# Ruteo de entrega de productos para una empresa de e-commerce

Juan Pablo Echeagaray González, Emily Rebeca Méndez Cruz,
Carolina Longoria Lozano, Verónica Victoria García De la Fuente
Optimización determinista
MA2001B.200
Dr. Fernando Elizalde Ramírez
Dr. Jaime Eduardo Martínez Sánchez

***Resumen*—Una empresa de e-commerce ha solicitado que alumnos del Tecnológico de Monterrey diseñen un programa de rutas para la entrega de los productos adquiridos por sus clientes en su tienda *online*. Con las bases de datos recibidas se formuló un modelo de CVRP, este modelo fue embebido y optimizado para minimizar las distancias recorridas por los camiones de entrega. El producto final despliega un reporte de las rutas así como una visualización de estas.**

***Index Terms*—CVRP, Programación Entera, Metaheurísticos, Open Source Software**

## I. Introducción

Uno de los procesos más importantes para cualquier empresa que tenga un proceso de distribución es el ruteo y asignación de vehículos para satisfacer las demandas de sus clientes. El problema de Ruteo de Vehículos (VRP) se centra en la generación de un conjunto de rutas para su flotilla que minimice el costo total de las entregas, se conocen de antemano las demandas de los clientes, y todos los vehículos de la flotilla deben salir y regresar al mismo centro de distribución; aunado a esto, cada cliente es visitado una sola vez, y a cada vehículo no se le pueden asignar entregas que sobrepasen su capacidad de carga máxima [1].

El VRP ha sido uno de los problemas más estudiados en el campo de la logística; a la fecha se han logrado implementar algoritmos que encuentran soluciones exactas a este problema, sin embargo, estas aproximaciones fallan al ser escaladas a un nivel industrial; las ganancias de tener una solución óptima se ven minimizadas cuando se toma en cuenta el tiempo y poder computacional requerido para obtenerlas. Por esto, es recomendable introducir heurísticos al proceso de optimización, que si bien, no llegarán al conjunto de rutas óptimos, estarán cercanos al mismo.

Este escrito se organiza de la siguiente manera: en las secciones II y III se delimita el caso de estudio a resolver y se plantea formalmente la problemática. En las secciones IV y V presentamos una justificación de nuestro estudio así como técnicas usadas con anterioridad para resolver este problema.

En las secciones VI y VII se definen los objetivos de nuestra implementación así como la hipótesis a probar. En las secciones VIII y IX se presenta la metodología seguida; en las secciones X y XI presentamos nuestros resultados experimentales y áreas de oportunidad descubiertas, y en las secciones XII y XIV presentamos el impacto que tendría nuestra implementación para los Objetivos de Desarrollo Sostenible y conclusiones del estudio.

En las secciones XIII y A se presentan datos técnicos del equipo utilizado y se proporciona una liga al código fuente implementado.

Verónica Victoria García De la Fuente, Emily Rebeca Méndez Cruz, Carolina Longoria Lozano, Juan Pablo Echeagaray González pertencen al Tec de Monterrey Campus Monterrey, N.L. C.P. 64849, Mexico

## II. Objeto de estudio

Nuestra implementación se enfoca en resolver el problema de ruteo para la entrega de productos, siendo esto un problema conocido como CVRP (Capacitated Vehicle RoutingProblem); en este se busca encontrar un conjunto de rutas para una flotilla de vehículos que tenga el menor costo de ruta en el que se realicen todas las entregas necesarias. Cada uno de los vehículos de la flotilla sale del mismo centro de distribución, y al final de su ruta debe de regresar a este [1].

Los vehículos solamente realizarán entregas a los clientes de la empresa, es decir, no habrá etapa alguna de carga de paquetes más que en el centro de distribución; aunado a esto, cada uno de los clientes se visitará una sola vez, esto hace referencia a que si uno de los clientes ordenó más de un producto, se entregarán todas sus ordenes en el momento en que sea visitado, considerando al conjunto de artículos como un solo objeto.

No se toma en cuenta un límite de distancia recorrida por vehículo ni tampoco se consideran ventanas de tiempo para realizar las entregas.

## III. Planteamiento del problema

Sean los conjuntos:

$$i \in \{1, \ldots, n\}$$
$$j \in \{1, \ldots, n\}$$
$$k \in \{1, \ldots, V\}$$

Sean los parámetros:

$V$ : Número de vehículos disponibles

$Q$ : Capacidad de carga máxima del vehículo en $m^3$

$n$ : Número de clientes a visitar

$c_{ij}$ : Distancia geodésica entre el punto $i$ y el punto $j$

$d_j$ : Demanda en $m^3$ del cliente $j$

Sean las variables:

$x_{ijk}$ : Variable binaria que toma el valor de 1
si el arco del punto i al punto j
forma parte de la ruta óptima recorrida por el vehículo $k$
$x_{ijk} \in \{0, 1\}$

Modelamos el problema de optimización como:

$$\begin{aligned}
\min \quad & z = \sum_{k=1}^{V}\sum_{j=1}^{n}\sum_{i=1}^{n} c_{ij}x_{ijk} \\
& \sum_{i=1}^{n} x_{ijk} = \sum_{i=1}^{n} x_{jik} \qquad \forall j, k \\
& \sum_{k=1}^{V}\sum_{i=1}^{n} x_{ijk} = 1 \qquad \forall j \\
\text{sujeto a:} \quad & \sum_{j=2}^{n} x_{1jk} = 1 \qquad \forall k \\
& \sum_{i=1}^{n}\sum_{j=2}^{n} d_j x_{ijk} \leq Q \qquad \forall k \\
& u_j - u_i \geq d_j - Q(1 - x_{ijk}) \qquad \forall i, j, k \\
& d_i \leq u_i \leq Q \qquad \forall i \\
& x_{ijk} \in \{0, 1\} \qquad \forall i, j, k
\end{aligned}$$

Las restricciones anteriores hacen referencia a:

1. Número de veces que un camión entra por un punto debe de ser igual que el número de veces que las que sale
2. Cada camión solamente puede entrar 1 vez a un punto
3. Todos los vehículos salen del CEDIS
4. Límite a la capacidad de carga máxima del vehículo
5. Las últimas 2 restricciones eliminan los *sub-tours*

Para la problemática actual, estamos planeando realizar 192 entregas $(n)$, se dispondrán de 9 vehículos $(V)$, cada uno de ellos tendrá una capacidad máxima de carga de 18 $m^3$ $(Q)$. Los valores numéricos para las demandas y distancias son calculados dentro de la aplicación.

El heurístico que usamos en nuestro modelo es el costo asociado de ir de un punto de la ciudad a otro, en nuestro modelo la distancia entre 2 puntos de entrega funge ese papel. No estamos modelando la distancia que hay entre cada punto como la distancia manejando, sino que estamos usando la distancia geodésica que hay entre cada punto, una de las principales virtudes de este heurístico es la sencillez con la que puede ser calculado y tampoco se requiere de software comercial para obtenerse.

Hemos decidido seguir adelante con este heurístico ya que hay varios proyectos reconocidos que lo usan para resolver problemas de ruteo de vehículos [2].

## IV. Justificación

Actualmente el e-commerce a crecido un 55 % debido a la pandemia y la tendencia es que se mantenga el número de ventas actual [3], siendo esto un reto para la logística de entrega, la cual afecta directamente al precio de venta, por lo tanto en los gastos de la empresa viendo reducidas sus utilidades, de ahí la necesidad del desarrollo de nuestro proyecto ya que consistes en generar soluciones de forma rápida a este tipo de problemas beneficiando a la empresa y al cliente, ya que se eficientiza el proceso, se optimizan los recursos y mejora la calidad de servicio hacia el cliente

## V. Marco teórico

El *Traveling Salesman Problem*, conocido como *Problema del Agente Viajero* en español, es un problema que se encarga de buscar la ruta más corta y eficiente para llegar a un destino. Esta se basa en que existen múltiples opciones de llegar a un mismo destino, pero enfocándose en reducir costos de transporte, por lo que obtenemos de solución la ruta más corta [4].

Para ésta problemática se tiene contemplado que entre más destinos haya automáticamente el nivel de complejidad para el cálculo de la solución óptima aumenta. Por esta razón TSP es clasificado como un problema de NP-Hard, de acuerdo con la teoría de la complejidad computacional [4].

Para la solución de este problema existen diferentes métodos debido a la popularidad y complejidad que tiene, estos son algunos de ellos:

### *V-A. Vecino más cercano*

También conocido como KNN, este es uno de los algoritmos más simples para resolver este problema. Se trata de un método de aprendizaje automático básico aplicado para la logística de transporte, en este caso, el conductor -o agente viajero- siempre comienza su recorrido con el destino más cercano. Aunque para este método la solución encontrada no siempre logra la optimización [4].

### *V-B. Ramificación y atadura*

El método Branch and Bound es un algoritmo complejo diseñado para solucionar problemas que cuentan con variables de decisión enteras; para este método, el problema se divide en múltiples sub-problemas donde cada uno de estos tiene varias soluciones posibles. Se debe destacar que al seleccionar una solución, esta puede afectar en las posibles soluciones de sub-problemas posteriores, ya que como su nombre lo indica, actúa de manera ramificada. Este método se aplica como solución del problema del vendedor viajero [4].

### *V-C. Fuerza bruta*

Consiste en la enumeración sistemática de todas las posibles rutas de distribución, se calcula y compara todas las posibles soluciones revisando cuál o cuáles de ellas cumplen mejor a los objetivos de la empresa. Esto con el fin de establecer una única solución, que para el caso de este problema, sería la más corta y por ende la óptima [4].

## VI. Objetivos

El objetivo principal de nuestro trabajo es generar una metodología para resolver el problema de generación de rutas de entrega para una empresa de e-commerce que se acerque a la optimalidad. De forma específica, se desea encontrar un

conjunto de $n$ rutas que realicen todas las entregas de 1 día para la empresa, estas rutas deben de minimizar la distancia total recorrida así como respetar las diversas restricciones propias del problema y de la empresa

## VII. Hipótesis

Proponemos una implementación que es capaz de obtener resultados cercanos a la solución óptima en tiempos aceptables, con un costo computacional bajo, en relación a otras implementaciones.

## VIII. Metodología

Se realizará una limpieza de las bases de datos proporcionadas por medio de un lenguaje de programación; con estas se determinarán las entregas a hacer en 1 día y se obtendrán los domicilios de los clientes, estos serán después geo-codificados para calcular una matriz de distancias todos contra todos.

En conjunto con la matriz de distancias y el volumen de los pedidos de cada cliente se realizará un proceso de optimización con el `solver` implementado en la librería `ortools` [5] de libre distribución. Finalmente se desplegará un reporte con el conjunto de rutas encontradas en un mapa en adición a datos generales de la solución propuesta, tales como el volumen muerto y el consumo de gasolina.

## IX. Propuesta metodológica

Para la solución de esta problemática hemos usado el lenguaje de programación Python; para los procesos de lectura, limpieza, exploración y agregación de datos hemos usado la librería `pandas` [6] en conjunto con `matplotlib` [7] y `seaborn` [8], para la solución del problema de programación lineal usamos la librería `ortools` [9] de Google, y para la visualización de los rutas generadas hemos usado `folium` [10]. Una liga al código fuente que hemos desarrollado se encuentra en el apéndice A.

### *IX-A. Limpieza de la base de datos*

Hemos recibido 4 bases de datos. Una conteniendo las direcciones de los centros de distribución de la empresa, un archivo que contiene un listado de todos los productos que vende más algunas de sus propiedades, en particular nos interesa el volumen de cada artículo. Disponemos también de un archivo que contiene todos los tipos de vehículos que la empresa usa para realizar las entregas, y una base de datos que contiene un listado de las entregas que se deben de realizar.

### *IX-B. Procesamiento de domicilios*

Para la geo-codificación de los domicilios hemos usado la librería `Photon` [11], desde el paso anterior hemos creado una columna que contiene todos los datos de referencias de los domicilios de los compradores. El proceso de geo-codificación toma un tiempo promedio de 15 minutos en la máquina que usamos, en promedio se logran codificar 25 % de los domicilios que recibe, el resto no contiene la información geográfica suficiente para ser identificada, por lo que no se toma en cuenta para la generación de rutas. Al final de este proceso tenemos las coordenadas asociadas a cada uno de los domicilios.

### *IX-C. Cálculo de matriz de distancias*

De las coordenadas de cada domicilio calculamos una matriz de distancia todos contra todos, la distancia a usar es la geodésica. Como función geodésica hemos usada la implementada en el módulo `distance` de la librería `Geopy` [12] usada para el procesamiento de domicilio, y la matriz de distancia se calcula con la misma a través de la función `cdist` implementada en el sub-módulo `spatial` de la librería de cómputo científico `SciPy` [13].

### *IX-D. Generación de rutas*

Para la solución del problema hemos usado la librería `ortools` [9] de Google. Hacemos uso de la matriz de distancias calculada en el paso anterior, un listado de cuánto volumen debe de ser entregado a cada uno de los clientes, un número de vehículos a mandar, y su capacidad máxima de carga en metros cúbicos. La solución generada consiste de un listado de las rutas a seguir por cada uno de los vehículos, el volumen total que llevarán, cuál fue la distancia total recorrida por todos los vehículos y cuánta carga llevaron.

### *IX-E. Visualización de rutas*

La visualización de rutas se realizó con la API de la librería `folium` [10], con ella hemos generado un mapa de Monterrey en la que se visualizan las rutas que siguen todos los vehículos; el mapa también despliega información del domicilio al que se realiza la entrega, y el volumen total de los productos que debe de recibir.

### *IX-F. Resumen de resultados*

Del resultado de generación de rutas se despliega también un tabulado en el que se concentran métricas para cada uno de los vehículos de la flotilla, el tabulado contiene información como:

1. Número de clientes visitados
2. Distancia total recorrida en $km$
3. Carga del vehículo en $m^3$
4. Volumen muerto en $m^3$
5. Gasolina consumida en $l$

Dada la instancia de VRP que intentamos resolver, suponemos que la flotilla es homogénea; es decir, todos los vehículos tienen la misma capacidad de carga y tienen el mismo rendimiento de combustible. Es por esto que podemos calcular un consumo de gasolina y una huella de carbono.

## X. Experimentación y resultados

La red con la que tratamos consta de 193 nodos, 192 referentes a los clientes, y 1 nodo más para el Centro de Distribución. El número de conexiones (aristas) dentro del grafo es $\binom{193}{2} = 18528$

Dada la complejidad computacional de resolver de forma exacta un VRP con 193 nodos, deben de utilizarse heurísticos en el proceso de solución. La librería `ortools` [9]le permite al usuario seleccionar 1 heurístico para la obtención de una solución inicial a mejorar, y 1 meta-heurístico para realizar el

proceso de optimización. Es también decisión del desarrollador imponer un límite al tiempo de ejecución del que dispone el algoritmo, este parámetro también deberá ser optimizado en base a los resultados generados.

Para obtener una solución básica inicial hemos optado por utilizar un algoritmo de naturaleza voraz que selecciona como siguiente nodo a visitar a aquel con el menor costo de ruta asociado; este proceso se repite para cada uno de los vehículos de la flotilla. Como metaheurístico hemos seleccionado una búsqueda local guiada, que de acuerdo a la documentación de la librería [5] suele ser la mejor opción para resolver problemas de ruteo de vehículos.

### *X-A. Ajuste de parámetros*

Para determinar cuál es el mejor tiempo límite se obtuvieron 10 muestras de resultados para tiempos máximos de cómputo de entre 1 y 10 segundos. Se recuperó la mejor distancia encontrada, así como la media, el máximo, la desviación estándar y la varianza. En la figura 1 se presentan las curvas de optimización generadas, se grafica la distancia mínima encontrada, la distancia media y la máxima del conjunto de 10 muestras.

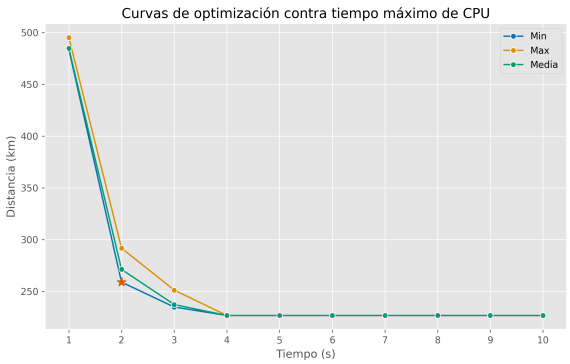


Figura 1: Curva de optimización con diferentes tiempos máximos de cómputo

También hemos generado una curva para varianza den cada una de las observaciones:

De estas curvas después se utilizó la librería de `Python`, `kneed` [14]. En ella se implementa el algoritmo propuesto por Satopaa en Finding a "Kneedle" in a Haystack: Detecting Knee Points in System Behavior [15]. En la figura 1 y en la figura 2 se visualiza el punto encontrado.

El punto de inflexión de la distancia mínima sugeriría que a partir de 2 segundos esta métrica no cambiará de forma drástica. Sin embargo, en la curva de la varianza este punto se localiza en 4 segundos. Con el fin de asegurar la mayor consistencia, sugerimos que para la generación de rutas se realicen 10 simulaciones con un tiempo máximo de cómputo por simulación de 4 segundos, para después escoger la que tenga la menor distancia asociada.

### *X-B. Resultados*

Después de correr la aplicación con los parámetros definidos en X-A, se desplegará el navegador el conjunto de rutas de

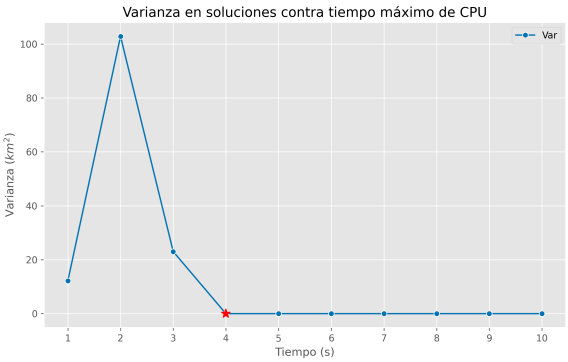


Figura 2: Varianza de soluciones encontradas para diferentes tiempos máximos de cómputo

todos los vehículos, en la figura 3 podemos ver 2 de las rutas encontradas.

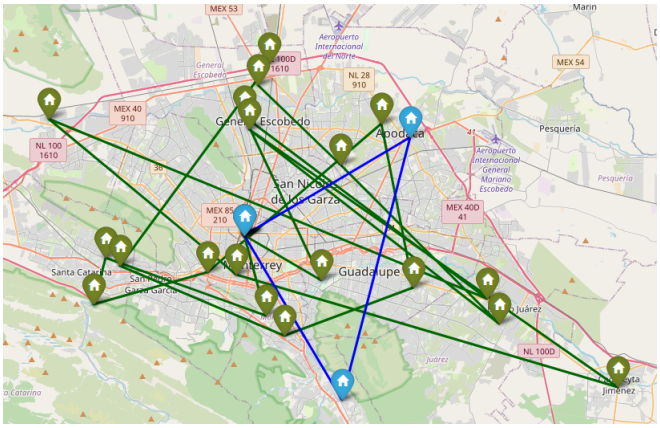


Figura 3: Ejemplo de rutas generadas

Como se puede apreciar en la tabla I, para esta simulación, la aplicación encontró un conjunto de rutas que minimizan la distancia que recorren los camiones en total hasta 226.79 km. En esta solución el vehículo con la ruta más corta viaja 12.86 $km$ y el que tiene la ruta más larga viaja 42.21 $km$, se tiene una huella de carbono de 55 km de $CO_2$; en este tabulado cada una de las filas representa una ruta óptima encontrada por la aplicación.

| N | Distancia ($km$) | Carga ($m^3$) | Vol. muerto ($m^3$) | Gas ($l$) | Huella $CO_2$ |
|---|---|---|---|---|---|
| 22 | 21.52 | 12.23 | 5.77 | 2.26 | 5.198 |
| 35 | 28.82 | 16.11 | 1.89 | 3.03 | 6.969 |
| 3 | 35.57 | 17.86 | 0.14 | 3.74 | 8.602 |
| 20 | 13.4 | 14.68 | 3.32 | 1.41 | 3.243 |
| 33 | 12.86 | 17.67 | 0.33 | 1.35 | 3.105 |
| 26 | 42.21 | 15.11 | 2.89 | 4.43 | 10.189 |
| 18 | 35.46 | 17.95 | 0.05 | 3.72 | 8.556 |
| 22 | 14.73 | 17.23 | 0.77 | 1.55 | 3.565 |
| 22 | 22.22 | 15.8 | 2.2 | 2.33 | 5.359 |

Tabla I: Rutas generadas por el programa

## XI. Discusión de resultados

A pesar de que el producto que hemos desarrollado ya es funcional, hemos encontrado algunas áreas de oportunidad que de ser resueltas incrementarían aún más el valor de la aplicación.

### XI-A. Geo-codificación de los domicilios

Actualmente la aplicación hace uso del motor de búsqueda de domicilios `Photon` [11] para geo-codificar los domicilios disponibles en la base de datos; consideramos que la adquisición de una licencia comercial podría aumentar aún más el número de domicilios que son geo-codificados con éxito.

### XI-B. Cálculo de distancia entre 2 puntos

La aplicación utiliza como métrica de distancia la distancia geodésica, que está relativamente lejos de ser una aproximación rigurosa para determinar la distancia entre 2 puntos. Sugerimos también que se adquiera una licencia comercial que calcule esta distancia. Su uso le daría una mayor validez a los resultados que encontramos.

### XI-C. Variabilidad de resultados

El límite de tiempo máximo impuesto al proceso de optimización en nuestra aplicación está ligado al número máximo de cálculos por segundo del equipo usado; en caso de que nuestro sistema sea implementado en un ordenador con un mayor poder de cómputo, el tiempo necesario para encontrar una solución óptima se vería reducido.

### XI-D. Tiempo de cómputo

El proceso de la aplicación podría ser condensado a 4 etapas, limpieza de datos, geo-codificación de domicilios y cálculo de matriz de distancias, optimización y despliegue de resultados. Se han realizado 10 simulaciones de las que se calculó el tiempo promedio de ejecución, dichos resultados se presentan en la tabla II:

| Proceso | Tiempo (s) |
|---|---|
| Limpieza de datos | 7.7 |
| Geo-codificación y matriz de distancias | 777 |
| Optimización | 42.3 |
| Despliegue de rutas | 1.18 |

Tabla II: Tiempos de cómputo de cada proceso

El tiempo total de ejecución es de 13 minutos con 48 segundos, tiempo del cual el proceso de geo-codificación representa el 93.8 % de la operación completa. Pensamos que el uso de un software más avanzado podría reducir drásticamente este tiempo de ejecución. Sin embargo, hay que destacar que el tiempo de geo-codificación puede ser mitigado con una base de datos en la que se lleve un registro de los domicilios de los clientes en conjunto con sus coordenadas asociadas.

## XII. ODS

Igual que encontrar la ruta más eficiente para la empresa, buscamos tener un impacto social por esta razón reconocemos la importancia que tiene que la innovación trascienda los objetivos personales, así como a los objetivos colectivos. Por eso mismo, tenemos la intención de tener un impacto positivo en los Objetivos de Desarrollo Sostenible propuestos por la ONU [16].

Una de las ODS implementadas en el proyecto es el objetivo número 9, Industria, Innovación e Infraestructura, ya que los valores de esta ODS se alinean perfectamente con los de nuestro proyecto. Este objetivo tiene como enfoque la construcción de infraestructuras resilientes en forma de rutas adaptables que se basan en las necesidades, la industrialización sostenible en forma de la minimización de kilómetros recorridos y por lo tanto energía ocupada, e innovación de un sistema ya existente.

Otro de los objetivos en los que impactamos positivamente es el número 11, Ciudades y Comunidades Sustentables, ya que reconocemos que el mundo está cada vez más urbanizado, y la sostenibilidad no se puede quedar atrás. El e-commerce está cada vez más presente en las vidas de la gente, y al minimizar los efectos negativos de esta industria es el mejor paso que podemos tomar para un futuro verde.

## XIII. Recursos utilizados

La implementación propuesta fue corrida en una máquina virtual de sistema operativo Ubuntu 20.04.3 LTS (Focal Fossa). El equipo disponía de un procesador Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz, tenía disponible 8 GB de RAM.

Especificaciones sobre los paquetes usados pueden ser encontrados en la documentación de nuestro repositorio A.

## XIV. Conclusiones

El ruteo de vehículos seguirá siendo uno de los desafíos más relevantes para la Investigación de Operaciones, el diseño de algoritmos eficientes que logren encontrar rutas de entrega óptimas con cantidades cada vez más grandes de puntos de entrega es un imperativo para cualquier departamento de logística.

Dado el gran número de puntos de entrega y de la enorme cantidad de posibles rutas a explorar, el uso de algoritmos heurísticos que logren encontrar rutas sub-óptimas es una alternativa a considerar para cualquier empresa; el que estos métodos puedan encontrar soluciones buenas en tiempos razonables hace que sean mucho más sencillos de implementar en producción.

La aplicación desarrollada es capaz de producir resultados cercanos a la optimalidad con recursos computacionales limitados, en un tiempo de operación aceptable, esta aplicación solamente se vería beneficiada si es que se ejecutase en un ordenador de carácter industrial, volviendo sencillo encontrar mejores soluciones en ventanas de tiempo cada vez más pequeñas, además que siguiendo la metodología y algoritmos que proponemos hemos encontrado que el mejor tiempo de procesamiento es de cuatro segundos, obteniendo resultados bastante favorables y una baja demanda en costo temporal y computacional.

## Apéndice A
## Código fuente

El código generado puede ser consultado aquí

## Referencias

[1] R. Elshaer and H. Awad, "A taxonomic review of metaheuristic algorithms for solving the vehicle routing problem and its variants," *Computers & Industrial Engineering*, vol. 140, p. 106242, 2020.

[2] G. Erdoğan, "An open source spreadsheet solver for vehicle routing problems," *Computers & Operations Research*, vol. 84, pp. 62–72, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0305054817300552
[3] J. Koetsier, "E-Commerce Jumped 55 % During Covid To Hit $1.7 Trillion," 04 2022. [Online]. Available: https://www.forbes.com/sites/johnkoetsier/2022/03/15/pandemic-digital-spend-17-trillion/?sh=35c76c7e5035
[4] M. M. Flood, "The traveling-salesman problem," *Operations Research*, vol. 4, no. 1, pp. 61–75, 1956. [Online]. Available: http://0-www-jstor-org.biblioteca-ils.tec.mx/stable/167517
[5] Google, "Routing Options — OR-Tools —," 2021. [Online]. Available: https://developers.google.com/optimization/routing/routing_options#local_search_options
[6] Wes McKinney, "Data Structures for Statistical Computing in Python," in *Proceedings of the 9th Python in Science Conference*, Stéfan van der Walt and Jarrod Millman, Eds., 2010, pp. 56 – 61.
[7] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.
[8] M. L. Waskom, "seaborn: statistical data visualization," *Journal of Open Source Software*, vol. 6, no. 60, p. 3021, 2021. [Online]. Available: https://doi.org/10.21105/joss.03021
[9] L. Perron and V. Furnon, "Or-tools," Google. [Online]. Available: https://developers.google.com/optimization/
[10] python visualization, "Folium." [Online]. Available: https://python-visualization.github.io/folium/
[11] C. Lingg, "photon," https://github.com/komoot/photon, 2022.
[12] K. Esmukov, "Geopy," https://github.com/geopy/geopy, 2022.
[13] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.
[14] K. Arvai, "kneed," https://github.com/arvkevi/kneed, 2022.
[15] V. Satopaa, J. Albrecht, D. Irwin, and B. Raghavan, "Finding a"kneedleïn a haystack: Detecting knee points in system behavior," in *2011 31st international conference on distributed computing systems workshops*. IEEE, 2011, pp. 166–171.
[16] United Nations, "THE 17 GOALS — Sustainable Development," 2015. [Online]. Available: https://sdgs.un.org/goals