Instituto Tecnológico y de Estudios Superiores de Monterrey

MA2001B.101 - Optimización determinista

Semestre Febrero - Junio 2024

Reporte Técnico
Optimización de planes de viajes turísticos
OPTIMUM ROUTE
Expo Ingenierías

Alumnos

Ana Karen Márquez Escobar	A01028413
Rodrigo Leal Torres	A00836930
Juan Pablo Guzmán Segura	A01039810
Valeria Mariane Cárdenas Rodríguez	A01721814
Pablo Pérez Sanadoval	A01710355

Profesor Responsable
Dr. Fernando Elizalde Ramírez

31 de mayo de 2024
Monterrey, Nuevo León

1. Resumen

El proyecto se enfoca en desarrollar un plan de visita automatizado para turistas que visitan la ciudad de Puebla, México. El objetivo es proporcionar una experiencia eficiente y adaptada a ciertas preferencias, maximizando así la satisfacción del viajero. El proyecto aborda la necesidad de optimizar la planificación de rutas turísticas en Puebla, considerando factores como el tiempo disponible, el presupuesto, las preferencias de atracciones y las restricciones logísticas. Para la realización del trabajo se emplea un modelo matemático y computacional, y se espera obtener rutas turísticas óptimas que maximicen la satisfacción del cliente y minimicen los tiempos de trayecto. El proyecto cuenta con recursos informáticos adecuados, como una PC con capacidad de procesamiento y memoria suficientes, así como softwares específicos para el desarrollo y la implementación de los modelos matemáticos. La solución propuesta mejorará la experiencia turística en Puebla al proporcionar itinerarios eficientes y altamente satisfactorios , lo que contribuirá al desarrollo económico y social de la región al promover el turismo de manera más efectiva.

2. Introducción

El presente proyecto aborda la necesidad de desarrollar un plan de visita personalizado y automatizado para turistas que visitan la ciudad de Puebla, México. Puebla. Este destino es conocido por sus múltiples atractivos naturales y culturales, por lo que se enfrenta el reto de optimizar la experiencia turística de sus visitantes. Nuestro objetivo es proporcionar una herramienta que maximice la satisfacción del viajero mediante la creación de itinerarios personalizados que consideren factores como el tiempo disponible, el presupuesto, las preferencias de atracciones y las restricciones logísticas.

Para alcanzar este objetivo, se utiliza un modelo matemático y computacional que genera rutas turísticas óptimas, minimizando los tiempos de trayecto y maximizando la satisfacción del cliente. La herramienta propuesta no solo mejorará la experiencia individual del turista, sino que también contribuirá al desarrollo económico y social de la región al promover el turismo de manera más efectiva.

El proyecto se apoya en recursos informáticos adecuados, como una PC con capacidad de procesamiento y memoria suficientes, así como un software específico y construido con Python para el desarrollo e implementación de los modelos matemáticos. La automatización de la planificación de rutas turísticas con tecnologías avanzadas permitirá atender eficientemente las necesidades de los turistas. Esto tendrá un impacto positivo en el desarrollo económico, turístico y cultural de nuestro país.

3. Contexto general

El turismo en México es una de las actividades económicas más importantes del país y del mundo. México es reconocido por su alto potencial y riqueza en recursos naturales y culturales, lo que ha generado amplias expectativas en torno al turismo. Esta industria representa una actividad primordial en la estrategia económica del país, contribuyendo significativamente al producto interno bruto y generando numerosos puestos de trabajo para los locales. El turismo es una necesidad para todas las naciones, ya que contribuye en gran medida al desarrollo y crecimiento de un país. En México, tanto el turismo internacional como el local son vitales. En 2022, el sector turístico produjo 211 mil 327 millones de pesos, representando

un aumento de más de 90 mil millones de pesos respecto al año anterior (Wortev Capital, 2022). Esto resalta la importancia del turismo como un motor para el desarrollo sostenible de México.

Dentro del panorama turístico de México, la ciudad de Puebla destaca como un destino de gran relevancia. Con su rica historia, arquitectura colonial y una oferta gastronómica de renombre internacional, Puebla atrae a numerosos visitantes tanto nacionales como extranjeros cada año. En el primer semestre de 2023, la derrama económica fue de 9 mil 506 millones de pesos, lo cual representa un incremento del 33.3 % respecto al mismo periodo del año anterior. En cuanto a turistas, la capital del estado reportó alrededor de 5.57 millones de turistas, lo que representó un aumento del 26.2 % en comparación con el año anterior. Esta actividad turística generó una derrama económica de 6 mil 656 millones de pesos (Gobierno de Puebla, 2023).

Además de su capital, Puebla cuenta con la mayor cantidad de pueblos mágicos en México, incluyendo Cuetzalan, Pahuatlán y Zacatlán, entre otros. Estos pueblos no solo son destinos turísticos importantes, sino que también generan ingresos significativos que contribuyen al desarrollo sostenible de sus comunidades, proporcionando empleo y beneficios económicos para sus habitantes.

En este contexto, nuestro proyecto busca facilitar la planificación de rutas turísticas personalizadas en Puebla. Dado los numerosos lugares para visitar, los turistas enfrentan el desafío de decidir cuáles son prioritarios y en qué orden conviene visitarlos. Nuestra herramienta de automatización de rutas turísticas ayudará a simplificar esta decisión, mejorando la experiencia del usuario y aumentando su satisfacción. Al optimizar las visitas, esperamos contribuir positivamente al desarrollo económico, sostenible y turístico de Puebla.

4. Delimitación del objeto de estudio

La propuesta aborda el estudio de modelos matemáticos y computacionales destinados a mejorar el enrutamiento de viajes para los usuarios. Se tiene como enfoque la creación de un modelo y una plataforma que faciliten una experiencia de viaje más satisfactoria. Este estudio se centrará en viajes a una única ciudad y en la visita de una selección específica de lugares turísticos dentro de dicha ciudad. Además, se parte de la premisa de que los usuarios se hospedarán en un hotel especificado por el programa y deberán regresar a él al final de cada día. Asimismo, el usuario cuenta con la opción de ingresar el total de días de estancia, el presupuesto total del viaje y la cantidad de horas destinadas para turistear cada día.

La ciudad seleccionada para este estudio es Puebla de Zaragoza, en Puebla, México. El estudio incluye un total de 20 lugares de interés (sin contar el hotel) que formarán parte de las diferentes rutas planificadas. Estos lugares incluirán puntos destacados como el centro histórico, lugares característicos y sitios de interés cultural, entre otros. Con este enfoque se busca optimizar el itinerario de los usuarios en su viaje a Puebla, asegurando que puedan disfrutar al máximo de su estancia en la ciudad y visitar los lugares más relevantes de manera satisfactoria.

5. Planteamiento del problema

El presente proyecto tiene como objetivo desarrollar un modelo matemático y computacional que permita generar un plan de visita eficiente y personalizado para turistas en la ciudad de Puebla, México. El modelo computacional se encargará de crear una secuencia de lugares

a visitar, considerando el tiempo y el presupuesto disponibles, así como la satisfacción que genera visitar cada lugar. Por su parte, el modelo matemático optimizará el orden en que estos lugares deben ser visitados para maximizar la satisfacción del cliente.

El modelo computacional se basa en algoritmos Greedy, que siguen un método heurístico. Estos algoritmos eligen la mejor opción local en cada paso con el objetivo de encontrar una solución globalmente óptima. En nuestro caso, el algoritmo Greedy se utiliza para maximizar la satisfacción del turista, realizando pasos secuenciales hasta alcanzar un límite, definido por el tiempo máximo disponible para el recorrido. Posteriormente, se utiliza el modelo matemático del Problema del Viajante (TSP) para optimizar las rutas y minimizar el tiempo necesario para visitar todos los lugares seleccionados.

El problema central que se busca resolver es cómo tomar decisiones óptimas sobre las rutas turísticas para maximizar la satisfacción del usuario. Esto implica considerar restricciones establecidas por los usuarios, como el presupuesto y la duración del viaje, y desarrollar un algoritmo que genere rutas óptimas. La pregunta clave es cómo definir y calcular estas rutas para proporcionar la mejor experiencia posible al turista.

Con la combinación del enfoque Greedy y el modelo TSP, se busca maximizar la satisfacción del turista y minimizar el tiempo necesario para completar el itinerario, ofreciendo una solución integral y eficaz.

6. Justificación

El Estado de Puebla se destaca por tener incontables atractivos naturales y una amplia gama de actividades turísticas. La Secretaría de Turismo promueve destinos que contribuyen a la reactivación económica y ofrecen una experiencia turística segura, bajo los principios de solidaridad y respeto hacia las comunidades y el medio ambiente.

Al ser Puebla un destino turístico tan variado y diverso en la página web Visit Puebla, podemos encontrar guías turísticas para todo tipo de turismo. El turismo Outdoor en Puebla es una de las principales fuentes de ingreso del sector turístico, ya que existen actividades que van desde el rappel, parapente, parques de aventuras hasta hikes y visitas a lagunas vírgenes o lagunas termales. Otro de los llamativos más grandes de la ciudad de Puebla es el recorrido de sus 12 pueblos mágicos y el centro histórico, siendo este un recorrido lleno de color e historia. Actividades que nos muestran desde el inicio de las civilizaciones en Puebla hasta sus pueblos llenos de color y sabor como Atlixco o Zacatlán de las Manzanas. La automatización de este proceso a través de la tecnología permitirá mejorar la experiencia del turista y promover el turismo en Puebla de manera más efectiva.

7. Marco Teórico

Para iniciar con este problema tuvimos que aprender acerca de los conceptos básicos de la optimización determinista, teoría de grafos y solución de problemas mediante heurísticas.

Para mostrar un conocimiento más profundo, debemos conocer otros conceptos como el TSP o Problema del Agente Viajero (Travelling Salesman Problem) el cual es un problema clásico en el área de optimización que busca encontrar la ruta más corta posible que un vendedor debe seguir para visitar una serie ciudades y regresar a la ciudad de origen, visitando cada ciudad una sola vez, este es el ejemplo general del problema, sin embargo, se puede adaptar a distintos contextos para soluciones diferentes problemas. Este método de

optimización de tiempo es el métodos que utilizamos para reducir los tiempos de recorridos entre las distintas atracciones turísticas.

Para la selección de los puntos más satisfactorios utilizamos un métodos heurísticos. El método heruístico que nosotros utilizamos para la resolución del problema es conocido como búsqueda local codiciosa o greedy. Este método heurístico conciste en buscar el siguiente punto con el valor más alto de satisfacción. Tomando esto en cuenta podemos observar que la ruta arrojada no tiene en cuenta la optimización de los tiempos, ahí es donde aplicamos el TSP.

8. Objetivos

8.1. Objetivo General

Optimizar la satisfacción del usuario a lo largo de un recorrido turístico en Puebla, México, mediante el desarrollo de un modelo matemático y computacional.

8.2. Objetivos Específicos

- Seleccionar lugares turísticos que maximicen la satisfacción del usuario, basándose en sus preferencias y opiniones.
- Desarrollar un algoritmo que respete y tome en cuenta las restricciones de presupuesto y tiempo de viaje de cada usuario.
- Asegurar que la propuesta sea amigable y fácil de usar para los turistas que utilicen nuestro algoritmo.
- Implementar un sistema que genere itinerarios personalizados y eficientes, optimizando tanto la satisfacción del usuario como el tiempo disponible.

9. Hipótesis

La implementación de un algoritmo de búsqueda Greedy combinado con el algoritmo del Problema del Viajante (TSP) en Python permitirá generar una ruta óptima que maximice la satisfacción del usuario, considerando restricciones de costo y tiempo.

10. Metodología

10.1. Elección de ciudad

Lo primero que realizamos fue un análisis de las ciudades alrededor de México. En este análisis nos enfocamos en encontrar una ciudad que tuviera los suficientes puntos turísticos, estuvieran a distancias considerables entre si y que la movilidad no fuera una limitante, así es como nos optamos por la ciudad de Puebla.

Después de esta elección investigamos los 20 puntos turísticos más emblemáticos de la ciudad además de un hotel, que nos servirá como inicio y final de nuestro viaje. A partir de

la búsqueda, encontramos las siguientes atracciones turísticas las cuales están enumeradas en base a su nodo de identificación.

0. Hotel Casareyna
1. Catedral Basílica de Puebla
2. Capilla del Rosario
3. Mercado el Parián
4. Barrio del Artista
5. Zocalo de Puebla
6. Callejón de los Sapos
7. Iglesia de Nuestra Señora de los Remedios
8. Zona Arquelógica de Cholula
9. Templo de San Francisco Acatepec
10. Museo Fuertes de Loreto
11. Africam Safari
12. Calle de los Dulces
13. Casa del Alfeñique
14. Biblioteca Palafoxiana
15. Museo Amparo
16. Museo Internacional del Barroco
17. Pasaje Histórico del 5 de Mayo
18. Museo Regional de Cholula
19. Volcán Cuexcomate
20. Museo Regional De La Revolución Mexicana Casa De Los Hermanos Serdán

10.2. Construcción de la base de datos

Por la naturaleza del problema, la base de datos necesitaba contener las siguientes características:

- Satisfacción: La satisfacción se midió por medio de la combinación de las calificaciones otorgadas en TripAdvisor y Google Maps. Después de conseguir estos datos, tomamos el valor absoluto del promedio de calificaciones y lo dividimos entre el promedio de respuestas de cada lugar, posteriormente, le aplicamos a este valor un logaritmo natural y así conseguimos nuestra satisfacción.

$$Satisfaccion = |\frac{Promediodevaloración}{Promedioderespuestas}|$$

- Tiempo de recorrido: Con la ayuda de Google Maps calculamos el tiempo promedio en llegar de un lugar a otro.
- Distancia: Con la ayuda de Google Maps tomamos la ruta más corta. En caso de que la distancia fuera menor a 1.5 km decidimos que el individuo caminara. Y las distancias mayores a este valor serán recorridas en taxi.
- Costo del viaje: Con la ayuda de la base de datos de distancia creamos una nueva matriz de costos de transporte. Como lo explicamos en la distancia, los recorridos menores a 1.5 km serán hechos a pie, mientras que los trayectos más largos serán hechos en taxi.

  Para calcular la tarifa del taxímetro utilizamos la siguiente fórmula:

  $$40 + (cantidaddeKM * 5)$$

  Además del costo de transporte sumamos el costo por entrada a la atracción visitada.
- Tiempo: Con la ayuda de herramientas como Google Maps hicimos el cálculo del tiempo de punto A a punto B. Es importante mencionar que esta base de datos también se ve afectada por la distancia, ya que no es el mismo tiempo el que se tarda una persona en caminar una distancia al que se tarda en taxi.

10.3. Creación de modelo

Para la correcta utilización del modelo necesitamos especificar ciertas restricciones.

1. Los lugares solo pueden ser visitados una sola vez. A excepción del hotel que sera visitado todos los días dos veces. Una en la salida del recorrido y otra en el regreso después del recorrido.
2. La cantidad de días y el tiempo por día que el usuario querrá turistear por día, serán restablecidos por el usuario antes de iniciar su viaje.
3. El presupuesto del viaje completo tendrá que ser especificado antes de iniciar el viaje para después hacer un aproximado de lo que el usuario gastará diariamente en su viaje.
4. Se darán alrededor de 2 horas de tiempo de holgura para contabilizar el tiempo perdido en trayectos, atracciones o comidas.

Como justificación teórica a estas cuestiones decidimos apoyarnos con el modelo matemático del TSP. En donde las restricciones para este modelo estan expresadas de la siguiente forma.

- Representación de la función objetivo:

$$\text{mín} \sum_{i=0}^{n} \sum_{j \neq i, j=0}^{n} c_{ij} x_{ij} \tag{1}$$

- Los nodos solo se pueden visitar una vez:

$$\sum_{i=0}^{n} x_{ij} = 1 \quad j = 0, \ldots, n \tag{2}$$

$$\sum_{j=0, j\neq i}^{n} x_{ij} = 1 \quad i = 0, \ldots, n \tag{3}$$

- Ecuación de Subtours:

$$u_i - u_j + n x_{ij} \leq n - 1 \quad 1 \leq i \neq j \leq n \tag{4}$$

Este modelo matemático es la justificación teórica del modelo TSP y como es que este funciona de una manera tan eficiente.

10.4. Método de resolución

Para resolver este modelo fue necesario generar una búsqueda local codiciosa, comunmente conocida como Greedy. La implementación de este método fue con la ayuda de Python, creando una función totalmente desde cero para encontrar el siguiente punto más satisfactorio del arreglo de datos.

Además, gracias a la simpleza del método y a la construcción desde cero de este programa pudimos enfocar a que el programa se modele alrededor de las restricciones y los nodos propuestos en la ciudad, logrando una mejor solución a nuestro problema.

Al nosotros poder personalizar el modelo fácilmente, decidimos aventurarnos a la creación de un código que fue utilizado para minimizar los tiempos entre los recorridos arrojados por el Greedy. El modelo que utilizamos fue el TSP.

La personalificación del modelo nos permitió agregar directamente las restricciones como parte de la heurística utilizada para encontrar la combinación de puntos más satisfactoria con el tiempo y el presupuesto requerido.

Este modelo consiste en encontrar la ruta más corta entre una lista de puntos. Con la restricción de siempre empezar y terminar en el nodo 0 (Hotel Reina Sofía). A partir de toda esta planificación pudimos encontrar la ruta que llevó menos tiempo recorrerla para así optimizar el tiempo del traslado.

11. Propuesta metodológica a utilizar

Para la realización de nuestro proyecto, optamos por utilizar el algoritmo Greedy. Este método se basa en un proceso en el cual se toma la mejor decisión local en cada paso, con la intención de encontrar una solución global óptima. En el contexto de nuestro proyecto, el método Greedy se utiliza para seleccionar iterativamente los puntos turísticos más satisfactorios para el usuario.

El proyecto se desarrolla en el contexto del Problema del Viajante (TSP), que implica encontrar la ruta más corta que pasa por una serie de destinos y regresa al punto de inicio. El TSP realiza una serie de acciones basadas en criterios como la minimización de la distancia desde la posición actual.

El equipo decidió utilizar el método Greedy porque es útil cuando se combina con otras técnicas más sofisticadas, como los heurísticos, permitiéndonos mejorar la calidad de las soluciones en problemas complejos como el que abordamos en este proyecto. La combinación de Greedy con el TSP nos permite no solo maximizar la satisfacción del usuario seleccionando los mejores puntos turísticos, sino también optimizar el tiempo de recorrido entre estos puntos.

12. Técnicas y herramientas de ingeniería empleadas

En el proyecto descrito se implementaron diversas técnicas y herramientas de ingeniería, especialmente en el área de optimización de rutas. Se utilizó principalmente el algoritmo Greedy, que permite buscar de manera iterativa los lugares que mayor satisfacción brindarán al usuario, guiando así la planificación de la ruta en esa dirección. Este proceso se realizó de manera eficiente utilizando la herramienta Google Colab, una plataforma de máquinas virtuales que facilitó la implementación y prueba de nuestro algoritmo. Todo el código fue desarrollado en el lenguaje de programación Python (versión 3.10.12).

Adicionalmente, se aplicó la técnica del Problema del Viajante (TSP), la cual se basa en permutaciones combinatorias de los nodos (puntos de interés) en la ruta diaria. Esta técnica permitió buscar de manera iterativa la combinación de nodos que minimice el tiempo total de recorrido. Se emplearon estructuras de control como ciclos "forz "while", así como condicionales, para implementar el modelo. También se integraron bases de datos dentro del mismo algoritmo para gestionar y procesar la información relevante de manera eficiente.

13. Infraestructura

El desarrollo del proyecto se centró en la implementación de un código programado en Python, un lenguaje elegido por su amplia gama de librerías que facilitan la manipulación de datos y permiten la creación de métodos eficientes de búsqueda. Entre las librerías clave empleadas en el código se encuentran numpy, pandas e itertools, cada una desempeñando un papel crucial en el procesamiento y análisis de datos.

Para garantizar una experiencia óptima para el usuario, se adoptaron métodos avanzados de búsqueda y optimización. En primer lugar, se utilizó el método heurístico de búsqueda Greedy, reconocido por su efectividad en la resolución de problemas de optimización, ya que selecciona la mejor opción en cada paso de la búsqueda. Además, se implementó el método de optimización del Problema del Viajante (TSP), diseñado para encontrar la ruta más eficiente que pasa por una serie de puntos, optimizando así la satisfacción del usuario en términos de distancia y tiempo.

El proyecto se desarrolló utilizando Google Colab, una plataforma de máquinas virtuales que facilita el desarrollo colaborativo y permite el uso de recursos de cómputo avanzados sin necesidad de hardware especializado. Google Colab proporciona un entorno interactivo para escribir y ejecutar código Python, lo que resulta ideal para la implementación y prueba de algoritmos.

14. Recursos utilizados

Para la realización del proyecto, se requirieron diversos recursos digitales y materiales esenciales para la creación del programa. A continuación, se detallan los principales recursos utilizados:

14.1. Hardware

14.1.1. PC

- **Modelo:** HP Victus

- **Sistema Operativo:** Windows 11 PRO
- **Capacidad de Disco Duro:** SSD PCIe® NVMe™ TLC M.2 de 512 GB
- **Memoria RAM** 16 GB de RAM
- **Capacidad de Almacenamiento Adicional:** 1 TB
- **Tipo de procesador:** Intel core i7
- **Número de núcleos:** 8 núcleos

14.2. Software

- **Lenguaje de Programación:** Python 3.10.12
- **Entorno de Desarrollo:** Google Colab Versión 1.5.3

Para cargar las bases de datos correspondientes a las distancias, tiempos, costos y satisfacciones, se utilizó la PC mencionada, la cual cuenta con un sistema de procesamiento robusto, garantizando así una experiencia satisfactoria para el usuario. La ejecución del programa requirió el uso de varias librerías y métodos específicos en Python, incluyendo numpy, pandas e itertools.

Estas herramientas y recursos fueron fundamentales para el desarrollo del proyecto, permitiendo la implementación eficiente del algoritmo y el procesamiento de grandes volúmenes de datos.

15. Resultados

Después haber trabajado, en el desarrollo de nuestra propuesta se tuvo como resultado un algoritmo computacional en python en donde logramos unificar el método heurístico como el modelo matemático de optimización. La creacion primero de una función que busque la maximización de la satisfacción a los lugares a lo que ira a visitar. Siendo esto posible con nuestro algoritmo de greedy. Para despues poder tener los nodos que se van a visitar en ese dia para con la ayuda de nustro algoritmo de TSP heuristico poder ordenar estos lugares de manera que se reduzca el tiempo de trayecto en ese dia.
A continuacion se mostrara un ejemplo tangible de como neustra propuesta trabaja.
Nuestro algoritmo de phyton es nuestro resultado final en el cual es necesario poder proporcionar las bases de datos necesarias para hacer este ruteo de esa cuidad o localidad escogida. La idea de nuestros resultados es que esta idea que se ejemplifico en la cuidad de Puebla se pueda hacer en muchas mas ciudades de la república mexicana. Como también en otras partes del mundo.

Este resultado ejemplifica de manera muy sencilla la optimizacion de este proceso de turismo en las diferentes localidades. Y con ayuda de una base de datos mas grande y con convenios con cuidades se podria crear una red de ayuda a turistas. Pues nuestra propuesta es que el algoritmo se pueda implementar en cuidades al rededor de la república.
Nuestro algoritmo heutistico y matematico es un resultado conveniente para nustra propuesta.

Como resultado del proyecto obtuvimos un script de python en donde podemos introducir los valores personalizados de cantidad de días, tiempo por día y costo máximo por día.

Este Script lo que realiza es la ruta más satisfactoría con base a la investigación realizada anteriormente. Después de realizar la herística lo mejora mediante el TSP

Después de realizar este programa nosotros decidimos hacer una prueba, la realizamos con los siguientes:

- Cantidad de días: 3
- Cantidad de horas por día: 8 horas
- Costo máxmio del viaje: 4000

El resultado que nos arrojó el algoritmo con estos valores fueron los siguientes:

Día 1:

- Hotel - Zócalo - Africam Safari - Hotel
- Tiempo acumulado por día: 7.6 horas
- Costo acumulado por día: 623.5 pesos
- Satisfacción acumulada del día: 17.6814

Día 2:

- Hotel - Mercado del Pairan - Zona arqueológica de Cholula - Hotel
- Tiempo acumulado por día: 6.183333333333334 horas
- Costo acumulado por día: 392.5 pesos
- Satisfacción acumulada del día: 16.405

Día 3:

- Hotel - Callejón de los Sapos - Museo Amparo - Museo Internacional del Barroco - Capilla del Rosario - Hotel
- Tiempo acumulado por día: 5.95 horas
- Costo acumulado por día: 273.5 pesos
- Satisfacción acumulada del día: 28.0425

- Satisfacción acumulada del viaje: 62.1289
- Costo acumulada del viaje: 1289.5 (Presupuesto sobrante: 2710.5)

16. Conclusiones

A partir del proyecto elaborado pudimos observar que la combinación del método Greedy con modelos heurísticos fue fundamental para que la idea que planteamos tuviera una ejecución efectiva y viable para situaciones donde se prioriza la optimización de procesos. Al pesar de reconocer a este método como un proceso simple, la combinación de ideas y conocimientos del equipo permitió que se planteara un método más complejo y útil para el mundo contemporáneo, donde las decisiones basadas en procesos óptimos se ha convertido en prioridad para cada ser humano.

El uso de tecnologías ha transformado a la humanidad, por lo que utilizarlas es algo crucial para cualquier emprendimiento hoy en día. El equipo demostró esta capacidad al desarrollar el algoritmo a partir de una herramienta tecnológico como lo es Python. La implementación de conocimientos a esta herramienta no solo nos ayudó a simplificar operaciones matemáticas de larga duración de ejecución, sino también nos permitió generar un algoritmo complejo que demostrara el esfuerzo y los aprendizajes aplicados a este proyecto.

Comprendemos que tenemos áreas de mejora por trabajar como la implementación de otros tipos de heurísticas que permitan simplificar y sofisticar el proceso, mejorando la calidad de las soluciones, sin embargo, consideramos que el trabajo desarrollado por el equipo fue determinante para futuros retos o proyectos.

17. Bibliografía

Referencias

[1] Wortev Capital. *Turismo en México: tendencias que marcan al sector en este 2023*. 30 de noviembre de 2022. Recuperado de: https://wortev.capital/consumo-5-0/turismo-en-mexico-tendencias/

[2] Gobierno de Puebla. *Capta Puebla casi 9 millones de visitantes en siete meses gracias a estrategia estatal*. 17 de septiembre de 2023. Recuperado de: https://puebla.gob.mx/index.php/noticias/item/13427-capta-puebla-casi-9-millones-de-visitantes-en-siete-meses-gracias-a-estrategia-estatal#:~:text=Respecto\%20a\%20la\%20capital\%20del,ciento\%20con\%20relaci\%C3\%B3n\%20al\%20a\%C3\%B1o

[3] Kuo, M. *Algorithms for the Travelling Salesman Problem*. 8 de noviembre de 2023. Routific. Recuperado de: https://www.routific.com/blog/travelling-salesman-problem#:~:text=The\%20Traveling\%20Salesman\%20Problem\%20(TSP,and\%20optionally\%20an\%20ending\%20point.

[4] *LAS 10 MEJORES cosas que ver en Puebla (Actualizado 2024)*. 2024. Tripadvisor. Recuperado de: https://www.tripadvisor.com.mx/Attractions-g152773-Activities-Puebla_Central_Mexico_and_Gulf_Coast.html

[5] Néstor, G. *Las matemáticas y el turismo*. 6 de junio de 2023. LinkedIn. Recuperado de: https://es.linkedin.com/pulse/las-matem\%C3\%A1ticas-y-el-turismo-gustavo-n\%C3\%A9stor-fern\%C3\%A1ndez

[6] *Python and Greedy Algorithms*. 12 de octubre de 2023. Reintech. Recuperado de: https://reintech.io/blog/python-and-greedy-algorithms-tutorial

[7] Wortev Capital. *Turismo en México: tendencias que marcan al sector en este 2023*. 30 de noviembre de 2022. Recuperado de: https://wortev.capital/consumo-5-0/turismo-en-mexico-tendencias/

[8] Gobierno de Puebla. *Capta Puebla casi 9 millones de visitantes en siete meses gracias a estrategia estatal*. 17 de septiembre de 2023. Recuperado de: https://puebla.gob.mx/index.php/noticias/item/13427-capta-puebla-casi-9-millones-de-visitantes-en-siete-meses-gracias-a-estrategia-estatal#:~:text=Respecto\%20a\%20la\%20capital\%20del,ciento\%20con\%20relaci\%C3\%B3n\%20al\%20a\%C3\%B1o

18. Anexos

18.1. Código en Python

```
[ ]  # Importar librerias
     import pandas as pd
     import numpy as np
     import itertools

     # Importar las bases de datos necesarias para el ejemplo
     sat = pd.read_csv('sat.csv')
     tiempo = pd.read_csv('tiempo.csv')
     costo = pd.read_csv('costo.csv')

     # Convertir en string los indices de neustras bases de datos para ser manejadas de mejor manera
     sat.index = sat.index.astype('str')
     tiempo.index = tiempo.index.astype('str')
     costo.index = costo.index.astype('str')
```

Figura 1: Librerias y Datasets a utilizar

```
# Crear el diccionario con los lugares según cada nodo
diccionario = {
    '0' : ['Hotel'],
    '1' : ['Catedral Basílica de Puebla'],
    '2' : ['Capilla del Rosario'],
    '3' : ['Mercado el Parián'],
    '4' : ['Barrio del Artista'],
    '5' : ['Zócalo Puebla'],
    '6' : ['Callejón de los Sapos'],
    '7' : ['Iglesia de Nuestra Señora de los Remedios'],
    '8' : ['Zona Arqueológica de Cholula'],
    '9' : ['Templo de San Francisco Acatepec'],
    '10' : ['Museo Fuertes de Loreto'],
    '11' : ['Africam Safari'],
    '12' : ['Calle de los Dulces'],
    '13' : ['Casa del Alfeñique'],
    '14' : ['Biblioteca Palafoxiana'],
    '15' : ['Museo Amparo'],
    '16' : ['Museo Internacional del Barroco'],
    '17' : ['Pasaje Histórico del 5 de Mayo'],
    '18' : ['Museo Regional de Cholula'],
    '19' : ['Volcán Cuexcomate'],
    '20' : ['Museo Regional De La Revolución Mexicana Casa De Los Hermanos Serdán']
}

diccionario = pd.DataFrame(diccionario, index=['0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12', '13', '14', '15', '16', '17', '18', '19', '20'])
```

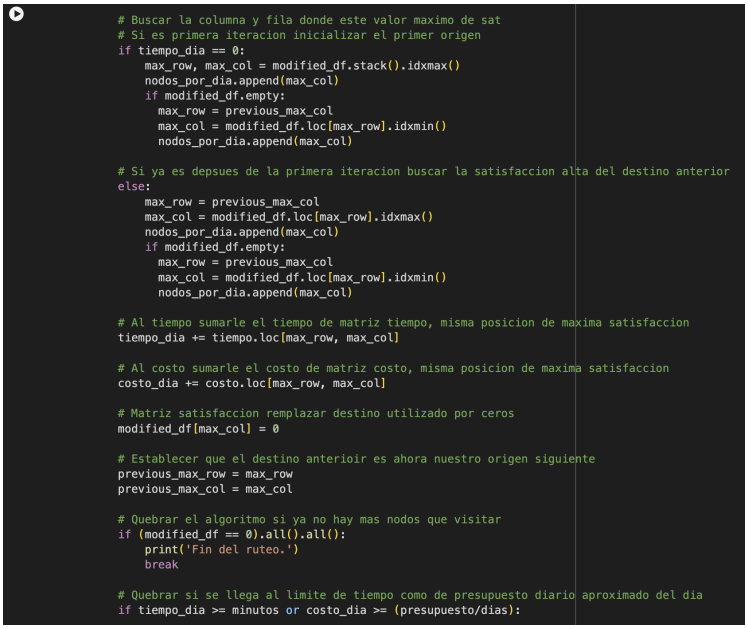
Figura 2: Diccionario

```python
def tsp(nodos_por_dia, tiempo_data):
    # Creacion de matriz solo con los nodos de ese dia
    tiempo_por_dia = tiempo_data.loc[nodos_por_dia, nodos_por_dia]

    # Remove node 0 from nodos_por_dia
    nodos_sin_cero = [n for n in nodos_por_dia if n != '0']

    # Generar todos las las combinaciones entre los nodos
    all_permutations_sin_cero = itertools.permutations(nodos_sin_cero)

    # Generar las variables vacias que se usaran
    min_route = None
    min_tiempo = float('inf')

    # Checar cada permutacion
    for permutation_sin_cero in all_permutations_sin_cero:
      # Meter el nodo 0 (hotel) and principio y al final del ruteo
      permutation = ('0',) + permutation_sin_cero + ('0',)

      # Inicializar la variable
      current_tiempo = 0

      # Ver el tiempo de ese ruteo
      for i in range(len(permutation) - 1):
        current_tiempo += tiempo_por_dia.loc[permutation[i], permutation[i + 1]]

      # Ver si es el tiempo mas chico se que puede tener
      if current_tiempo < min_tiempo:
        min_tiempo = current_tiempo
        min_route = permutation

    return min_route, min_tiempo
```

Figura 3: Función TSP

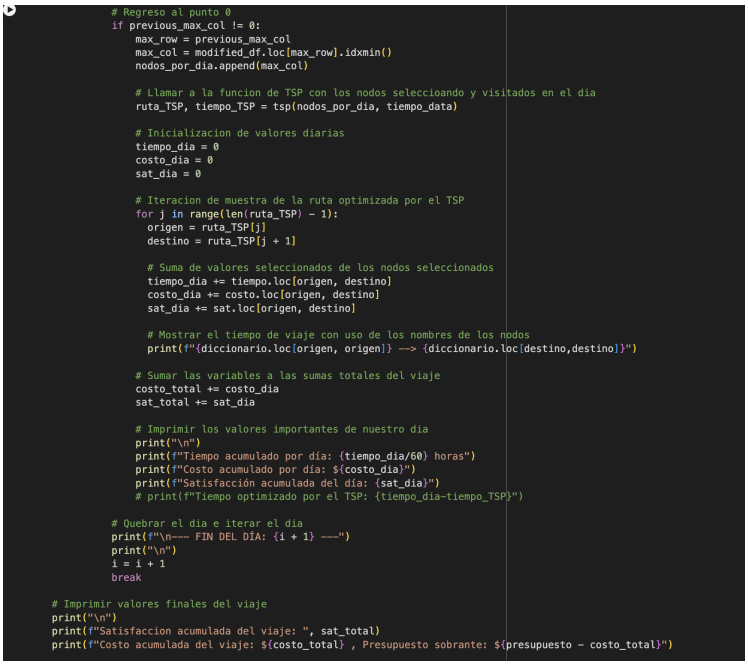
```python
def Max_Satisfaccion(sat, tiempo, costo, diccionario, horas, presupuesto, dias):
    # Creacion de una copia de matriz de tiempo para el uso correcto del TSP
    tiempo_data = pd.DataFrame(tiempo, index=[str(i) for i in range(tiempo.shape[0])])

    # Inicializacion de valores iniciales
    sat_total = 0
    tiempo_total = 0
    costo_total = 0

    # Obtencion de la media de los regesos al hote, colchon de tiempos
    media_regreso_hotel = tiempo.iloc[1:, 0].mean()
    # Conversion de el tiempo de restrinccion diaria
    minutos = ((horas - 2)*60) -  media_regreso_hotel*2

    # Copia que se modificara de la matriz de satisfaccion
    modified_df = sat.copy()
    modified_df = pd.DataFrame(modified_df)

    # Hacer la iteracion la canitdad de dias que el usuario decidio
    for i in range(dias):
      # Inicializacion de valores diarias
      tiempo_dia = 0
      costo_dia = 0
      sat_dia = 0

      # Parar la creacion de rutas
      if (modified_df == 0).all().all():
        print('Ya no se pueden crear mas rutas.')
        break

      # Creacion del conjunto para gurdar los nodos visitados diarios
      nodos_por_dia = []

      print(f"--- COMIENZO DEL DÍA: {i + 1} --- \n")

      # Ciclo por dia para elgoritmo de greedy
      while tiempo_dia < minutos and not modified_df.empty:

          # Enontrar el valor mas alto en la matriz de satisfaccion
          overall_max_value = modified_df.values.max()
```

Figura 4: Función Max Satisfacción 1.



```
            # Buscar la columna y fila donde este valor maximo de sat
            # Si es primera iteracion inicializar el primer origen
            if tiempo_dia == 0:
                max_row, max_col = modified_df.stack().idxmax()
                nodos_por_dia.append(max_col)
                if modified_df.empty:
                  max_row = previous_max_col
                  max_col = modified_df.loc[max_row].idxmin()
                  nodos_por_dia.append(max_col)

            # Si ya es depsues de la primera iteracion buscar la satisfaccion alta del destino anterior
            else:
                max_row = previous_max_col
                max_col = modified_df.loc[max_row].idxmax()
                nodos_por_dia.append(max_col)
                if modified_df.empty:
                  max_row = previous_max_col
                  max_col = modified_df.loc[max_row].idxmin()
                  nodos_por_dia.append(max_col)

            # Al tiempo sumarle el tiempo de matriz tiempo, misma posicion de maxima satisfaccion
            tiempo_dia += tiempo.loc[max_row, max_col]

            # Al costo sumarle el costo de matriz costo, misma posicion de maxima satisfaccion
            costo_dia += costo.loc[max_row, max_col]

            # Matriz satisfaccion remplazar destino utilizado por ceros
            modified_df[max_col] = 0

            # Establecer que el destino anterioir es ahora nuestro origen siguiente
            previous_max_row = max_row
            previous_max_col = max_col

            # Quebrar el algoritmo si ya no hay mas nodos que visitar
            if (modified_df == 0).all().all():
                print('Fin del ruteo.')
                break

            # Quebrar si se llega al limite de tiempo como de presupuesto diario aproximado del dia
            if tiempo_dia >= minutos or costo_dia >= (presupuesto/dias):
```

Figura 5: Función Max Satisfacción 2.



```
            # Regreso al punto 0
            if previous_max_col != 0:
                max_row = previous_max_col
                max_col = modified_df.loc[max_row].idxmin()
                nodos_por_dia.append(max_col)

                # Llamar a la funcion de TSP con los nodos seleccioando y visitados en el dia
                ruta_TSP, tiempo_TSP = tsp(nodos_por_dia, tiempo_data)

                # Inicializacion de valores diarias
                tiempo_dia = 0
                costo_dia = 0
                sat_dia = 0

                # Iteracion de muestra de la ruta optimizada por el TSP
                for j in range(len(ruta_TSP) - 1):
                  origen = ruta_TSP[j]
                  destino = ruta_TSP[j + 1]

                  # Suma de valores seleccionados de los nodos seleccionados
                  tiempo_dia += tiempo.loc[origen, destino]
                  costo_dia += costo.loc[origen, destino]
                  sat_dia += sat.loc[origen, destino]

                  # Mostrar el tiempo de viaje con uso de los nombres de los nodos
                  print(f"{diccionario.loc[origen, origen]} --> {diccionario.loc[destino,destino]}")

                # Sumar las variables a las sumas totales del viaje
                costo_total += costo_dia
                sat_total += sat_dia

                # Imprimir los valores importantes de nuestro dia
                print("\n")
                print(f"Tiempo acumulado por día: {tiempo_dia/60} horas")
                print(f"Costo acumulado por día: ${costo_dia}")
                print(f"Satisfacción acumulada del día: {sat_dia}")
                # print(f"Tiempo optimizado por el TSP: {tiempo_dia-tiempo_TSP}")

            # Quebrar el dia e iterar el dia
            print(f"\n--- FIN DEL DÍA: {i + 1} ---")
            print("\n")
            i = i + 1
            break

    # Imprimir valores finales del viaje
    print("\n")
    print(f"Satisfaccion acumulada del viaje: ", sat_total)
    print(f"Costo acumulada del viaje: ${costo_total} , Presupuesto sobrante: ${presupuesto - costo_total}")
```

Figura 6: Función Max Satisfacción 3.

```python
def Optimum_Route():
    # Dar la bienvenida a los usuarios
    print(f"¡BIENVENIDO A OPTIMUM ROUTE!")
    print(f"Es un placer atenderte")

    nombre = input("¿Cuál es tu nombre? ")
    print('\n')
    print(f"Hola, {nombre}!")
    print('\n')

    # Pide al usuario que ingrese la duracion de su viaje
    while True:
      try:
        dias = int(input("¿Cuántos días quieres que dure el tour? "))
        print('\n')
        break
      except ValueError:
        print("Por favor, introduce un caracter numerico: ")
        print('\n')

    # Pide al usuario que ingrese las horas de turisteo diarias
    while True:
      try:
        horas = int(input("¿Cuántas horas al día te gustaría turistear? "))
        if 1 <= horas <= 24:
          print('\n')
          break
        else:
          print('Lo siento, introduce un número entre el 1 y el 24, por favor')
          print('\n')
      except ValueError:
        print("Lo siento, por favor introduce un caracter numerico: \n")
```

Figura 7: Función Optimum Route 1.

```python
    # Pide al usuario que ingrese la duracion de su viaje
    # Pide el presupuesto del viaje
    while True:
      try:
        presupuesto = float(input("¿De cuánto es tu presupuesto para movilidad y de paga de lugares a conocer?"))
        print('\n')
        break
      except ValueError:
        print("Lo siento, por favor introduce un caracter numerico: \n")

    print(f'Esta va a ser tu ruta de viaje {nombre}')
    print('\n')


    # Ahora puedes llamar a la función con los valores adecuados
    Max_Satisfaccion(sat, tiempo, costo, diccionario, horas, presupuesto, dias)

    print('\n')
    print(f'¡Muchas gracias por elegirnos, Optimun Routes está feliz de tenerte!')
```

Figura 8: Función Optimum Route 2.

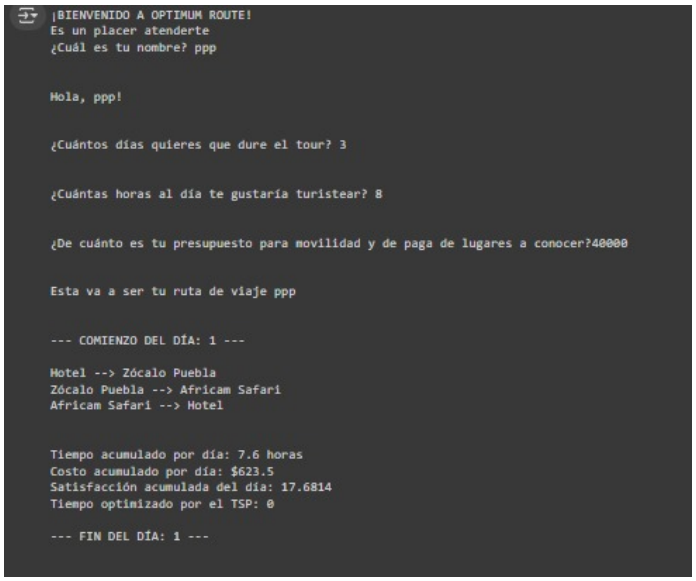


Figura 9: Resultados 1.

Figura 10: Resultados 2.

18.2. Objetivo

- Minimizar la distancia en la ruta.

18.3. Conjuntos

- **Vértices**: $V = \{1, 2, ..., n\}$, n = número de nodos
- **Aristas**: $A(i, j)$ donde $(i, j) \in V$

18.4. Restricciones

- Solo se puede llegar una vez a cada nodo (o lugar), a excepción del nodo incial (el hotel).
- Solo se puede salir una vez de cada nodo (o lugar), a excepción del nodo incial (el hotel).
- Un nodo será insertado dentro de una arista del subtour solo si esa arista se generó por una inserción previa y está disponible.

18.5. Parámetros

- $d(i, j)$: distancia por transitar el arco (i, j)

18.6. Variables

- $x(i, j) = \left\{ \begin{array}{l} 1 \text{ Si se va de i a j} \\ 0 \text{ otro caso} \end{array} \right\}$
- $u(i)$: momento en que fue visitado i

18.7. Modelo matemático

$$\min z = \sum_{i=1}^{n} \sum_{j=1}^{n} d(i, j) * x(i, j)$$

s.a.:

$$\sum_{j=1}^{n} x(i, j) = 1 \ \ \forall \ \ i \neq 0$$

$$\sum_{i=1}^{n} x(i, j) = 1 \ \ \forall \ \ j \neq 0$$

$$u(i) - u(j) + N * x(i, j) \leq N - 1 \ \ \forall \ \ (i, j) \ \ 2 \leq \ i \neq j \ \ < N$$