Estudio de Optimización del TSP Dinámico para la Planeación de Rutas de Entrega de Mercancía.

Galván García, José Adrián; Juan Martín Carpio Valadez; Zamudio Rodríguez Víctor M.; Carlos Lino Ramírez; Héctor José Puga Soberanes
Tecnológico Nacional de México, Instituto Tecnológico de León,
Departamento de Estudios de Posgrado e Investigación, León, Gto, México
pugahector@yahoo.com

Recibido: *7 de octubre de 2022*
Aceptado: *15 de noviembre de 2022*

RESUMEN

El problema del vendedor ambulante dinámico (DTSP) en un ambiente experimental es estocástico y dinámico. La capacidad de cambio requiere que el algoritmo que lo soluciona tenga la capacidad de adaptarse rápidamente. La mayoría de los científicos llaman la atención sobre la correlación entre la diversidad de la población y la convergencia al óptimo. El control de la variación de la población permite una convergencia estable del algoritmo al óptimo y proporciona un buen mecanismo para evitar el estancamiento. En este trabajo se hace un análisis del algortimo Genético y herramientas para la resolución del problema TSP dinámico, incluyendo las librerías de TSPLIB. La libreria TSPLIB consiste de un conjunto de instancias de las cuales se conoce la ruta optima y se utilizan para calibrar el algoritmo Genetico basico con elitismo (SGA). Las herramientas de programación revisadas fueron: Django, Xamarin, React, Flutter para el análisis de elaboración de la aplicación. Se optó por Flutter por la autoria de google (Google maps) y la curva de aprendizaje de menor tiempo para su elaboración. Adicionalmente se utilizó el lenguaje Python para realizar una simulación controlada de puntos al azar, el movimiento al azar del centroide de los puntos con radio fijo especificado (controlado por el usuario), y la generación de rutas incluyendo y excluyendo puntos en movimiento.

Palabras claves: TSP; DTSP; Optimización; Algoritmos Genéticos.

ABSTRACT

The dynamic traveling salesman problem (DTSP) in an experimental environment is stochastic and dynamic. The ability to change requires that the algorithm that solves it has the ability to adapt quickly. Most scientists draw attention to the correlation between population diversity and convergence to the optimum. Controlling population variation allows stable convergence of the algorithm to the optimum and provides a good mechanism to avoid stagnation. In this work, an analysis of the genetic algorithm and tools for solving the dynamic TSP problem is made, including the TSPLIB libraries that indicates which is the optimal route to be able to evaluate the proposed Genetic algorithm as a strategy for solving the problem. The TSPLIB library consists of a set of instances of which the optimal path is known and used to calibrate the basic genetic algorithm with elitism (SGA). The programming tools reviewed were Django, Xamarin, React, Flutter for the application development analysis. Flutter was chosen because of it google authoring (Google maps) and the learning curve of less time for its development. Additionally, a controlled simulation of random points was performed using the Python language, and the random movement with fixed radius (controlled by the user) specified to verify the response time of route generation including and excluding moving points.

1. INTRODUCCIÓN

El problema del agente viajero (TSP) es uno de los problemas más intensamente estudiados en la optimización combinatoria (Babel L. , 2020). Si la distancia entre dos puntos cualesquiera es la distancia euclidiana, el problema se denomina problema del vendedor ambulante euclidiano (ETSP) (Babel L., 2020). En este problema, un viajante tiene que visitar un conjunto de puntos (ciudades), terminando el recorrido en el punto inicial, de forma que pase solamente una vez por cada punto y que el trayecto total realizado sea mínimo. El problema del vendedor viajero asimétrico (ATSP) es un problema donde la distancia entre dos nodos no es simétrica (Nalepa, 2019). Otras variaciones del TSP incluyen el Problema del vendedor ambulante con los vecindarios (TSPN) (Restrepo J. H., 2004), el Problema del vendedor ambulante de cuello de botella (BTSP), (Riaño, 2018), entre otros (S. Dokania, 2017).

Los problemas de planificación de rutas mencionadas anteriormente tienen como objetivo encontrar los mejores recorridos posibles sin tener en cuenta las características del vehículo. Hay restricciones y características que para este trabajo no se están considerando, pero cuando se trabaja con vehículos del mundo real, hay que tener en cuenta. Por ejemplo, las restricciones cinemáticas como el radio de giro mínimo (Guntsch M. &., 2001) (Otto, 2018). Los vehículos con restricciones de movimiento impuestas por el mecanismo de dirección satisfacen una restricción. Tales vehículos no son capaces de seguir caminos obtenidos de las soluciones de los problemas clásicos de TSP. Otros casos que no se están considerando son los siguientes: los robots móviles similares a automóviles o los vehículos aéreos de ala fija que avanzan a una velocidad constante y giran con curvatura acotada superior pueden modelarse como un vehículo (Otto, 2018). Estos casos mencionados anteriores se conocen como DTSP o TSP con Curvatura (LaValle, 2006).

El DTSP ha atraído considerable atención debido a muchas aplicaciones civiles y militares (Otto, 2018). Una configuración típica es monitorear una colección de puntos distribuidos espacialmente por un vehículo aéreo no tripulado (UAV) (S. Dokania, 2017). Esto podría referirse al control del tráfico en lugares específicos, la recopilación de inteligencia y el reconocimiento de puntos sospechosos para operaciones antiterroristas, misiones de seguridad y vigilancia de infraestructuras críticas y otros puntos de interés, el apoyo a las misiones de combate mediante operaciones de inteligencia, vigilancia y reconocimiento, la evaluación de los daños de batalla (confirmación de un punto objetivo y verificación de su destrucción), entre otros. Para otras aplicaciones (Nalep, 2019).

Normalmente, cuando tenemos un problema de optimización, podemos recurrir a dos tipos de algoritmos para solucionar el problema, los algoritmos exactos y los algoritmos heurísticos. Para el primer caso, hay algoritmos que pueden encontrar la mejor solución (solución óptima) de manera determinista para dicho problema. La desventaja de este tipo de algoritmos es que, en primera, suelen ser difíciles de modelar e implementar, ya que requieren de una capacidad analítica, además de conocimientos sobre matemáticas aplicadas y programación. Otra desventaja de estos métodos exactos es que estos exigen una cantidad de recursos computacionales considerable. Sin embargo, a pesar de todas estas desventajas, utilizar este tipo de algoritmos siempre que sea factible es la mejor opción. Por otro lado, puede haber casos donde definitivamente el tamaño del problema sea muy grande y no tengamos la posibilidad de utilizar un método exacto.

Estando en esta situación, la única opción que tenemos es optar por alguna técnica que nos brinde una solución que muy posiblemente no sea la óptima pero que al menos nos asegure que la solución brindada sea una solución factible y de calidad, es aquí cuando los algoritmos heurísticos y metaheurísticos toman relevancia.

Como parte de la investigación en la que estamos interesados, se tienen dos escenarios para la aplicación del problema DTSP: En la primero, una persona necesita conocer diferentes sitios en donde se encuentran los productos que desea adquirir y generar la ruta óptima para adquiridos con la libertad de recorrer toda o parte de la ruta (Galván, 2022). En el segundo, una tienda online debe de entregar los productos en cierta zona de influencia y generar la ruta óptima sujeta a las restricciones: a) cancelación de pedidos o b) se añaden nuevos productos a la ruta de entrega. En el presente trabajo se reportan los resultados enfocados en el segundo caso mencionado.

El TSP Dinámico (DTSP) fue introducido en 1998 por Psaraftis (Psaraftis, 1988). El DTSP es un TSP en el cual ciudades pueden ser agregadas o eliminadas en tiempo real, o cuando el costo de viajar entre ciudades puede cambiar.

Un TSP Dinámico es un TSP determinado por una matriz de costos dinámica (Whigham, 2006), como se muestra a continuación:

$$D(t) = \{d_{ij}(t)\}_{n(t)\times n(t)} \tag{1}$$

d_{ij} $Es\ el\ costo\ de\ viajar\ de\ la\ ciudad\ C_i\ a\ la\ ciudad\ C_j$

$t = Es\ un\ tiempo\ determinado$

Donde $d_{ij}(t)$ es el costo desde el punto (ciudad) c_i al punto ciudad c_j, y t es el tiempo real de recorrido. En esta definición, el número de ciudades $n(t)$ y el costo de la matriz son dependientes del tiempo. El problema dinámico del agente viajero es encontrar el mínimo costo de la ruta que contiene todos los $n(t)$ nodos.

Dados los $n(t)$ nodos $(P_1, P_2, \dots, P_{n(t)})$, y la correspondiente matriz de costo $D = \{d_{ij}(t)\}, i, j = 1,2, \dots, n(t)$, encontrar la ruta de mínimo costo que contiene todos los $n(t)$ puntos, donde t es el tiempo entre el punto P_i y el punto P_j, y $d_{ij}(t) = d_{ji}(t)$.

En la definición consideremos el cambio del DTSP de la matriz de costos con el tiempo continuo del proceso. Prácticamente discretizamos este proceso de cambio, Así, un DTSP se convierte en una serie de problemas de optimización TSP.

$$D(t_k) = \{d_{ij}(t_k)\}_{n(t_k)\times n(t_k)} \tag{2}$$

$k = 0,1,2, \dots, m-1$, con la ventana de tiempo $[t_k, t_{k+1}]$, donde $\{t_k\}_{k=0}^{m}$ es una secuencia de TSP´s.

2. METODOLOGIA

Un algoritmo Genético básico con elitismo (SAG por sus siglas en inglés), está basado en la generación de población aleatoria inicial y un conjunto de operadores como son, selección, cruce y mutación, y la optimización gradual de la población a un estado que contenga la solución óptima aproximada (Chura, 2015).

A. Evaluación

Los algoritmos Genéticos son potentes herramientas para resolver problemas de TSP, y se tomó como base para resolver el problema del DTSP con la estrategia de almacenar la última ruta optima durante su trayecto, introducirlo a la población y obtener más rápido una manera más eficiente el resultado.

En el algoritmo 1 se muestra el pseudocódigo del SGA utilizado.

Algoritmo **1** Pseudocódigo de un Simple Algoritmo Genético

```
/* Parámetros de un AG */
t=0 // número de generación
P (0)
Evaluar P (0)
While not (Condición de terminación) do
  P´(t) = seleccionar(P(t))
  P´(t) = aplicar operadores de cruce(P(t))
  P(t+1) = reemplazar (P(t), P'(t))
  Evaluar P(t+1)
  t=t+1
  return Mejor Solución Encontrada
end while
```

2.1 DISEÑO

La utilización de los archivos de TSPLIB (Heidelberg, s.f.) proporciona las coordenadas de las ciudades y la ruta óptima para la entrada de datos para la utilización del simple algoritmo genético para generar la ruta optima que, por medio de la indicación de generaciones o paro por tiempo, nos da el resultado ya que el planteamiento es el DTSP es añadir y eliminar ciudades se almacena la ruta óptima para que el algoritmo sea eficiente en el resultado.

2.2 MATERIALES

El algoritmo propuesto utilizo instancias de prueba tomadas de la librería TSPLIB. Dichos algoritmos se ejecutaron en una laptop (Intel(R) Core (TM) i7-5500U CPU 2.40GHz: 16GB),
Utilizando el lenguaje de programación Python 3.10.6.

La simulación de productividad fue implementada utilizando la librería Turtle (para el procesamiento de los gráficos. Los experimentos fueron evaluados mediante SGA propuesto para resolver problema del DTSP, utilizando el parámetro de tiempo como criterio de paro del algoritmo para mostrar la ruta.

2.3 ENTORNO

El entorno para generar la población inicial de los puntos fue controlado con dos opciones: La primera es tomando las coordenadas de la librería TSPLIB, y la segunda es generando aleatoriamente los puntos. Una vez generada la población inicial, se toma un punto al azar como punto de inicio en ambas opciones. Tomando como centroide el punto de inicio, el usuario decide el tamaño del radio de selección de puntos para construir la ruta optima, utilizando el algoritmo SGA propuesto (ver Figura 1). Posteriormente se establece un vecindario alrededor del centroide y selecciona un nuevo centroide dentro de este vecindario. Se vuelve aplicar el algoritmo SGA para generar la nueva ruta,

introduciendo en la nueva población la ruta óptima que se generó anteriormente, acompañado de un proceso de análisis para eliminar puntos de esta ruta optima que están fuera del nuevo vecindario de selección de puntos.

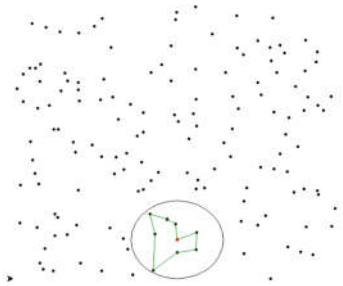

Figura 1: Fuente: Elaboración propia código Python, 900x900, puntos 150, Ch150.lib, radio 100

2.4 TRATAMIENTO

Se programo un algoritmo SGA utilizando el lenguaje Python 3.10.6. Se leen los archivos TSPLIB o se generar puntos aleatoriamente, el algoritmo SGA procesa las coordenadas de cada punto para medir las distancias euclídeas entre puntos consecutivos, la suma de las distancias de la ruta nos da la instancia del recorrido donde buscamos el mínimo para optimizar recursos.

2.5 ANALISIS DE INFORMACIÓN

Para contrastar el resultado entre la ruta optima de TSPLIB y la generada por el algoritmo SGA propuesto, se utilizo la Prueba de Kendall (A. Jay, 1998), la cual se plantea a continuación.

Hipótesis nula.
Ho: La ruta generada por el algoritmo SGA $=$ La ruta optima de TSPLIB.
H1: La ruta generada por el algoritmo SGA $\neq$ La ruta optima de TSPLIB.

Al aplicar la Prueba de Kendall se obtuvieron los siguientes resultados.
Kendall coeficiente de correlación es: 0.1423525
z = 2.5936, p-value = 0.009498
0.009498 < 0.05 (se acepta la hipótesis alternativa)
τ = 0.1423525

3. RESULTADOS.

Los parámetros que se establecieron para el entrenamiento del algoritmo SGA fueron: Población inicial 50, Probabilidad de cruza 0.9, Probabilidad de mutación 0.5, 2000 generaciones. Los resultados se muestran en la tabla 1.

Tabla 1: Evaluación del TSP, Experimentos 33 veces y el resultado es el siguiente

TSPLIB	Optima Distancia	Distancia obtenida (la mejor, AG articulo)	Diferencia distancia optima y AG	Tiempo Concorde	Tiempo de ejecución (AG)
A280	2579	2582	3	5.04	187.4927
ATT48	33522	33580	58	0.50	17.64321
BERLIN52	7542	7542	0	0.20	20.7606
CH130	6110	6195	85	0.40	91.7263
ELI76	538	538	0	0.30	35.1462

Tabla 2: Evaluación del dinámico TSP, Experimento del resultado es el siguiente

TSPLIB	A280	ATT48	BERLIN52	CH130	ELI76
Tiempo para inserción de puntos	90.7958	12.6625	9.1090	30.8054	21.5682
Tiempo para eliminación de puntos	87.6544	10.1032	8.7077	30.3712	18.9942

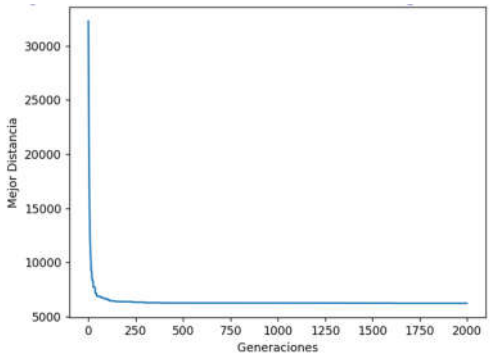

Figura 1: Fuente: Elaboración propia código Python, Mejor distancia contra el número de generaciones

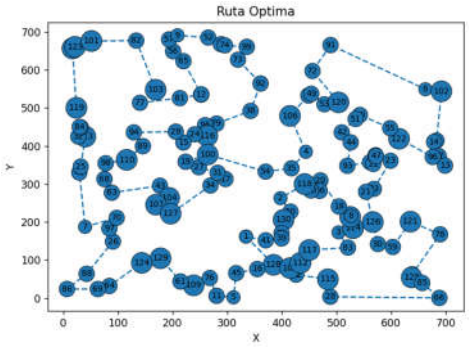

Figura 2: Fuente: Elaboración propia código Python, TSPLIB CH130.TSP, representa la ruta optima

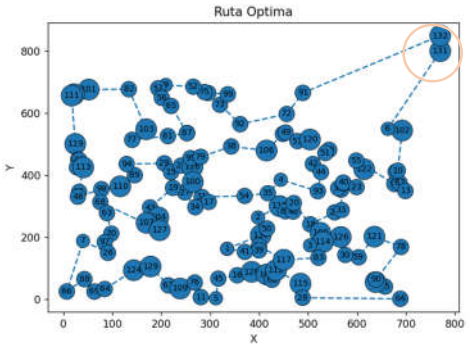

Figura 3: Fuente: Elaboración propia código Python, Añadiendo 2 puntos P1(770,800) y P2(770,850)

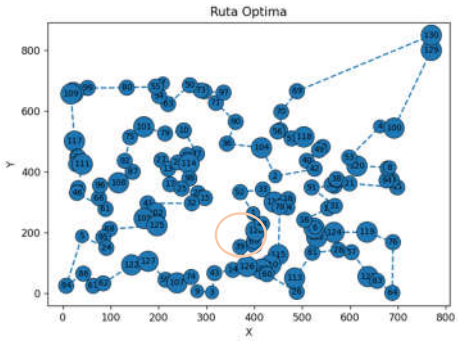

Figura 4: Fuente: Elaboración propia código Python, borrando los 2 primeros puntos

P1(334.5909,161.7809) y P2(397.6446,262.81653)

4. DISCUSIÓN

Se trabajo con el problema del agente viajero con el contexto de entrega de mercancías. El problema nos permitir identificar la entrega de la mercancía en los puntos especificados con la ruta más corta. Con el algoritmo SGA pudimos obtener una respuesta rápida sobre la ruta. Dado que el repartidor (centroide) se encontraba en movimiento, se añadió un módulo para eliminar o insertar puntos en la trayectoria del repartidor y con ello se generó la ruta nueva de manera efectiva.

Utilizamos instancias de la librería de TSPLIB, para la calibración de los parámetros del algoritmo SGA propuesto en la generación de la ruta optima. Comparamos el tiempo de respuesta del algoritmo SGA propuesto contra el tiempo que le tomo al algoritmo Concord. Ambas comparaciones se muestran en la tabla 1.

Al experimentar con el algoritmo SGA pudimos observar que éste encuentra la ruta optima indicándonos los puntos donde pueden entregar la mercancía.

Se logró un desempeño en la generación de la ruta optima en la simulación del movimiento dinámico, donde se mostró la ruta en el siguiente cambio de posición y se generó la nueva ruta.

5. CONCLUSIONES Y/O PROYECTOS FUTUROS.

Con la utilización de las instancias tomadas de la librería TSPLIB fue posible calibrar los parámetros del algoritmo SGA propuesto para la generación de la ruta optima. Con la comparación con el algoritmo Concord pudimos calibrar el tiempo de respuesta. Una vez calibrado en algoritmo SGA, los resultados que se obtuvieron nos permite concluir que la aplicación de este algoritmo es útil para la generación de la ruta optima factible y de calidad que solucione el problema DTSP.

De acuerdo con los resultados obtenidos al realizar las pruebas, se propone como trabajo futuro evaluar otros algoritmos metaheurísticos evolutivos para resolver el problema del TSP y DTSP y contrastar el algoritmo SGA propuesto en este trabajo.

AGRADECIMIENTO.

El autor agradece al Consejo Nacional de Ciencia y Tecnología (CONACyT) por el apoyo económico brindado a través de becas de posgrado CVU 1109353. También al Tecnológico Nacional de México, en particular a la División de Estudios de Posgrado e Investigación (DEPI) del Instituto Tecnológico de León donde se desarrolló este trabajo.

6. REFERENCIAS BIBLIOGRAFICAS

A. Jay, S. S. (1998). *Estadística No Paramètrica Aplicada a las ciencias de la conducta.* Mexico: Trillas.

Adrián Quispe Andía, K. M. (2019). *Estadística no parametrica aplicada a la investigación científica con software.* Colombia: Eidec.

Babel. (2020). *New heuristic algorithms for the Dubins traveling salesman problem.* Journal of Heuristics, 1-28.

Boryczka, U. &. ((2015, March).). *Diversification and entropy improvement on the DPSO algorithm for DTSP. In Asian Conference on Intelligent Information and Database Systems (pp. 337-347).* Cham.: Springer.

C. Archetti, D. F. (2020). Dynamic travelling Salesman problem with stochastic release dates. *spring*, 5-6.

Chura, H. E. ((2015)). Aplicación del algoritmo de colonia de hormigas al problema del agente viajero. *Ciencia & Desarrollo*, 98-102.

García, J. A. (2022). TSP dinámico para la toma de decisiones en escenarios de e-commerce. *5 congreso Estudiantil de inteligencia artificial aplicada a la ingenieria y tecnologia (CEIAAIT)* (pág. 4). Unam Cuatitlan: Universidad NAcional autónoma de México .

Guntsch, M. &. (2001). Pheromone modification strategies for ant algorithms applied to dynamic TSP. In Workshops on applications of evolutionary computation. *Springer*, pp. 213-222.

Guntsch, M. M. (2001). An ant colony optimization approach to dynamic TSP. *In Proceedings of the 3rd annual conference on genetic and evolutionary computation*, pp. 860-867.

Heidelberg, U. (s.f.). *http://comopt.ifi.uni-heidelberg.de/*. Obtenido de http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp/

Hoboken, N. J. (2009). Aplicación del algoritmo de colonia de hormigas al problema del agente viajero. *Ciencia & Desarrollo*, 98-102.

Jakub, N. (s.f.). *Smart Delivery Systems , Solving complex vehicle Routing Problems, intelligent Data centric System.* Elservier.

Kopel, A. S. (2019). Solving dynamic TSP by parallel and adaptive ant colony communities. *Journal of Intelligent & Fuzzy Systems*, 1-4.

LaValle, S. (2006). *Planning Algorithms. Cambridge University Press.* Cambridge.

Nalep, J. (2019). *Smart Delivery Systems , Solving complex vehicle Routing Problems.* Elservier.

Nalepa, J. (2019). *Smart Delivery Systems , Solving complex vehicle Routing Problems, intelligent Data centric System.* Elservier.

Otto, A. A. (2018). *Optimization approaches for civil applications of unmanned aerial vehicles (UAVs) or aerial drones: a survey.* https://doi.org/10.1002/net.21818.

Petr Stodola, K. M. (2020). Hybrid algorithm based on ant colony optimization and simulated anneling applied to the dynamic travelling salesman problem. *entropy*, 1-7.

Psaraftis. (1988). Dynamic vehicle routing problems. *Vehicle Routing*, 223–248.

Ragav Sachdeva, F. N. (2020). The dynamic travelling thief problem: Benchmarks and Performance of Evolutionary Algorithms. *ResearchGate*, 1-5.

Restrepo, J. H. (2004). Aplicación de la teoría de grafos y el algoritmo de Dijkstra para determinar las distancias y las rutas más cortas en una ciudad. *Scientia et technica*, 121-126.

Riaño, E. R.-B. (2018). Árbol de caminos mínimos: enrutamiento, algoritmos aproximados y complejidad. *REVISTA COLOMBIANA DE TECNOLOGIAS DE AVANZADA*, 11-21.

S. Dokania, S. B. (2017). Opportunistic Self Organizing Migrating Algorithm for real-time Dynamic Traveling Salesman Problem. *2017 51st Annual Conference on Information Sciences and Systems (CISS)*, pp. 1-6.

Whigham, G. D.-D. (2006). Simulated Evolution and Learning 6th International Conference. *Springer*, 15-18,.