Novel condición de paro y algoritmo para el problema del agente viajero

Carlos Barrón-Romero

cbarron@azc.uam.mx

Universidad Autónoma Metropolitana, Unidad Azcapotzalco, México
Av. San Pablo No. 180, Col. Reynosa Tamaulipas, C.P. 02200
Ciudad de México, México

Resumen: El problema del agente de viajero de ventas en el espacio euclidiano 2D se resuelve mediante el uso de propiedades geométricas: la curva simple de Jordan y colorear con dos colores un mapa. Está comprobado que, localmente, una trayectoria con intersecciones es más grande que una trayectoria sin estas. Además, una trayectoria o ciclo cerrado sin cruzar dentro de una región apropiada, es decir, un plano apropiado que contiene una curva simple de Jordan corresponde a una coloración del mapa con solo dos colores. Este enfoque es una aplicación novedosa de la visión artificial para resolver un problema de búsqueda. Varios ejemplos numéricos y sus resultados se presentan utilizando programas ad-hoc y el software gratuito Concorde del Departamento de Matemáticas de la Universidad de Waterloo.

Palabras claves: Algoritmos, TSP, Optimización Numérica.

Abstract: *The Travel Salesman Problem in the Euclidian 2D space is solved by using geometric properties: Jordan's simple Curve and Two Coloring Maps. It is proved that, locally, a crossed trajectory is larger than a not crossed trajectory. Moreover, a closed trajectory or cycle without not crossed inside of an appropriate region, i.e. an appropriate plane containing a Jordan Simply Curve corresponds to a coloring the map with only two colors. This approach is a novel application of Machine Vision in solving a search problem. Several numerical examples, and their results are presented using ad-hoc programs and the free software Concorde from Department of Mathematics of the University of Waterloo.*

Keywords: Algorithms, TSP, Numerical optimization.

1. Introducción

La construcción de software es un proceso intelectual similar y relacionado con la actividad matemática de construir y probar teoremas; es decir, estudiar, pensar, planear, revisar, corregir, cortar, cambiar, entender, para conseguir una belleza estética en un procedimiento eficiente para resolver problemas o preguntas de ciencia y tecnología.

Para el problema del agente viajero (Travel Salesman Problem o TSP) y problemas relacionados, existen varios artículos y publicaciones al respecto [4,7,10].

Este documento describe y establece las condiciones de paro para el TSP bajo la distancia euclidiana (donde la ley de coseno se cumple) y una curva simple de Jordan es una condición necesaria en un plano de tener una solución pequeña. Esto es similar a usar la condición necesaria $f'(x^*) = 0$, para resolver un problema de optimización global con función objetivo f, que es derivable. Esta es la conocida condición necesaria de primer orden, en optimización. No implica la optimización local o global de un punto x^* (otros puntos podrían aparecer como ceros de f' sin ser óptimos locales, como por ejemplo los puntos de inflexión). Para el TSP, la curva simple de Jordan proporciona una condición de paro (véase Prop. 3). Una curva simple con otras propiedades geométricas como la distancia de sus conexiones con sus vecinos y la falta de cruces en la ruta, serán las condiciones necesarias (véase Prop. 4). Un breve panorama de la literatura se tiene en la sección 2. La parte experimental se da en la sección 3 y finalmente la sección 3 presenta conclusiones y trabajos futuros.

2. Teoría del dominio y trabajos previos

Una revisión amplia de literatura de TSP se tiene en [8]. Una propuesta heurística de IA basada en animales sencillos (hormigas) simuladas en un ambiente distribuido se encuentra en [9], un trabajo previo similar es [3].

Una fuente de información, de software libre y problemas de prueba lo ofrece la universidad de Waterlo en su página dedicada al problema TSP [2]. La página de TSP incluye la liga a la página del software Concorde [1] y destaca la bibliografía del periodo de 1990 a 1930, premios, records, noticias y libros.

La mayoría de los algoritmos de búsqueda de soluciones del TSP y de grafos utilizan la propiedad de las soluciones estacionarias, o sea que una posible solu-

ción local o global se manifiesta por la repetición de su valor objetivo varias veces, lo cual no garantiza tener la solución óptima, sino un candidato a serlo. La desventaja, por decir lo menos, es que la verificación por repetición consume tiempo de máquina y dependiendo del criterio del usuario, el tiempo consumido por estas repeticiones puede ser mucho mayor que el que le tomo a su algoritmo llegar a la posible solución. La importancia de nuestra propuesta es que elimina el tiempo de repeticiones y da una solución local que cumple una condición necesaria de optimalidad, que en algunos tipos de problemas puede ser necesaria y suficiente para la optimalidad global de la solución obtenida. En este trabajo, por primera vez, bajo nuestro criterio, se utilizan las propiedades de la curva simple de Jordan y el colorear una región con dos colores como una condición de paro para el problema TSP. La propiedad de la curva simple de Jordan proviene de [4] figuras 1.21 y 1.22 (pág. 24), como segmentos sin cruzar de la ruta. Por lo que sabemos, es la primera vez que la curva simple de Jordan se combina con técnicas de visualización de datos (colorear una región con dos colores) como condición necesaria para detener los algoritmos de TSP. Esto será abordado en las siguientes secciones 2.1 y 2.2.

2.1. Problema del agente viajero

El conocido problema del agente viajero (TSP), se representa gráficamente por una función de distancia y una gráfica, es decir, $G = \{V, A\}$, donde V es un conjunto de ciudades y A son los enlaces o vértices relacionados con la distancia euclidiana entre las ciudades, es decir, la $d(i, j)$ es la distancia euclidiana entre $i, j \in V$.

El modelado del TSP está relacionado con la teoría de grafos [6]. "Euler llamó a esta nueva matemática sin números, en la cual solo la estructura de la configuración juega un papel, pero no el tamaño y la forma: *"geometria situs"*, geometría de posición (tomó el concepto de una carta de Leibnitz del año 1679)".

Aquí, el TSP es un problema con ciudades como puntos en $\mathbb{R}^2$ y los bordes provienen de sus distancias euclidianas entre todas las ciudades, es decir, la gráfica de las ciudades es una gráfica completa.

Proposición 1. *Cualquier TSP tiene solución, es decir, existe un ciclo hamiltoniano con mínimo costo.*

Demostración. La gráfica asociada a las n ciudades es una gráfica completa. Entonces, por comparación de todos los costos de los $(n-1)!$ *ciclos hamiltonianos, se puede seleccionar el mínimo costo y se obtiene la solución.*

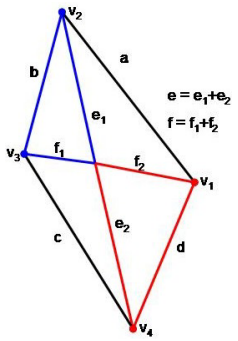

Figura 1. Cuadrilátero arbitrario.

La proposición establece la existencia de la solución, pero el tiempo para hacer una búsqueda exhaustiva es abrumador. Es deseable encontrar un ciclo hamiltoniano mínimo o cercano a él en un tiempo pequeño. Esto significa determinar y probar la optimalidad del supuesto ciclo hamiltoniano óptimo eficientemente, es decir, evitar la búsqueda exhaustiva o esperar por siempre una respuesta o usar el criterio de valor estacionario (cuando un valor es repetido k veces) o simplemente asignar un tiempo de ejecución al algoritmo.

2.2. Algoritmo de paro para el TSP

La siguiente proposición (Prop. 6.11 de [5]) representa que para cualquier cuadrilátero, sus lados son menores que su diagonal.

Proposición 2. *Dados cuatro puntos como vértices en 2D, el perímetro del cuadrilátero es menor que la longitud del ciclo euclidiano usando las diagonales y los lados opuestos.*

Demostración. La fig. 1 representa un cuadrilátero arbitrario con vértices v_1, v_2, v_3, *y* v_4, *y bordes euclidianos de longitud* a, b, c, *y* d. *La longitud de sus diagonales son* e, *y* f. *Sin pérdida de generalidad los lados opuestos son* a, *y* c. *Las siguientes relaciones son válidas, por la desigualdad del triángulo, para el triángulo azul y el triángulo rojo:*

$$b \leq e_1 + f_1$$
$$d \leq e_2 + f_2.$$

Entonces, estas relaciones y el perímetro del cuadrilátero cumplen:

$$a+b+c+d \leq a+e_1+f_1+c+e_2+f_2 = a+e+c+f$$

Suponiendo que se tienen un algoritmo para resolver TSP, hay una imagen en blanco y negro del supuesto ciclo hamiltoniano óptimo para un TSP determinado. Dicha imagen de la solución actual se puede obtener mediante una interfaz gráfica como la de Concorde o utilizando la herramienta gráfica de MATLAB® u Octave. Como ejemplo véase la fig. 2.

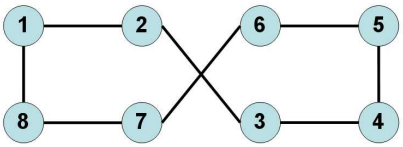

Figura 2. Ciclo hamiltoniano con cruce entre las ciudades: v_2, v_3, v_6, v_7.

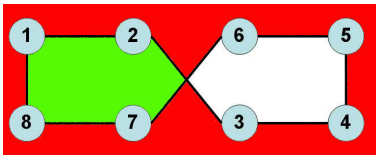

Figura 3. Mapa de colores de un ciclo hamiltoniano con-cruce entre las ciudades: v_2, v_3, v_6, v_7.

Algoritmo 1 ***entrada:*** *Una imagen en blanco y negro del ciclo hamiltoniano actual,* $v = (v_1, v_2, \ldots, v_n, v_1)$ *donde las ciudades están en orden consecutivo.*

salida: *detener, la imagen del ciclo hamiltoniano está coloreada por dos colores. De lo contrario, el ciclo hamiltoniano tiene un cruce entre las ciudades* $v_i, v_{i+1}, v_j, v_{j+1}$.

usando v_1 *detecta su ubicación en la imagen de entrada.*
usando v_1 *detecta su frontera en la imagen para seleccionar dos puntos, uno dentro y otro fuera de* v_1.
usando *una herramienta de pintura de inundación con la punta al exterior de* v_1, *colorear la imagen con verde.*
usando *una herramienta de pintura para inundación con la punta al interior de* v_1, *colorear la imagen con rojo.*
dos_colores="sí";
para $i := 1$ ***hasta*** n
  usando v_i *detecta su ubicación en la imagen de entrada coloreada;*
  si *la proximidad de* v_i *tiene rojo, verde y blanco* ***entonces***
    continua;
  de otro modo
    dos_colores="no";
    marcar v_i;
fin de para i;
si *dos_colores es "sí"* ***entonces***
    detener *"La imagen del ciclo hamiltoniano puede ser coloreada por dos colores,*$v = (v_1, v_2, \ldots, v_n, v_1)$ *es un ciclo hamiltoniano como la curva simple de Jordan, es decir, sin cruzar";*
seleccionar *vértice marcado, supongamos que es* v_i
usando v_i *detecta su ubicación en la imagen de entrada de color;*
usando v_i *detecta sus cuatro vecinos marcados;*
detener *hay un cruce entre las ciudades:* $v_i, v_{i+1}, v_j, v_{j+1}$.

El algoritmo anterior responde si el ciclo actual es una curva simple o las cuatro ciudades de las esquinas de un cuadrilátero tienen un cruce. El último caso lo resuelve el algoritmo. 2.

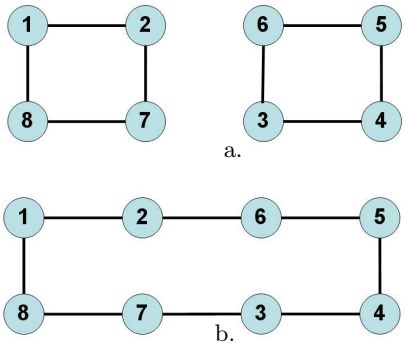

Figura 4. a) Conexión incorrecta de los cruces entre las ciudades: v_2, v_3, v_6, and v_7, y b) Ciclo hamiltoniano sin el cruce entre las ciudades: v_2, v_3, v_6, y v_7.

Un algoritmo de pintura para inundación es una herramienta común en casi cualquier software gráfico. Tiene la complejidad del tamaño de la imagen $\mathbf{O}(kn^2)$ donde kn Está relacionado con la resolución de la proximidad de los n vértices. El valor del factor k depende de la distancia mínima de los vértices. Se debe permitir que k tenga una imagen en blanco y negro donde las líneas del ciclo hamiltoniano se distinguen claramente. La figura 3 representa cuando hay un cruce. Representa que el ciclo no es óptimo ya que no corresponde con una curva simple de Jordan.

Asumiendo que los vértices del ciclo hamiltoniano están en orden, la única solución para la figura 3 se muestra en la figura 4. La figura 4 muestra la conexión errónea. Existen dos posibilidades cuando se encuentra un cruce, pero solo una es la correcta en el ciclo hamiltoniano actual. Al quitar un cruce, el costo del ciclo siempre disminuye. Por lo tanto, si hay cruces en las ciudades $v_i, v_{i+1}, v_j, v_{j+1}$ el ciclo hamiltoniano debe conectar v_i a v_j y v_{i+1} a v_{j+1} y siempre hay una garantía que se disminuirá el costo del ciclo por la Prop. 2.

Algoritmo 2 ***entrada:*** $V = (v_1, v_2, \ldots, v_n, v_1)$ *un ciclo hamiltoniano, donde las ciudades están en orden consecutivo y con un cruce en las cuatro ciudades:* $v_i, v_{i+1}, v_j, v_{j+1}$.

salida: $V = (v_1, v_2, \ldots, v_n)$ *un ciclo hamiltoniano sin el cruce en las ciudades* $v_i, v_{i+1}, v_j, v_{j+1}$.

$V' = V$;

$l = j$
para k:= i+1 ***hasta*** j
$V(k) = V'(l)$
$l = l - 1$;
fin de para k;
detener *El ciclo hamiltoniano sin los cruces es*
$V = (v_1, \ldots, v_i, v_j, v_{j-1}, \ldots, v_{i+1}, v_{j+1}, \ldots, v_n, v_1)$.

Algoritmo 3 *El algoritmo completo para resolver el TSP.*
entrada: *TSP.*
salida: $V = (v_1, v_2, \ldots, v_n, v_1)$ *un ciclo hamiltoniano como la curva simple de Jordan, es decir, sin ningún cruce en la ruta de las ciudades.*

repetir
ejecuta un algoritmo ad-hoc para el TSP;
ejecuta algoritmos *1 y 2.*
hasta obtener un mapa de dos colores del ciclo hamiltoniano actual.
detener *el ciclo hamiltoniano es la solución supuesta del TSP.*

Proposición 3. *Dado un TSP_n, donde las ciudades son puntos en $\mathbb{R}^2$, y la matriz de costo corresponde a distancias euclidianas entre ciudades. Entonces, una curva simple de Jordan es una condición necesaria para la reducción del costo del ciclo hamiltoniano.*

Demostración. Sin perder generalidad, si el supuesto ciclo hamiltoniano óptimo tiene un cruce, entonces por Prop. 2, el ciclo hamiltoniano obtenido por el algoritmo 2 tiene un valor menor.

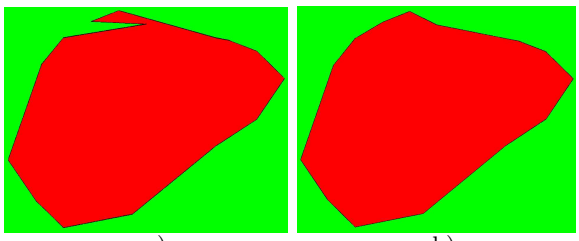

Figura 5. TSP_{17}. a) Mapa de dos colores de un ciclo hamiltoniano no óptimo con su curva simple de Jordan, y b) Mapa de dos colores del ciclo hamiltoniano óptimo con su curva simple de Jordan.

Proposición 4. *Dado un TSP_n con ciudades ubicadas alrededor de un área convexa cerrada, un algoritmo glotón o codicioso que selecciona la siguiente ciudad utilizando la distancia euclidiana más cercana, encuentra el ciclo hamiltoniano de costo mínimo.*

Demostración. La optimalidad del ciclo hamiltoniano proviene de la configuración de vértices alrededor del área convexa, quizás puntos en un círculo, elipse, n-poligonal. El algoritmo seleccionado combinado con los algoritmos 1, y 2 en el algoritmo 3. Por construcción selecciona las ciudades vertices más cercanos en un ciclo cerrado. El ciclo hamiltoniano corresponde a la solución del problema variacional de la curva de longitudes mínimas (curva simple de Jordan) con el área convexa máxima (Teorema Variacional de la Isoperimétrica).

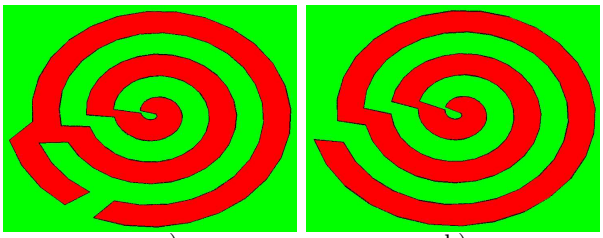

Figura 6. TSP_{152} en espiral. Mapa de colores de dos curvas simples de Jordan: a) Ciclo hamiltoniano no óptimo, con coste =21,1661, y b) ciclo hamiltoniano óptimo con costo = 21,1451.

La fig. 5 describe un caso en el que el algoritmo 1 no puede detectar la zona donde es posible descender el costo. El ciclo corresponde a una curva simple, pero no es la curva del casco convexo alrededor de las ciudades. Una posible solución para este caso es comparar el área en rojo de los dos mapas de color. El ciclo óptimo corresponde a la imagen con el área grande en rojo. Esto se desprende del conocido teorema de la Isoperimétrica).

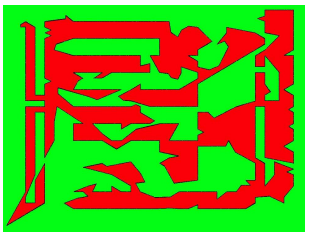

Figura 7. TSP_{442}. Mapa final de dos colores del ciclo hamiltoniano con coste = 56, 601,1738. El coste inicial de un ciclo aleatorio = 221, 435,5555.

3. Experimentos y Resultados

Las figuras 5 a) y b) representan un caso en el que se verifica la proposición anterior para un TSP_{17} pequeño. El área roja de la figura anterior tiene 415, 570 pixels

con un costo = 15,5100,y el área roja de la siguiente figura tiene 417, 719 pixels con un costo = 13,5849. Estas figuras presentan curvas simple de Jordan, pero el caso b) es el óptimo global por ser de mayor área y ser una región convexa, aquí la curva de Jordan y la convexidad corresponden con una propiedad de optimalidad necesaria y suficiente.

El siguiente experimento numérico es para TSP con 152 ciudades ubicadas en una curva en espiral. TPS_{152}. La figura 6 representa dos ciclos hamiltonianos muy cerrados. En a) no es el óptimo, su área roja tiene 161, 279 pixeles, y en b) es el supuesto óptimo, su área roja tiene 158, 813 pixeels. La diferencia de costo entre los dos ciclos hamiltonianos es solo 0,021. Además, las ubicaciones de las ciudades no cumplen con la hipótesis de la proposición 4, por lo tanto el tamaño de sus áreas rojas no implica optimalidad por no ser regiones convexas, como en las figuras 5 a) y b).

De [1] el problema fig. 7 tiene un costo inicial para un ciclo aleatorio de 221, 435,5555. La reducción de costo es 164, 834,3817. La figura 7 representa el supuesto ciclo óptimo, sin cruces, con un costo de 56, 601,1738.

Conclusiones y trabajos futuros

La propiedad de la curva simple de Jordan no se puede aplicar directamente para todos los TSP, donde el objetivo es obtener un costo mínimo del ciclo hamiltoniano relacionado con el dinero, el tiempo de viaje, valores arbitrarios o la distancia máxima $d_m(x, y) = \text{máx}\{|x_1 - y_1|, |x_2 - y_2\}$, donde $x = (x_1, x_2)$, y $y = (y_1, y_2) \in \mathbb{R}^2$.

Estamos en una fase de nuestro método científico donde parte del trabajo futuro es explorar cómo incorporar esta propiedad para la creación de algoritmos muy eficientes para otros tipos de TSP en la elaboración de rutas completas con costo muy cercano o posiblemente igual al mínimo.

La eficiencia de los nuevos algoritmos se deberá a que la curva simple de Jordan es un ejemplo de una propiedad necesaria para detener los algoritmos de TSP sobre ciclos completos de costo cercano al mínimo en un tiempo eficiente.

Por otro lado, mediante algoritmos de grafos se busca adaptar la propiedad visual de detección de dónde hay un cruce y de dónde un ciclo podría disminuir su costo. Nuevamente, buscaremos que el ciclo aceptado sea una solución cercana o posiblemente igual a la de ciclo de mínimo costo.

Los resultados presentados en este trabajo son muy alentadores de que vamos por un camino apropiado de nuestra metodología científica computacional para la construcción de algoritmos muy eficientes en donde por primera vez, bajo nuestro punto de vista y con base en muestra exploración de la literatura, se incorporan propiedades necesarias para la resolución eficiente de problemas TSP.

Es conocido que las soluciones estacionarias, o sea una posible solución local o global se manifiesta por la repetición de su valor objetivo varias veces, en particular las soluciones locales se pueden descartar si se exploran un gran número de alternativas, donde posiblemente se avanza a otra solución local o quizás a la solución global. Esta propiedad necesaria no distingue, ni garantiza estar en la solución global aunque el número de repeticiones k sea muy grande. Se usa en muchos algoritmos como condición de paro.

Usar detección de soluciones estacionarias, adolece de un extra consumo de tiempo por realizar iteraciones innecesarias. Pero salvo ésta, no se había sugerido nunca antes una condición de paro tan eficiente como la curva de Jordan y el colorear una región con dos colores. Nuestra propuesta es eficiente porque una vez que se detectan estas dos condiciones se tiene la certeza de que la solución es un mínimo local, o quizá se trate de la solución global.

Referencias bibliográficas

1. Concorde Home. Concorde TSP Solver. Department of Mathematics of the University of Waterlo. http://www.math.uwaterloo.ca/tsp/concorde.html, 2015.
2. The Traveling Salesman Problem. Department of Combinatorics and Optimization of the University of Waterlo. http://www.math.uwaterloo.ca/tsp/index.html, 2018.
3. J. Aguilar. The Combinatorial ANT System for Dynamic Combinatorial Optimization Problems. *Revista de Matemática: Teoría y Aplicaciones*, 12:51–60, 2005.
4. D. L. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook. *The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics)*. Princeton University Press, Princeton, NJ, USA, 2007.
5. C. Barrón-Romero. The Complexity of the NP-Class. *arXiv*, http://arxiv.org/abs/1006.2218, 2010.
6. R. Borndörfer, M. Grötschel, and A. Löble. *Mathematics Everywhere*, chapter 4. The Quickest Path to the Goal. Spain - AMS, 2010.
7. D. E. Knuth. *Selected Papers on Fun and Games*, volume 192 of *CSLI lecture notes series*. Cambridge University Press, 2011.
8. J. I. Pérez Rave and G. P. Jaramillo Álvarez. Espacio literario relevante sobre el problema del vendedor viajero (TSP): contenido, clasificación, métodos y campos de inspiración. *Production*, 23:866 – 876, 12 2013.
9. F. Riveros, N. Benítez, J. Paciello, and B. Barán. A Many-objective Ant Colony Optimization applied to the Traveling Salesman Problem. *Journal of Computer Science & Technology*, 16(2):89–94, 2016.
10. E. Sandifer. How Euler Did It. Knight's Tour. http://eulerarchive.maa.org/hedi/HEDI-2006-04.pdf, 2006.