Aplicación del TSP para la optimización de rutas de metro en Ciudad de México usando Recocido Simulado y Búsqueda Tabú

Semiramís G. de la Cruz[a,1] **y Federico Salinas Samaniego**[b,2]

[a] *CIMAT-MTY, semiramis.garcia@cimat.mx*
[b] *CIMAT-MTY, federico.salinas@cimat.mx*

Martha Selene Casas Ramírez, Alejandro Rosales Pérez

1. Introducción

La Ciudad de México enfrenta desafíos significativos en movilidad urbana debido a su denso y complejo sistema de transporte público. Según la Encuesta Origen-Destino de 2017 del INEGI, en la Zona Metropolitana del Valle de México, el transporte público es el medio más utilizado para ir al trabajo, con un 45 % de los viajes [1]. Sin embargo, las personas se enfrentan a una gran cantidad de opciones de rutas y conexiones, lo que complica la identificación de la mejor ruta.

Figura 1. Sistema de Transporte Colectivo Metro

El Problema del Viajante (TSP, por sus siglas en inglés) es un clásico problema de optimización que busca determinar la ruta más corta posible que visite un conjunto de ubicaciones y regrese al punto de partida. Aplicar el TSP al General Transit Feed Specification (GTFS) de la Ciudad de México [2] representa una oportunidad para optimizar las rutas del metro, como lo sugieren otros trabajos. El GTFS es un formato estándar que permite el intercambio de información sobre horarios y rutas de transporte público, facilitando analizar y mejorar la eficiencia en la planificación de rutas. [3]

La optimización se ha aplicado a problemas de transporte desde sus inicios, pero el abordaje con datos GTFS específicos de una ciudad es relativamente nuevo. Trabajos como el de Sikora (2018), que analiza la aplicación del TSP a sistemas de metro en otras ciudades, muestran que es posible reducir significativamente el número de pasos necesarios para cubrir todas las líneas de metro [4]. Así, el objetivo en este caso es que personas que tomen el servicio (sean usuarios recurrentes o no) hagan sus recorridos en el menor tiempo.

2. Descripción del problema

Como se ha mencionado con anterioridad, el TSP busca hallar la ruta más eficiente en términos de distancia o tiempo para visitar un conjunto de localizaciones establecidas, considerando que el agente viajero se traslada en un grafo completo, véase la figura 2, dado que se puede arribar a cualquier ubicación a partir del resto de las ubicaciones. Sin embargo, un sistema tipo metro no posee esta característica, porque no es posible trasladarse desde una estación a cualquier otra estación que no esté directamente conectada, lo que nos conduce a un grafo incompleto, véase la figura 3.

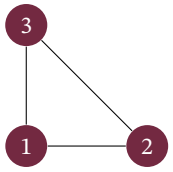

Figura 2. Ejemplo de grafo completo.

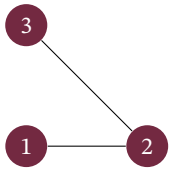

Figura 3. Ejemplo de grafo incompleto.

Por tanto, para calcular las *distancias* (o tiempos) se recurrió al uso del algoritmo de Dijkstra [5] para completar el grafo. Esto se logra asignando vértices para los pares de nodos no adyacentes con el costo de la ruta más corta obtenida mediante dicho algoritmo. Una vez obtenida una versión completa de nuestro grafo representativo del metro, el TSP puede ser descrito matemáticamente con los siguientes componentes.

2.1. Parámetros

- **Conjunto de nodos:** $N = \{1, 2, ..., n\}$ un conjunto de las estaciones que se deben visitar.
- **Matriz de tiempos de traslado:** $T = [t_{ij}]_{n\times n}$ una matriz $n \times n$, donde t_{ij} representa el tiempo de traslado entre la estación i y la estación j.

2.2. Variables de decisión

Serán de la forma x_{ij}, variables binarias, que serán 1 si el camino entre las estaciones i y j se incluye en la solución y 0 en caso contrario.

2.3. Función objetivo

Para el TSP, se busca minimizar el tiempo total de recorrido, que puede representarse como

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} t_{ij} x_{ij}$$

2.4. Factibilidad

Las restricciones del problema son las siguientes:

- Cada nodo es visitado exactamente una vez

$$\sum_{j=1, j\neq i}^{n} x_{ij} = 1, \forall i \in N$$

- Se regresa a la ciudad de origen

$$\sum_{i=1, i\neq j}^{n} x_{ij} = 1, \forall j \in N$$

- No existen sub-tours, es decir, viajes que no incluyan todas las ciudades.

$$u_i - u_j + n x_{ij} \leq n - 1, \forall i, j \in N, i \neq j, j > 1, u_1 = 1$$

Donde u_i es una variable que indica el orden en el cual se visitan las ciudades. Esta formulación garantizada que se recorran todas las estaciones con libertad de que algunas sean visitadas más de una vez y regrese al punto de partida, sin formar sub-tours. Nótese que esta formulación, aunque consiste en el TSP original, parte de que se haya realizado la completez del grafo bajo los resultados que el algoritmo de Dijkstra pueda brindar.

3. Metodología

3.1. Preparación de los datos

Previo al desglose de la propia metodología llevada a cabo para encontrar soluciones del TSP, es importante mencionar que los datos GTFS utilizados fueron pre procesados por partes con el fin de filtrar todas las instancias referentes a viajes, rutas, ubicaciones de parada y tiempos de parada asociados al metro. Posteriormente, con los datos extraídos se procedió a la construcción del grafo incompleto de todas las estaciones del sistema metro de la Ciudad de México. Para asignar los pesos de los vértices pertenecientes a una misma línea del sistema, se calculó la diferencia en tiempo de los pares consecutivos de estaciones por línea y sus valores absolutos fueron los utilizados como pesos de dichos vértices.

3.2. Métodos heurísticos

Las heurísticas son métodos de resolución de problemas que utilizan técnicas simplificadas y aproximadas para encontrar soluciones satisfactorias, si bien no necesariamente óptimas, en un tiempo razonable. Suelen aplicarse a problemas conocidos NP-duros, lo que significa que no se conoce un algoritmo que pueda resolver todos los casos de manera eficiente (en tiempo polinómico) para entradas de tamaño grande. El TSP entra dentro de la clasificación de problemas NP-duros, porque el número de posibles rutas crece factorialmente con el número de ciudades.

En el presente trabajo, se implementaron y evaluaron diversas heurísticas, destacando la recomendación de la literatura de utilizar Búsqueda Tabú y Recocido Simulado. [6]

3.2.1. Recocido Simulado

La meta-heurística del Recocido Simulado es usado, usualmente, como un procedimiento Monte Carlo con cierta semejanza al algoritmo Metropolis[7]. En este procedimiento, se intenta realizar una analogía en la cual las configuraciones de nuestras soluciones candidatas se consideraran como átomos de un sistema físico y en el cual se busca hallar aquella configuración que brinde el estado base o el estado de menor energía[8] que vendrían siendo configuraciones/soluciones óptimas de nuestro problema original.

En términos de optimización, el mecanismo del algoritmo permite que se acepten incrementos en la función de costo del TSP de una forma elegante utilizando evaluaciones de probabilidad para considerar o no soluciones con peores evaluaciones en dicha función aunque esperando que estas soluciones permitan evadir malos mínimos locales [9]; y al poder controlar estas probabilidades, mediante temperaturas artificiales, es que se podrán hallar más soluciones candidatas bajo un entorno controlado. Un pseudocódigo del recocido simulado (1) se muestra a continuación.

Algorithm 1 Algoritmo de Recocido Simulado

Require: $T_0 \geq 0 \quad \alpha \geq 0 \quad N \geq 0 \quad T_f \geq 0$
  $T \leftarrow T_0$
  $S_{act} \leftarrow$ Solución inicial
  while $T \geq T_f$ **do**
    for $k \leftarrow 1$ to N **do**
      $S_{cand} \leftarrow$ Nueva Solución(S_{act})
      $\delta \leftarrow Costo(S_{cand}) - Costo(S_{act})$
      if $U(0,1) < \exp(-\delta/T) \quad$ **or** $\quad \delta < 0$ **then**
        $S_{act} \leftarrow S_{cand}$
      end if
    end for
    $T \leftarrow \alpha T$
  end while
  return S_{act}

En el algoritmo tenemos un total de 4 parámetros de entrada que establecen la extensión de la búsqueda de soluciones candidatas:

- T_0 corresponde la temperatura inicial del sistema de soluciones candidatas.
- α es la razón de enfriamiento del sistema.
- N establece el número máximo de iteraciones por realizar, como máximo, para la búsqueda de candidatas.
- T_f es la temperatura mínima en la que se seguirán buscando soluciones.

Y respecto a las demás variables, S_{cand} y S_{act} corresponden a las soluciones candidata y actual del sistema por cada iteración realizada y δ es la diferencia de las evaluaciones del par de soluciones sobre la función de costo [10]. Para la generación de nuevas soluciones se optó por la búsqueda local **2-opt** [11] que consiste en capturar una sub-ruta aleatoria que podría, o no, estarse cruzando consigo misma y reordenarla para que no sea así; este procedimiento se realizará una única vez y no de forma iterada hasta hallar aquel intercambio más óptimo, puesto que se plantea que el mismo algoritmo de recocido simulado realice esta labor.

Algorithm 2 Algoritmo de búsqueda 2-opt

Require: ruta: La ruta actual
  NumNodos ← Tamaño(ruta)
  nueva_ruta ← ruta
  i ← Aleatorio(NumNodos)
  j ← Aleatorio(NumNodos)
  nueva_ruta[i:j] ← Revertir(nueva_ruta[i:j])
  return nueva_ruta

3.2.2. Búsqueda Tabú

La Búsqueda Tabú (TS del inglés *Tabu Search*) es un método meta-heurístico que puede utilizarse para resolver problemas de optimización combinatoria, como el TSP. Se caracteriza por permitir empeoramientos para escapar de óptimos locales y emplear mecanismos de reinicialización para mejorar la capacidad de exploración del espacio de búsqueda.

El esquema general de la búsqueda tabú, empieza las iteraciones escogiendo una solución local inicial y a partir de ésta se encontrará al conjunto de vecinos de la solución S, denotado por $N(S) \subset X$. En cada iteración se acepta siempre el mejor vecino de dicho entorno,

tanto si es peor como si es mejor que el de la solución actual S_{act}, mientras no sea una solución tabú.

Se define la lista de soluciones tabú como aquellas ya visitadas, son marcadas para no volver a ellas, eliminándolas del vecindario $N(S_{act})$. De ahí el término de memoria a corto plazo que identifica esta metaheurística que prohíbe determinados movimientos utilizados en las últimas iteraciones. El proceso se repite hasta el criterio de paso, que en este caso son las iteraciones máximas permitidas, como se muestra en el pseudocódigo descrito en (3).

Algorithm 3 Algoritmo de Búsqueda Tabú

Require: $max\_iters \geq 0 \quad tabu\_size \geq 0 \quad tabu\_list \leftarrow []$
  $S_{act} \leftarrow$ Solución inicial
  $S_{best} \leftarrow S_{act}$
  for $iter \leftarrow 1$ to max_iters **do**
    $N(S_{act}) \leftarrow$ GenerarVecinos(S_{act})
    for $S \in N(S_{act})$ **do**
      if $S \notin tabu\_list$ **then**
        Elegir mejor solución $S \in N(S_{act})$
        if $Costo(S) < Costo(S_{best})$ **then**
          $S_{best} \leftarrow S$
        end if
      end if
    end for
    $S_{act} \leftarrow S$
    Agregar a $tabu\_list(S_{act})$
    if $|tabu\_list| > tabu\_size$ **then**
      Eliminar el primer elemento de $tabu\_list$
    end if
  end for
  return S_{best}

4. Experimentos

4.1. Recursos

Para realizar los experimentos, se utilizaron dos equipos de cómputo con las siguientes características:

Equipo	Memoria (GB)	Procesador	OS
1	16	Intel Core i7-12650H	Windows 11
2	16	Intel Core i7-13620H	Fedora 40

Cuadro 1. Especificaciones del equipo de cómputo

Relacionado con el software, el lenguaje de programación utilizado para el desarrollo del proyecto fue Python y las paqueterías utilizadas en el mismo son las siguientes: **GeoPandas**, **Matplotlib**, **Networkx**, **Numpy**, **Pandas**, **Plotly** y **Shapely**.

4.2. Caso a evaluar

La instancia propuesta incluye todos los nodos cuyo grado es igual a 1, es decir, se visitarán todas las estaciones que solo tienen una estación conectada a ellas. Esto implica que solo habrá una vía de entrada a estas estaciones que cumplan con esta condición. A continuación, se muestra la lista de nodos de la ruta descrita, exceptuando la estación Buenavista, ya que esta se encuentra en el área de mayor concentración de estaciones.

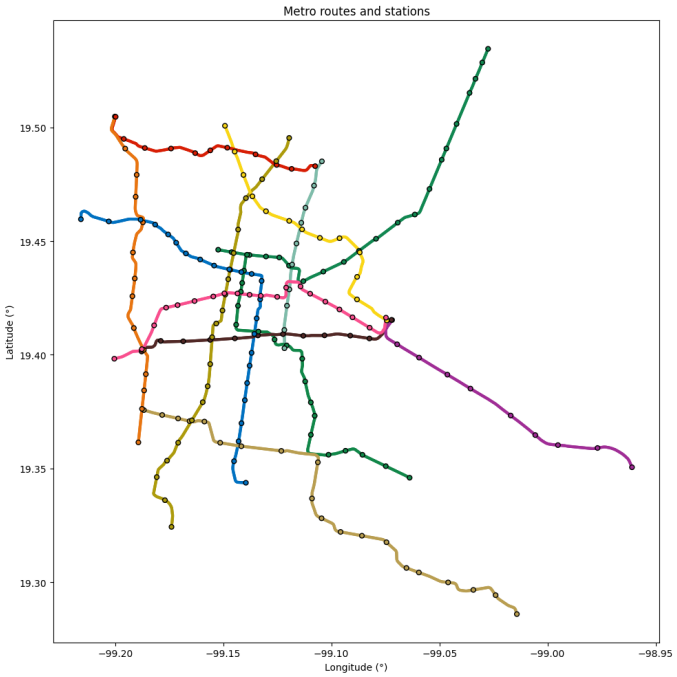

Figura 4. Grafo del Sistema de Transporte Colectivo Metro (elaboración propia)

- Observatorio
- Cuatro Caminos
- Politécnico
- Indios Verdes
- Ciudad Azteca
- La Paz
- Constitución de 1917
- Tláhuac
- Taxqueña
- Universidad
- Barranca del Muerto

4.2.1. Recocido Simulado

Para tener un entendimiento del comportamiento de las soluciones para el problema dada la heurística de recocido simulado, se plantea establecer los parámetros de temperatura inicial y temperatura mínima, T_0 y T_f, valores de 1000 y 1e-12, respectivamente. Para los parámetros restantes se tendrán un par de sub modelos que se muestran en seguida:

1. $\alpha = 0{,}85, N = 500$
2. $\alpha = 0{,}99, N = 3500$

Y tras realizar la búsqueda de soluciones con los hiperparámetros fijos y variables mencionados, se obtienen los siguientes resultados:

# Solución	Costo (Horas)	Tiempo
1	5.832222	0.013311
2	6.245000	0.013149
3	5.693056	0.013445
4	5.693056	0.012782
5	5.993056	0.013167

Cuadro 2. Resultados de 5 soluciones obtenidas mediante recocido simulado con parámetros $\alpha = 0{,}85$ y $N = 500$.

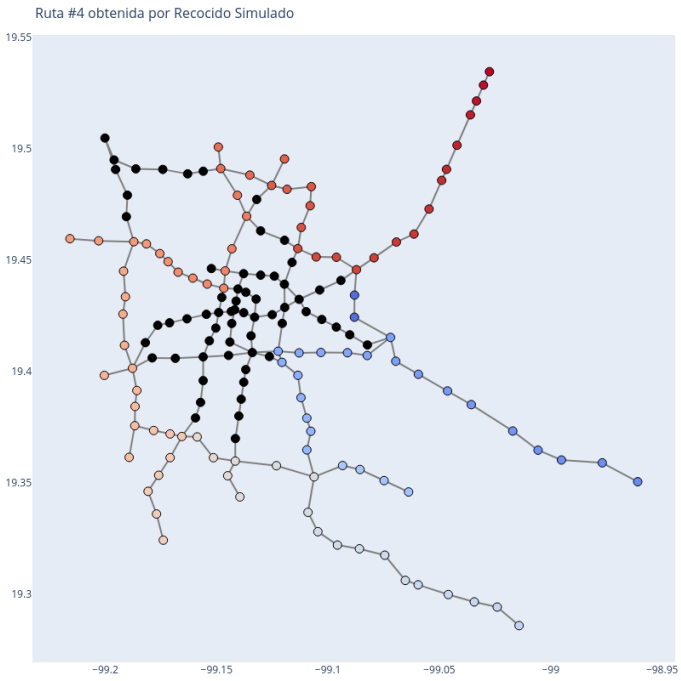

Figura 5. Mejor solución encontrada de la tabla 2 para los nodos de visita planteados mediante recocido simulado.

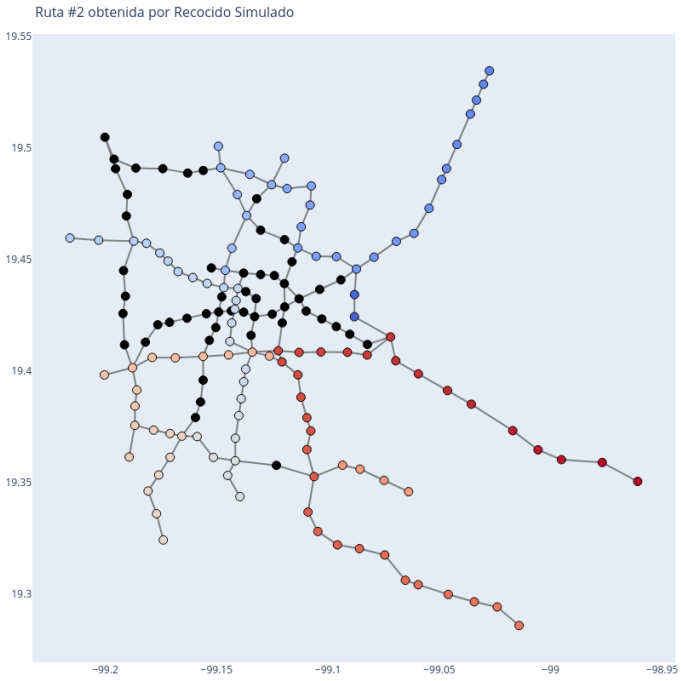

Figura 6. Peor solución encontrada de la tabla 2 para los nodos de visita planteados mediante recocido simulado.

# Solución	Costo (Horas)	Tiempo
1	5.693056	0.177517
2	5.693056	0.174258
3	5.693056	0.188916
4	5.693056	0.196538
5	5.693056	0.183417

Cuadro 3. Resultados de 5 soluciones obtenidas mediante recocido simulado con parámetros $\alpha = 0{,}99$ y $N = 3500$.

Respecto a estos mismos resultados es que se obtienen las siguientes métricas: tiempo de cómputo promedio y el promedio de la función de costo sobre las soluciones obtenidas por sub modelo.

# Modelo	Costo Promedio (Horas)	Tiempo Promedio
1	5.831277	0.013170
2	5.693056	0.184129

Cuadro 4. Costo promedio y tiempo de cómputo promedio del par de sub modelos del recocido simulado.

4.2.2. Búsqueda Tabú

En el caso de la Búsqueda Tabú, la programación solicita dos parámetros:

- $tabu\_size$: el tamaño de la lista tabú,
- max_iters: el criterio de paro.

Para ilustrar el efecto de los estos en el método, se probaron distintas combinaciones de pares de valores.

- $tabu\_size = \{5, 10, 15\}$
- $max\_iters = \{5, 10, 15\}$

Aplicando el método en el caso evaluado, se utiliza la Búsqueda Tabú (TS) en cinco trayectorias diferentes, cambiando la estación de inicio en cada caso. A continuación, se presentan los mejores y peores resultados.

# Solución	Costo (Horas)	Tiempo (Horas)
1	6.276389	0.017797
2	6.047222	0.018772
3	5.979722	0.018454
4	6.113333	0.019072
5	6.006944	0.019015

Cuadro 5. Resultados de 5 soluciones obtenidas mediante búsqueda tabú con $tabu\_size = 5$ y $max\_iters = 5$.

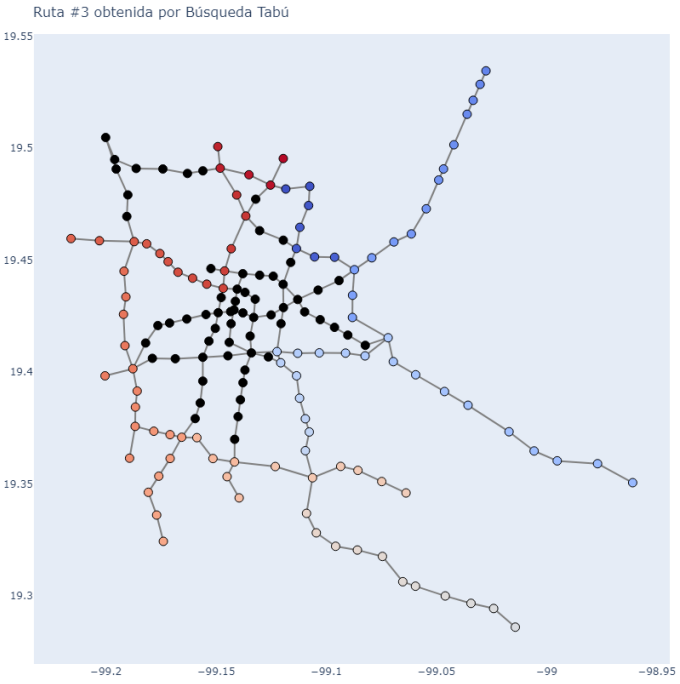

Figura 7. Mejor solución encontrada de la tabla 5 para los nodos de visita planteados mediante búsqueda tabú.

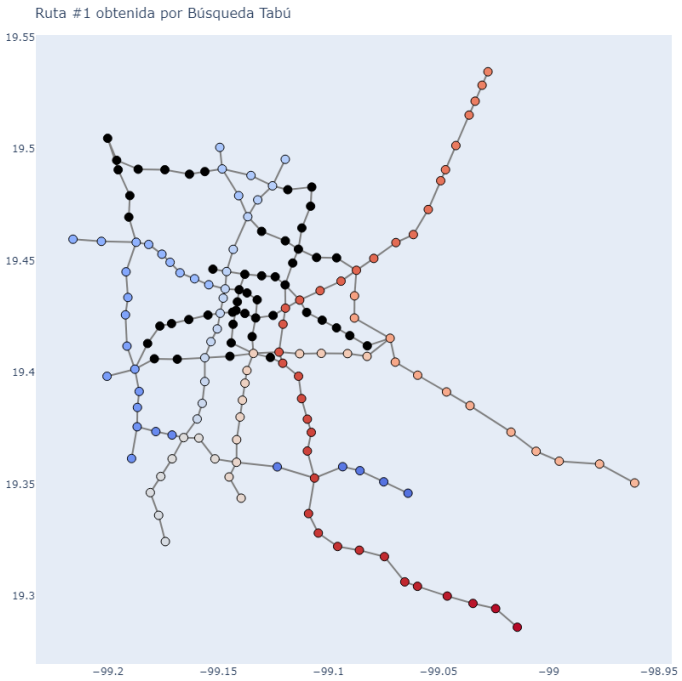

Figura 8. Peor solución encontrada de la tabla 5 para los nodos de visita planteados mediante búsqueda tabú.

# Solución	Costo (Horas)	Tiempo (Horas)
1	5.693056	0.045797
2	5.693056	0.046560
3	5.693056	0.046212
4	5.693056	0.048388
5	5.693056	0.051388

Cuadro 6. Resultados de 5 soluciones obtenidas mediante búsqueda tabú con $tabu\_size = 5$ y $max\_iters = 15$.

El tamaño de la lista tabú parece ser adecuado con 5 elementos y se observa que con 15 iteraciones se alcanza el tiempo óptimo. Podemos comparar el mejor y peor escenario en términos computacionales en el siguiente cuadro.

# Modelo	Costo Promedio (Horas)	Tiempo Promedio
1	6.084722	0.01825
2	5.693056	0.04766

Cuadro 7. Costo promedio y tiempo de cómputo promedio del par de submodelos de búsqueda tabú.

5. Conclusiones

En el presente trabajo, se presentaron dos métodos para optimizar las rutas del Sistema de Transporte Colectivo Metro de la Ciudad de México, facilitando los trayectos diarios de los usuarios. En primer lugar, se implementó la meta-heurística de Recocido Simulado, en el cual se propusieron dos submodelos con los que se revisó la sensibilidad del método para hallar soluciones a la ruta planteada y notando que una variación significativa en los hiperparámetros aseguraban la convergencia a una solución óptima que se logra encontrar de igual manera con la búsqueda tabú aunque con la diferencia de que los tiempos de cómputo es alrededor de 5 veces mayor respecto a la otra heurística.

En segundo lugar, se presentó el método de Búsqueda Tabú, que se adapta adecuadamente al TSP debido a su capacidad para evitar ciclos y estancamientos en la búsqueda.

Podemos llegar a varias conclusiones al comparar ambos algoritmos en términos de eficiencia y calidad de las soluciones obtenidas. Por un lado, la Búsqueda Tabú (Tabu Search, TS) puede ofrecer una convergencia más rápida, requiriendo un menor número de iteraciones para alcanzar una solución óptima o cercana a la óptima. Sin embargo, los experimentos muestran que este método presenta un mayor costo computacional en comparación con el Recocido Simulado (Simulated Annealing, SA). Mientras que SA puede ser más lento en términos de convergencia, su simplicidad y menor costo computacional lo hacen una alternativa viable en escenarios donde los recursos computacionales son limitados.

Referencias

[1] D. López, A. Lozano, H. González, A. Guzmán y F. Maldonado, «La movilidad en la Ciudad de México: Impactos, conflictos y oportunidades», *Universidad Nacional Autónoma de México*, 2018, Primera edición. dirección: http://www.publicaciones.igg.unam.mx/index.php/ig/catalog/view/149/138/712-2.

[2] Gobierno de la Ciudad de México, *GTFS Dataset - Datos Abiertos de la Ciudad de México*, [Accessed: 20-Abril-2024], 2024. dirección: https://datos.cdmx.gob.mx/dataset/gtfs.

[3] A. Martínez-Rebollar, C. A. Ruiz-Gutierrez, H. Estrada-Esquivel e Y. Hernández-Pérez, «Planificador de viajes de transporte público utilizando el estándar GTFS», *Boletín Científico INVESTIGIUM de la Escuela Superior de Tizayuca*, vol. 8, n.º Especial, págs. 102-110, 2022, Primer Congreso de Investigación e Innovación en Tendencias Globales, 26-28 de octubre de 2022. dirección: https://repository.uaeh.edu.mx/revistas/index.php/investigium/issue/archive.

[4] F. Sikora, «The shortest way to visit all metro lines in a city», *arXiv preprint arXiv:1709.05948*, 2018, Version 3, last revised 10 Apr 2018. dirección: https://arxiv.org/abs/1709.05948.

[5] E. W. Dijkstra, «A note on two problems in connexion with graphs», *Numerische Mathematik*, vol. 1, n.º 1, págs. 269-271, dic. de 1959, ISSN: 0945-3245. DOI: 10.1007/BF01386390. dirección: https://doi.org/10.1007/BF01386390.

[6] E. S. López, Ó. Murillo y A. Álex, «El problema del agente viajero: Un algoritmo determinístico usando búsqueda tabú», Español, *Revista de Matemática: Teoría y Aplicaciones*, 2014, ISSN: 1409-2433. dirección: https://www.redalyc.org/articulo.oa?id=45331281008.

[7] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller y E. Teller, «Equation of state calculations by fast computing machines», *The journal of chemical physics*, vol. 21, n.º 6, págs. 1087-1092, 1953.

[8] S. Kirkpatrick, C. D. Gelatt y M. P. Vecchi, «Optimization by Simulated Annealing», *Science*, vol. 220, n.º 4598, págs. 671-680, mayo de 1983. DOI: 10.1126/science.220.4598.671.

[9] C. C. Skiscim y B. L. Golden, «Optimization by simulated annealing: A preliminary computational study for the tsp», Institute of Electrical y Electronics Engineers (IEEE), inf. téc., 1983.

[10] «Simulated Annealing», en *Engineering Optimization*. John Wiley Sons, Ltd, 2010, cap. 12, págs. 181-188, ISBN: 9780470640425. DOI: https://doi.org/10.1002/9780470640425.ch12. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9780470640425.ch12. dirección: https://onlinelibrary.wiley.com/doi/abs/10.1002/9780470640425.ch12.

[11] G. A. Croes, «A Method for Solving Traveling-Salesman Problems», *Operations Research*, vol. 6, n.º 6, págs. 791-812, 1958. DOI: 10.1287/opre.6.6.791. eprint: https://doi.org/10.1287/opre.6.6.791. dirección: https://doi.org/10.1287/opre.6.6.791.