Universidad del Valle de Guatemala

FACULTAD DE CIENCIAS Y HUMANIDADES

THE TRAVELING SALESMAN PROBLEM

Proyecto final de Métodos Numéricos 2

Autor:
José Eduardo López Gómez - 181045
Javier Enrique Mejía Esquivel - 18638

1 de diciembre de 2021

I. Introduction

DADA una lista de ciudades y la distancia entre cada par de ellas, ¿cuál es la ruta más corta posible que visita cada ciudad exactamente una vez y al finalizar regresa a la ciudad origen? El enunciado aparece por primera vez en 1832 en una guía para vendedores ambulantes en Europa, y fue atacado por matemáticos en 1930. Es un problema popular dentro del área de optimización con combinatoria, por la simplicidad del enunciado y la alta complejidad de la solución.

Estamos ante un problema NP, es decir que no tenemos una solución que se pueda calcular en tiempo polinomial. Existen múltiples formas de atacar este problema, y en general, varias formas de atacar un problema tan difícil, nos enfocaremos en dos.

Cuando se busca abordar un problema de combinatoria "imposible", existen al menos dos posibilidades. Una posibilidad es hacer una búsqueda exhaustiva, probando todas (o casi todas) las posibles combinaciones. Esta posibilidad se descarta cuando el problema es desamasado grande. Por ejemplo, el problema del TSP tiene $n!$ combinaciones, por lo que tan solo con 20 ciudades que visitar, no es factible hacer una búsqueda exhaustiva. La segunda posibilidad es aproximar una solución, lo que se conoce como un PTAS (Esquema de aproximación en tiempo polinomial). Esta segunda opción ofrece una solución rápida, pero sub-óptima. Finalmente tenemos la posibilidad de combinar ambos esquemas de solución. Esto es un método popular para la solución de problemas NP, en el que primero se aproxima una solución inicial, y luego se hace una búsqueda exhaustiva de forma ordenada, limitando las posibilidades que se prueban. En este trabajo solo exploramos las alternativas aisladas, para entrar en detalle sobre el funcionamiento de esos algoritmos, y aspectos a considerar cuando se considera cualquiera de las dos posibilidades.

El primer algoritmo que se presentará es el algoritmo genético. Esta es una forma de búsqueda exhaustiva que genera nuevas soluciones desde una solución raíz o padre, y busca imitar el funcionamiento biológico de evolución y el principio de selección natural, en donde solo los más "aptos" sobreviven.

El segundo algoritmo es el algoritmo de Christofides, un PTAS determinista que ofrece una solución 3/2-aproximada en tiempo polinomial. Este fue expuesto por Nicos Christofides en 1976 para este problema en particular. El algoritmo utiliza como herramienta a los árboles de expansión, a partir de los que define una ruta.

Elaboramos animaciones para mostrar el funcionamiento de los algoritmos usando el lenguaje de programación Python, y comparamos el tiempo de ejecución de ambos algoritmos hasta su convergencia. Nuestros hallazgos fueron coherentes con la teoría, el tiempo de ejecución del algoritmo de Christofides es mucho menor que el del algoritmo genético, y en ocasiones la solución del algoritmo de Christofides no es óptima.

El algoritmo genético implementado tuvo problemas en varias ocasiones para hallar la solución óptima. Suponemos que esto es un problema de implementación y de ajuste de parámetros dado que el algoritmo genético, por lo general, depende de muchos factores aleatorios que deben ajustarse correctamente para que el algoritmo mejore.

En el algoritmo de Christofides hay trabajo teórico que resolver, por ejemplo, demostrar que la solución es 3/2-aproximada, y que se resuelve en tiempo polinomial.

Por último, debemos notar que el Problema del Viajero (TSP) es altamente utilizado en la vida cotidiana por empresas que buscan precisamente optimización de rutas. Tiene mayor aplicación en cadenas de distribución y cadenas de suministro, por lo que se han desarrollado múltiples formas de resolver el problema, entre ellas, algoritmos específicos para atacar y dar solución a dicho problema. Por el momento, solo nos enfocaremos en dos de ellos, pero se debe de tomar en consideración que en la vida real, probablemente se implementa más de un algoritmo para encontrar la solución óptima. Solo como mención general, y para conocimiento del lector se citan nombres de algunos algoritmos que pueden servir para encontrar soluciones aproximadas u óptimas: **algoritmo Greedy**, **algoritmo Nearest Neighbors**, **algoritmo 2 - opt**, y claramente, lo dos explorados en este documento **algoritmo genético** y **algoritmo de Christofides**. Cabe resaltar que estos son los más comunes pero existen una alta variedad de ellos, fácilmente se pueden encontrar más de diez.

II. Métodos y Expresiones

Enunciado formal:

Considere el caso de N puntos, y sea $d(i,j)$ la distancia directa entre las ciudades i y j entonces se busca minimizar lo siguiente:

$$min\left\{ z = \sum_{i=1}^{N}\sum_{j=1}^{N} x(i,j)d(i,j) \right\}$$

sujeto a las restricciones:

$$\sum_{j=1}^{N} x_{ij} = 1, \quad i = 1, ..., N \qquad \sum_{i=1}^{N} x_{ij} = 1, \quad j = 1, .., N$$

con $x_{ij} \in \{0, 1\}$ y con $i \neq j$

Definición: Grafo

Un grafo no dirigido es una pareja (V, E) donde V es un conjunto de vértices y $E \subseteq V \times V$ es un conjunto de aristas que conectan a los vértices de V.

A cada vértice v en V se le asigna un grado, $deg(v)$ que es el número de aristas que conectan con v.

Definición: Grafo conexo

Un grafo se dice conexo si para cada par de vértices a, b existe una arista que los conecta.

Definición: Camino
Un camino es una sucesión de aristas conectadas. Usualmente se representa por la sucesión de vértices que recorre, por ejemplo

$$v_0 \to v_1 \to ... \to v_n$$

Definición: Circuito
Un circuito es un camino tal que el último vértice y el primero coinciden, es decir $v_0 = v_n$

Definición: Circuito Euleriano
Un circuito es llamado Euleriano si recorre todas las aristas del grafo exactamente una vez.

Definición: Circuito Hamiltoniano
Un circuito se llama hamiltoniano si recorre todos los vértices del grafo exactamente una vez (sin contar el vértice final e inicial).

Definición: Árbol de expansión
Un árbol de expansión es un subconjunto de las aristas de un grafo tal que todos los vértices están en alguna de las aristas. El árbol de expansión se llama mínimo si la suma de la longitud de sus aristas es mínima.

Definición: Perfect matching
Un perfect matching de un conjunto de vértices es un conjunto de aristas tales que cada vértice está conectado exactamente a 1 vértice por 1 arista. Se le llama de peso mínimo cuando la suma de la longitud de sus aristas es mínimo. Dado que solo podemos conectar parejas de vértices y éstas se van descartando, esto se puede hallar con una búsqueda exhaustiva en tiempo polinomial

Algoritmo Genético:
Este algoritmo busca simular el proceso de codificación genética. Se busca evolucionar cromosomas a través de las iteraciones llamadas generaciones. En todas las generaciones los cromosomas son evaluados con una métrica de aptitud. Luego se aplican operaciones genéticas repetidamente.

1) Inicialización: se genera aleatoriamente una población inicial. Debemos notar que esto no es necesariamente estricto. Es decir, podemos empezar con una población predefinida, pero la ventaja de hacerlo aleatoriamente es que nos garantiza con mayor probabilidad que la solución óptima se encuentra dentro del rango de búsqueda.
2) Evaluación: a cada uno de los cromosomas de esta población se le aplicara la función de aptitud. Esta puede ser cualquier tipo de función, siempre y cuándo cumpla con lo que estamos buscando optimizar.
3) Criterio de paro: generalmente se coloca un máximo de generaciones o un máximo de generaciones sin mejora. Mientras el algoritmo no termine se aplican operadores genéticos.

Debemos notar que, por lo general, los operadores genéticos pueden aplicarse cuántas veces se desee. Además, no es necesario agregar todos en la implementación pero por lo menos el algoritmo debe contener uno de ellos. Los más comunes son:

- Selección: basándose en la aptitud de los cromosomas se selecciona a los que serán cruzados, esto para garantizar que los más aptos pasen a la siguiente generación.
- Cruce: genera una nueva generación tomando como antecesores a la población con mejor aptitud. Generalmente este proceso se hace aleatoriamente para que el método funcione mejor.
- Mutación: modifica al azar parte de los cromosomas. Este cambio está ligado a una variable que se conoce como índice de mutación, por lo general, es un valor entre (0,1) que nos índica que tanto debe mutar la población. Por lo que, un índice de mutación alto indica mutación alto indica un mayor número de mutaciones por generación, mientras que, un índice de mutación bajo indica un menor número de mutaciones por generación.

Algoritmo de Prim:
Dado un grafo no dirigido (V, E):

1) Seleccionamos un vértice inicial $v \in V$.
2) Iniciamos $M = \{v\}, T = \{\}$.
3) Mientras $M \neq V$:
   a) Tomar u tal que $d(M, u) = \min\{d(M, x), x \in V\}$
   b) Concatenar (M, u) a T, y hacer $M = M \cup \{u\}$
4) Devuelve el árbol mínimo T.

Algoritmo de Christofides:
Dado un grafo no dirigido (V, E):

1) Calculamos el árbol de expansión mínimo MST del grafo.
2) Identificamos los vértices de grado impar en MST, llamados O.
3) Hacemos un perfect matching de peso mínimo de los elementos de O, denotado M.
4) Unimos M con MST y hallamos un circuito euleriano.
5) Convertimos el euleriano en un hamiltoniano eliminando repeticiones.

III. Teoría

Características del Problema:

Debemos notar que este problema tiene diferentes versiones dependiendo de las características que se estén trabajando en ese preciso momento. Algunos versiones del problema solo requieren un conjunto de aristas y tienen restricciones con respecto a qué cambios pueden realizarse. Además, versiones más abstractas del problema también conllevan a trabajar con distintos tipos de distancias, e incluso en algunos casos se puede agregar más condiciones al problema. Por ello, creemos que es importante aclarar bajo que condiciones se trabajó el Traveling Sales Man Problem.

Condiciones

- Comenzando desde la primer ciudad se debe recorrer el resto de ciudades exactamente

una vez antes de regresar al punto de partida.

- La distancia entre las ciudades es conocida y se supone que es la misma en ambas direcciones.

- Es posible tomar cualquier ruta entre puntos, es decir, podemos seleccionar cualquier arista de todas las que existen.

- Se trabaja con la distancia euclideana.
- un punto puede tener únicamente una arista de entrada y una arista de salida (es decir, cada punto puede tener dos aristas asociadas como máximo).

Aplicación del algoritmo genético a TSP

- Gen: es un punto representado con sus coordenadas (x, y).
- Cromosoma: es representado por una ruta entre dos puntos.
- Función de aptitud: es la función que determina que tan buena es una ruta, es decir, que tan corta es:

$$f(x) = 1/x$$

  donde x representa la distancia de la ruta. Esto porque mientras más corta es una ruta entonces más grande es el valor de la función de aptitud.
- Población: es la colección de todas las posibles conexiones que existen entre los n puntos.
- criterios de paro: generaciones máximas y cierta cantidad de generacioens sin mejora.

El Algoritmo de Christofides es 3/2 aproximado

Teorema:

Sea (V, E) un grafo conexo, y C el circuito hamiltoniano resultante del algoritmo de Christofides. Sea $C*$ la solución óptima del TSP en (V, E), entonces se asegura que:

$$cost(C) \leq \frac{3}{2} cost(C*)$$

Demostración:

Nótese que la longitud de la solución óptima no puede ser menor que la longitud de un MST en (V, E), pues en ese caso el árbol no sería minimal en longitud. Entonces

$$cost(MST) \leq cost(C*)$$

Por otro lado, nótese que $cost(M)$ es máximo en el caso en el que todos los vértices del MST tienen grado impar. En este caso, los matching constituyen la mitad del paseo, y dado que son perfect matches

$$cost(M) \leq \frac{1}{2} cost(C*)$$

Luego, notamos que

$$cost(C) \leq cost(MST) + cost(M) \leq \frac{3}{2} cost(C*)$$

IV. Resultados

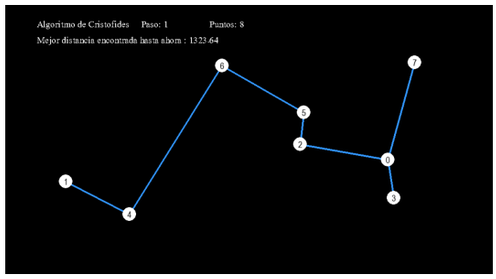

Fig. 1. Ejemplo para $n = 8$ con algoritmo de Christofides

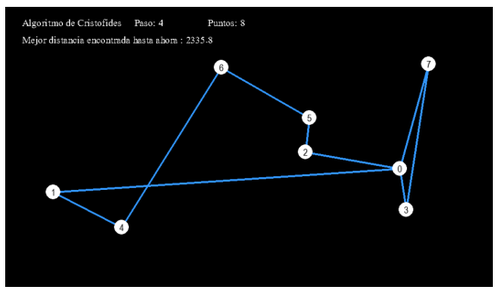

Fig. 2. Solución para $n = 8$ con algoritmo de Christofides

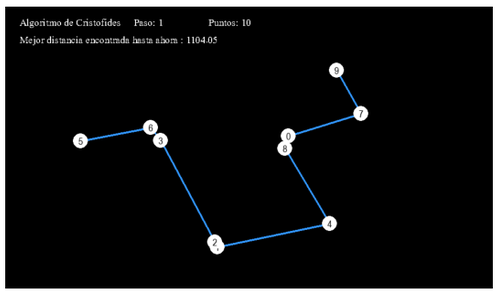

Fig. 3. Ejemplo para $n = 10$ con algoritmo de Christofides

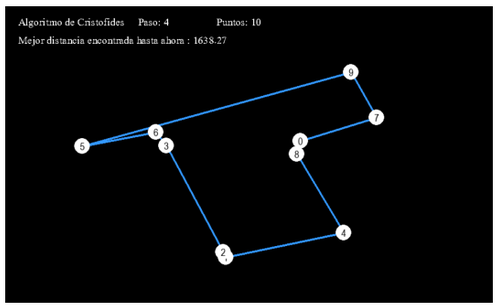

Fig. 4. Solución para $n = 10$ con algoritmo de Christofides

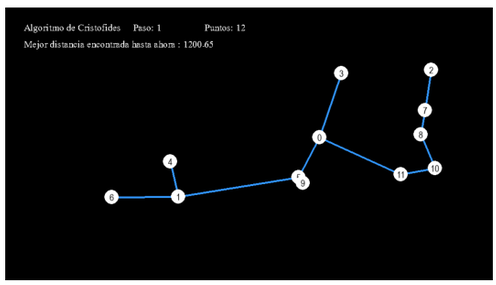

Fig. 5. Ejemplo para $n = 12$ con algoritmo de Christofides

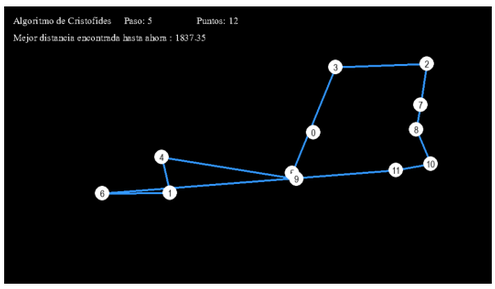

Fig. 6. Solución para $n = 12$ con algoritmo de Christofides

Para las siguientes figuras ilustrativas se utilizó un índice de mutación del 90%, con un criterio de iteraciones máximas (generaciones máximas) de 300 y otro criterio de paro denominado ”generaciones sin mejora” de 50. Este último solo se refiere a que si el algoritmo está evolucionado de forma demasiado lenta, entonces para.

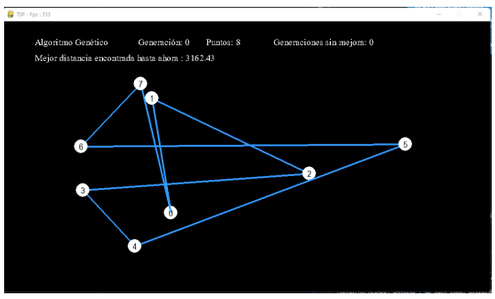

Fig. 7. Ejemplo para $n = 8$ con algoritmo genético

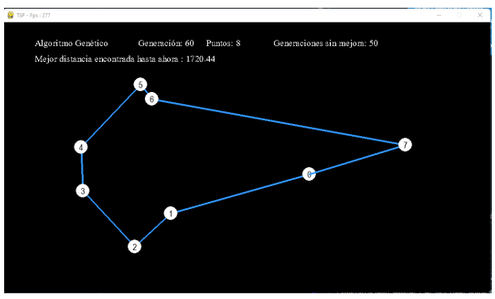

Fig. 8. Solución para $n = 8$ con algoritmo genético

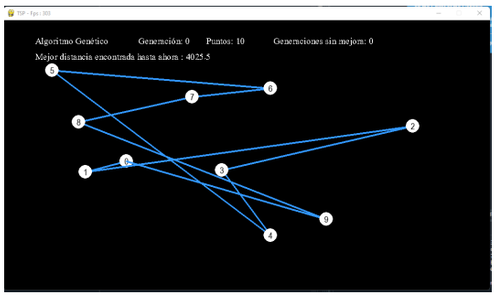

Fig. 9. Ejemplo para $n = 10$ con algoritmo genético

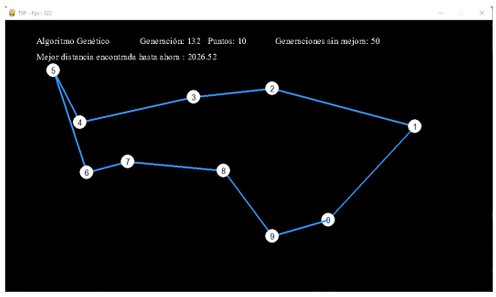

Fig. 10. Solución para $n = 10$ con algoritmo genético

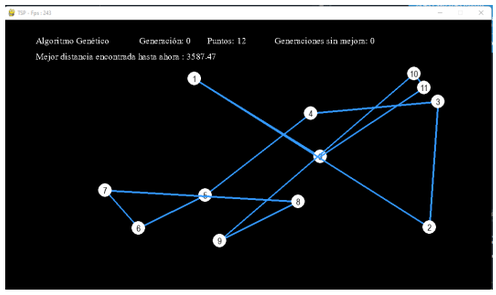

Fig. 11. Ejemplo para $n = 12$ con algoritmo genético

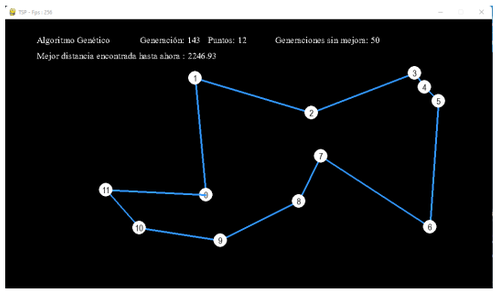

Fig. 12. Solución para $n = 12$ con algoritmo genético

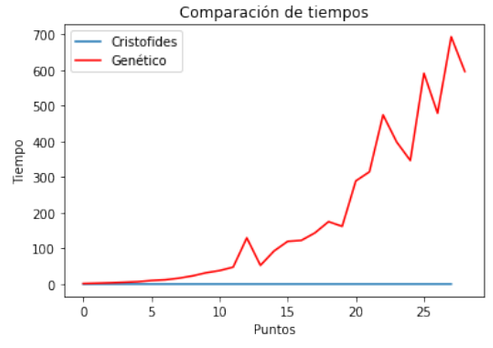

Fig. 13. Comparación de tiempos de ejecución, en segundos

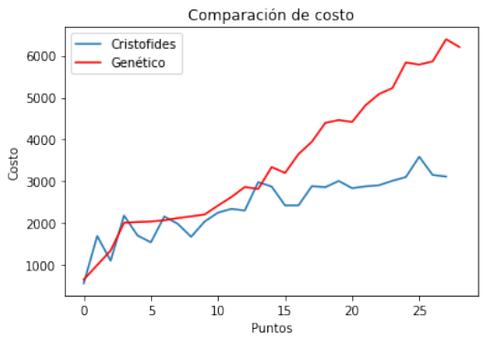

Fig. 14. Comparación de costo de la solución encontrada

Tabla 1: de tiempos de ejecución

puntos	Genético (s)	Christofides (s)
5	4.98	2.97
10	22.77	3.18
15	52.39	3.65
20	174.81	3.98
25	397.91	4.19
30	595.68	4.27

V. Discusión

Empezamos la discusión notando los resultados de Fig. 1, que ilustra la diferencia de tiempos de los algoritmos. Notamos que el algoritmo genético crece exponencialmente, mientras que el algoritmo de Christofides crece en tiempo polinomial, y apenas se nota en la gráfica. Sin embargo esta comparación no es justa, pues un algoritmo es de búsqueda exhaustiva, y está diseñado para buscar la solución óptima sin detenerse; el otro es un algoritmo determinista, diseñado para hallar una aproximación rápida.
Notamos en la Fig. 2 la diferencia en las soluciones encontradas. Antes de los 15 puntos las soluciones son idénticas o casi idénticas. Luego de los 15 puntos el algoritmo genético no llegaba a converger, por lo que se detiene en una solución subóptima.

De igual manera, debemos notar que el algoritmo genético requiere mucho tiempo para converger a una solución óptima. Esto se representa de mejor manera en la **Tabla 1**, donde vemos los tiempos que le toma converger a una solución subóptima. Considerando esto, fácilmente el algoritmo puede llegar a tardar el doble o incluso el triple de tiempo para converger a la menor distancia posible. Mientras que el algoritmo de Christofides tiene una clara ventaja dado que tiene una convergencia mucho más rápida, sin embargo, cómo se establece en la sección de **Teoría**, estas son aproximaciones, y a pesar de que sean bastante precisas, el **algoritmo de Christofides** rara vez encontrará una solución óptima. Además, se debe de tomar en consideración que la forma en que se seleccionan los criterios de paro para el algoritmo genético afectan directamente los resultados. Esto lo podemos observar en la **Figura 14**, en donde claramente el **algoritmo de Christofides** está aproximando de mejor la solución una vez pasado los 12 o 13 puntos. Esto se puede argumentar debido a que el **algoritmo genético** se quedaba sin tiempo debido a los criterios de paro empleados en esta implementación.

Además, podemos observar en las **Figura 1** a la **Figura 12** que los algoritmos funcionan bastante bien con una pequeña cantidad de puntos. Sobretodo, el **algoritmo genético**, el cuál a pesar de demorarse más tiempo ilustraba mejores soluciones a los problemas presentados. Es bastante interesante ver los resultados después de correr cada uno de los algoritmos, y claramente podemos observar que en algunas ocasiones las soluciones de el **algoritmo de Christofides** presentan aristas que se cruzan, por lo que no estaba convergiendo a una solución óptima en todos los casos debido a que la solución óptima de este problema siempre se encuentra cuando no se presenta ningún par de aristas cruzadas. Es evidente, que el conjunto de puntos que atacó el **algoritmo genético** convergió a las soluciones óptimas, al menos para las corridas de $n = 8, 10, 12$ puntos.

En general, notamos que ninguno de los dos algoritmos es suficientemente bueno por si solo, y que si se quiere una solución buena en un tiempo razonable lo mejor sería utilizar una combinación de ambos. Esto se ha hecho anteriormente, primero se calcula una aproximación con el algoritmo de Christofides y luego se inicia un algoritmo genético sobre esa solución. Esto ayuda a reducir el tiempo de ejecución, aproximando la solución con el **algoritmo de Christofides**, y una vez reducido el rango de búsqueda, se puede utilizar el **algoritmo genético** para encontrar la solución óptima. No obstante, si se quiere reducir aún más el tiempo de búsqueda de la solución óptima, se recomienda emplear más de dos algoritmos, ya que esto permite encontrar soluciones óptimas mucho más rápido. La ventaja de esto se evidencia cuando se comienza a trabajar con una gran cantidad de puntos.

VI. Conclusiones

Como primera conclusión, tenemos que el tiempo de ejecución de el **algoritmo genético** es significativamente mayor comparado con el **algoritmo de Christofides**. Esto hace sentido debido a para este problema, el **algoritmo genético** implementado de esta forma representa un problema NP. En contraste, el **algoritmo de Christofides** realiza aproximaciones del problema en tiempo polinomial, por lo que estamos sacrificando exactitud para ganar velocidad de ejecución. Además, podemos ver que la diferencia es notablemente mayor, cuando examinamos con detenimiento la **Tabla 1**, en algunas ocasionas, sobre todo cuando la cantidad de puntos aumenta, la implementación del algoritmo genético puede llegar a tardar hasta 100 veces más que el algoritmo que implementa árboles.

Por otra parte, para valores pequeños de n, podemos tonar que el **algoritmo genético** si

llega a las soluciones óptimas del problema. Analizando detenidamente las **Figura 6** a la **Figura 12**, podemos visualizar que el algoritmo no termina con pares de aristas entrecruzadas, mientras que en el caso de el **algoritmo de Christofides**, esto no siempre ocurre. Esto es precisamente lo que esperamos, dado que Christofides no busca resolver el problema encontrando una solución óptima. Sin embargo, en algunos casos puede llegar a converger a la solución óptima. pero eso está ligado a la aleatoriedad de los puntos y su distribución. En cambio, el **algoritmo genético** si hace una búsqueda exhaustiva de esta solución óptima y por ende, tiende a tardarse un mayor tiempo de ejecución.

Por último, debemos notar que para valores de $n >= 12$ ninguno de los algoritmos está encontrando la solución óptima del problema en cuestión. Esto se puede evidenciar de mejor manera en la **Figura 14**, donde observamos que el costo de la función objetivo comienza a aumentar de forma significativa. En el caso de Christofides, esto se debe por lo establecido con anterioridad, dado que el algoritmo intenta aproximar la solución, y podemos ver que lo hace, pero cuando la muestra de puntos comienza a crecer, casi nunca lo consigue. En el caso de la implementación con genéticos, esto se debe principalmente a los criterios de paro. Es decir, si dejamos correr el algoritmo el tiempo que sea necesario, este encontrará la solución óptima al problema. Sin embargo, dado que lo que estábamos evaluando eran tiempos de ejecución y precisión, nos vimos obligados a colocar dichos criterios de para para realizar múltiples pruebas. Además, esto nos da una pauta bastante importante. De esto podemos concluir, que ninguno de los algoritmos es lo suficientemente bueno por si sólo. Cómo se ha establecido con anterioridad, uno carece de velocidad y el otro carece de exactitud, por ello, lo mejor es optar por una implementación de ambos algoritmos, para obtener mejores resultados de una forma rápida y exacta.

VII. Referencias

- Lawrence, W. (2021). 11 animated algorithms for the traveling salesman problem. Extraído de: https://stemlounge.com/animated-algorithms-for-the-traveling-salesman-problem/
- Korte, B., Vygen, J. (2008). Combinatorial Optimization: Theory and Algorithms (Algorithms and Combinatorics) (4th ed.). Springer.
- Trevisan, L. (2011). Combinatorial Optimization: Exact and Approximate Algorithms. Stanford University.