Digital Object Identifier: 10.24054/rcta.v1i45.3079

Solución al problema de generación de planes de trabajo basado en el problema del agente viajero utilizando el algoritmo genético Chu-Beasley

Solution to the work plan generation problem based on the traveling salesman agent problem using the genetic algorithm Chu-Beasley

PhD.(c) John Fredy Castaneda Londoño[1],
PhD. Ramón Alfonso Gallego Rendón[1],
PhD. Eliana Mirledy Toro Ocampo[2]

[1] ***Universidad Tecnológica de Pereira,*** *Facultad de Ingenierías, Pereira, Risaralda, Colombia.*
[2] ***Universidad Tecnológica de Pereira,*** *Facultad de Ciencias Empresariales, Pereira, Risaralda, Colombia.*

Correspondencia: *jfcastaneda@utp.edu.co*

Recibido: *16 agosto 2024.* ***Aceptado:*** *17 diciembre 2024.* ***Publicado:*** *01 enero 2025.*

Cómo citar: *J. F. Castañeda Londoño, R. A. Gallego Rendón, y E. M. Toro Ocampo, «Solución al problema de generación de planes de trabajo basado en el problema del agente viajero utilizando el algoritmo genético Chu-Beasley», RCTA, vol. 1, n.º 45, pp. 66–73, ene. 2025.*
Recuperado de https://ojs.unipamplona.edu.co/index.php/rcta/article/view/3079

Resumen: Este artículo introduce una metodología para abordar la gestión logística al generar planes de trabajo basado en el problema del Agente Viajero Múltiple (MTSP) y tienen como objetivo la minimización de costos, que puede manifestarse en dos formas: la distancia o el tiempo requerido para la ejecución de los planes de trabajo. Se emplean dos métodos para medir la distancia: el primero calcula distancias esféricas utilizando la fórmula de Haversine, y el segundo aprovecha datos de Google Maps para obtener información de tráfico asociada a tiempos de viaje. Uno de los propósitos de esta metodología es validar los beneficios de llevar a cabo optimizaciones en rutas múltiples que se ven afectadas por variables de tráfico, que se traducen en tiempos de recorrido, sobre una arquitectura de software moderna, modular y de rápida implementación. Para resolver el modelo matemático, se utiliza el algoritmo genético Chu-Beasley con mejoramientos a través del operador Or-Opt, con el objetivo de obtener soluciones de calidad con tiempos razonables en la planificación logística diaria. En la etapa de análisis de resultados, se llevaron a cabo pruebas con instancias obtenidas de ubicaciones reales de empresas cuyas operaciones se ven afectadas por el rendimiento logístico. Los resultados del estudio se compararon simulando un experto en el área utilizando el algoritmo del vecino más cercano como referencia y centrándose en la variable de distancia, evidenciando mejoras significativas y confirmando el beneficio del uso del planeamiento logístico usando algoritmos de optimización con variables de tráfico.

Palabras clave: problema de agente viajero, algoritmo genético, condiciones de tráfico.

Abstract: This article introduces a methodology to address logistics management by generating work plans based on the Multiple Traveling Salesman Problem (MTSP). The

primary objective is cost minimization, which can manifest in two forms: distance or the time required for executing work plans. Two methods are used to measure distance: the first calculates spherical distances using the Haversine formula, and the second leverages Google Maps data to obtain traffic-related travel time information. One of the goals of this methodology is to validate the benefits of carrying out optimizations on multiple routes affected by traffic variables, resulting in travel times, within a modern, modular, and rapidly implementable software architecture. To solve the mathematical model, the Chu-Beasley genetic algorithm is used with enhancements through the Or-Opt operator, aiming to obtain high-quality solutions with reasonable times in daily logistics planning. In the results analysis stage, tests were conducted using instances obtained from real company locations whose operations are affected by logistics performance. The study's results were compared to those of an expert in the field using the nearest neighbor algorithm as a reference, focusing on the distance variable, which demonstrated significant improvements and confirmed the benefits of using logistics planning with optimization algorithms considering traffic variables.

Keywords: traveling salesman problem, genetic algorithm, traffic.

1. INTRODUCCIÓN

El problema del agente viajero (TSP) es un problema fundamental de optimización combinatoria que ha recibido una atención significativa en las últimas décadas. Su relevancia se debe a su amplia gama de aplicaciones en el mundo real, que incluyen operaciones logísticas, servicios de mensajería y entrega de paquetes, perforación de placas de circuito, programación de atención para corregir fallos en sistemas de empresas de servicios públicos y muchos otros.

En términos matemáticos, el problema se formula de la siguiente manera: Dada una lista de n ciudades junto con las distancias entre cada par de ellas, el objetivo es encontrar una ruta que visite cada ciudad exactamente una vez y minimice la longitud total de dicha ruta. El TSP se clasifica como NP-completo, lo que implica que no existe un algoritmo que pueda resolverlo en un tiempo determinístico sin importar la cantidad de ciudades.

No obstante, existen diversos algoritmos aproximados que pueden encontrar soluciones aceptables en tiempos razonables, especialmente para instancias de tamaño moderado, dependiendo del tiempo que se considere como razonable. Cuando el problema requiere de encontrar una solución para más de un agente, este es denominado el problema del agente viajero múltiple (MTSP).

Para plantear el problema en escenarios logísticos reales, se utilizaron las distancias de Google Maps. Estas distancias se obtienen a través del algoritmo de Dijkstra, que es un algoritmo eficiente para calcular la distancia más corta entre dos puntos en un grafo. Una ventaja adicional de utilizar las distancias de Google Maps es la posibilidad de combinar los objetivos asociados con la distancia entre clientes, como el tiempo. El tiempo entre dos clientes se ve afectado por el tráfico, por lo tanto, la matriz de distancia de Google Maps aumenta la precisión para resolver problemas más cercanos a los escenarios logísticos comunes.

Una revisión breve del desarrollo del TSP a lo largo de los años desde su definición nos ubica en 1954 donde se introduce formalmente el Problema del Vendedor Viajero (TSP) por George Dantzig, Ray Fulkerson y Selmer Johnson. Esto marca el inicio de la investigación en problemas de optimización combinatoria en el contexto de la planificación de rutas. En 1956 Menger y Blumenthal extienden de manera independiente el TSP al Problema del Vendedor Viajero Múltiple (MTSP) al introducir el concepto de múltiples vendedores. También proponen una heurística para resolver el problema [1].

Décadas de 1970-1980: Se realiza un progreso significativo en el desarrollo de algoritmos para resolver el MTSP. Los investigadores exploran formas de optimizar la asignación de ciudades a vendedores, así como el orden en que cada vendedor visita las ciudades. También se aplican programación dinámica y técnicas de programación matemática al problema [2].

Década de 1990: Los investigadores se centran cada vez más en aplicaciones prácticas del MTSP, incluidos problemas de enrutamiento de vehículos y

logística de distribución. Esto lleva al desarrollo de algoritmos especializados y herramientas de software para resolver instancias del MTSP en el mundo real [3].

Década de 2000: Avances en la potencia de cómputo y técnicas de optimización conducen a algoritmos más eficientes y escalables para resolver instancias a gran escala del MTSP. Se popularizan enfoques metaheurísticos como algoritmos genéticos, recocido simulado y optimización por colonias de hormigas para abordar el problema [4]. Década de 2010: El MTSP sigue siendo un problema relevante y objeto de investigación activa, con aplicaciones en campos como transporte, telecomunicaciones y gestión de la cadena de suministro. Los investigadores exploran algoritmos híbridos que combinan múltiples técnicas de optimización para lograr mejores resultados [5].

2020 a hoy: El MTSP sigue siendo un área de investigación continua, con un enfoque en el desarrollo de algoritmos que pueden manejar instancias grandes y complejas del problema. Los investigadores también exploran variaciones del MTSP, como el MTSP Dependiente del Tiempo y el MTSP Estocástico, para abordar escenarios más realistas [6], [7], [8].

Es importante tener en cuenta que esta línea de tiempo proporcionada es una visión general amplia, y ha habido numerosas contribuciones individuales de investigación y desarrollos en el campo del MTSP a lo largo de los años.

Con respecto a aplicaciones identificadas en la literatura donde se aborda el TSP con las distancias de los servicios de Google Maps, se encuentra un trabajo inicial en [9] se discute un enfoque novedoso para optimizar el MTSP utilizando un algoritmo genético modificado. El enfoque introduce un cromosoma separado para cada vendedor, lo que permite una mejor representación del problema. El artículo también presenta una metodología completa para la planificación de rutas óptimas con varios vendedores y restricciones secundarias, utilizando Google Maps como punto de partida. El enfoque proporciona un marco completo para resolver problemas como MTSP con ventanas de tiempo.

[10] presenta el desarrollo de un sistema de pedidos de alimentos basado en web y en Android que utiliza algoritmos heurísticos para optimizar la entrega de los productos. Se plantea el TSP para encontrar la ruta más corta entre las direcciones de los clientes. Además, se aprovecha la tecnología de GPS y Google Maps para permitir la visualización de las rutas en un mapa.
[11] discute el desarrollo de una aplicación móvil para resolver el TSP en dispositivos con sistemas operativos Android y iOS. El estudio utiliza las API de Google Maps para obtener datos del mundo real sobre ubicaciones y distancias entre ellas. Emplea Algoritmos Genéticos (GA) y Optimización de Colonias de Hormigas (ACO) para encontrar las rutas óptimas para el TSP. El estudio probó la aplicación en diferentes conjuntos de datos y encontró que ACO proporciona mejores soluciones de ruta, mientras que GA es más rápido en cuanto al tiempo de computación.

[12] propone una metodología para obtener soluciones precisas para el TSP en instancias del mundo real utilizando las API de Google Maps. Las instancias del mundo real del TSP a menudo enfrentan desafíos relacionados con la precisión y las actualizaciones de mapas, especialmente en vecindarios. Los autores proponen una metodología que utiliza las API de Google Maps para obtener información espacial, preprocesa los datos para crear un grafo válido y luego aplica una metaheurística para resolver el TSP. Los resultados se comparan con mapas digitales, resaltando las ventajas de usar las API de Google Maps para soluciones precisas. Los autores enfatizan la importancia de obtener coordenadas precisas para los puntos de interés y la calidad de los datos de carreteras para una solución exitosa.

[13] emplea el algoritmo genético (GA) para abordar TSP. El estudio implica la recopilación y el preprocesamiento de las coordenadas geográficas de las zonas de la ciudad, que luego se utilizan como entrada para el algoritmo genético específico para el TSP (TSGA). El TSGA genera y evoluciona iterativamente poblaciones de posibles rutas, calculando su aptitud en función de la distancia. A través de operaciones de selección, cruce y mutación, explora y refina de forma eficiente las posibles soluciones, convergiendo finalmente hacia la ruta más corta para visitar todas las zonas. MATLAB se utiliza para la implementación y el análisis.

[14] desarrolla una aplicación móvil que emplea un algoritmo genético (GA) y el sistema de navegación de Google Maps para resolver el problema de optimización de rutas (ROP) para rutas de ventas. El ROP es un problema desafiante en la planificación de rutas y la logística, donde el objetivo es encontrar la ruta más corta para visitar un conjunto de clientes de manera eficiente. La aplicación, diseñada para

dispositivos Android, utiliza el GA para calcular rutas de ventas óptimas basadas en factores como la distancia y las ubicaciones de los clientes.

La contribución de este trabajo es realizar una comparación entre el uso de distancias de Haversine y las de la API de Google Maps desarrollar soluciones eficientes para el problema del agente viajero simulando escenarios reales en empresas, utilizando distancias y tiempos afectados por el tráfico como función objetivo y demostrar los beneficios que se obtienen de implementar estas aplicaciones en compañías donde su cadena de valor tiene un componente logístico importante es sus costos. Además, se pretende desarrollar una arquitectura moderna, modular y de rápida implementación para compañías de cualquier tamaño.

Con base en lo anterior se define como objeto de investigación de este artículo el desarrollo de una metodología para abordar el Problema del agente viajero Múltiple (MTSP) con que minimice el costo-tiempo en la planificación logística diaria utilizando dos matrices de distancia: una calculada mediante la fórmula de Haversine para distancias esféricas y otra que incorpora información de Google Maps para considerar variables de tráfico y tiempo de viaje sobre un plataforma de implementación eficiente.

2. METODOLOGÍA

La definición convencional de Problemas del agente viajero (TSP), se puede describir mediante un grafo completo no dirigido. En esta representación, el conjunto de clientes se denota como $V = \{0,..,n\}$ donde 0 designa el centro de distribución central o centro logístico. La colección de arcos que forman la estructura subyacente del grafo se denota como $A = \{(i, j) i \neq j, \forall i, j \in V\}$. Al ser un problema de optimización combinatorio, puede ser planteado como un problema de programación lineal entera mixta. Para ello se define:

Conjuntos:

I	Conjunto de centros logísticos o depósitos
J	Clientes o nodos
V	Conjunto $I \cup J$
G	Conjunto de agentes activos

Variables de decisión:

x_{ij}	Representa los arcos activos en el grafo (binaria).
y_i	Representa si un agente se encuentra activo (binaria).
f_{ij}	Un Cliente *j* atendido desde el depósito $i \in I$. (entera)

Parámetros:

d_{ij}	La distancia/costo entre los clientes *i-j* (real)
0_i	El costo de usar el grupo de trabajo *i*. *(O_i=1)* (real)
m	Número de agentes (entera)

El modelo se define como:

$$\min \sum_{\substack{i,\, j \in J \\ i \neq j}} d_{ij} x_{ij} + \sum_{j \in V} o_i y_i \qquad (1)$$

Sujeto a:

$$\sum_{\substack{i \in V,\, j \in J \\ i \neq J}} x_{ij} = 1 \qquad (2)$$

$$x_{ij} + x_{ji} \leq 1 \quad \forall i, j \in V \qquad (3)$$

$$\sum_{\substack{i \in V, j \in J \\ i \neq J}} x_{ij} \leq card(V) \quad (4)$$

$$\sum_{\substack{i \in I, j \in J \\ i \neq J}} f_{ij} \leq 1 \quad (5)$$

$$f_{ij} = x_{ij} \quad \forall i \in I, j \in J \quad (6)$$

$$\sum_{j \in J} x_{0j} \leq m \quad (7)$$

$$\sum_{i \in G} y_i \geq 1 \quad (8)$$

La ecuación (1) representa la función de costos que se simplifica con el uso de las distancias como la variable dependiente de la función $c_{ij} = f(d_{ij})$. Se adicionan además los costos de usar un grupo de trabajo específico. Las restricciones (2) y (3) aseguran que cada nodo del grafo, excepto el depósito, tenga dos aristas que lo conecten con otros nodos. Esto garantiza que cada nodo sea visitado exactamente una vez por cada ruta. La restricción (4) garantiza que el número de aristas del grafo sea igual al número de nodos como una función de cardinalidad del conjunto *V*. Las restricciones (5) y (6) aseguran que cada cliente sea atendido por un depósito específico. Esto es necesario para cumplir con los requisitos del caso de uso, que especifica que cada cliente debe ser atendido por un solo agente. La restricción (7) indica que el número de rutas debe ser igual al número de grupos de trabajo disponibles. Esto es necesario para garantizar que haya suficientes recursos para atender a todos los clientes. La restricción (8) garantiza que haya al menos un grupo de trabajo disponible. Esto es necesario para que el problema tenga al menos una solución factible.

El modelo anterior hace referencia al TSP, pero para obtener soluciones para el MTSP, en este caso se implementa una capa de clusterización a través del algoritmo K-means que genera grupos de clientes o nodos dependiendo de su ubicación geográfica.

Para obtener soluciones al modelo planteado, se propone una implementación de un algoritmo genético Chu-Beasly (CBGA). Las principales características del CBGA son las siguientes: (i) Población inicial: La población inicial se genera mediante la creación de una permutación aleatoria de los clientes. (ii) Selección: Los padres se seleccionan de un subconjunto de la población mediante selección aleatoria. Solo se eligen los dos individuos más aptos para el cruce. (iii) Cruce: El cruce se realiza utilizando el operador de Cruce Parcialmente Emparejado (PMX) [15]. Este operador garantiza que los descendientes resultantes sigan siendo factibles dentro de las restricciones definidas. (iii) Mutación: La mutación se realiza mediante la inversión de una sección aleatoria de la ruta de un individuo. (iv) Mejora: Después de la aplicación de la mutación, se utiliza el operador Or-opt para realizar mejoras locales en los individuos [16].

3. RESULTADOS

Para esta implementación, se utilizaron ubicaciones aleatorias simulando activos o clientes de una empresa de distribución de energía eléctrica. El conjunto de datos consta de instancias con 50, 100 y 150 clientes, distribuidos tanto en áreas urbanas como rurales. Además, se tuvieron en cuenta condiciones operativas específicas, como la existencia de un depósito u oficina central en las zonas urbanas más críticas. En este escenario, existen cuatro oficinas centrales, cada una asociada a un número constante de grupos de trabajo.

Tabla 1: *Instancias implementadas*

Instancia	Numero de grupos de trabajo
150	20
100	7
50	5

Bajo estas condiciones, el proceso de asignar ubicaciones a las oficinas centrales individuales, junto con un número designado de grupos de trabajo, involucra un procedimiento de agrupación de dos pasos: (1) Agrupación inicial: Se ejecuta un algoritmo de K-means con el número de grupos configurado para que coincida con el número de grupos de trabajo. Esta restricción operativa asegura la utilización de todos los grupos de trabajo disponibles. (2) Asignación de oficinas centrales: A continuación, se asigna una oficina central a cada grupo utilizando el algoritmo de los K- means. Este paso implica el uso de los centroides de los grupos iniciales en conjunto con las ubicaciones de las oficinas centrales. Estos experimentos se implementaron en Python 3.9 en una laptop con un procesador Intel i5 vPro de 8ª generación a 1.896

GHz y 4 núcleos, ejecutando Windows 10. El algoritmo CBGA se implementó utilizando la biblioteca NumPy.

En la tabla 2, 3 y 4 se muestran los resultados obtenidos para la instancia usadas en este trabajo y presentadas en la tabla 1. En los resultados para las distancias de Haversine como de la API de Google Maps tiene como referencia a los resultados usando el algoritmo del vecino más cercano o k-NN que simula la planeación empírica más eficiente. Esto quiere decir que para una compañía que su planeación está basada en conocimiento de experto sin ninguna regla o estrategia clara de programación, los beneficios serían mayores. Para las distancias de Haversine, la matriz de tiempo de viaje se calculó estableciendo una velocidad constante de 35 km/h. Este enfoque no es muy preciso, pero resulta útil como referencia en este experimento para hacer comparaciones con los tiempos entre ubicaciones afectados por el tráfico promedio.

De las tablas de resultados se obtiene: (1) Incumbente distancia: Que es el valor de la distancia en kilómetros de todas las rutas. Estas distancias para las calculadas como Haversine son siempre menores a las de Google ya que las primeras son lineales y las segundas son por carretera. (2) Ahorro: Corresponde a la comparación entre la incumbente calculada con la matriz de distancias de Google con respecto al uso del algoritmo K-NN. (3) Distancia clúster promedio: Es la distancia de Haversine entre el centroide del clúster y el depósito o centro logístico. (4) Incumbente tiempo: Corresponde al tiempo de todas las rutas. Para Haversine, es calculado estableciendo una velocidad constante de 35 km/h. En las de Google son basadas en la información de la API. (5) Ahorro en tiempo: Es calculado con la matriz de tiempos de Google usando como referencia los resultados obtenidos por el algoritmo de K-NN. (6) F.O Tiempo: Esta columna representa los resultados cuando en la función objetivo es usada la matriz de tiempos de Google. Es decir, se minimiza el tiempo de recorrido y no la distancia.

Tabla 2: *Resultados para la instancia de 150 ubicaciones y 20 grupos de trabajo disponibles.*

Tipo de distancia	Haversine	Google	F.O Tiempo
Incumbente distancia	1257.32 km	2771.00 km	2811.00 km
Ahorro	41.32 km	122.00 km	82.00 km
Ahorro (%)	3,18%	4,22%	2,83%
Distancia clúster promedio	22.60 km	22.60 km	22.60 km
Incumbente tiempo	83.83 Horas	95.30 Horas	94.33 Horas
Ahorro en tiempo	2.76 Horas	3.35 Horas	4.36 Horas
Tiempo de ejecución	74,7 seg	404,01 seg	541,52 seg

Tabla 3: *Resultados para la instancia de 100 ubicaciones y 7 grupos de trabajo disponibles.*

Tipo de distancia	Haversine	Google	F.O Tiempo
Incumbente distancia	689.63 km	1570.91 km	1583.91 km
Ahorro	63.90 km	178.60 km	165.60 km
Ahorro (%)	8,48%	10,21%	9,47%
Distancia clúster promedio	17.31 km	17.31 km	17.31 km
Incumbente tiempo	45.96 Horas	58.66 Horas	58.22 Horas
Ahorro en tiempo	4.25 Horas	6.71 Horas	7.15 Horas
Tiempo de ejecución	42,3 seg	285,44 seg	310,73 seg

Tabla 4: *Resultados para la instancia de 50 ubicaciones y 5 grupos de trabajo disponibles.*

Tipo de distancia	Haversine	Google	F.O Tiempo
Incumbente distancia	558.24 km	1191.15 km	1219.33 km
Ahorro	21.96 km	36.34 km	8.16 km
Ahorro (%)	3,79%	2,96%	0,66%
Distancia clúster promedio	17.73 km	17.73 km	17.73 km
Incumbente tiempo	37.22 Horas	46.00 Horas	45.27 Horas
Ahorro en tiempo	1.47 Horas	2.09 Horas	2.83 Horas
Tiempo de ejecución	21,99 seg	97,88 seg	91,99 seg

A partir de estos resultados, el usar la información de la API de Google es una ventaja competitiva para obtener un planeamiento de rutas óptimo cercano a escenarios logísticos reales. La minimización de tiempo se realiza usando la matriz de tiempos de la API de Google Maps en vez de la de distancias, donde este enfoque es el más adecuado para obtener las rutas. La ejecución usando la matriz de distancias de Haversine es generalmente menor debido a que se realiza propiamente en el algoritmo a diferencia de las de la API de Google Maps que dependen de la respuesta que se realiza a la misma a través de internet usando el protocolo HTTP. La Fig. 1 muestra la solución visual para los resultados de la instancia 150-20 utilizando distancias de Google, minimizando la distancia y se pueden observar las 4

oficinas centrales o depósitos. La Fig. 2 muestra una ruta preliminar en Google Maps para la ruta 6.

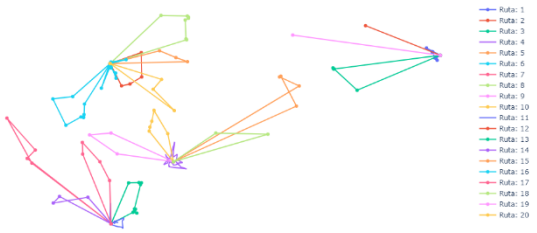

Fig. 1. *Resultados para la instancia de 150 ubicaciones con 20 grupos de trabajo.*

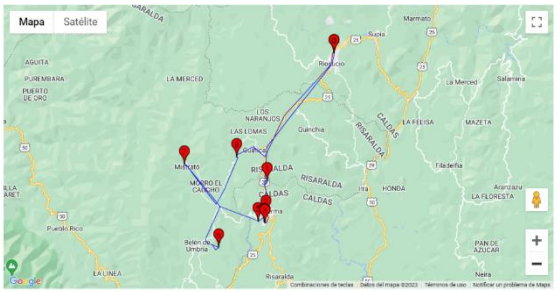

Fig. 2. *Ruta 6 de la solución de la instancia 150-20 como un bosquejo de la ruta in Google Maps.*

Para la implementación se usó la arquitectura de software que se muestra en la Fig. 3. Allí se identifica la construcción modular con un front-end para la interfaz o interacción y un back-end que procesa la información. La carga computacional es realizada desde una API RESTFUL alojada en los servicios de nube de Microsoft Azure, por lo que puede ser implementada fácilmente en cualquier tipo de dispositivo, móvil o fijo sin requerir de hardware de alto rendimiento.

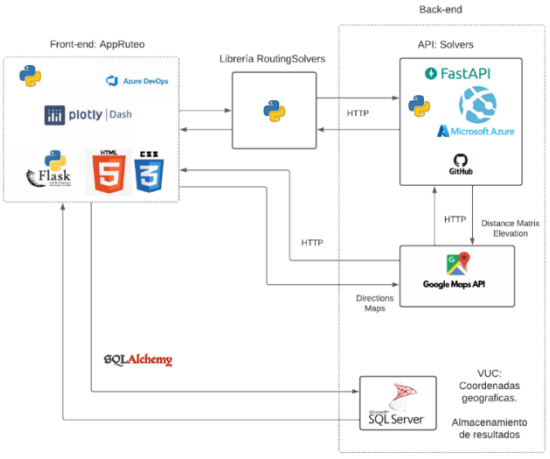

Fig. 3. *Arquitectura de implementación*

4. CONCLUSIONES

Este trabajo aborda el problema del agente viajero que se considera importante en la industria logística e incluso en empresas de servicios públicos domiciliarios. Se presenta una metodología que utiliza dos matrices de distancia: una basada en la fórmula de Haversine para obtener distancias esféricas y otra que aprovecha la información de Google Maps para incorporar variables de tráfico y tiempo de viaje. Esta metodología tiene como objetivo validar los beneficios de la optimización de rutas afectadas por el tráfico.

Se utiliza un algoritmo genético denominado Chu-Beasley con el operador Or-Opt para obtener soluciones de calidad en tiempos razonables. Se utilizaron instancias de prueba con ubicaciones reales pero aleatorias para simular escenarios logísticos reales.

Los resultados obtenidos en los experimentos fueron satisfactorios en comparación con el algoritmo del vecino más cercano, que simula una planificación empírica con un rendimiento aceptable. La utilización de la información de Google Maps para obtener distancias y tiempos reales en carreteras entre clientes, considerando el tráfico, se destacó como una ventaja competitiva para obtener una planificación de rutas óptima en escenarios logísticos realistas. Además, se propone una arquitectura de implementación modular que permite hacer despliegues rápidos y versátiles.

Para trabajos futuros, se pretende desarrollar una metodología bi-objetivo que además del costo, permita minimizar las emisiones de gases de efecto invernadero. También se planea incluir un algoritmo de solución adicional que incorpore mecanismos de búsqueda local e incremente el rendimiento en la calidad y tiempo de las soluciones.

5. AGRADECIMIENTOS

Los autores agradecen a la Universidad Tecnológica de Pereira por su apoyo en el proyecto 7-22-1 y a Central Hidroeléctrica de Caldas por el apoyo y retroalimentación en la implementación.

6. REFERENCIAS

[1] W. R. Abel and L. M. Blumenthal, “Distance Geometry of Metric Arcs,” *Am. Math. Mon.*, vol. 64, no. 8P2, pp. 1–10, Oct. 1957, doi: 10.1080/00029890.1957.11989113.

[2] K. C. Gilbert and R. B. Hofstra, “A New Multiperiod Multiple Traveling Salesman

Problem with Heuristic and Application to a Scheduling Problem," *Decis. Sci.*, vol. 23, no. 1, pp. 250–259, 1992, doi: 10.1111/J.1540-5915.1992.TB00387.X.

[3] A. Langevin, F. Soumis, and J. Desrosiers, "Classification of travelling salesman problem formulations," *Oper. Res. Lett.*, vol. 9, no. 2, pp. 127–132, Mar. 1990, doi: 10.1016/0167-6377(90)90052-7.

[4] T. Bektas, "The multiple traveling salesman problem: an overview of formulations and solution procedures," *Omega*, vol. 34, no. 3, pp. 209–219, Jun. 2006, doi: 10.1016/J.OMEGA.2004.10.004.

[5] R. I. Bolaños, E. M. Toro O, and M. Granada E, "A population-based algorithm for the multi travelling salesman problem," *Int. J. Ind. Eng. Comput.*, vol. 7, no. 2, pp. 245–256, Mar. 2016, doi: 10.5267/J.IJIEC.2015.10.005.

[6] O. Cheikhrouhou and I. Khoufi, "A comprehensive survey on the Multiple Traveling Salesman Problem: Applications, approaches and taxonomy," *Computer Science Review*, vol. 40. 2021. doi: 10.1016/j.cosrev.2021.100369.

[7] C. Colombaroni, M. Mohammadi, and G. Rahmanifar, "Makespan minimizing on multiple travel salesman problem with a learning effect of visiting time," *WSEAS Trans. Syst. Control*, vol. 15, pp. 477–489, 2020, doi: 10.37394/23203.2020.15.50.

[8] R. G. Mbiadou Saleu, L. Deroussi, D. Feillet, N. Grangeon, and A. Quilliot, "The Parallel Drone Scheduling Problem with Multiple Drones and Vehicles," *Eur. J. Oper. Res.*, vol. 300, no. 2, pp. 571–589, Jul. 2022, doi: 10.1016/j.ejor.2021.08.014.

[9] A. Király and J. Abonyi, "A Google Maps based novel approach to the optimization of multiple Traveling Salesman problem for limited distribution systems," *Acta Agrar. Kaposváriensis*, vol. 14, no. 3, pp. 1–14, 2010, [Online]. Available: https://journal.uni-mate.hu/index.php/aak/article/view/1952

[10] R. D. H. Tobing, "A food ordering system with delivery routing optimization using global positioning system (GPS) technology and google maps," *Internetworking Indones. J.*, vol. 8, no. 1, pp. 17–21, 2016.

[11] İ. İlhan, "An Application on Mobile Devices with Android and IOS Operating Systems Using Google Maps APIs for the Traveling Salesman Problem," *Appl. Artif. Intell.*, vol. 31, no. 4, pp. 332–345, 2017, doi: 10.1080/08839514.2017.1339983.

[12] L. G. Hernández-Landa and R. E. Mata-Martínez, "Accurate solutions for real instances of the traveling salesman problem using Google Maps APIs," *Proc. Int. Conf. Ind. Eng. Oper. Manag.*, vol. 2018, no. SEP, pp. 837–843, 2018.

[13] Z. Al-Jabbar, "Using Genetic Algorithm to Solve Travelling Salesman Optimization Problem Based on Google Map Coordinates for Duhok City Areas," *Acad. J. Nawroz Univ.*, vol. 7, no. 3, pp. 99–114, 2018, doi: 10.25007/ajnu.v7n3a207.

[14] C. Zambrano-Vega, G. Acosta, J. Loor, B. Suárez, C. Jaramillo, and B. Oviedo, "A Sales Route Optimization Mobile Application Applying a Genetic Algorithm and the Google Maps Navigation System," *Adv. Intell. Syst. Comput.*, vol. 918, pp. 517–527, 2019, doi: 10.1007/978-3-030-11890-7_50.

[15] D. E. Goldberg and R. Lingle, "Alleles, loci, and the traveling salesman problem," in *Proceedings of an International Conference on Genetic Algorithms*, 1985, pp. 10–19.

[16] R. A. Gallego Rendón, E. M. Toro Ocampo, and A. H. Escobar Zuluaga, *Técnicas Heurísticas y Metaheurísticas*. Universidad Tecnológica de Pereira. Vicerrectoría de Investigaciones, Innovación y Extensión. Ingenierías Eléctrica, Electrónica, Física y Ciencias de la Computación, 2015. Accessed: Sep. 03, 2023. [Online]. Available: https://unilibros.co/gpd-tecnicas-heuristicas-y-metaheuristicas.html