Universidad Complutense
Madrid

Facultad de Informática

Trabajo Optativo

Travelling Salesman Problem

Autores:
Guillermo Cortina Fernández
Iván Fernández Mena
Alberto Pastor Moreno
Montserrat Sacie Alcázar

Profesora:
Rosa María Ramos Domínguez

Modelos Operativos de Gestión

21 de diciembre de 2018

Índice

1. Introducción

1.1. Sobre este documento

En este documento se expone el problema del Viajante de comercio (*TSP* por sus siglas en inglés, *Travelling Salesman Problem*) con la intención de:

- Definir su origen técnico y la motivación que llevó a su resolución.
- Exponer brevemente el contexto histórico en el cual fue planteado.
- Conocer qué necesidades reales dan lugar a su origen y su relación directa con los problemas de rutas vistos en la asignatura.
- Formular y modelizar el problema.
- Clasificar los algoritmos que han sido desarrollados para su resolución y profundizar en algunos de ellos, entendiendo paso a paso su ejecución.
- Ofrecer una aplicación informática que implemente un algoritmo de resolución de los expuestos.
- Exponer la complejidad computacional del problema TSP

Finalmente, serán descritas una serie de conclusiones que han sido obtenidas en el desarrollo de este proyecto, planteando en ellas futuros retos los cuales quedan actualmente por resolver, para los apasionados de este problema matemático.

1.2. Sobre el problema

En el problema del Viajante de comercio se plantea la siguiente situación:
Un comerciante desea visitar n ciudades, comenzando y terminando el viaje en la misma ciudad Conocemos la distancia entre cada ciudad además de sus conexiones. Se desea conocer el recorrido a seguir para que la distancia total recorrida sea mínima visitando todas las ciudades, el cual queda expuesto en el contexto del problema como el camino óptimo.

Se engloba dentro de los "Problemas de Rutas" en los que se busca una ruta óptima para satisfacer la demanda de un conjunto de clientes.

1.3. Historia del problema

El problema del viajante de comercio fue definido en los años 1800s por el matemático irlandés William Rowan Hamilton y por el matemático británico Thomas Kirkman. Sin embargo, no es hasta 1930 cuando varios matemáticos en Viena y Harvard plantean una formulación matemática acorde a la problemática correspondiente, naciendo así la cuestión que aquí tratamos: ¿Cómo encontrar el camino óptimo? Hassler Whitney de la Universidad de Princeton introdujo el nombre "travelling salesman problem", TSP, poco tiempo después. Durante los años 1950 a 1960, el problema incrementó su popularidad entre el círculo de científicos de Europa y Estados Unidos. En este período de tiempo se expresa este problema como un problema de Programación Lineal con enteros, para el cual desarrollaron una solución envuelta por el método de Planos Cortantes. Con este nuevo método, resolvieron una instancia con 49 ciudades, óptimamente, mediante la construcción de un recorrido y probando que no había un recorrido que pudiera ser más corto. En las siguientes décadas, el problema fue estudiado por muchos investigadores, matemáticos, científicos de la computación, químicos, físicos, etc.
Observamos que ha sido uno de los problemas más desafiantes en Investigación Operativa y estudiados históricamente por su aplicación real, principalmente en la logística y distribución de mercancías.

1.4. Aplicaciones

A pesar de que la problemática propuesta plantea una situación muy concreta, podemos asemejar las ciudades del problema así como las distancias de las conexiones entre ellas a otra serie de problemas comunes en el contexto de la modelización de problemas de gestión. Por ello, una gran serie de problemas han sido asemejados al TSP. Desarrollamos a continuación brevemente las aplicaciones más importantes del TSP:

- Problema de scheduling: hay T tareas que realizar y m procesadores. Se busca una planificación en m procesadores para T minimizando el tiempo.Además se pueden plantear restricciones de precedencia entre las tareas, por ejemplo, dando lugar a otras variantes del problema. Para plantear el problema como un TSP, las tareas equivaldrían a las ciudades y los tiempos a las distancias entre ellas.
- Problema de la placa de circuitos impresos: un problema de gran importancia que puede enfocar en dos subproblemas; el orden óptimo de taladrar las placas y los caminos óptimos que comunican los chips. Para el primer subproblema, las posiciones a perforar corresponderían a las ciudades del TSP y las distancias entre ellas serían el tiempo que tarda la máquina de perforar en trasladarse de una posición a otra. El punto origen y destino del camino óptimo será un punto adicional donde descansa la máquina fuera de la placa.
  Por su lado, el objetivo del problema de conexión de chips es minimizar la cantidad de cable necesaria para unir todos los puntos de la placa evitando cualquier interferencia
- Estructura de la red de Internet: En internet los bits de datos que circulan por la red pueden visualizarse como el viajante de comercio del TSP y las ciudades a su vez equivalen a los servidores web distribuidos por todo el planeta. Se pretende usar de forma óptima esta plataforma masiva de distribución estableciendo rutas óptimas que conectan a conjuntos de servidores.
- Problema de red de basuras: tenemos varios puntos de recogida de basura y pueden plantearse diversos objetivos como minimizar ruta de recogida de residuos, minimizar número de camiones que atienden diferentes puntos, etc.

En definitiva , existen innumerables aplicaciones que solucionan problemas cotidianos. La atención de llamadas de emergencia, rutas de buses escolares, secuenciamiento de genes u ordenamiento de observaciones en telescopios, aplicado por la NASA; junto con las aplicaciones explicadas anteriormente son solo algunas de ellas.

2. Modelización y Formulación

2.1. Modelización

Una vez hemos profundizado en la historia del problema del viajante y qué problemática pretende resolver, continuamos nuestro proyecto analizando como podemos modelizarlo para su posterior formulación y resolución, En la modelización vamos a representar las ciudades a visitar como nodos de un grafo completo(en caso contrario, siempre podemos añadir una arista ficticia entre dos vértices con coste del camino más corto que los une) y las aristas que unen los nodos del grafo serán las calles por las que puede pasar el viajante para ir de una ciudad a otra. Por último los pesos de las aristas corresponden con las distancias de las calles a recorrer.
Así seleccionamos un nodo origen y buscamos el recorrido mínimo que pase una única vez por cada nodo del grafo, volviendo al nodo origen, formando un tour. Esto es lo que conocemos de manera general como ciclo Hamiltoniano.

2.2. Formulación

Una de las formulaciones más importantes y extendidas es la planteada a continuación: Sea G = (V, A) un grafo completo. Los vértices i = 2, . . . , n se corresponden con las ciudades a visitar y el vértice 1, con la ciudad de origen y destino. A cada arco (i, j), se le asocia un valor no negativo cij , que representa el coste de viajar del nodo i al j con i n $i \neq j$ para toda n.
Si G es un grafo dirigido, la matriz de costes C es asimétrica, sin embargo si cij = cji para todo (i, j) $\in$ A entonces la matriz de costes será simétrica, y el problema recibirá el nombre de TSP simétrico (STSP).

En este caso, el conjunto A se sustituye por el conjunto E de arcos no dirigidos i, j con i menor que j. El problema es encontrar el orden en que visitamos cada nodo, de forma que la suma de costos de las aristas que se recorren sea mínimo. Una cuestión más es que el nodo origen que elijamos es inmaterial y sin pérdida de generalidad podemos elegir cualquier nodo para que sea el origen, y de ahí podemos seguir a n - 1 cualesquiera nodos.A su vez para cada uno de esos n - 1 nodos que se pueden elegir, se podría viajar a cualquiera de los n - 2 nodos restantes.De modo que el número total de posibles tours en un problema de n ciudades es (n-1)(n-2)...1=(n-1)!

Vamos a definir un conjunto de variables de decisión xij $\in$ 0, 1 para cada (i, j) $\in$ A, tal que xij = 1 si el arco (i, j) forma parte de la solución, xij = 0 en caso contrario. Encontramos ya entonces la función objetivo:

$$min \sum_{i \neq 1} c_{ij} xij,$$

sujeto a las condiciones

$$\sum_{j \in \delta^-(i)} x_{ij} = 1 \forall i \in V,$$

$$\sum_{j \in \delta^+(i)} x_{ij} = 1 \forall i \in V,$$

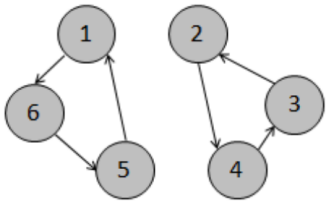
donde

$$\delta^-(i) = \{a = (j, i) \in A\},$$

$$\delta^+(i) = \{a = (i, j) \in A\},$$

La primera restricción nos dice, que a cada vértice, solo puede llegar un arco y la segunda determina que de cada vértice, solo puede salir un arco. Estas restricciones, son necesarias pero no suficientes, pues podría dar lugar a subcircuitos dentro del grafo.

En un caso ejemplo, las variables son $x_1 2 = x_2 3 = x_3 4 = x_4 5 = x_5 6 = x_6 7 = 1$, luego se cumplen las restricciones arriba mencionadas, es decir, a cada vértice solo entran y salen como mucho un arco, pero sin embargo no es un camino válido para el TSP.



Necesitamos por lo tanto añadir alguna restricción más al problema. En el ejemplo anterior, vemos que para el subconjunto 1, 2, 3, hay tres arcos que unen los nodos entre sí. Si limitamos el número de arcos a dos en todos estos subconjuntos, entonces evitaríamos subcircuitos dentro del grafo. Para modelizar estas situaciones, es necesario definir un nuevo conjunto:

$$\forall W \subsetneq V, \quad A(W) = \{a = (i,j) \in A | i,j \in W\}$$

Así, las restricciones de ruptura de subcircuito son las siguientes:

$$\sum_{i,j \in A(W)} X_i j \leq |W| - 1 \quad \forall W \subseteq V, 2 \leq |W| \leq n/2,$$

equivalente a

$$\sum_{i \in W, j \notin W} X_i j \geq 1 \quad \forall W \subseteq V, 2 \leq |W| \leq n/2,$$

La primera restricción obliga a que cualquier subconjunto de vértices del grafo ≤ —W— tenga a lo sumo ≤ —W—-1 aristas. Esta restricción es equivalente a la segunda pues para cada subconjunto de vértices, al menos uno de los arcos debe llegar a un nodo que no se encuentra en ese subconjunto, evitando así que ≤ —W— queden dentro del subconjunto.

2.3. Variantes

El TSP planteado hasta ahora puede resolverse con un único viajante o vehículo con suficiente capacidad para visitar cada una de las ciudades llegando finalmente al punto de origen. Sin embargo, podemos plantear una nueva situación en la que no sea suficiente con un vehículo. Esta y otras nuevas restricciones dan lugar a nuevos problemas de rutas, de gran utilidad en la realidad y modelizables a partir del modelo planteado, añadiendo algunas varibles y restricciones nuevas a las ya formuladas. Distinguimos algunas de las principales variantes:

- Problemas de rutas de vehículos con capacidades idénticas(CVRP, Capacitated Vehicle Routing Problem): varios vehículos los cuales disponen de igual capacidad parten de un nodo central o almacén.A cada uno se le asigna una ruta, respetando que la carga de cada vehículo no exceda su capacidad máxima y que la distancia recorrida por los vehículos sea mínima.
- Problema con recogidas(VRPB, Vehicle Routing Problem with Backhauls): en el cual los vehículos pueden vaciarse a lo largo del recorrido pero también pueden existir clientes para los que tengamos que hacer recogidas, por lo tanto en este caso aparecen nuevos problemas de llenado/vaciado del vehículo.
- Problemas de vehículos con ventanas horarias(VRPTW, Vehicle Routing Problem with Time Windows): en el cual cada cliente tiene asociada una franja horaria en la que puede ser visitado, aparecen cálculos y factibilidad de rutas de gran complejidad.
- Problemas con múltiples almacenes(VRPMD, Vehicle Routing Problem with Multiple Depots): en este caso hay varios almacenes y cada ruta puede partir y terminar en un almacén.

Todas estas variantes del problema del viajante pueden formularse añadiendo restricciones y variables nuevas al modelo original del TSP.

3. Algoritmos resolución TSP

La complejidad del cálculo del problema del agente viajero ha despertado múltiples iniciativas por mejorar la eficiencia en el cálculo de rutas. El método más básico es el conocido con el nombre de fuerza bruta, que consiste en el cálculo de todos los posibles recorridos, lo cual se hace extremadamente ineficiente y casi que se imposibilita en redes de gran tamaño.

- **Método de fuerza bruta:**
  El método de la fuerza bruta no implica la aplicación de ningún algoritmo sistemático, tan solo consiste en explorar todos los recorridos posibles. Considerando la siguiente red simétrica, los caminos posibles se reducen a la mitad.

Desde los años 50, muchos métodos han sido aplicados para la resolución del TSP. En general estos métodos pueden ser clasificados en dos categorías: algoritmos aproximados y algoritmos exactos.

Los algoritmos exactos usan modelos matemáticos, mientras que los algoritmos aproximados buscan soluciones usando heurísticas o metaheurísticas y mejoras iterativas. Ejemplos de algoritmos exactos son: Ramificación y poda (Branch and Bound), Relajación Lagrangiana y Programación Entera. Dentro de los algoritmos aproximados tenemos, por ejemplo, la heurística del vecino más próximo, heurística de Greedy o heurísticas de inserción, entre otras. Por último, en cuanto a ejemplos de heurísticas de mejora tenemos k-opt, Recocido Simulado, Búsqueda Tabú o Algoritmos Genéticos.

Se denomina heurística al arte de inventar. En programación se dice que un algoritmo es heurístico cuando la solución no se determina en forma directa, sino mediante ensayos, pruebas y reensayos.

El método consiste en generar candidatos de soluciones posibles de acuerdo a un patrón dado, luego los candidatos son sometidos a pruebas de acuerdo a un criterio que caracteriza a la solución.

3.1. Algoritmos heurísticos de construcción

- **Método del vecino más cercano:**
  El método del vecino más cercano es un algoritmo heurístico diseñado para solucionar el problema del agente viajero, no asegura una solución óptima, sin embargo suele proporcionar buenas soluciones, y tiene un tiempo de cálculo muy eficiente. El método de desarrollo es muy similar al utilizado para resolver problemas de árbol de expansión mínima. El método consiste en una vez establecido el nodo de partida, evaluar y seleccionar su vecino más cercano.

3.2. Algoritmos exactos

- **Método de branch and bound - WINQSB**
  El método de branch and bound (ramificación y poda), nos proporciona una solución óptima del problema del agente viajero, calculando mediante el algoritmo simplex la solución del modelo.
  A medida que aumente el tamaño de la red el método puede tardar gran cantidad de tiempo en resolverse, sin embargo para redes de mediano tamaño es una excelente alternativa.

3.3. Algoritmos heurísticos de mejora

- **K-opt:**
  Se trata de una heurística de mejora, y por lo tanto, se ha de partir de una solución factible ya conocida. Se basa en la sustitución de k aristas de una ruta por otras k aristas de la misma. Se dice que un tour es k-optimo (k-opt) si es imposible obtener una ruta con menor distancia (o coste) reemplazando k de sus arcos por otros k arcos distintos.

  Aunque la capacidad computacional ha aumentado mucho en los últimos años, la heurística 2-opt es interesante en grafos que tienen dos aristas que se cruzan, ya que pueden ser sustituidas por otras dos que no se crucen, mejorando así el problema.

3.4. Método génerico de Tucker, Miller y Zemlin

Este apartado no trata un algoritmo como tal pero sí un problema clásico y genérico dentro de todos los modelos dictados. Este tiene la siguiente estructura:

Se pide a un viajante que visite cada una de las n ciudades. Sle de la "ciudad base" marcada con 0 , para visitar cada una de las ciudades una sola vez, teniendo que regresar a la ciudad 0 finalmente. Durante sus viajes, debe retornar a la ciudad 0 exactamente "t" veces, incluyendo su regreso final y no debe visitar más de "p" ciudades en cada vez o vuelta. Se requiere encontrar un itinerario tal que minimice la distancia total recorrida por el viajante.

Para este problema se han planteado varios modelos que dictan la resolución del mismo.

3.5. Otros algoritmos

Hay otras tres heurísticas muy conocidas que debido a su extendido uso en la resolución del TSP y de otros muchos problemas, merecen tener cabida en este apartado. Hablamos de las metaheurísticas Búsqueda Tabú, Algoritmos Genéticos y GRASP.

- **Búsqueda Tabú:**
  La búsqueda Tabú es un tipo de búsqueda por entornos, que según su definidor, "guía un procedimiento de búsqueda local para explorar las soluciones más allá del óptimo local".

  La Búsqueda Tabú permite moverse a una solución del entorno que no sea tan buena como la actual, de modo que se pueda escapar de óptimos locales y continuar estratégicamente la búsqueda de soluciones aún mejores. Para ello, se clasifica un número determinado de movimientos recientes como "movimientos tabú". Es decir, se mantiene una memoria de eventos (por ejemplo soluciones o algún otro elemento propio de las soluciones visitadas) que sean de interés en relación con el pasado. Así, al explorar el "entorno" de la solución actual, estará prohibido aceptar como la siguiente solución un elemento de la búsqueda tabú.

- **Algoritmos genéticos:**
  Los Algoritmos Genéticos imitan el proceso darwiniano de selección natural. En cada iteración el algoritmo procesa no solo una solución sino en conjunto de ellas, llamado población.

  Esta población inicial se genera de manera aleatoria, y se trataría de un conjunto de individuos (formados por cromosomas) que representan posibles soluciones del problema. Después, en cada iteración y utilizando tres elementos clave (operador de selección, operador de cruce y operador de mutación), la población de soluciones es transformada en la siguiente población de la siguiente manera:

  En primer lugar, las soluciones de la población inicial se agrupan en parejas. De esos pares de individuos se eligen a los mejores mediante el operador de selección, dando lugar a la "población de padres".

  Después pasaríamos a la fase de reproducción. En la que de nuevo agrupamos la población de padres en parejas. Cada par de padres es transformado en dos nuevas soluciones llamados descendientes.

  Tras realizar el cruce de las parejas, cada uno de los individuos de la población de descendientes podrá sufrir mutación con una pequeña probabilidad p (operador de mutación).

  El último paso será quedarnos con las mejores soluciones.

- **GRAPS:**
  Su nombre proviene del acrónimo de Greedy Randomized Adaptative Search Procedure, cuya traducción literal podría ser Procedimientos de Búsqueda Ávidos, Aleatorizados y Adaptativos.

  Se trata de un método multi-start en el cual cada iteración consta de dos pasos: la fase de construcción y la fase de búsqueda local. Con estas dos fases se obtiene una solución óptima que después se almacena y se procede a efectuar una nueva iteración, guardando cada vez la mejor solución que se haya

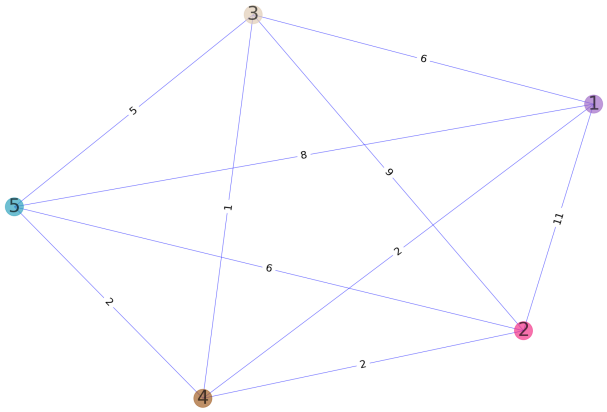
encontrado hasta el momento.

En definitiva, GRAPS dirige la mayor parte de su esfuerzo a construir soluciones de alta calidad que posteriormente son procesadas con el fin de conseguir otras aún mejor.

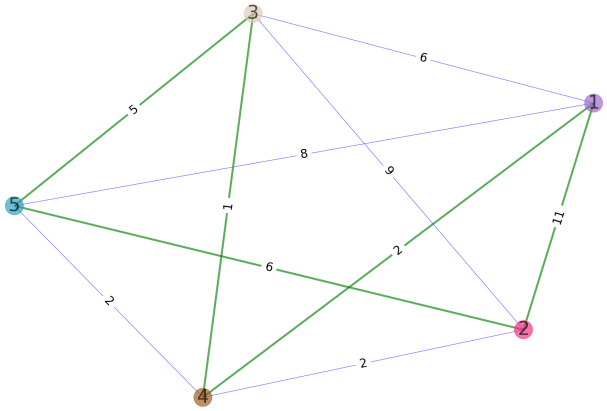
4. Implementación

En la implementación del algoritmo del vecino más cercano hemos utilizado el lenguaje de programación *Python* junto a las librerías *matplotlib, numpy y networkx*.

El grafo utilizado para ejemplificar nuestra implementación es el siguiente:



Una vez procesado este grafo con el algoritmo seleccionado, el resultado, remarcando el camino óptimo, es el siguiente:



La implementación propuesta para este problema puede ser encontrada junto a todos los archivos necesarios para su ejecución en el siguiente enlace: https://github.com/albertopastormr/tsp-mog. En esta url se encuentra además una implementación del algoritmo *branch and bound* utilizando el lenguaje de programación *C++*.

5. Complejidad computacional

5.1. Problemas NP-Completos

En este capítulo vamos a tratar algunos de los aspectos más relevantes para el TSP en el contexto de la teoría de la complejidad computacional. Para ello,

veremos una de las cuestiones más estudiadas de la teoría de la complejidad, que consiste con- cretamente en determinar si un determinado problema pertenece alguno de los dos grupos de

problemas P (Polinómico) y NP (No-determimístico Polinómico) en función de si conocemos un algoritmo eficiente para el mismo o no (veremos con mayor detenimiento las definiciones de P y NP a lo largo del capítulo). Este hecho que a simple vista puede no parecer tan complicado, ha sido y todavía es, una de las grandes cuestiones de la computación y la teoría de la complejidad.

Todavía no se sabe si en realidad P y NP coinciden, argumento que ha creado mucha contro- versia entre matemáticos ampliamente reconocidos y que han trabajado en este campo, muchos de

los cuales se centraron en el problema del viajante, pues es posiblemente el más representativo de todos estos problemas. Esta conjetura de descubrir si P=NP ha adquirido tal relevancia que el Instituto Clay de Matemáticas ofrece un millón de dólares al creador de un algoritmo en P para resolver el problema del viajante, o al que por el contrario demuestre la no existencia del mismo (www.claymath.org/millennium-problems/p-vs-np-problem).

Otro resultado muy importante relacionado con este tema procede de un artícu- lo publicado por S.Cook en (1971), en el que prueba que muchos problemas de decisión consi- derados "difíciles o no eficientes", son computacionalmente equivalentes. A estos problemas los

llamaremos NP-Completos y tienen la característica de que cualquier problema de NP puede transformarse en un proceso de tiempo polinómico en él. Así, cualquier método que resuelva un problema NP-Completo, podrá resolver mediante esta transformación cualquier problema de NP.

Cuando ya no solo hablamos de problemas de decisión sino también de optimización, apa- recerá la noción de NP-Duro y veremos que el TSP es uno de ellos.

5.2. Categorización TSP

El TSP es un problema NP-duro. Igual que ocurre con otros problemas en el ámbito de la Organización Industrial, el TSP se caracteriza por ser fácilmente descriptible pero difícil de resolver, por lo que ha despertado un gran interés a lo largo de la historia no sólo en su estudio sino también en la obtención de heurísticas para su resolución. Su dificultad de resolución radica en que el número de posibles soluciones aumenta significativamente al aumentar el tamaño de la muestra (n). Si pensamos en el concurso propuesto por "Procter and Gamble" en el 1962, en el cual se ofrecía 10.000$ para quien pudiera ayudar a Toody y Muldoon a decidir la ruta más corta entre 33 ciudades; a priori podemos pensar que en lugar de hacer complicados cálculos, uno podría haber comprobado los distintos caminos y luego elegir la mejor solución. Sin embargo, veamos como no resultaría sencillo. Como el origen viene determinado, restarían (n - 1) puntos para empezar, a continuación tendríamos que elegir cualquiera de los (n - 2) restantes y así sucesivamente. De esta forma, multiplicando todas estas cantidades obtenemos el número total de rutas o soluciones posibles que han de ser evaluadas: (n - 1)! En el caso particular de 33 ciudades esto significaría evaluar $32! = 2,63 * 10^35$; es decir, para un problema no excesivamente grande el número de pruebas aumenta considerablemente. Así, aunque el problema puede resolverse en un número finito de pasos, esto no es suficiente y se precisa buscar otros algoritmos que reduzcan este número de pruebas. La única reducción que se puede hacer es en el caso que la dirección de ida sea igual a la de vuelta; es decir que no importe la dirección en la que viajemos. Sin embargo, el número de soluciones posibles seguiría siendo abrumador. Para nuestro caso particular:

$$\frac{32!}{2} = 131565418466846765053609080000000$$

Para esta cifra necesitaríamos alrededor de 28 trillones de años para comprobar todas las posibles soluciones. Por esta razón, el problema del viajante ha alcanzado una importante posición en la teoría de la complejidad. Según esta teoría, los problemas para los que existe un buen algoritmo de resolución son clasificados como P, de tiempo polinómico. Por el contrario, los problemas para los que aún no se ha encontrado un algoritmo eficiente son clasificados como NP, de tiempo polinómico no determinista.

Además, se ha demostrado que muchos de los problemas denominados como "difíciles" son computacionalmente equivalentes. Es decir, un algoritmo polinómico que puede usarse para resolver uno de estos problemas sería válido para resolver el resto de los englobados en esta categoría. Estos problemas reciben el nombre de NP-duro (más conocidos por su nombre inglés NP-hard). Por esta razón no es de extrañar que el Instituto Clay [11] haya ofrecido un millón de dólares a quien descubra un algoritmo eficiente para el TSP o demuestre la no existencia del mismo. De existir, todos los problemas NP podrían resolverse en un tiempo polinómico y se concluiría que P y NP son iguales.

6. Conclusión

Tras aprender sobre todo aquello que ha sido expuesto en este documento sobre el problema del viajante de comercio, podemos concluir que este problema ha sido muy relevante históricamente pero aún sigue siendo crucial para resolver algunas de las problemáticas que más preocupación causan mundialmente. Es por ello por lo cual consideramos que este problema de optimización combinatoria mantiene actualmente un entorno científico muy importante que debe ser apoyado y seguido de cerca.

El estudio de este problema requiere de conocimiento en múltiples areas algorítmicas, ya que han sido propuestas a lo largo de los años diferentes soluciones utilizando varias metodologías algorítmicas. Por lo tanto, consideramos que este problema podría satisfacer la curiosidad científica de muchos futuros investigadores.

Referencias

[1] Salvador Peñalva García, *El problema del viajante. Métodos de resolución y un enfoque hacia la Teoría de la Computación*, Facultad de Ciencias, Estudios Agroalimentarios e Informática, Universidad de La Rioja, 2015.

[2] Beatriz Pérez de Vargas Moreno, *Resolución del Problema del Viajante de Comercio (TSP) y su variante con Ventanas de Tiempo (TSPTW) usando métodos heurísticos de búsqueda local*, Escuela de ingenierías industriales, Universidad de Valladolid, 2015.