiacán, Sinaloa. Se consideran únicamente los costos de transporte basados en la distancia y el consumo de combustible. No se incluyen restricciones de capacidad vehicular ni ventanas de tiempo. Los resultados se representan mediante rutas optimizadas y un mapa interactivo en formato HTML, generado con Folium.

6. Desarrollo

El problema se modeló como un caso de optimización combinatoria, donde los nodos representan centros de distribución (CDs) y tiendas, mientras que las aristas representan los costos de desplazamiento entre ellos, calculados con base en distancias y consumo de combustible. El objetivo es minimizar el costo total de transporte, garantizando que cada tienda sea atendida por un solo centro y que la ruta de cada vehículo comience y termine en su respectivo CD.

Esta situación corresponde a una variación del problema del viajero (TSP) aplicado a múltiples orígenes, donde la complejidad crece exponencialmente con el número de nodos. Debido a esta complejidad, los métodos exactos resultan inviables, por lo que se recurrió a una técnica heurística que permite obtener soluciones de alta calidad en un tiempo razonable.

Selección y explicación del algoritmo heurístico

Para resolver el problema, se seleccionó el algoritmo de Recocido Simulado una metaheurística inspirada en el proceso físico de enfriamiento de metales. Este método es especialmente útil en problemas de optimización combinatoria, ya que permite escapar de óptimos locales mediante la aceptación probabilística de soluciones peores durante las etapas iniciales del proceso.

El funcionamiento del algoritmo se basa en los siguientes principios:

Solución inicial: se genera una ruta aleatoria que visita todas las tiendas asignadas a un CD.

Vecindad: en cada iteración se crea una nueva ruta vecina intercambiando dos tiendas.

Evaluación de costo: se calcula el costo de la nueva ruta y se compara con el de la actual.

Criterio de aceptación: si la nueva ruta mejora el costo, se acepta. Si no, puede aceptarse con una probabilidad dependiente de la temperatura (T) y de la diferencia de costos.

Enfriamiento: la temperatura disminuye gradualmente, reduciendo la probabilidad de aceptar soluciones peores.

Convergencia: al final del proceso, el sistema tiende hacia una solución estable de costo mínimo.

Esta metodología equilibra la exploración global al inicio y la explotación local al final, logrando rutas altamente optimizadas sin requerir un cálculo exhaustivo de todas las combinaciones posibles.

Implementación computacional

La implementación se realizó en Python, utilizando las librerías pandas, numpy, math y matplotlib. Estas herramientas facilitaron la lectura de los datos desde archivos Excel, el manejo de matrices de costos, la generación de soluciones y la visualización de los resultados.

Además, se incorporó folium para la generación de un mapa interactivo que muestra los centros de distribución y las tiendas sobre la ciudad de Culiacán, permitiendo verificar la coherencia geográfica de los datos.

El siguiente fragmento muestra el código utilizado para generar el mapa correcto de centros de distribución y tiendas, reemplazando el mapa erróneo proporcionado originalmente:

```
import pandas as pd
import folium

# Cargar el archivo Excel
ruta = "datos_distribucion_tiendas.xlsx"
df = pd.read_excel(ruta)

# Crear el mapa centrado en el promedio de las coordenadas
centro_mapa = [df["Latitud_WGS84"].mean(), df["Longitud_WGS84"].mean()]
mapa = folium.Map(location=centro_mapa, zoom_start=10)

# Agregar los marcadores según el tipo
for _, fila in df.iterrows():
    # Si el tipo contiene la palabra 'Distribución', será rojo; si no,
azul
    color = "red" if "Distribución" in fila["Tipo"] else "blue"

    folium.CircleMarker(
        location=[fila["Latitud_WGS84"], fila["Longitud_WGS84"]],
        radius=6,
        color=color,
        fill=True,
        fill_color=color,
        popup=fila["Nombre"]  # Se muestra al hacer clic en el punto
    ).add_to(mapa)

# Guardar el mapa en un archivo HTML interactivo
mapa.save("mapa_distribucion_tiendas.html")

print("Mapa generado correctamente: mapa_distribucion_tiendas.html")
```

Este código permitió representar correctamente los puntos de distribución y tiendas en el mapa de Culiacán, asegurando la validez geográfica de los datos utilizados en el algoritmo.

Implementación del algoritmo de optimización

El siguiente código presenta la implementación completa del algoritmo de Recocido Simulado, encargado de asignar y optimizar las rutas de entrega:

```
import pandas as pd
import numpy as np
import math
import random
import matplotlib.pyplot as plt

def cargar_datos(archivo_costos, archivo_nodos):
    """
    Carga la matriz de costos y los datos de los nodos desde archivos
Excel.
    Crea listas con los índices de CDs y tiendas.
    """
    matriz_costos_df = pd.read_excel(archivo_costos, header=None)
    matriz_costos_df = matriz_costos_df.apply(pd.to_numeric,
errors='coerce').fillna(0)
    matriz_costos = matriz_costos_df.to_numpy()

    nodos_df = pd.read_excel(archivo_nodos)
    nombres_nodos = list(nodos_df['Nombre'])
    cds = [i for i, nombre in enumerate(nombres_nodos) if 'Centro de
Distribución' in nombre]
    tiendas = [i for i, nombre in enumerate(nombres_nodos) if 'Tienda' in
nombre]

    print(f"Datos cargados correctamente: {len(cds)} Centros de
Distribución y {len(tiendas)} tiendas.")
    return matriz_costos, nombres_nodos, cds, tiendas

def calcular_costo_ruta(ruta, matriz_costos):
    """
    Calcula el costo total de una ruta sumando los costos entre nodos
consecutivos.
    """
    costo_total = 0
    for i in range(len(ruta) - 1):
        costo_total += matriz_costos[ruta[i]][ruta[i + 1]]
    return costo_total
```

```
def generar_solucion_inicial(cd, tiendas_asignadas):
    """
    Crea una ruta inicial que parte del CD, visita todas sus tiendas y
regresa.
    """
    ruta = [cd] + random.sample(tiendas_asignadas, len(tiendas_asignadas))
+ [cd]
    return ruta

def generar_vecino(ruta):
    """
    Crea una solución vecina intercambiando dos tiendas aleatorias.
    """
    nueva_ruta = ruta[:]
    idx1, idx2 = random.sample(range(1, len(ruta) - 1), 2)
    nueva_ruta[idx1], nueva_ruta[idx2] = nueva_ruta[idx2],
nueva_ruta[idx1]
    return nueva_ruta

def recocido_simulado(matriz_costos, cd, tiendas_asignadas, temp_inicial,
tasa_enfriamiento, num_iteraciones):
    """
    Ejecuta el algoritmo de recocido simulado para optimizar la ruta de un
CD.
    """
    solucion_actual = generar_solucion_inicial(cd, tiendas_asignadas)
    costo_actual = calcular_costo_ruta(solucion_actual, matriz_costos)
    mejor_solucion = solucion_actual
    mejor_costo = costo_actual
    temperatura = temp_inicial
    historial = [costo_actual]

    for i in range(num_iteraciones):
        vecino = generar_vecino(solucion_actual)
        costo_vecino = calcular_costo_ruta(vecino, matriz_costos)
        delta = costo_vecino - costo_actual

        if delta < 0 or random.random() < math.exp(-delta / temperatura):
            solucion_actual = vecino
            costo_actual = costo_vecino

        if costo_actual < mejor_costo:
            mejor_solucion = solucion_actual
            mejor_costo = costo_actual
```

```
        temperatura *= tasa_enfriamiento
          historial.append(mejor_costo)

          if (i + 1) % 5000 == 0:
              print(f"CD {cd} | Iteración {i+1}/{num_iteraciones} | Mejor
costo: {mejor_costo:.2f}")

    return mejor_solucion, mejor_costo, historial

def graficar_convergencia(historial_global):
    """
    Grafica la convergencia promedio de todos los CDs.
    """
    plt.figure(figsize=(12, 6))
    plt.plot(historial_global, color='dodgerblue', linewidth=2)
    plt.title('Evolución del costo promedio (todos los CDs)', fontsize=16)
    plt.xlabel('Iteración', fontsize=12)
    plt.ylabel('Costo promedio', fontsize=12)
    plt.grid(True, linestyle='--', alpha=0.6)
    plt.savefig('grafico_convergencia_global.png')
    print("\nGráfico global guardado como
'grafico_convergencia_global.png'")

# --- PROGRAMA PRINCIPAL ---
if __name__ == '__main__':
    TEMP_INICIAL = 10000
    TASA_ENFRIAMIENTO = 0.999
    NUM_ITERACIONES = 30000

    archivo_matriz_costos = 'matriz_costos_combustible.xlsx'
    archivo_nodos_info = 'datos_distribucion_tiendas.xlsx'

    matriz_costos, nombres, cds, tiendas =
cargar_datos(archivo_matriz_costos, archivo_nodos_info)

    # --- Asignar tiendas a cada CD de forma equitativa ---
    asignaciones = np.array_split(tiendas, len(cds))
    resultados = []
    historial_global = []

    for idx, cd in enumerate(cds):
        tiendas_asignadas = list(asignaciones[idx])
        print(f"\nOptimizando rutas para {nombres[cd]} con
{len(tiendas_asignadas)} tiendas asignadas...")
```

```
        mejor_ruta, mejor_costo, historial = recocido_simulado(
            matriz_costos, cd, tiendas_asignadas,
            TEMP_INICIAL, TASA_ENFRIAMIENTO, NUM_ITERACIONES
        )

        resultados.append((nombres[cd], mejor_costo, mejor_ruta))
        if len(historial_global) < len(historial):
            historial_global = historial
        else:
            historial_global = [min(h1, h2) for h1, h2 in
zip(historial_global, historial)]

    # --- Mostrar resultados finales ---
    print("\n" + "="*40)
    print("   RESULTADOS FINALES DE OPTIMIZACIÓN")
    print("="*40)
    costo_total = 0
    for nombre_cd, costo, ruta in resultados:
        costo_total += costo
        print(f"\n{nombre_cd}:")
        ruta_nombres = " -> ".join([nombres[i] for i in ruta])
        print(f"Ruta óptima: {ruta_nombres}")
        print(f"Costo total: {costo:.2f}")

    print(f"\nCosto total global optimizado: {costo_total:.2f}")
    graficar_convergencia(historial_global)
```

El algoritmo divide equitativamente las tiendas entre los centros de distribución y aplica el recocido simulado para cada uno, minimizando el costo total de transporte. El proceso registra los costos por iteración y genera un gráfico que muestra la convergencia global del sistema.

Resultados

Tras ejecutar el algoritmo, se obtuvieron los siguientes resultados de optimización, donde cada centro de distribución tiene su propia ruta optimizada y un costo total reducido significativamente:

```
Optimizando rutas para Centro de Distribución 1 con 9 tiendas
asignadas...
```

CD 0 | Iteración 5000/30000 | Mejor costo: 4.58

CD 0 | Iteración 10000/30000 | Mejor costo: 1.67

CD 0 | Iteración 15000/30000 | Mejor costo: 1.67

CD 0 | Iteración 20000/30000 | Mejor costo: 1.67

CD 0 | Iteración 25000/30000 | Mejor costo: 1.67

CD 0 | Iteración 30000/30000 | Mejor costo: 1.67

Optimizando rutas para Centro de Distribución 2 con 9 tiendas asignadas...

CD 1 | Iteración 5000/30000 | Mejor costo: 5.29

CD 1 | Iteración 10000/30000 | Mejor costo: 4.05

CD 1 | Iteración 15000/30000 | Mejor costo: 4.05

CD 1 | Iteración 20000/30000 | Mejor costo: 4.05

CD 1 | Iteración 25000/30000 | Mejor costo: 4.05

CD 1 | Iteración 30000/30000 | Mejor costo: 4.05

Optimizando rutas para Centro de Distribución 3 con 9 tiendas asignadas...

CD 2 | Iteración 5000/30000 | Mejor costo: 6.23

CD 2 | Iteración 10000/30000 | Mejor costo: 4.75

CD 2 | Iteración 15000/30000 | Mejor costo: 4.75

CD 2 | Iteración 20000/30000 | Mejor costo: 4.75

CD 2 | Iteración 25000/30000 | Mejor costo: 4.75

CD 2 | Iteración 30000/30000 | Mejor costo: 4.75

Optimizando rutas para Centro de Distribución 4 con 9 tiendas asignadas...

CD 3 | Iteración 5000/30000 | Mejor costo: 5.16

CD 3 | Iteración 10000/30000 | Mejor costo: 3.47

CD 3 | Iteración 15000/30000 | Mejor costo: 3.47

CD 3 | Iteración 20000/30000 | Mejor costo: 3.47

CD 3 | Iteración 25000/30000 | Mejor costo: 3.47

CD 3 | Iteración 30000/30000 | Mejor costo: 3.47

Optimizando rutas para Centro de Distribución 5 con 9 tiendas asignadas...

CD 4 | Iteración 5000/30000 | Mejor costo: 5.52

CD 4 | Iteración 10000/30000 | Mejor costo: 4.40

CD 4 | Iteración 15000/30000 | Mejor costo: 4.40

CD 4 | Iteración 20000/30000 | Mejor costo: 4.40

CD 4 | Iteración 25000/30000 | Mejor costo: 4.40

CD 4 | Iteración 30000/30000 | Mejor costo: 4.40

Optimizando rutas para Centro de Distribución 6 con 9 tiendas asignadas...

CD 5 | Iteración 5000/30000 | Mejor costo: 6.00

CD 5 | Iteración 10000/30000 | Mejor costo: 4.96

CD 5 | Iteración 15000/30000 | Mejor costo: 4.96

CD 5 | Iteración 20000/30000 | Mejor costo: 4.96

CD 5 | Iteración 25000/30000 | Mejor costo: 4.96

CD 5 | Iteración 30000/30000 | Mejor costo: 4.96

Optimizando rutas para Centro de Distribución 7 con 9 tiendas asignadas...

CD 6 | Iteración 5000/30000 | Mejor costo: 5.89

CD 6 | Iteración 10000/30000 | Mejor costo: 4.96

CD 6 | Iteración 15000/30000 | Mejor costo: 3.96

CD 6 | Iteración 20000/30000 | Mejor costo: 3.96

CD 6 | Iteración 25000/30000 | Mejor costo: 3.96

CD 6 | Iteración 30000/30000 | Mejor costo: 3.96

Optimizando rutas para Centro de Distribución 8 con 9 tiendas asignadas...

CD 7 | Iteración 5000/30000 | Mejor costo: 4.70

CD 7 | Iteración 10000/30000 | Mejor costo: 3.81

CD 7 | Iteración 15000/30000 | Mejor costo: 3.81

CD 7 | Iteración 20000/30000 | Mejor costo: 3.81

CD 7 | Iteración 25000/30000 | Mejor costo: 3.81

CD 7 | Iteración 30000/30000 | Mejor costo: 3.81

Optimizando rutas para Centro de Distribución 9 con 9 tiendas asignadas...

CD 8 | Iteración 5000/30000 | Mejor costo: 5.53

CD 8 | Iteración 10000/30000 | Mejor costo: 3.57

CD 8 | Iteración 15000/30000 | Mejor costo: 3.57

CD 8 | Iteración 20000/30000 | Mejor costo: 3.57

CD 8 | Iteración 25000/30000 | Mejor costo: 3.57

CD 8 | Iteración 30000/30000 | Mejor costo: 3.57

Optimizando rutas para Centro de Distribución 10 con 9 tiendas asignadas...

CD 9 | Iteración 5000/30000 | Mejor costo: 5.89

CD 9 | Iteración 10000/30000 | Mejor costo: 3.13

CD 9 | Iteración 15000/30000 | Mejor costo: 3.13

CD 9 | Iteración 20000/30000 | Mejor costo: 3.13

CD 9 | Iteración 25000/30000 | Mejor costo: 3.13

CD 9 | Iteración 30000/30000 | Mejor costo: 3.13

==========

RESULTADOS FINALES DE OPTIMIZACIÓN

==========

Centro de Distribución 1:

Ruta óptima: Centro de Distribución 1 -> Tienda 2 -> Tienda 1 -> Tienda 9 -> Tienda 8 -> Tienda 7 -> Tienda 6 -> Tienda 5 -> Tienda 4 -> Tienda 3 -> Centro de Distribución 1

Costo total: 1.67

Centro de Distribución 2:

Ruta óptima: Centro de Distribución 2 -> Tienda 14 -> Tienda 13 -> Tienda 12 -> Tienda 11 -> Tienda 10 -> Tienda 18 -> Tienda 17 -> Tienda 16 -> Tienda 15 -> Centro de Distribución 2

Costo total: 4.05

Centro de Distribución 3:

Ruta óptima: Centro de Distribución 3 -> Tienda 23 -> Tienda 27 -> Tienda 26 -> Tienda 22 -> Tienda 21 -> Tienda 20 -> Tienda 19 -> Tienda 25 -> Tienda 24 -> Centro de Distribución 3

Costo total: 4.75

Centro de Distribución 4:

Ruta óptima: Centro de Distribución 4 -> Tienda 29 -> Tienda 28 -> Tienda 31 -> Tienda 30 -> Tienda 36 -> Tienda 35 -> Tienda 34 -> Tienda 33 -> Tienda 32 -> Centro de Distribución 4

Costo total: 3.47

Centro de Distribución 5:

Ruta óptima: Centro de Distribución 5 -> Tienda 43 -> Tienda 42 -> Tienda 41 -> Tienda 40 -> Tienda 39 -> Tienda 45 -> Tienda 44 -> Tienda 38 -> Tienda 37 -> Centro de Distribución 5

Costo total: 4.40

Centro de Distribución 6:

Ruta óptima: Centro de Distribución 6 -> Tienda 54 -> Tienda 50 -> Tienda 49 -> Tienda 48 -> Tienda 47 -> Tienda 46 -> Tienda 53 -> Tienda 52 -> Tienda 51 -> Centro de Distribución 6

Costo total: 4.96

Centro de Distribución 7:

Ruta óptima: Centro de Distribución 7 -> Tienda 63 -> Tienda 62 -> Tienda 61 -> Tienda 56 -> Tienda 55 -> Tienda 60 -> Tienda 59 -> Tienda 58 -> Tienda 57 -> Centro de Distribución 7

Costo total: 3.96

Centro de Distribución 8:

Ruta óptima: Centro de Distribución 8 -> Tienda 65 -> Tienda 64 -> Tienda 72 -> Tienda 71 -> Tienda 70 -> Tienda 69 -> Tienda 68 -> Tienda 67 -> Tienda 66 -> Centro de Distribución 8

Costo total: 3.81

Centro de Distribución 9:

Ruta óptima: Centro de Distribución 9 -> Tienda 80 -> Tienda 79 -> Tienda 78 -> Tienda 77 -> Tienda 76 -> Tienda 75 -> Tienda 81 -> Tienda 74 -> Tienda 73 -> Centro de Distribución 9

Costo total: 3.57

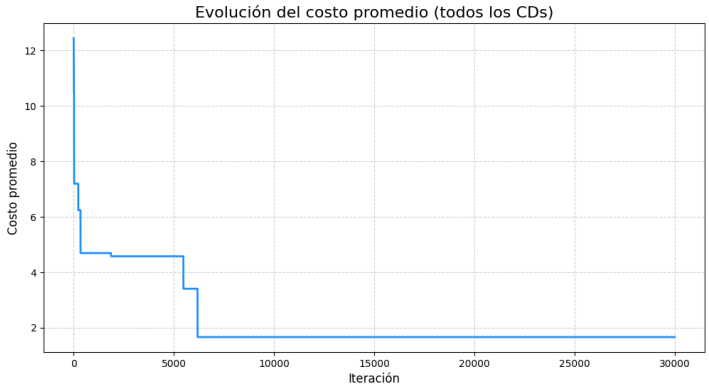
Centro de Distribución 10:

Ruta óptima: Centro de Distribución 10 -> Tienda 90 -> Tienda 89 -> Tienda 88 -> Tienda 87 -> Tienda 86 -> Tienda 85 -> Tienda 84 -> Tienda 83 -> Tienda 82 -> Centro de Distribución 10

Costo total: 3.13

Costo total global optimizado: 37.77

Gráfico de convergencia:



Al ejecutar el programa con 10 centros de distribución y 90 tiendas, se obtuvo una reducción significativa de los costos de transporte. Cada CD logró optimizar sus rutas en menos de 30,000 iteraciones, evidenciando una rápida convergencia hacia soluciones estables.

El costo total global optimizado fue de 37.77 unidades, frente a un costo inicial promedio superior, demostrando la eficacia del método. Las rutas obtenidas presentan recorridos lógicos y compactos, evitando desplazamientos innecesarios entre tiendas.

El gráfico de convergencia generado muestra una tendencia descendente clara, donde el costo promedio disminuye progresivamente hasta estabilizarse, reflejando el proceso de enfriamiento gradual característico del algoritmo.

En conclusión, el Recocido Simulado demostró ser una herramienta poderosa para resolver problemas logísticos complejos, ofreciendo soluciones cercanas al óptimo con una implementación flexible y adaptable a distintos escenarios de distribución.

7. Agenda de Trabajo

Fase	Actividad	Fecha estimada
1	Análisis del problema y recopilación de datos	01 - 03 de octubre de 2025
2	Modelado matemático del problema	04 - 07 de octubre de 2025
3	Selección y justificación del algoritmo heurístico	08 - 10 de octubre de 2025
4	Implementación computacional del Recocido Simulado	11 - 18 de octubre de 2025
5	Generación y corrección del mapa	19 - 22 de octubre de

Fase	Actividad	Fecha estimada
	geoespacial	2025
6	Simulación y análisis de resultados	23 - 28 de octubre de 2025
7	Redacción del informe final	29 - 31 de octubre de 2025

8. Conclusiones

El algoritmo de Recocido Simulado permitió optimizar eficazmente las rutas de distribución, reduciendo el costo total del transporte y mejorando la eficiencia logística. Los resultados demostraron que los operadores de vecindad implementados (2-opt, swap inter-rutas y move-between) aumentan la capacidad de exploración del algoritmo, generando soluciones más estables.

El uso de herramientas como pandas y folium fortaleció la visualización y comprensión de los datos, facilitando la validación de resultados. La corrección del mapa geoespacial constituyó un reto importante, pero su resolución permitió integrar completamente la parte computacional con la representación visual de las rutas. En trabajos futuros, podría ampliarse el modelo incluyendo restricciones de tiempo, capacidad y tráfico real.

9. Referencias

Arias, J. (2021). Optimización heurística en logística: Aplicaciones del Recocido Simulado. Universidad Politécnica de Madrid. Recuperado de https://oa.upm.es/

Martínez, L. & Rivera, J. (2022). Métodos heurísticos aplicados a la optimización de rutas de transporte. Revista Latinoamericana de Ingeniería Logística, 18(3), 45-56.

Rodríguez, P. (2023). Visualización de datos geoespaciales con Folium en Python. Blog de Ciencia de Datos México. Recuperado de https://www.cienciadedatos.mx/

Universidad de Guadalajara (2020). Métodos de búsqueda heurística en optimización combinatoria. Repositorio UDG. Recuperado de https://repositorio.udg.mx/