

Instituto Tecnológico de León

**“Técnicas de softcomputing aplicadas al TSP dinámico
Para la toma de decisiones en escenarios de e-commerce”**

Tesis

Que presenta:

JOSÉ ADRIÁN GALVÁN GARCÍA

**Para obtener el grado de
Maestro en Ciencias de la Computación**

Con la Dirección de:

Dr. Juan Martín Carpio Valadez

Y Co Dirección de:

Dr. Víctor Manuel Zamudio Rodríguez

Revisores

Dr. Carlos Lino Ramírez

Dr. Héctor José Puga Soberanes

León Guanajuato diciembre 2022

León, Guanajuato, 11/enero/2023
OFICIO No. DEPI-005-2023


C. JOSÉ ADRIÁN GALVÁN GARCÍA
PRESENTE

De acuerdo al fallo emitido por la Comisión Revisora, integrada por los: **Dr. Juan Martín Carpio Valdez**, **Dr. Víctor Manuel Zamudio Rodríguez**, **Dr. Carlos Lino Ramírez** y **Dr. Héctor José Puga Soberanes**, y considerando que cubre todos los requisitos establecidos en los Lineamientos Generales para la Operación del Posgrado del Tecnológico Nacional de México, **se autoriza la impresión** del trabajo de tesis titulado: "Técnicas de soft computing aplicadas al TSP dinámico para la toma de decisiones en escenarios de e-commerce".

Lo que hacemos de su conocimiento para los efectos y fines correspondientes.

ATENTAMENTE

Excelencia en Educación Tecnológica®
Ciencia Tecnología y Libertad


LIC. PATRICIA DONATO JIMÉNEZ
SUBDIRECTORA ACADÉMICA

C.c.p. Expediente


JMCV/CLDG



Av. Tecnológico s/n Fracc. Industrial Julián de Obregón C.P.37290 León, Guanajuato.
Tel.: 477 710 5200 e-mail: tecleon@leon.tecnm.mx tecnm.mx | leon.tecnm.mx



León, Guanajuato, a 11 de enero del 2023

C. ING. LUIS ROBERTO GALLEGOS MUÑOZ
JEFE DE SERVICIOS ESCOLARES
PRESENTE

Por este medio hacemos de su conocimiento que la tesis titulada "Técnicas de softcomputing aplicadas al TSP dinámico Para la toma de decisiones en escenarios de e-commerce", ha sido leída y aprobada por los miembros del Comité Tutorial para su evaluación por el jurado del acto de examen de grado al alumno (a) C. José Adrián Galván García con número de control M93240319 como parte de los requisitos para obtener el grado de Maestro(a) en Ciencias de la Computación (MCCOM-2011-05).

Sin otro particular por el momento, quedamos de Usted.

ATENTAMENTE
COMITÉ TUTORIAL



Dr. Juan Martín Carpio Valadez

DIRECTOR



Dr. Víctor Manuel Zamudio Rodríguez

CODIRECTOR



Dr. Carlos Lino Ramírez

REVISOR



Dr. Héctor José Puga Soberanes

REVISOR





DECLARACION DE AUTENTICIDAD Y DE NO PLAGIO

Yo, José Adrián Galván García identificado con No. Control M93240319, alumno (a) del programa de la **Maestría en Ciencias de la Computación**, autor (a) de la Tesis titulada: "Técnicas de softcomputing aplicadas al TSP dinámico Para la toma de decisiones en escenarios de e-commerce" DECLARO QUE:

1.- El presente trabajo de investigación, tema de la tesis presentada para la obtención del título de **MAESTRO (A) EN CIENCIAS DE LA COMPUTACIÓN** es original, siendo resultado de mi trabajo personal, el cual no he copiado de otro trabajo de investigación, ni utilizado ideas, fórmulas, ni citas completas "stricto sensu", así como ilustraciones, fotografías u otros materiales audiovisuales, obtenidas de cualquier tesis, obra, artículo, memoria, etc. en su versión digital o impresa.

2.- Declaro que el trabajo de investigación que pongo a consideración para evaluación no ha sido presentado anteriormente para obtener algún grado académico o título, ni ha sido publicado en sitio alguno.

3.- Declaro que las pruebas o experimentos derivados de esta investigación fueron realizados bajo el consentimiento de los involucrados y con fines estrictamente académicos conforme a criterios éticos de confidencialidad.

Soy consciente de que el hecho de no respetar los derechos de autor y hacer plagio, es objeto de sanciones universitarias y/o legales por lo que asumo cualquier responsabilidad que pudiera derivarse de irregularidades de la tesis, así como de los derechos sobre la obra presentada.

Asimismo, me hago responsable ante el Tecnológico Nacional de México/Instituto Tecnológico de León o terceros, de cualquier irregularidad o daño que pudiera ocasionar por el incumplimiento de lo declarado.

De identificarse falsificación, plagio, fraude, o que el trabajo de investigación haya sido publicado anteriormente; asumo las consecuencias y sanciones que de mi acción se deriven, responsabilizándome por todas las cargas pecuniarias o legales que se deriven de ello sometiéndome a las normas establecidas en los Lineamientos y Disposiciones de la Operación de Estudios de Posgrado en el Tecnológico Nacional de México.

León, Guanajuato a 19 del mes de Enero de 2023

Nombre y firma del autor

José Adrián Galván García



ACUERDO PARA USO DE OBRA (TESIS DE GRADO)

A QUIEN CORRESPONDA

PRESENTE

Por medio del presente escrito, José Adrián Galván García (en lo sucesivo el AUTOR) hace constar que es titular intelectual de la obra denominada: "Técnicas de softcomputing aplicadas al TSP dinámico Para la toma de decisiones en escenarios de e-commerce", (en lo sucesivo la OBRA) en virtud de lo cual autoriza al Tecnológico Nacional de México/Instituto Tecnológico de León (en lo sucesivo TECN/IT León) para que efectúe resguardo físico y/o electrónico mediante copia digital o impresa para asegurar su disponibilidad, divulgación, comunicación pública, distribución, transmisión, reproducción, así como digitalización de la misma con fines académicos y sin fines de lucro como parte del Repositorio Institucional del TECN/ITLeón.

De igual manera, es deseo del AUTOR establecer que esta autorización es voluntaria y gratuita, y que de acuerdo a lo señalado en la Ley Federal del Derecho de Autor y la Ley de Propiedad Industrial el TECN/IT León cuenta con mi autorización para la utilización de la información antes señalada, estableciendo que se utilizará única y exclusivamente para los fines antes señalados. El AUTOR autoriza al TECN /IT León a utilizar la obra en los términos y condiciones aquí expresados, sin que ello implique se le conceda licencia o autorización alguna o algún tipo de derecho distinto al mencionada respecto a la "propiedad intelectual" de la misma OBRA; incluyendo todo tipo de derechos patrimoniales sobre obras y creaciones protegidas por derechos de autor y demás formas de propiedad intelectual reconocida o que lleguen a reconocer las leyes correspondientes. Al reutilizar, reproducir, transmitir y/o distribuir la OBRA se deberá reconocer y dar créditos de autoría de la obra intelectual en los términos especificados por el propio autor, y el no hacerlo implica el término de uso de esta licencia para los fines estipulados. Nada de esta licencia menoscaba o restringe los derechos patrimoniales y morales del AUTOR.

De la misma manera, se hace manifiesto que el contenido académico, literario, la edición y en general de cualquier parte de la OBRA son responsabilidad de AUTOR, por lo que se deslinda al (TECN/ITLeón) por cualquier violación a los derechos de autor y/o propiedad intelectual, así como cualquier responsabilidad relacionada con la misma frente a terceros. Finalmente, el AUTOR manifiesta que estará depositando la versión final de su documento de Tesis, OBRA, y cuenta con los derechos morales y patrimoniales correspondientes para otorgar la presente autorización de uso.

En la ciudad de León, del estado de Guanajuato a los 19 días del mes de Enero de 2023.

Atentamente,

Nombre y firma autógrafa de EL AUTOR

José Adrián Galván García

Índice

1	Introducción	12
1.1	Aplicaciones de TSP	14
1.1.1	Transporte de mercancías y similares	14
1.1.2	Taladrado de placas de circuitos.....	14
1.1.3	Orden de recogida en un almacén	15
1.1.4	Cristalografía de rayos-X.....	15
1.1.5	Aplicaciones a e-commerce y otras aplicaciones industriales	16
1.2	Definición del problema	18
1.3	Justificación	20
1.4	Objetivos	21
1.4.1	Objetivo General	21
1.4.2	Objetivos Específicos	21
1.5	Hipótesis	22
2	Marco Teórico	23
2.1	Aspectos fundamentales de teoría de grafos	23
2.1.1	Definiciones de teoría de grafos.....	25
2.1.1.1	Grafo.....	25
2.1.1.2	Grafos conexos y no conexos	26
2.1.1.3	Grafos dirigidos	27
2.1.1.4	Grafos ponderados	28
2.1.1.5	Grafo completo.....	29
2.1.2	Vértices o nodos adyacentes y matriz de adyacencia	29
2.1.3	Caminos y ciclos en un grafo	31
2.1.4	Ciclo Hamiltoniano y el problema del agente viajero.	33
2.2	Optimización combinatoria.....	35
2.3	Programación heurística.....	35
2.3.1	Heurística	36

2.3.2	Búsqueda local.....	40
2.3.3	Búsqueda local <i>k-opt</i>	41
2.3.4	Metaheurísticas.....	42
2.3.4.1	Algoritmos metaheurísticos poblacionales.....	43
2.3.4.2	Algoritmos metaheurísticos trayectoriales.....	45
2.3.4.3	Algoritmo de búsqueda local iterada.....	45
2.3.4.4	Algoritmo del vecino más cercano.....	48
3	Estado del arte.....	50
3.1	Modelo formal de TSP.....	53
3.2	El problema dinámico del agente viajero.....	56
3.3	Otros problemas sobre rutas.....	57
4	Metodología.....	61
4.1	Librería TSPLIB.....	61
4.1.1	Problema del viajante de comercio simétrico (TSP).....	61
4.1.2	Problema de ordenación secuencial (SOP).....	61
4.1.3	Problema de enrutamiento de vehículos capacitados (CVRP).....	62
4.2	Diseño.....	66
4.3	Materiales.....	66
4.4	Entorno.....	66
4.5	Tratamiento.....	67
4.6	Análisis de información.....	67
5	Experimentación.....	69
5.1	Resultados de la experimentación del algoritmo genético con elitismo en la solución al problema de agente viajero dinámico (DTSP por sus siglas en inglés)	69
5.1.1	Evaluación con la librería TSPLIB.....	72
5.2	Resumen y conclusiones del estado del arte.....	74
5.3	Simulación DTSP.....	76
6	Resultados.....	82
6.1	Tabla 1: Evaluación del TSP, Experimentos 33 veces y el resultado es el siguiente.....	82

6.2	Tabla 2: Evaluación del dinámico TSP, Experimento del resultado es el siguiente.....	82
7	Conclusiones.....	83
8	Referencias	91

Índice Figuras

Figura 2.1:	Königsberg problema del puente (adaptado de [7]).	24
Figura 2.2:	Algunos grafos (adaptado de [10]).	25
Figura 2.3:	Grafos conexo y no conexo (adaptado de [10])	26
Figura 2.4:	Grafo dirigido (adaptado de [10]).	27
Figura 2.5:	Se muestran tres ejemplos de grafos completos (adaptado de [10]). ..	29
Figura 2.6:	Matriz de adyacencia correspondiente (adaptado de [4])......	30
Figura 2.7:	Un camino euleriano recorre todas las aristas exactamente una vez (puede repetir vértices) (adaptado de [4]).	31
Figura 2.8:	Un camino hamiltoniano no recorre todos los vértices exactamente una vez (adaptado de [4])......	32
Figura 2.9:	Orden en que se recorre un grafo en una búsqueda en profundidad (adaptado de [4])......	32
Figura 2.10:	Un ciclo euleriano pasa por todas las aristas exactamente una vez, regresando al punto de partida (adaptado de [4]).	32
Figura 2.11:	Un ciclo hamiltoniano pasa por todos los vértices exactamente una vez, regresando al punto de partida (adaptado de [4]).	33
Figura 2.12:	El grafo del Icosian Game (adaptado de [4]).	33
Figura 2.13:	Clasificaciones heurísticas constructivas (adaptado de [12]).	38
Figura 2.14:	Clasificaciones heurísticas de dos fases (adaptado de [13])......	39

Figura 2.15: Clasificaciones heurísticas de mejora (adaptado de [13]).....	40
Figura 2.16: Búsqueda local (adaptado de [5]).....	47
Figura 3.2: El CVRP y sus variantes (adaptado de [27]).	59
Figura 5.3: Coordenadas de prueba (elaboración propia).	69
Figura 4.4: Mejor distancia contra el número de generaciones (elaboración propia).	
71	
Figura 5.5: Grafica puntos a recorrer (elaboración propia).....	72
Figura 5.6: Mejor distancia contra número de generaciones (elaboración propia).	73
Figura 5.7: Ruta optima (elaboración propia).....	73
Figura 5.8: Ruta optima generada por el Algoritmo Genético con elitismo (elaboración propia).	74
Figura 5.9: Primer movimiento (elaboración propia).....	77
Figura 5.10: Se genera la ruta en el primer movimiento (elaboración propia).	78
Figura 5.11: Segundo movimiento (elaboración propia).	78
Figura 5.12: Ruta generada en el segundo movimiento (elaboración propia).....	79
Figura 5.13: Tercer movimiento (elaboración propia).....	80
Figura 5.14: Ruta generada en el tercer movimiento (elaboración propia).....	80
Figura 5.15: Aplicación en dispositivo móvil Flutter (elaboración propia).	81

Índice Tablas

Tabla 3.1 Simétrica de distancias de 15 puntos	70
Tabla 3.2 Suma de distancias entre cada nodo, ruta	71

Índice Algoritmo

Algoritmo 2.1 Dijistra	28
Algoritmo 2.2 Método de búsqueda local	46
Algoritmo 2.3 Vecino más cercano	50

Agradecimientos

Quiero agradecer 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 al Departamento de Posgrado e Investigación (DEPI) del Instituto Tecnológico de León donde se desarrolló este trabajo.

A mis compañeros y colegas de posgrado, porque de ellos obtuve una amistad, que como compañeros mostraron un buen equipo de trabajo, y tener ganas de seguir trabajando como equipo para futuros proyectos.

A mis hermanos que con su apoyo día a día me mostraron que la familia es el empujón a mi hermana porque estuvo hay en sustitución de mi mama, a mis hermanos que con sus palabras me motivaron a seguir luchando.

A mis profesores que estuvieron apoyándome en la realización de la investigación y me guiaron para la elaboración de la investigación que me motivo para que lo realizara, y la oportunidad de conocer el ramo de la inteligencia artificial que para mí es una actualización en la Ingeniería En Sistemas Computacionales.

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 algoritmo Genético y herramientas para la resolución del problema TSP dinámico, incluyendo las librerías de TSPLIB. La librería TSPLIB consiste de un conjunto de instancias de las cuales se conoce la ruta óptima y se utilizan para calibrar el algoritmo Genético básico 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 autoría 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.

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 its 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.

Capítulo I

Introducción

El problema consiste en la necesidad de saber cuál es la ruta óptima para comprar los productos que se encuentran en venta online donde el comprador tenga la información en ese instante de su solicitud teniendo en cuenta que el comprador marcara un límite o radio donde la información se tomara y se procesara para dar el resultado de los puntos a visitar y que pueda tener un ahorro en las compras de los productos que están buscando en su recorrido, dado el caso que si el comprador se mueve en su trayecto se mostraran alternativas de los precios de los productos donde se incluirán o se eliminaran dependiendo de precio en este caso el precio más bajo.

La relevancia de esta investigación en el uso de la computadora es la posibilidad de conocer la ruta optima donde el comprador pueda recorrer los puntos en el menor distancia posible para ahorrar consumos de gasolina y costos de los productos que tendrá un ahorro en su consumo donde pasaría a comprar los productos donde se le mostrara la ruta ahí es donde la computadora realiza el procesamiento de la información que por medio de las diferentes rutas posibles que existan la computadora por medio de su operación nos mostrara la ruta optima con ayuda de un algoritmo que es eficiente su operación ya que si no tuviera un algoritmo el procesamiento de la computadora elevaría el tiempo de procesamiento y no arrojaría un resultado optimo y con el algoritmo se puede calibrar para nos dé resultados óptimos y el comprador obtenga una información donde él pueda elegir donde comprar.

La importancia de la aplicación utilizando los recursos computacionales así como el algoritmo de optimización de la ruta optima a visitar los puntos mostrados en su

recorrido así como los productos que se está buscando es mostrar en un dispositivo móvil la ruta en tiempo real y nos indique la posibilidad de tomar una decisión de compras para tener un ahorro en nuestra economía y tener un amplio panorama de los productos que están en nuestro alrededor y saber con la información donde poder comprar y ahorrar en el consumo de gasolina o si es caminando ir rápido a los puntos donde se elija visitar y poder comprar el producto.

Se tendría una base de datos de los vendedores y de los compradores donde la información de la predilección de los productos se tendría, con los puntos de las tiendas, productos se tomaría la información para trazar la ruta optima, con ayuda del algoritmo y el usuario tendría la información en su dispositivo móvil para tomar la decisión de comprarlo o no.

El problema del agente viajero por sus siglas en inglés (TSP) es uno de los problemas más intensamente estudiados en la optimización combinatoria [1]. Si la distancia entre dos puntos cualesquiera es la distancia euclidiana, el problema se denomina problema del vendedor ambulante euclidiano (ETSP) [1]. 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 [10]. Otras variaciones del TSP incluyen el Problema del vendedor ambulante con los vecindarios (TSPN) [19], el Problema del vendedor ambulante de cuello de botella (BTSP) [18], entre otros [15].

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 además de otros parámetros como fricción, potencia, masa variable, etc. [9]. 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 [12]. Estos casos mencionados anteriores se conocen como DTSP o TSP con Curvatura [13].

El DTSP ha atraído considerable atención debido a muchas aplicaciones civiles y militares [10]. Una configuración típica es monitorear una colección de objetivos distribuidos espacialmente por un vehículo aéreo no tripulado (UAV) [13]. Esto podría referirse al control del tráfico en lugares específicos, la recopilación de inteligencia y el reconocimiento de objetivos 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 objetivo y verificación de su destrucción), entre otros.

1.1 Aplicaciones de TSP

1.1.1 Transporte de mercancías y similares

La aplicación más obvia del TSP es justo el planteamiento del que nace. Imaginemos que tenemos que repartir mercancías en una serie de ciudades y no queremos tener que pasar dos veces por la misma. Tratando las ciudades como vértices de un grafo, las carreteras que las unen como aristas y tomando la distancia que las separa como costes del problema ya tenemos planteado nuestro TSP [1].

1.1.2 Taladrado de placas de circuitos

Una aplicación directa del TSP es el problema del taladrado de placas de circuito impresas -Printed Circuit Boards, PCBs -. Para conectar dos conductores o insertar

partes del circuito se taladran una serie de agujeros en la placa del circuito. Estos agujeros pueden tener distintos tamaños y para realizarlos la máquina tiene que desplazarse para cambiar la broca. Este cambio de broca, evidentemente, consume tiempo de producción y aunque la heurística nos dice que lo más sencillo es realizar todos los agujeros del mismo diámetro y después cambiar de broca, el problema puede ser tratado como un TSP y vemos que se obtienen mejores resultados.

Para formularlo correctamente debemos considerar como nodos -o ciudades- los puntos a taladrar y la ponderación de los arcos el tiempo que el taladro tardaría en cambiar de broca y alcanzar el siguiente objetivo, tomando como punto de partida la posición de reposo del taladro [1].

1.1.3 Orden de recogida en un almacén

Este problema se asocia con el tratamiento del material de un almacén. Asumiendo que llega al almacén un pedido de un cierto número de objetos almacenados y que un vehículo tiene que recoger todos estos elementos para llevárselos al cliente, podemos apreciar una relación directa con el TSP si consideramos como nodos los puntos del almacén donde se encuentran los objetos del pedido y la ponderación de los arcos como el tiempo que tardaría el vehículo en ir de un punto al otro [1].

1.1.4 Cristalografía de rayos-X

El análisis de estructuras cristalinas es una importante aplicación del TSP. En este caso un difractómetro de rayos-X se utiliza para medir la intensidad de la refracción de los rayos-X en el cristal desde varias posiciones. Estas mediciones se pueden realizar en un tiempo breve, pero en comparación lo que consume más tiempo es la colocación del instrumento de medición en las distintas posiciones necesarias.

Como el orden en el que se realicen las mediciones no afectan al resultado del experimento, tratándolo como un TSP podemos reducir significativamente el tiempo que tarda en realizarse [1].

1.1.5 Aplicaciones a e-commerce y otras aplicaciones industriales

Describiremos ahora algunos problemas que se presentan de forma natural en algunas empresas. El primero tiene que ver con la programación de tareas en una máquina. Muchas veces en algún taller de manufactura, se cuenta con una sola máquina en la cual se pueden procesar diferentes tareas, una a la vez. Ahora bien, para procesar cada una de estas tareas, la máquina requiere de cierta configuración característica de la tarea, pueden ser: número y tamaño de diferentes dados, colocación de cuchillas a cierta distancia unas de otras, colorantes para alguna fibra, etc. De manera que una vez que una tarea ha sido terminada, es necesario preparar la máquina para procesar una nueva tarea, aquí será necesario invertir un cierto tiempo, y este tiempo dependerá de la tarea recién procesada y de la próxima. Si las características de una tarea son similares a las de otra, es plausible pensar que el tiempo que se requiere para pasar de una configuración a otra será pequeño, en comparación del tiempo requerido para pasar de una tarea a otra con características muy diferentes [2].

Desgraciadamente durante las labores de preparación de la máquina, ninguna de las tareas se puede ejecutar, así que este tiempo es tiempo perdido, y se está desaprovechando la capacidad de la máquina, esto representa un costo de oportunidad para la empresa. Es importante entonces encontrar el orden en el cual se deben de procesar estas tareas con el fin de reducir al mínimo todo este tiempo perdido.

Aun cuando este problema parezca no tener ninguna relación con el TSP, se puede formular de la misma manera. Cada tarea puede ser vista como una de las ciudades a visitar, y el tiempo necesario para cambiar la configuración de la máquina

corresponde a la distancia que hay entre una ciudad y otra. Encontrar la manera de ordenar las tareas para minimizar el tiempo total de preparación es equivalente a diseñar la ruta, esto es, el orden en el cual se deben de visitar las ciudades para minimizar la distancia total recorrida. Esto nos da una idea de lo crucial que resulta tener buenas soluciones para el TSP en un ambiente de manufactura.

Una aplicación que utiliza TSP, se encuentra en el mercado, una aplicación de e-commerce que utilizan el Uber Eats, Didi Food, Rappi y otras, para la entrega de mercancía ya que tienen personas que utilizan moto para su repartición y tienen la necesidad de que se les muestre la ruta para entregar los productos que tienen así la moto usuario con la aplicación mira el camino a seguir y regresar al lugar de inicio.

Para la empresa es muy útil saber la ruta optima porque se consume menos gasolina y el tiempo de entrega es rápido.

Un segundo ejemplo lo podemos encontrar dentro de la logística de distribución de mercancía a los clientes. Generalmente, algunas empresas que distribuyen bienes perecederos necesitan hacerlo en un tiempo corto, un esquema muy común es que la empresa disponga de un almacén central, en el cual se concentran los bienes a distribuir, y una flotilla de unidades de transporte se encarga de visitar a los clientes para hacer entrega de la mercancía [3].

Analicemos los componentes de este problema, en primer lugar, tenemos que las unidades de servicio son limitadas, la forma en la que se podría efectuar la entrega de mercancías en el menor tiempo posible sería enviar una unidad a cada uno de los clientes. Pero, lo más realista sería pensar que no se tienen tantas unidades como clientes, ya que esto resultaría sumamente oneroso. Si la empresa dispone de una sola unidad el costo fijo se reduce bastante, y el problema de determinar la ruta que debe de seguir el vehículo para entregar en el menor tiempo toda la mercancía es ni más menos que el TSP. Pero aquí hay dos problemas en los que tenemos que pensar: en primer lugar, puede ser que el tiempo mínimo (si es que se puede determinar) resulte demasiado largo, p. ej. si se trata de entrega de leche, esta debe de estar entregada por la mañana, que es cuando los clientes la

requieren, y con una sola unidad de entrega, podría darse el caso que los últimos clientes la fueran recibiendo por la tarde. Por otro lado, las unidades tienen una cierta capacidad de almacenamiento, y puede ser que se necesiten varias para poder cargar con toda la mercancía que debe de ser entregada.

Así pues, vemos que este problema contiene dentro de sí muchos más. Primero: determinar cuál es el tamaño ideal de la flota de vehículos. Segundo: determinar cuáles son los clientes que deben de ser asignados a cada unidad para hacer la entrega. Y finalmente: cuál es la ruta que debe de seguir cada una con el fin de terminar con el reparto en el menor tiempo posible (TSP). Para complicar más las cosas estos problemas no son independientes, sino que la solución de uno determina la de otro. Este problema se conoce como el problema de ruteo de vehículos (VRP: Vehicle Routing Problem). Muchas aplicaciones más pueden encontrarse en [4]el libro de Lawler. La figura 1.1 ilustra un ejemplo de ruteo.



Figura 1.1 Ejemplo Problema de ruteo de vehículos

1.2 Definición del problema

Al encontrarse en un punto cualquiera en las coordenadas x , y que es el punto inicial donde se quiere encontrar los puntos de venta de los productos que uno está buscando, se quiere en este caso recorrer cada punto y regresar al punto de partida, para así adquirir los productos que se están buscando y como el punto está en

movimiento se requiere un radio para delimitar los productos que se están buscando esto nos da un cambio en las entradas y salidas de los productos, se encuentra un problema de optimización de rutas así como un movimiento dinámico donde nos permite cambiar la ruta dependiendo del movimiento que se realice, esto da pauta a investigar un algoritmo de optimización de las rutas posibles en movimiento para nos dé un resultado rápido.

Nos encontramos en el problema de investigación en las diferentes formas de trazar los caminos de inicio y pasar por los puntos y regresar al inicio ya que nos muestran diferentes combinaciones donde elegir y no sabemos cuál tomar ,entonces se presenta una manera de optimizar o reducir esas combinaciones y por medio de interacciones saber la mínima distancia a recorrer, al investigar las heurísticas que contienen algoritmos se tomó el algoritmo genético simple con elitismo para que nos indique cual es el camino optimo a seguir y reducir la distancia mínima.

La optimización es un proceso matemático para obtener la mejor solución de un conjunto de ciertas soluciones posibles [5]. A lo largo de los años, se han propuesto varios algoritmos inspirados como Redes Neuronales, Inteligencia de enjambre, Algoritmos evolutivos, en fenómenos naturales para resolver de manera eficiente varios problemas de optimización. Los algoritmos evolutivos se clasifican como algoritmos de búsqueda [5]heurísticos, donde los posibles elementos de la solución abarcan el espacio de búsqueda n-dimensional para encontrar la solución óptima global. Estos algoritmos funcionan eficientemente en varios problemas del mundo real. La mayoría de estos algoritmos implican la creación de nuevas soluciones y descartan aquellas que no producen resultados adecuados [7].

Para resolver el problema del vendedor ambulante dinámico (DTSP por sus siglas en inglés), la entropía, también son utilizadas para medir la diversidad de la población para el estudio de la diversidad de una población, que un único valor de función fitness no implica necesariamente un único fenotipo, del mismo modo que un fenotipo no implica necesariamente un único genotipo. Sin embargo, un genotipo sí implica un único fenotipo y un valor fitness [7]. Un ambiente experimental es estocástico y dinámico. La capacidad de cambio requiere que el algoritmo 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 el control de una convergencia estable del algoritmo al óptimo y proporciona un buen mecanismo para evitar el estancamiento.

Un problema donde los datos de entrada son cambiables dependiendo del tiempo se denomina un problema de optimización dinámica (DOP) [5]. El propósito de la optimización para los DOP continuamente realizar un seguimiento y adaptarse a los cambios a través del tiempo y encontrar rápidamente la mejor solución en la actualidad.

El problema de la falta de una ruta de acceso cerrada más breve y limitada a través de un conjunto de objetivos en el plano y en movimiento es conocido problema del agente viajero dinámico (DTSP por sus siglas en inglés). Las aplicaciones del DTSP incluyen la planificación de movimiento para robots móviles con restricciones cinemáticas y vehículos aéreos de ala fija no tripulados. La dificultad del DTSP es encontrar simultáneamente un orden de los objetivos y los rumbos adecuados (ángulos de orientación) del vehículo al pasar los objetivos. Dado que se sabe que el DTSP es NP-hard [9], existe la necesidad de algoritmos heurísticos que proporcionen soluciones de buena calidad en un tiempo razonable. Los algoritmos se basan en una técnica que optimiza los encabezados de los objetivos de un subtour abierto o cerrado con un orden dado [16]. Esto se hace discretizando los encabezados, construyendo una red auxiliar y calculando un camino más corto en la red.

1.3 Justificación

El problema del agente viajero, conocido como TSP (Traveling Salesman Problem), ha sido estudiado por muchos investigadores en las áreas de inteligencia artificial y optimización con enfoques de programación matemática y de programación heurística. Variantes de este problema también han sido discutidas en el estado del arte, dos de ellas son el problema de múltiples agentes viajeros conocido como

mTSP (multiple Traveling Salesmen Problem) y el problema dinámico del agente viajero conocido como DTSP (Dynamic Traveling Salesman Problem) [8]. En el caso del DTSP se han encontrado artículos que se presentan técnicas que se han utilizado para resolver DTSP, tales como algoritmos genéticos, heurísticas constructivas y de mejoramiento, recocido simulado, búsqueda tabú e incluso redes neuronales. Presentar un modelo de programación heurística que resuelva el problema DTSP, así como presentar el problema y como se podría resolver. El modelo de DTSP puede ser utilizado en entornos dinámicos similares a los escenarios del e-commerce, como por ejemplo, en un sistema de distribución donde múltiples vehículos que reparten un mismo producto, tienen que visitar un conjunto de destinos específicos donde el costo de tiempo entre viajar de un lugar a otro cambia en diferentes horas del día, encontrar el camino más corto de alguna compra por realizar desde el punto de geolocalización en donde nos encontremos, para obtener un ahorro en las decisiones de gastar el dinero de manera inteligente con ayuda de la inteligencia artificial que nos mostrará el camino óptimo para adquirir nuestro producto y tener un ahorro en nuestra economía.

1.4 Objetivos

1.4.1 Objetivo General

- Generar un modelo de programación Heurística Softcomputing para optimizar la solución del problema dinámico de agentes viajeros en la obtención de lugar más cercano para comprar el producto

1.4.2 Objetivos Específicos

- Plasmar los resultados en una tesis y publicación de al menos un artículo
- Generar un modelo de Programación heurística para el DTSP

- Evaluar el modelo generado
- Seleccionar e implementar algoritmos heurísticos que den solución a TSP, DTSP a la solución del lugar más cercano
- Evaluar e identificar cuál de los algoritmos heurísticos aplicados a TSP, DTSP

1.5 Hipótesis

Es posible generar un modelo que permita optimizar la solución de DTSP mediante algoritmos de programación heurística para obtener la ruta optima.

Capítulo II

Marco Teórico

2.1 Aspectos fundamentales de teoría de grafos

La teoría de grafos es un área relativamente nueva de las matemáticas, siendo considerado el trabajo de Leonhard Euler, sobre el problema de los puentes de Königsberg (1736), el primero en este tema [25]. A partir de ese momento, hemos presenciado un vertiginoso crecimiento gracias a importantes aportes que han hecho matemáticos, ingenieros y otros científicos, quienes han encontrado en esta área las herramientas necesarias para modelar y resolver problemas de muy distinta índole. Es, sin duda, la simpleza de los conceptos estudiados lo que ha motivado esta importante evolución. Sin embargo, esta simpleza contrasta fuertemente con las dificultades a las cuales nos enfrentamos al estudiar las distintas conjeturas planteadas desde el primer trabajo de Euler, conjeturas que abarcan desde determinar si es posible colorear un mapa con solo cuatro colores, probada solo a mediados del siglo XX con la ayuda de los computadores, hasta la conjetura $\mathcal{P} \neq \mathcal{NP}$ [25] [6].

El estudio de la teoría de grafos ha ido de la mano con el desarrollo de algoritmos, para resolver los variados tipos de problemas que aparecen naturalmente en esta temática. Esto explica porque la teoría de grafos ha sido también de vital importancia para las ciencias de la computación que, gracias al explosivo desarrollo que han tenido los computadores desde mediados del siglo XX, se ha consolidado como una ciencia fuerte e independiente, incentivando y potenciando, a su vez, la investigación en teoría de grafos.

La teoría de grafos comenzó en 1736 cuando Leonhard Euler (1707-1783) resolvió el conocido problema del puente de Königsberg. Este problema pedía un recorrido circular a través de la ciudad de Königsberg (ahora Kaliningrado) de tal manera que

cruzar cada uno de los siete puentes que cruzan el río Pregel una vez, y sólo una vez. una vez; véase la figura 2.1 para un esquema de la situación.

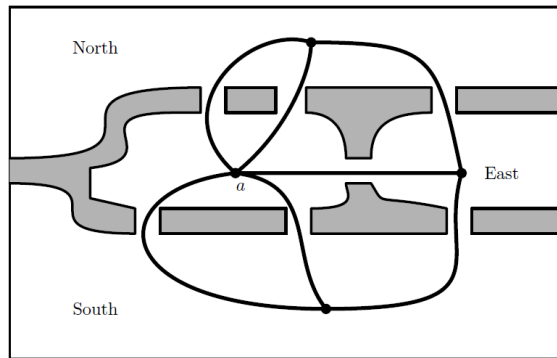


Figura 2.1: Königsberg problema del puente (adaptado de [7]).

Al tratar de resolver este problema uno pronto tiene la sensación de que no hay solución. Pero ¿cómo se puede demostrar esto? Euler se dio cuenta de que las formas precisas de la isla y de los otros tres territorios implicados no son importantes; La solvencia depende únicamente de sus propiedades de conexión. Representemos los cuatro territorios por puntos (llamados vértices), y los puentes por curvas que unen los puntos respectivos. Intentando organizar un paseo circular, comenzamos ahora un recorrido, digamos, en el vértice llamado a. Cuando volvemos a por primera vez, hemos utilizado dos de los cinco puentes conectados con a. En nuestro siguiente regreso a hemos utilizado cuatro puentes. Ahora podemos volver a salir de a utilizando el quinto puente, pero no hay posibilidad de volver a sin utilizar uno de los cinco puentes por segunda vez. Esto demuestra que el problema es efectivamente irresoluble. Utilizando un argumento similar, vemos que también es imposible encontrar cualquier paseo - no necesariamente circular, por lo que el recorrido podría que termine en un vértice diferente al que comenzó- que utilice cada puente exactamente una vez. Euler

demonstró aún más: dio una condición necesaria y suficiente para que un gráfico arbitrario admita un recorrido circular.

2.1.1 Definiciones de teoría de grafos

Un grafo es una dupla $G = (V, E)$ compuesta por un conjunto finito $V = V(G)$ de vértices, también llamados nodos y típicamente dibujados por círculos y un conjunto $E = E(G) \subseteq V_*^2$, que llamaremos conjunto de aristas ¹, donde hemos definido $V_*^2 = (V \times V) \setminus (\cup_{v \in V} (v, v))$, (v, v) , también conocidas como bucles [8].

2.1.1.1 Grafo

Los vértices de un gráfico $G = (V, E)$ se representan con puntos (en negrita) y las aristas con líneas (preferentemente rectas) que unen los puntos extremos. Damos algunos ejemplos en la figura 1.2. Insistimos en que en estas imágenes las líneas sólo sirven para indicar los vértices con los que inciden. En particular, los puntos interiores de estas líneas, así como los posibles puntos de intersección de dos aristas (como en la figura 1.2 para los gráficos K_5 y $K_{3,3}$) no son significativos. [9]

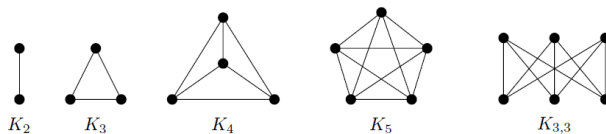


Figura 2.2: Algunos grafos (adaptado de [10]).

2.1.1.2 Grafos conexos y no conexos

Grafos conexos Un grafo es conexo si cada par de vértices está conectado por un camino; es decir, si para cualquier par de vértices (a, b) , existe al menos un camino posible desde a hacia b . Un grafo es doblemente conexo si cada par de vértices está conectado por al menos dos caminos disjuntos; es decir, es conexo y no existe un vértice tal que al sacarlo el grafo resultante sea desconexo [2].

Es posible determinar si un grafo es conexo usando un algoritmo Búsqueda en anchura (BFS) o Búsqueda en profundidad (DFS) [25]. En términos matemáticos la propiedad de un grafo de ser (fuertemente) conexo permite establecer con base en él una relación de equivalencia para sus vértices, la cual lleva a una partición de éstos en "componentes (fuertemente) conexas", es decir, porciones del grafo, que son (fuertemente) conexas cuando se consideran como grafos aislados. Esta propiedad es importante para muchas demostraciones en teoría de grafos.

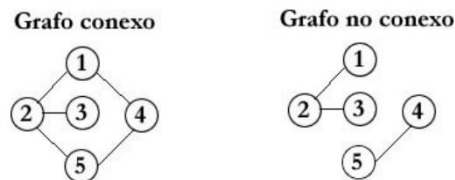


Figura 2.3: Grafos conexo y no conexo (adaptado de [10])

Los vértices constituyen uno de los dos elementos que forman un grafo. Como ocurre con el resto de las ramas de las matemáticas, a la Teoría de Grafos no le interesa saber qué son los vértices. Diferentes situaciones en las que pueden identificarse objetos y relaciones que satisfagan la definición de grafo pueden verse como grafos y así aplicar la Teoría de Grafos en ellos.

2.1.1.3 Grafos dirigidos

Un Grafo Dirigido G consiste en dos conjuntos:

- $V(G)$: un conjunto finito no vacío cuyos elementos son llamados Vértices de G . $E(G) \subseteq V(G) \times V(G)$, donde el operador \times denota al Producto Cartesiano de Conjuntos.
- Nótese que por la definición se tiene que la arista (u, v) no es equivalente a la arista (v, u) , i.e. $(u, v) \neq (v, u)$. Es decir, el orden en que son listados los vértices indica la dirección de la arista [3].

Los grafos dirigidos son representados por diagramas en el plano de manera natural. Específicamente, cada vértice v en $V(G)$ es representado por un círculo y cada arista $(v_1, v_2) \in E(G)$ es representada por una curva dirigida la cual sale del vértice v_1 y llega al vértice v_2 , tal que v_1, v_2 están en $V(G)$. Por ejemplo, en la Figura 5.1 se presenta un grafo dirigido G para el cual:

- $V(G)$ consiste en los vértices A, B, C, D , es decir, $V(G) = A, B, C, D$.
- $E(G)$ consiste en las aristas $(A, B), (B, C), (C, D), (A, C)$ y (B, D) .
Formalmente, $E(G) = (A, B), (B, C), (C, D), (A, C), (B, D)$

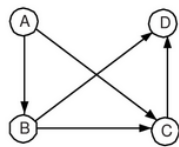


Figura 2.4: Grafo dirigido (adaptado de [10]).

2.1.1.4 Grafos ponderados

En un grafo ponderado a las aristas se les asigna un valor al que se le llama ponderación y que podría representar la distancia que hay de un nodo a otro, o bien el costo de transportarse de una ciudad a otra [2].

Determinar la ruta más corta es un problema típico de la teoría de grafos, y consiste en encontrar el camino más corto para ir de una ciudad origen w a una ciudad destino x . Pueden existir distintas rutas para ir de un nodo a otro, pero el objetivo es encontrar la más corta o bien la más económica, si es que la ponderación representa un costo.

El método más utilizado para encontrar la ruta más corta de un nodo cualquiera w a cualquier nodo de la red, es por medio del algoritmo de Dijkstra [4] el cual consta de los siguientes pasos:

Algoritmo 2.1: Dijkstra

```
1 procedure DIJKSTRA_SINGLE_SOURCE_SP ( $V, E, w, s$ )
2 begin
3    $V_T := s$  :
4   for all  $v \in (V - V_T)$  do
5     if  $(s, v)$  exists set  $l[v] := w(s, v)$ ;
6     else set  $l[v] := \infty$ ;
7   while  $V_T \neq V$  do
8     begin
9       find a vertex  $u$  such that  $l[u] := \min\{l[v] \mid v \in (V - V_T)\}$  :
10       $V_T := V_T \cup u$ ;
11      for all  $v \in (V - V_T)$  do
12         $l[v] := \min\{l[v], l[u] + w(u, v)\}$ ;
13      endwhile
14 end DIJKSTRA_SINGLE_SOURCE_SP
```

2.1.1.5 Grafo completo

Es el grafo en donde cada vértice está relacionado con todos los demás, sin lazos ni lados paralelos. Se indica como K_n , en donde n es el número de vértices del grafo.

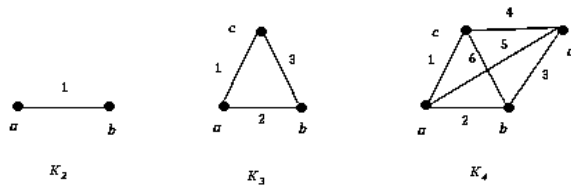


Figura 2.5: Se muestran tres ejemplos de grafos completos (adaptado de [10]).

La valencia en cada uno de los vértices de los grafos completos es $(n - 1)$, y el número de lados está dado por la expresión

$$\text{Núm. de lados} = \frac{n(n - 1)}{2} \quad (1)$$

en donde n es el número de vértices del grafo.

2.1.2 Vértices o nodos adyacentes y matriz de adyacencia

Vértice: Los vértices constituyen uno de los dos elementos que forman un grafo. Como ocurre con el resto de las ramas de las matemáticas, a la Teoría de Grafos no le interesa saber qué son los vértices. Diferentes situaciones en las que pueden

identificarse objetos y relaciones que satisfagan la definición de grafo pueden verse como grafos y así aplicar la Teoría de Grafos en ellos.

1. Un grafo consiste en un conjunto finito de puntos llamados vértices y un conjunto finito de aristas, cada una de las cuales conecta dos vértices.
2. Se dice que dos vértices son adyacentes, si están conectados por una arista.

Matriz de adyacencia

Es una matriz cuadrada en la cual los vértices del grafo se indican como filas y como columnas: el orden de los Vértices es el mismo que guardan las filas y las columnas de la matriz. Se coloca un 1 como elemento de la matriz cuando existe una relación entre uno y otro Vértice, o bien un 0 cuando no exista relación alguna.

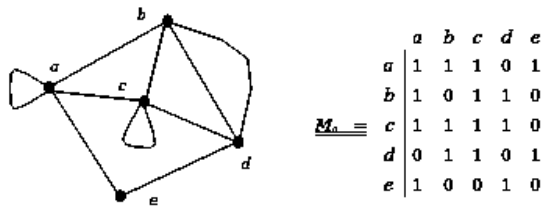


Figura 2.6: Matriz de adyacencia correspondiente (adaptado de [4]).

Algo que se puede observar en la matriz de adyacencia es que no se pueden representar en ella los lados paralelos, como ocurre con el par de aristas que unen los nodos b y d. En esta matriz también la mayoría de las aristas están repetidas, como ocurre con la arista que une a los vértices b y c que tiene un 1 en la línea b

columna c , pero que también tiene un 1 en la fila c columna b . Por último, los lazos, a diferencia de las aristas normales solamente se representan una sola vez. Se puede concluir que la matriz de adyacencia es buena para llevar a cabo operaciones con relaciones, pero que no permite registrar en ella toda la información del grafo [11].

2.1.3 Caminos y ciclos en un grafo

Un camino es una sucesión de vértices tal que de cada uno de sus vértices existe una arista hacia el vértice sucesor. Un camino simple es aquel que no repite vértices en su recorrido.

Dos caminos son ajenos o independientes si no tienen ningún vértice en común excepto el primero y el último.

La longitud de un camino es el número de aristas que usa dicho camino, contando aristas recorridas varias veces el mismo número de veces que las recorramos. En el ejemplo, (1, 2, 5, 1, 2, 3) es un camino con longitud 5, y (5, 2, 1) es un camino simple de longitud 2.

Camino euleriano

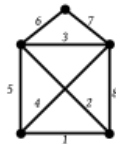


Figura 2.7: Un camino euleriano recorre todas las aristas exactamente una vez (puede repetir vértices) (adaptado de [4]).

Camino hamiltoniano

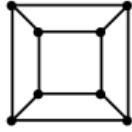


Figura 2.8: Un camino hamiltoniano no recorre todos los vértices exactamente una vez (adaptado de [4]).

Ciclo simple

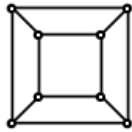


Figura 2.9: Orden en que se recorre un grafo en una búsqueda en profundidad (adaptado de [4]).

Ciclo euleriano

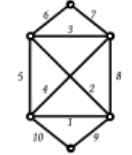


Figura 2.10: Un ciclo euleriano pasa por todas las aristas exactamente una vez, regresando al punto de partida (adaptado de [4]).

Ciclo hamiltoniano

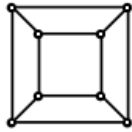


Figura 2.11: Un ciclo hamiltoniano pasa por todos los vértices exactamente una vez, regresando al punto de partida (adaptado de [4]).

2.1.4 Ciclo Hamiltoniano y el problema del agente viajero.

Deseamos encontrar un ciclo, pero esta vez le pediremos al ciclo que visite todos los vértices del grafo, pasando solo una vez por cada uno. A este tipo de ciclo lo llamaremos *ciclos Hamiltonianos*.

El nombre de este tipo de ciclo viene de un famoso “juego” inventado por William Hamilton en 1857 y comercializado en Europa bajo el nombre “The Icosian Game”. El problema era encontrar en el grafo de la Figura 2.12, un ciclo Hamiltoniano, es decir, un ciclo que visite cada uno de los vértices exactamente una vez.

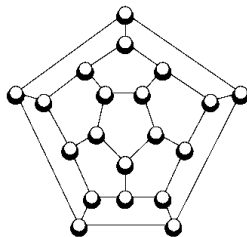


Figura 2.12: El grafo del Icosian Game (adaptado de [4]).

Notemos la similaridad entre la definición de ciclos Eulerianos (ciclo que visita todas las aristas exactamente una vez) y un ciclo Hamiltoniano (ciclo que visita todos los vértices exactamente una vez). De la misma forma que antes, si para un grafo existe un ciclo Hamiltoniano, diremos que es un *grafo Hamiltoniano*. Al contrario de los grafos Eulerianos, el poder caracterizar si un grafo es hamiltoniano o no es un problema “difícil”. Pero ¿qué significa que un problema sea difícil? Efectivamente, el intentar definir esta noción de “dificultad” ha originado una nueva área de la ciencia llamada “complejidad computacional”, cuyos inicios se remontan a los años 1950 y que ha dado origen a algunos de los problemas abiertos más famosos de la matemática. Retomaremos este tema en la siguiente sección. El ejemplo más simple de un grafo Hamiltoniano es cuando el grafo es un ciclo, pues el mismo es un ciclo que recorre todos los vértices. Por otro lado, si un grafo es Hamiltoniano y le agrego una arista, el grafo resultante sigue siendo Hamiltoniano. En particular, este argumento permite probar que los grafos completos son Hamiltonianos. [8]

Travelling Salesman Problem, TSP.

Considere el gráfico completo K_n junto con una función de peso $w: E \rightarrow \mathbb{R}^+$, Buscando una permutación cíclica $(1, \pi(1), \dots, \pi^{n-1}(1))$ del conjunto de verices $\{1, \dots, n\}$ tal como

$$w(\pi) := \sum_{i=1}^n w(\{i, \pi(i)\}) \quad (2)$$

es mínima. Llamamos a cualquier permutación cíclica π of $\{1, \dots, n\}$ así como el ciclo hamiltoniano correspondiente

$$1 - \pi(1) - \dots - \pi^{n-1}(1) - 1$$

En K_n un tour. Un recorrido óptimo es un recorrido π tal que $w(\pi)$ es mínimo entre todos los recorridos.

2.2 Optimización combinatoria

Consiste en encontrar un objeto entre un conjunto (o al menos contable) de posibilidades. Este objeto suele ser un número o conjunto de naturales, una permutación o una estructura de grafo (o subgrafo) [22].

Los problemas combinatorios presentan como particularidad que siempre existe un algoritmo exacto que permite obtener la solución óptima. Este método consiste en la exploración de forma exhaustiva del conjunto de soluciones (enumeración). Este algoritmo suele ser extremadamente ineficiente, ya que para la mayoría de los problemas de interés que se pueden encontrar en el ámbito de la optimización combinatoria el tiempo que emplearía en encontrar una solución crece de forma exponencial con el tamaño del problema. En general, este tipo de problemas se caracterizan por tener espacios de soluciones con una cardinalidad muy elevada.

2.3 Programación heurística

La Programación Heurística implica una forma de modelizar el problema en lo que respecta a la representación de su estructura, estrategias de búsqueda y métodos de resolución, que configuran el Paradigma Heurístico [20].

La Programación Heurística implica una forma de modelizar el problema en lo que respecta a la representación de su estructura, estrategias de búsqueda y métodos de resolución, que configuran el Paradigma Heurístico.

La programación heurística aborda la idea de la inteligencia artificial resolviendo problemas utilizando reglas o protocolos basados en la experiencia.

En general, la palabra "heurística" en informática se refiere a una filosofía que es diferente de los procesos informáticos cuantificados y basados en la lógica que impulsaron el avance de las computadoras primitivas en las últimas décadas.

Contrariamente al principio de usar computación estricta basada en algoritmos, la heurística es, en muchos sentidos, un atajo a un tipo de programación de lógica cuantificada. La programación heurística busca alcanzar una meta mediante la sustitución de ciertos tipos de programas de aprendizaje automático por algoritmos lógicos.

Otra forma de decir esto es que, si bien los algoritmos operan en sistemas conocidos y principios lógicos, la programación heurística opera en una serie de "conjeturas inteligentes" u operaciones informadas que no se basan completamente en números o datos duros.

Un ejemplo de un proceso de programación heurística es un programa que analizará el contenido de una unidad o sistema de archivos. El programa lógico buscaría de forma preprogramada, por ejemplo, alfabéticamente o en términos de modificación reciente de datos, donde el sistema de programación heurística podría programarse para que se realice de acuerdo con búsquedas anteriores que originó un usuario.

Aquí, la máquina está aprendiendo del usuario. Otro buen ejemplo de programación heurística es el uso de herramientas de procesamiento de lenguaje natural. Además de algoritmos sofisticados, muchos de estos programas están utilizando principios de aprendizaje automático o de programación heurística, donde el programa analiza los aportes anteriores del usuario y los incorpora a los procesos centrales que proporcionan resultados.

2.3.1 Heurística

Para la mayoría de problemas de interés no se conoce un algoritmo exacto con complejidad polinomial que encuentre la solución óptima a dicho problema, y la cardinalidad del espacio de búsqueda suele ser enorme, lo cual hace generalmente

inviabile el uso de algoritmos exactos, fundamentalmente porque la cantidad de tiempo que se necesitaría para encontrar una solución es completamente inaceptable, se deben de utilizar métodos aproximados o heurísticas que permitan obtener una solución de calidad en un tiempo razonable a estos problemas.

El término de heurística del vocablo griego heuriskein que podría traducirse como encontrar, descubrir o hallar [5].

Desde el punto de vista científico, el termino heurística se debe al matemático G. Polya quien empleó por primera vez en su libro How to solve it, Con este término Polya quería expresar las reglas con las que los humanos gestionan el conocimiento común [19].

Definición por Zanakis "Procedimientos simples, a menudo basados en el sentido común, que se supone que obtendrán una buena solución (no necesariamente óptima) a problemas difíciles de un modo sencillo y rápido" [1].

Los métodos heurísticos tienen su principal limitación en su incapacidad para escapar de óptimos locales. Esto se debe, fundamentalmente, a que estos algoritmos no utilizan ningún mecanismo que les permita perseguir la búsqueda del óptimo en el caso de quedar atrapados en un óptimo local. Para solventar este problema, se introducen otros algoritmos de búsqueda más inteligentes (denominados metaheurísticas que evitan, en la medida de lo posible, este problema).

Heurísticas constructivas

Los procedimientos constructivos, son métodos iterativos mediante los cuales se crea una solución de manera gradual. Esto es, se parte de una solución inicial y en cada paso se añade un elemento hasta completar una solución. Se basan en seleccionar en cada iteración, el elemento con mejor evaluación.

Los métodos constructivos más usados son “El Algoritmo de Ahorros de Clarke & Wright”, las variantes de éste, y “Las heurísticas de inserción secuencial”. En la Figura 2.13 se muestra una clasificación de las heurísticas constructivas que se acababan de mencionar.

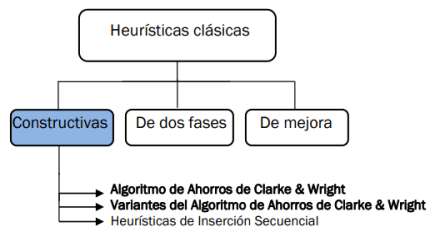


Figura 2.13: Clasificaciones heurísticas constructivas (adaptado de [12]).

Heurísticas de dos fases

En los procedimientos de dos fases, se descompone el problema en dos subproblemas, agrupación de vértices en rutas factibles y construcción de la ruta, que se resuelven de manera secuencial.

Las heurísticas de dos fases se dividen en dos clases: métodos de asignar primero-rutear después (First cluster then routing) y métodos de rutear primero asignar después (First routing then cluster).

Dentro del primer tipo, los métodos más usados son “El algoritmo de Barrido”, “El algoritmo de Pétalos”, “El algoritmo de Fisher y Jaikuman” y “El algoritmo de Bramel y Simchi- Levi”. En la Fig. 2.14 se puede observar un diagrama con las clasificaciones heurísticas de dos fases.

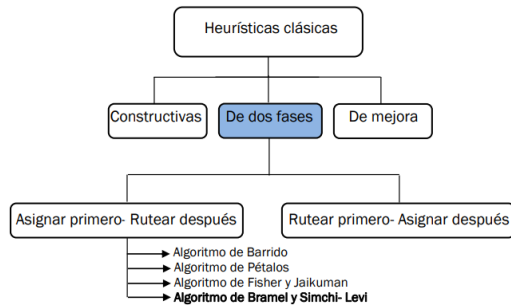


Figura 2.14: Clasificaciones heurísticas de dos fases (adaptado de [13]).

Heurísticas de mejora

Los métodos de mejora intentan mejorar la calidad de cualquier solución factible realizando una serie de intercambios de vértices, ya sea dentro de una misma ruta o entre distintas rutas de vehículos. Se trata de métodos de búsqueda local que parten de una solución ya completa y, usando el concepto de vecindario, recorren parte de espacio de búsqueda hasta encontrar un óptimo local. El vecindario de una solución s , denotado como $N(s)$, es el conjunto de soluciones se pueden construir a partir de s aplicando algún movimiento específico. Un óptimo local es una solución mejor o igual que cualquier otra solución de su vecindario. De esta forma, estos métodos, partiendo de una solución inicial, examinan su vecindario y se quedan con el mejor vecino, continuando el proceso hasta que encuentran un óptimo local. El principal inconveniente de este tipo de heurísticas es precisamente que la solución final va a depender del punto de partida. Pueden ser exactos o heurísticos. Dentro de estos últimos, existen del tipo intra- ruta, que mueven arcos dentro de una misma ruta, entre los que se encuentran las heurísticas 2- opt y 3- opt, y extra-ruta, que los intercambian entre dos o más rutas distintas, como la heurística 2- swap. En la Fig. 2.15 se muestra 2-opt es un algoritmo iterativo, en cada paso, eliminamos dos aristas de la solución actual y volvemos a conectar las dos vueltas así formadas, El

movimiento 3-OPT nos obliga a eliminar tres aristas y sustituirlas por tres nuevas para acortar el recorrido lo máximo posible, 2-swap Seleccionar e intercambiar cuatro procesos de diferentes máquinas.

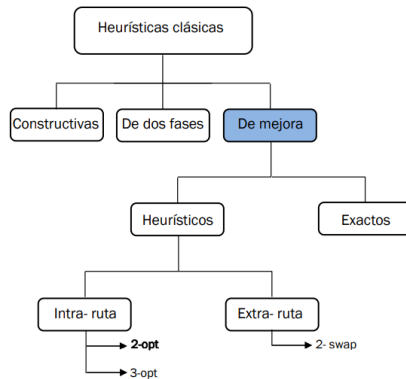


Figura 2.15: Clasificaciones heurísticas de mejora (adaptado de [13]).

2.3.2 Búsqueda local

Búsqueda local es la base de muchos de los métodos usados en problemas de optimización. Se puede ver como un proceso iterativo que empieza en una solución y la mejora realizando modificaciones locales. Básicamente empieza con una solución inicial y busca en su vecindad por una mejor solución. Si la encuentra, reemplaza su solución actual por la nueva y continúa con el proceso, hasta que no se pueda mejorar la solución actual. Claramente el diseño de la vecindad es crucial para el desempeño de éste, y de muchos otros, algoritmos. La vecindad son todas las posibilidades de soluciones que se consideran en cada punto.

El cómo se busca la vecindad y cuál vecino se usa en el reemplazo a veces se conoce como la regla de pivoteo (pivoting rule), que en general puede ser:

- Seleccionar el mejor vecino de todos (best-improvement rule).

- Seleccionar el primer vecino que mejora la solución (first-improvement rule).

Búsqueda local tiene la ventaja de encontrar soluciones muy rápidamente. Su principal desventaja es que queda atrapada fácilmente en mínimos locales y su solución final depende fuertemente de la solución inicial. Una búsqueda local es un método determinístico y sin memoria. Dada una misma entrada, regresa la misma salida. De hecho, un conjunto de entradas nos genera la misma salida (mínimo local). Esto se puede ver como un pozo de atracción. Esto sigue siendo muy superior a búsqueda aleatoria y el ingrediente principal es la definición de vecindad que permite moverse de una cierta solución a una mejor solución.

2.3.3 Búsqueda local *k-opt*

Los métodos de la familia *k-Opt*, conforman los algoritmos de búsqueda local de implementación más sencilla. La técnica de mejora de la que hacen uso recibe el nombre de *k-change*, Dada una instancia $J = (G, C)$ donde $G = (V, E)$ es un grafo de n vértices, y sea R una ruta sobre G . Una modificación de R que da como resultado otra ruta R' sobre G se dice que es de tipo *k-change* si a partir de R dado por

$$R = (v_1^1, v_2^1, \dots, v_{n_1}^1, v_1^2, v_2^2, \dots, v_{n_2}^2, \dots, v_1^k, v_2^k, \dots, v_{n_k}^k, v_1^1) \quad (3)$$

Se obtiene la ruta

$$R' = (v_1^1, v_2^1, \dots, v_{n_1}^1, v_{n_2}^2, v_{n_2-1}^2, \dots, v_1^2, v_{n_3}^3, \dots, v_1^{k-1}, v_{n_k}^k, v_{n_k-1}^k, \dots, v_1^k, v_1^1) \quad (4)$$

Consiste en suprimir k aristas de la ruta actual reemplazándolas por otras k que dan como resultado una nueva ruta. El valor de k es fijo para cada iteración del algoritmo

y dependiente de la heurística de la familia k-Opt utilizada. Para el 2-Opt, $k = 2$; para el 3-Opt $k = 3$; y así sucesivamente [22].

Los algoritmos k-Opt son ampliamente utilizados por obtener en general buenas soluciones sin requerir para ello un elevado coste computacional. Sin embargo, en el Capítulo 2 se ha probado la existencia de instancias del problema para las que la ejecución del 2-Opt puede llegar a requerir un coste computacional de orden exponencial.

A pesar de hacer uso de una mecánica similar, el comportamiento de cada uno de estos algoritmos difiere del de los demás y por tanto se predispone que alcanzarán soluciones de distintas características y con diferente coste computacional. Esto dependerá del valor de k elegido y, por supuesto, del número de vértices de la instancia y los valores de la matriz de costes asociada.

2.3.4 Metaheurísticas

El término metaheurística fue acuñado por F. Glover en el año 1986 [18]. Etimológicamente, deriva de la composición de dos palabras con origen griego, que son "meta" y "heurística", mientras que el prefijo meta (en inglés) se podría traducir como más allá de, en un nivel superior.

Glover pretendía definir un procedimiento nuestro de alto nivel que guía y modifica otras heurísticas para explorar soluciones más allá de la simple optimalidad local.

"Las metaheurísticas son una clase de métodos aproximados que están diseñados para resolver problemas difíciles de optimización combinatoria en los que los heurísticos clásicos no son efectivos. Las metaheurísticas proporcionan un marco general para crear nuevos algoritmos híbridos combinando diferentes conceptos derivados de la inteligencia artificial, la evolución biológica y los mecanismos estadísticos".

2.3.4.1 Algoritmos metaheurísticos poblacionales

A lo largo de la historia diversas teorías han intentado explicar el funcionamiento de la evolución de los seres vivos; pero fue a mediados del siglo XIX cuando se postularon los principios básicos, que en gran medida han condicionado la visión actual de la Evolución, Estos principios se asientan sobre la Teoría de la Selección Natural de Darwin [14] y sobre Herencia Genética de Mendel G. [15]. hace el siguiente resumen de dichos principios: "

- i) La evolución es un proceso que no opera directamente sobre organismos, sino sobre cromosomas. Estos son unos instrumentos orgánicos mediante los cuales se codifica la estructura de un ser vivo, un ser vivo se crea mediante la decodificación de un conjunto de cromosomas. Estos cromosomas (más concretamente, la información que contienen) pasan de una generación a otra mediante los procesos reproductivos.
- ii) El proceso evolutivo propiamente dicho tiene lugar durante la etapa de reproducción. En la naturaleza existe una gran cantidad de mecanismos reproductivos. Los más básicos son los de mutación (que modelan la variabilidad del material genético de los individuos) y los de recombinación (que combinan los materiales genéticos de los padres para producir el de los descendientes).
- iii) La selección natural es el mecanismo que permite relacionar los cromosomas con la eficiencia de las entidades que representan, provocando que aquellos organismos eficaces y adaptados al medio se reproduzcan con mayor probabilidad que aquellos que no lo están, ocasionando de esta manera la extinción de los organismos inadaptados o poco eficientes." La sencillez de estos principios ha favorecido que muchos investigadores hayan intentado trasladarlos al campo de la algoritmia. Así los Algoritmos Evolutivos se inspiran en ellos, modificándolos para obtener sistemas eficientes para la resolución de diferentes problemas.

Existe un enfoque diferente que intenta reproducir fielmente los principios naturales, simula la naturaleza; esto ha originado el campo denominado Vida Artificial [16] con el que los Algoritmos Evolutivos han tenido una gran interacción, gracias a la cual, se han producido avances notables en ambos campos. Un Algoritmo Evolutivo es un proceso estocástico e iterativo que opera sobre un conjunto P de individuos que constituyen lo que se denomina población; cada individuo contiene uno o más cromosomas que le permiten representar una posible solución al problema, la cual se obtiene gracias a un proceso de codificación/decodificación. La población inicial es generada aleatoriamente o con la ayuda de algún heurístico de construcción. Cada individuo es evaluado a través de una función de adecuación fitness. Estas evaluaciones se usan para predisponer la selección de cromosomas de forma que los superiores (aquellos con mayor evaluación) se reproduzcan más a menudo que los inferiores [17].

El algoritmo se estructura en tres fases principales que se repiten de forma iterativa, lo que constituye el ciclo reproductivo básico o generación. Dichas fases son: selección, reproducción y reemplazo. A finales de los 60 y principios de los 70 comenzaron a existir los algoritmos evolutivos tal y como hoy los conocemos. El biólogo Alexander S. Fraser [18] [19] [20] publicó una serie de trabajos sobre la evolución de sistemas biológicos en una computadora digital, los principios de la evolución al campo de la algoritmia originando diferentes modelos que pueden agruparse en tres grandes familias: las estrategias evolutivas, la programación evolutiva y los algoritmos genéticos. Esta familia de algoritmos tiene su origen en el trabajo de [21], y ponen un especial énfasis en la adaptación de los individuos más que en la evolución del material genético de éstos. Ello implica una visión mucho más abstracta del proceso, en la cual se modifica directamente el comportamiento de los individuos en lugar de trabajar sobre sus genes. Dicho comportamiento se modela mediante estructuras de datos relativamente complejas como pueden ser autómatas finitos. Tradicionalmente, estas técnicas emplean mecanismos de reproducción asexual y técnicas de selección mediante competición directa entre individuos. Las diferencias fundamentales con los algoritmos genéticos son dos: no se impone restricción sobre la representación y la operación de mutación

simplemente modifica aspectos de la solución según una distribución estadística que pondera como muy probables variaciones poco importantes en la descendencia y como cada vez menos probables, variaciones considerables a media que se vaya aproximando al óptimo [9] [22].

2.3.4.2 Algoritmos metaheurísticos trayectoriales

La representación de las soluciones se realiza mediante una cadena de enteros donde cada entero denota un vértice o punto de visita determinado. La posición que ocupa cada vértice indica el orden en el que será visitado. Se deben desarrollar operadores de cruce y mutación especiales para producir nuevas soluciones. Se han propuesto algunos operadores en la literatura, por ejemplo, en el trabajo [20] [21] trabajan con el operador de cruce denominado order crossover (OX) [19]: se seleccionan dos puntos de corte aleatoriamente y la cadena situada entre esos dos puntos se asigna a la nueva solución o solución hija. El resto de las posiciones se rellenan de una en una, comenzando después del segundo punto de corte, considerando los vértices en el orden que se encuentran en la segunda solución (padre), evitando duplicaciones. Se puede crear una segunda nueva solución invirtiendo el papel de los padres. Con el OX, las nuevas soluciones tienden a heredar el orden relativo de puntos de visita de las dos soluciones originales (padres).

2.3.4.3 Algoritmo de búsqueda local iterada

Un método básico de mejora de soluciones, que además es de gran importancia en muchas de las técnicas heurísticas más recientes, es la búsqueda local. Los procedimientos de búsqueda local, también llamados de mejora, se basan en explorar el entorno o vecindario de una solución con el fin de encontrar otra mejor.

Utilizan una operación básica llamada movimiento que, aplicada sobre los diferentes elementos de una solución, proporciona las soluciones de su entorno. Un procedimiento de búsqueda local parte de una solución inicial s_0 , explora su entorno, y escoge en él una nueva solución s_1 que mejore la actual.

Vamos a considerar un problema de optimización combinatoria en el que se trata de minimizar el valor de la función objetivo. Considerando un determinado número de iteraciones, llamado *LSITER* y un parámetro de vecindad de búsqueda δ , el procedimiento para encontrar el valor óptimo local se lleva a cabo de la siguiente manera: el punto x^p es asignado a una variable temporal y para almacenar la información inicial. A continuación, para una coordenada dada d de una partícula, un número aleatorio (λ_1) es seleccionado y combinado con δ obteniendo la longitud de paso de búsqueda. El punto y es entonces desplazado en la dirección que la longitud de paso indique, el signo que esta dirección tendrá, también se calcula de manera aleatoria (λ_2). Si el valor obtenido tras evaluar y en la función a optimizar es mejor después de haber realizado *LSITER* iteraciones, el punto x^p es reemplazado por y , terminando así la búsqueda en la vecindad p , de otra forma x^p conserva su valor, el pseudocódigo es el siguiente [22].

Algoritmo 2.2 Método de búsqueda local

```

1: contador  $\leftarrow$  1
2: longitud  $\leftarrow$   $\delta(\max\{u_d - l_d\})$ 
3: for  $p = 1$  to  $m$  do
4:   for  $d = 1$  to  $n$  do
5:      $\lambda_1 \leftarrow U(0,1)$ 
6:     while contador < LSITER do
7:        $y \leftarrow x^p$ 
8:        $\lambda_2 \leftarrow U(0,1)$ 
9:       if  $\lambda_1 < 0.5$  then
10:         $y_d \leftarrow y_d + \lambda_2 (\text{longitud})$ 
11:       Else
12:         $y_d \leftarrow y_d - \lambda_2 (\text{longitud})$ 
13:       end if
14:       if  $f(y) < f(x^p)$  then
15:         $x^p \leftarrow y$ 
16:        contador  $\leftarrow$  LSITER -1
17:       end if
18:     contador  $\leftarrow$  contador +1

```

```

19:     end while
20:   end for
21: end for
22:  $\mathbf{x}^{mejor} \leftarrow \arg \min \{f(\mathbf{x}^p), \forall p\}$ 

```

Un procedimiento de búsqueda local queda completamente determinado al especificar un entorno y el criterio de selección de una solución dentro del entorno.

Las formas más usuales de seleccionar $s \in N(s_0)$ son: explorar en todo $N(s_0)$ y tomar el correspondiente al menor valor de f (mayor descenso), o seleccionar el primero que mejora la solución actual (primer descenso). La búsqueda se detiene cuando la solución no puede ser mejorada. A la solución encontrada se la denomina óptimo local respecto al entorno definido.

La definición de entorno/movimiento, depende en gran medida de la estructura del problema a resolver, así como de la función objetivo. También se pueden definir diferentes criterios para seleccionar una nueva solución del entorno. Uno de los criterios más simples consiste en tomar la solución con mejor evaluación de la función objetivo, siempre que la nueva solución sea mejor que la actual. Este criterio, conocido como greedy, permite ir mejorando la solución actual mientras se pueda. El algoritmo se detiene cuando la solución no puede ser mejorada. A la solución encontrada se le denomina óptimo local respecto al entorno definido ver figura 2.16.

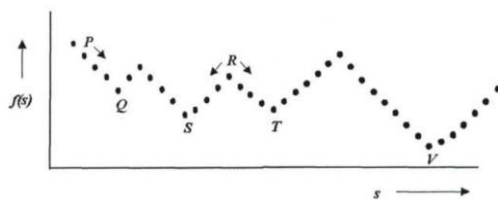


Figura 2.16: Búsqueda local (adaptado de [5]).

Para evitar estos problemas se han sugerido algunas posibles soluciones: repetir el algoritmo con diferentes puntos de partida, o considerar una estructura vecinal más compleja (aumentar el vecindario). Pero ninguna de estas variantes ha resultado ser totalmente satisfactoria, bien por la escasa mejoría en las soluciones finales o por el excesivo aumento en el tiempo de computación. [19]

2.3.4.4 Algoritmo del vecino más cercano

Para obtener un valor de k local asociado a cada ejemplo del conjunto de entrenamiento (a partir de ahora llamaremos prototipo a cada ejemplo del conjunto de entrenamiento). El método aprende este valor a partir del propio conjunto de entrenamiento. Para cada prototipo, evaluamos el rendimiento de k -NN con todos los valores de $[K]$ en un intervalo $[K_{min}, K_{max}]$ tomamos el mejor valor.

El enfoque del método es el siguiente, asignaremos a cada prototipo un valor de k , cuyo valor ideal será el número óptimo de vecinos a tener en cuenta para clasificar un ejemplo que esté en su vecindario (que un ejemplo esté en el vecindario de un prototipo significa que ése prototipo es su vecino más cercano). En el método k -NN estándar los prototipos son de la forma:

$$(x_1, \dots, x_n, y) \quad (5)$$

Donde x_1, \dots, x_n son los atributos del prototipo e y es la clase a la que pertenece. Sin embargo, con este método modificaremos los prototipos para que sean de la forma:

$$(x_1, \dots, x_n, y, k) \quad (6)$$

Donde k será el valor local de k asociado al prototipo que utilizaremos para clasificar los ejemplos de su vecindario. Para obtener el valor local óptimo de k de cada prototipo necesitaremos 2 medidas distintas: Accuracy Global: esta medida se

calculará ejecutando el algoritmo k-NN estándar sobre todo el conjunto de entrenamiento, con todos los valores de k desde 1 hasta k_{max} . Para ello, haremos una validación cruzada de 10 particiones sobre el conjunto de entrenamiento y ejecutaremos el algoritmo k-NN estándar para clasificar cada una de esas 10 particiones utilizando como clasificador las otras 9. Con ello, obtendremos la precisión del algoritmo k-NN en cada una de las 10 particiones, haciendo la media obtendremos un vector al que llamaremos acc_{global} que será una estimación de la precisión del algoritmo k-NN estándar con todos los valores de k para el conjunto de entrenamiento. Accuracy Local: para calcular esta medida también ejecutaremos el algoritmo k-NN estándar sobre todo el conjunto de entrenamiento, pero esta vez nos fijaremos en el efecto de cada valor de k desde 1 hasta k_{max} en el vecindario de cada prototipo. En principio, el vecindario de cada prototipo son sólo aquellos prototipos para los cuales éste es el vecino más cercano. Sin embargo, algunos prototipos no son el vecino más cercano de ningún otro o de muy pocos. Para evitar el efecto negativo de que haya muy pocos prototipos en un vecindario, el vecindario de cada prototipo estará formado por aquellos prototipos para los cuales éste sea uno de los 3 vecinos más cercanos. Así nos aseguramos vecindarios de tamaño mayor. El resultado será una matriz $n_{prototipos} \times k_{max}$ a la que llamaremos acc_{Local} , donde tendremos la precisión del k-NN estándar sobre el vecindario de cada prototipo con todos los valores de k. Ahora tenemos una medida que contiene la precisión del algoritmo k-NN sobre todo el conjunto de entrenamiento y otra que contiene la precisión del algoritmo k-NN sobre el vecindario de cada prototipo. Necesitamos tener en cuenta ambas medidas para obtener el valor óptimo de k que irá asociado a cada prototipo, por lo que calcularemos una nueva medida llamada $eval$ donde tendremos ambas medidas. Para ello, a la precisión que teníamos del k-NN sobre cada prototipo le vamos a sumar la precisión del k-NN sobre todo el conjunto de entrenamiento, es decir, a cada fila de la matriz acc_{local} le sumaremos el vector acc_{global} :

$$eval = aCC_{local} + aCC_{global} \quad (7)$$

Esta nueva medida eval será una matriz $n_{\text{prototipos}} \times k_{\text{max}}$, donde tendremos la precisión total del algoritmo k-NN sobre todo el conjunto de entrenamiento. Cada fila representa un prototipo, por tanto, la posición cuyo valor de precisión sea más alto de cada fila, será el valor local de k asociado a ese prototipo. Con el valor local de k ya calculado y añadido a cada prototipo, sólo quedaría clasificar nuevos ejemplos. Calculamos para cada nuevo ejemplo la distancia a todos los prototipos y clasificamos el ejemplo teniendo en cuenta para ello tantos vecinos como indique la k local de su prototipo más cercano.

Algoritmo 2.3 Vecino más cercano

```
1 procedure Mejor Vecino
2   Definir ListaTrabajos ordenadas de 1 a  $n$ .
3   while (ListaTrabajos no vacía) do
4     Asignar primer trabajo de la lista como primer trabajo de la secuencia.
5     while(Secuencia no completa)
6       Agregar a secuencia trabajo no asignado que origina menor setup.
7     endwhile
8     Calcular  $C_{\text{max}}$  de secuencia
9     Eliminar primer trabajo de ListaTrabajos
10  endwhile
11 Solución  $\leftarrow$  Secuencia de menor  $C_{\text{max}}$ 
```

Capítulo III

Estado del arte

Optimización de colonias de hormigas (ACO por sus siglas en inglés) se ocupa del complejo comportamiento social de las hormigas, lo que proporciona modelos para resolver problemas difíciles de optimización combinatoria. Optimización por enjambre de partículas (PSO) es un algoritmo de inteligencia de enjambres basado

en el comportamiento social de un grupo de individuos como el comportamiento de bandada de aves, la escuela de peces. Estos individuos, llamados partículas, se mueven a través de un espacio de búsqueda n-dimensional y comparten su información con el fin de encontrar el óptimo global. La evolución diferencial (DE) es un algoritmo metaheurístico estocástico basado en la población en el que el azar [23].

Sigue un comportamiento cooperativo-competitivo que difiere de otros algoritmos en su enfoque para encontrar nuevas soluciones. Hay varias estrategias utilizadas para Algoritmo migratorio autoorganizado (SOMA por sus siglas en inglés), como All-toOne y All-to-All. En el enfoque all-to-one, el líder actúa como un punto de referencia para todos los demás individuos activos. En el enfoque All-toAll, cada individuo se mueve hacia todos los demás agentes, lo que puede conducir a una mayor complejidad computacional.

Un algoritmo genético de migración autoorganizado codificado binario que es una hibridación de GA codificado binario simple con SOMA codificado real. Han desarrollado una hibridación de SOMA con operador cruzado de aproximación cuadrática y operador de mutación logística para mantener la diversidad de la población y aumentar la capacidad de búsqueda

Soma se basa en la cooperación entre una población de individuos (agentes) donde se emplea una técnica de búsqueda estocástica para lograr la optimización global. Se clasifica como un algoritmo evolutivo; sin embargo, no se crean nuevas personas durante el proceso de búsqueda. La robustez de este algoritmo es evidente desde su rápida convergencia hasta el extremo global.

Cada iteración en SOMA se denomina bucle de migración, en el que cada individuo localiza la mejor posición compitiendo contra otros individuos. El individuo con el mejor valor de fitness se convierte en el líder y los individuos activos viajan en el espacio de búsqueda siguiendo al líder.

El TSP dinámico se expresa a través de cambios tanto en el número de vértices como en una matriz de costes. Cada cambio en los datos de entrada puede implicar el cambio óptimo. La fórmula (8) describe la matriz de distancia en el problema.

$$D(t) = \{d_{ij}(t)\}_{n(t) \times n(t)} \quad (8)$$

donde t denota el parámetro de tiempo o iteración, i y j son extremos y n denota el recuento de vértices. Este problema tiene que ser entendido como una serie de problemas estáticos TSP.

La diversidad define cómo se varía la población. La relación precisa entre la diversidad y las estrategias de explotación/exploración en el espacio de solución es realmente importante. Cuanto más cerca esté el algoritmo de un óptimo local o global, más individuos son similares entre sí. Hay muchas medidas de diversidad en la población. Proponemos dos clases de medidas - basadas en la desviación estándar y en la entropía.

Engelbrecht y Olorunda [24] sugirieron la siguiente medida de diversidad: el diámetro del enjambre y el radio del enjambre, la distancia promedio alrededor del centro del enjambre, la distancia promedio normalizada alrededor del centro del enjambre, el promedio de la distancia alrededor de todas las partículas en el enjambre y la coherencia del enjambre. Los estudios han demostrado que la mejor medida es la distancia desde el centro de enjambre. Los investigadores Ismail y Engelbrecht desarrollaron el optimizador cooperativo de enjambre de partículas para el problema de optimización continua [4]. Otros científicos propusieron una variación de diferentes medidas para el PSO basadas en la posición de la partícula, la velocidad y el término cognitivo (todas las referencias a la posición actual de la partícula se reemplazan con su mejor posición personal).

El DTSP se evalúa su rendimiento empíricamente de los algoritmos se inspiran en métodos estándar para el TSP and base en varias variantes del SPPMD que se resuelvan discretizando las cabezas, construyendo una red falsa y encontrando una ruta más corta de la red. La heurística propuesta es adecuada para resolver grandes

instancias del DTSP con otros objetivos, manteniendo los tiempos de competencia. Probados en una gran variedad de escenarios aleatorios con diferentes radios de giro del vehículo Dubins, los métodos muestran una mejora significativa del rendimiento con respecto a los métodos estándar conocidos por la literatura. Por otra parte, los métodos parecen competitivos con el algoritmo de última generación de Le Ny [7], aunque con un tiempo de cálculo mucho menor.

El enfoque basado en la red para resolver el DSPP se analiza numéricamente el rendimiento con respecto a los diferentes niveles de discretización de los encabezados. Los algoritmos para el DTSP basado en el orden del blanco del TSP euclidiano. Una clase de algoritmos que extiende un subtour abierto agregando sucesivamente un nuevo destino, proponemos otra clase de algoritmos que comienza con un subtour cerrado que consiste en pocos objetivos y sucesivamente inserta un nuevo objetivo en el recorrido. analiza el equilibrio entre la calidad de las soluciones y el tiempo de ejecución.

3.1 Modelo formal de TSP

Formulaciones matemáticas que cubren el problema del vendedor. Dado un gráfico completo, $G = (N, A)$ minimice el costo total del recorrido de tal manera que un vendedor comience desde un nodo inicial, visite un subconjunto de nodos en $N' \subset N$ el que ningún nodo se visite más de una vez y regrese al nodo inicial, de modo que cada uno de los nodos que no están en gira estén a una distancia predeterminada de al menos uno de los nodos visitados. La formulación matemática de este problema puede expresarse como:

$$\text{Minimizar } Z = \sum_{i=1}^{|N|} \sum_{j=1}^{|N|} c_{ij} x_{ij} \quad (9)$$

$$\sum_{i=1}^{|N|} \sum_{j \in D_i} x_{ij} \geq 1, \forall l \in N \quad (10)$$

$$\sum_{i=1}^{|N|} x_{ik} = \sum_{j=1}^{|N|} x_{kj} = 0 \text{ or } 1, \forall k \in N \quad (11)$$

$$x_{ij} \in \{0,1\} \quad (12)$$

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1, \quad \forall S \subset N' \subset N, \quad 2 \leq |S| \leq |N'| - 2 \quad (13)$$

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1, \quad \forall S \subset N' \subset N, \quad 2 \leq |S| \leq |N'| - 2 \quad (14)$$

donde, es el N' conjunto de nodos visitantes, c_{ij} es el costo del nodo i al nodo j ,

$$x_{ij} = \begin{cases} 1, & \text{si } \exists \text{ una ruta desde el nodo } i \text{ al nodo } j \\ 0, & \text{de lo contrario.} \end{cases}$$

$D_i = \{j: d_{ij} \leq \Delta_j\}$, d_{ij} = Distancia más corta entre i y j , Δ_j = distancia máxima de cobertura en el nodo j .

La ecuación (1) minimiza el costo total del tour. La ecuación (2) asegura que cada nodo del grafo sea visitado o esté cubierto por los nodos visitados. La ecuación (3) indica que, para un vértice, en grado y en grado de salida son los mismos. La ecuación (4) implica el carácter binario de x_{ij} y (5) es la restricción de eliminación del subtour, es decir, si un nodo está en tour, no se visita más de una vez (solo el nodo inicial se visita dos veces).

La ecuación anterior (1) - (5) se puede reescribir de la siguiente manera: Sea $N = \{x_1, x_2, x_3, \dots, x_{|N|}\}$ el conjunto de nodos. Determine un recorrido completo $(x_{\alpha_1}, x_{\alpha_2}, x_{\alpha_3}, \dots, x_{\alpha_m}, x_{\alpha_1})$, $m \leq |N|$ a

$$\text{Minimizar } \sum_{i=1}^{m-1} c(x_{\alpha_i} x_{\alpha_{i+1}}) + c(x_{\alpha_m} x_{\alpha_1}) \quad (15)$$

$$\text{tal que, } x_j \in \bar{B}(x_{\alpha_i}, \Delta_{\alpha_i}), \forall x_j \in N \text{ y para algunos } \dot{x}; \quad (16)$$

donde $\alpha_i \in 1,2,3, \dots, |N|$ y $\alpha_i \neq \alpha_j$ para $i \neq j$, $c(i, j) = c_{ij}$, $\bar{B}(a, r)$ significa disco cerrado con centro a y radio de r , $\Delta_j =$ distancia máxima de cobertura en el nodo j .

Cubriendo el problema del vendedor viajero sólido El problema se puede plantear de la siguiente manera: Dado un gráfico completo $G = (N, A)$, minimice el costo total del recorrido cuando un vendedor comienza desde un nodo inicial y visita un subconjunto de nodos $N' \subset N$ exactamente una vez utilizando cualquiera de los vehículos disponibles en cada nodo y regresa al nodo inicial de modo que todos los nodos no visitados se cubran dentro de una distancia predeterminada de los nodos visitados. Sea en cada nodo p el número de medios de transporte disponible y $c(i, j, k)$ sea el costo de nodo i a nodo j usando el vehículo k -ésimo, ($k=1,2,3, \dots, p$). A continuación, la formulación matemática se da a continuación:

Sea el conjunto de $N = \{x_1, x_2, x_3, \dots, x_{|N|}\}$ nodos y $V = \{v_1, v_2, v_3, \dots, v_p\}$ sea el conjunto de vehículos. Determine un recorrido completo $(x_{\alpha_1}, x_{\alpha_2}, x_{\alpha_3}, \dots, x_{\alpha_m}, x_{\alpha_1})$, $m \leq |N|$ a

$$\text{Minimizar } \sum_{i=1}^{m-1} c(x_{\alpha_i} x_{\alpha_{i+1}}, v'_{\alpha_i}) + c(x_{\alpha_m} x_{\alpha_1}, v'_{\alpha_m}); \quad (17)$$

Tal que,

$$x_j \in \bar{B}(x_{\alpha_i}, \Delta_{\alpha_i}), \quad \forall x_j \in N \text{ and for some } i. \quad (18)$$

donde $\alpha_i \in 1,2,3, \dots, |N|$ y $\alpha_i \neq \alpha_j$ para $i \neq j, v'_{\alpha_i} \in V, \forall \alpha_i \in 1,2,3, \dots, |N|, c(i, j, k) = c_{ijk}, \bar{B}(a, r)$ significa disco cerrado con centro a y radio de $r, \Delta_j =$ distancia máxima de cobertura en el nodo j .

3.2 El problema dinámico del agente viajero

La tecnología ha cambiado rápidamente y tenemos la necesidad de tener la información más rápida posible esto nos lleva a encontrar algoritmos para resolver problemas de optimización para garantizar que los sistemas funcionen de forma eficaz y fiable nos conduce a un modelo matemático teórico el TSP dinámico (DTSP).

El objetivo es encontrar una solución óptima en cada momento, a medida que el tiempo se requiere soluciones rápidas y su calidad se reducirá, los objetivos de calidad en la solución y tiempo de respuesta está en conflicto.

La estrategia de resolver el problema del DTSP es utilizar el algoritmo genético y almacenar la ruta óptima y si se mueve o se quita un elemento entra de nuevo al algoritmo y trabajaría optimizando la ruta que tenía como óptima y así se reduce el tiempo de respuesta y el trabajo de cómputo.

Descripción del DTSP

Un TSP dinámico es un TSP determinado por un costo dinámico (distancia) matriz.

$$D(t) = \{d_{ij}(t)\}_{n(t) \times n(t)} \quad (19)$$

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

Dados todos $n(t)$ ($P_1, P_2, \dots, P_{n(t)}$) puntos, y la correspondencia de costo matriz $D = \{d_{ij}(t)\}, i, j = 1, 2, \dots, n(t)$, encontrar un mínimo costo de la ruta que contiene todos los $n(t)$ puntos, donde t se encuentra en el momento del tiempo t se encuentra por la distancia entre el punto del objetivo P_i y el punto objetivo P_j , y $d_{ij}(t) = d_{ji}(t)$.

Por ejemplo

$$\text{Min}(d(T(t))) = \min \left(\sum_{i=1}^{n(t)} d_{T_i, T_{i+1}}(t) \right) \quad (20)$$

Donde $T \in \{1, 2, \dots, n(t)\}$ if $i \neq j$, then $T_i \neq T_j, T_{n(t)+1} = T$

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.

$$D(t_k) = \{d_{ij}(t_k)\}_{n(t_k) \times n(t_k)} \quad (21)$$

$k = 0, 1, 2, \dots, m - 1$, con la ventana de tiempo $[t_k, t_{k+1}]$, donde $\{t_k\}_{i=0}^m$ es una secuencia de puntos de puntos de muestreo de tiempo del mundo real [25].

3.3 Otros problemas sobre rutas

La trascendencia económica del sector del transporte genera costos de gran envergadura. La distribución física representa para las empresas entre la sexta y la cuarta parte de las ventas y entre uno y dos tercios del total de los costos logísticos [26].

Problema de Rutas de Vehículos resulta extremadamente útil en una gran variedad de casos reales. A continuación, se comentan brevemente algunas de sus aplicaciones más importantes:

- Logística. - Las aplicaciones más directas y abundantes del VRP se centran en el campo de la logística. Entre ellas, se encuentran: rutas de vendedores, rutas escolares, reparto de mercancías y correo o sistema de recogidas de basura.
- Industria. - Las aplicaciones en industria no son tantas como en logística. Un ejemplo de dicha aplicación es la secuenciación de tareas.

Los Problemas de Rutas de Vehículos (PRV, o VRP del inglés Vehicle Routing Problem) son una serie de problemas que hacen referencia al transporte de mercancías a ciertos puntos con una flota de vehículos. Formalmente, se definen sobre un grafo completo $G = (V, A)$, con costes asociados a los arcos (A) y donde los clientes representan los vértices (V), siendo el vértice 1 el depósito o almacén donde se encuentran los vehículos de transporte y la mercancía.

Problema de Rutas de Vehículos Capacitado (PRVC, o CVRP del inglés Capacitated Vehicle Routing Problem), cuya finalidad es determinar las rutas que una flota de vehículos, con capacidad dada y que salen de un punto común o depósito, debe seguir para cubrir la demanda de una serie de clientes, existen muchas extensiones de éste (se muestran las más usuales en aplicaciones reales.

- VRP con ventanas de tiempo (time windows) (VRPTW): el servicio a cada cliente tiene que realizarse en un intervalo horario determinado, teniendo que permanecer el vehículo en el lugar del cliente durante dicho servicio.
- VRP con retornos (backhauls) (VRPWB): también conocido como Linehaul Backhaul Problem, en este tipo de problemas, los clientes se dividen en dos grupos. El primero de ellos, denominado "linehaul", engloba aquellos clientes a los que hay que entregar una cantidad determinada de producto. El segundo, "backhaul", contiene a aquellos a los que hay que recoger una cantidad determinada de

producto. La principal restricción es que, para cada ruta, todas las entregas deben hacerse antes que las recogidas.

- VRP multi- depósito: los vehículos no son diáfanos, sino que tienen divisiones interiores. La mayoría de las veces, la dificultad de este problema reside en que cada producto tiene que ir en una de esas dependencias, no pudiéndose mezclar entre ellos.
- VRP periódico (VRPP): el servicio a los clientes no se realiza todos los días sino de forma periódica (cada M días).
- VRP con inventarios (VRPI): la cantidad demandada por cada cliente no es fija, sino que dependerá del nivel mínimo de inventario que éste quiera mantener.

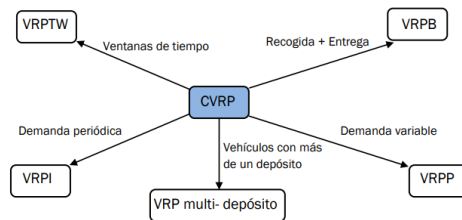


Figura 3.1: El CVRP y sus variantes (adaptado de [27]).

A pesar de la aparente sencillez de su planteamiento, es muy complejo de resolver: siendo n el número de clientes a visitar, el número total de posibles rutas es $(n - 1)!/2$ [27].

Tipos de Problemas de Rutas de Vehículos Capacitado

Se distinguen dos tipos de CVRP, en función de la naturaleza de la red:

- Problema de Rutas de Vehículos Capacitado Simétrico (SCVRP): si el grafo es no dirigido, es decir, si la matriz de costes es simétrica ($c_{ij} = c_{ji}$).

- Problema de Rutas de Vehículos Capacitado Asimétrico (ACVRP): si el grafo es dirigido, es decir, si la matriz de costes es asimétrica.

Al igual que se expuso en el Problema del Viajero, dado que todo grafo no dirigido puede transformarse en uno dirigido, el SCVRP puede verse como un caso particular del ACVRP.

Capítulo IV

Metodología

4.1 Librería TSPLIB

TSPLIB es una biblioteca de instancias de ejemplo para el TSP (y problemas relacionados) de varias fuentes y de varios tipos. Están disponibles instancias de las siguientes clases de problemas.

4.1.1 Problema del viajante de comercio simétrico (TSP)

Dado un conjunto de n nodos y las distancias para cada par de nodos, encontrar un viaje de ida y vuelta de longitud total mínima que visite cada nodo exactamente una vez. La distancia del nodo i al nodo j es la misma que la del nodo j al nodo i .

Problema del ciclo hamiltoniano (HCP)

Dado un gráfico, comprueba si el gráfico contiene un ciclo hamiltoniano o no.

Problema del viajante de comercio asimétrico (ATSP)

Dado un conjunto de n nodos y las distancias para cada par de nodos, encontrar un viaje de ida y vuelta de longitud total mínima que visite cada nodo exactamente una vez. En este caso, la distancia del nodo i al nodo j y la distancia del nodo j al nodo i pueden ser diferentes.

4.1.2 Problema de ordenación secuencial (SOP)

Este problema es un problema de viajante de comercio asimétrico con restricciones adicionales. Dado un conjunto de nodos y distancias para cada par de nodos, encontrar un camino hamiltoniano desde el nodo 1 al nodo n de longitud mínima

que tenga en cuenta las restricciones de precedencia dadas. Cada restricción de precedencia requiere que algún nodo i tenga que ser visitado antes que algún otro nodo j .

4.1.3 Problema de enrutamiento de vehículos capacitados (CVRP)

Se dan $n-1$ nodos, un depósito y distancias de los nodos al depósito, así como entre los nodos. Todos los nodos tienen demandas que pueden ser satisfechas por el depósito. Para la entrega en los nodos se dispone de camiones con idéntica capacidad. El problema consiste en encontrar recorridos para los camiones de longitud total mínima que satisfagan las demandas de los nodos sin violar la restricción de capacidad de los camiones. No se especifica el número de camiones. Cada recorrido visita un subconjunto de red y comienza y termina en el depósito. (Nota: En algunos archivos de datos se da una colección de depósitos alternativos. El CVRP se obtiene seleccionando uno de estos depósitos).

4.1 La parte del pliego de condiciones

Cada archivo consta de una parte de especificación y de una parte de datos. La parte de especificación contiene información sobre el formato del archivo y sobre su contenido. La parte de datos contiene datos explícitos.

Todas las entradas de esta sección tienen la forma $\langle \text{palabra clave} \rangle \langle \text{valor} \rangle$, donde $\langle \text{palabra clave} \rangle$ denota una palabra clave alfanumérica y $\langle \text{valor} \rangle$ denota datos alfanuméricos o numéricos. Los términos $\langle \text{cadena} \rangle$, $\langle \text{entero} \rangle$ y $\langle \text{real} \rangle$ denotan una cadena de caracteres, un entero o un dato real, respectivamente. El orden de especificación de las palabras clave en el archivo de datos es arbitrario (en principio), pero debe ser coherente, es decir, siempre que se especifique una palabra clave, debe conocerse toda la información necesaria para la correcta interpretación de la misma. A continuación, se ofrece una lista de todas las palabras clave disponibles.

4.1.1 NOMBRE:

Identifica el archivo de datos.

4.1.2 TIPO:

MAX 2D Las ponderaciones son las distancias máximas en 2D
MAX 3D Las ponderaciones son las distancias máximas en 3D
MAN 2D Las ponderaciones son las distancias Manhattan en 2D
MAN 3D Las ponderaciones son las distancias Manhattan en 3D

CEIL-2D Las ponderaciones son distancias euclidianas en 2D redondeadas

Los GEOWeights son distancias geográficas

ATT Función de distancia especial para los problemas att48 y att532

XRAY1 Función especial de distancia para problemas de cristalografía (Versión 1)

XRAY2 Función especial de distancia para problemas de cristalografía (Versión

2) ESPECIAL Existe una función especial de distancia documentada en otro lugar

Acceda a

TSPLIB está disponible en

<http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/>

TSPLIB es una biblioteca de instancias de muestra para el TSP (y problemas relacionados) de varias fuentes y de varios tipos.

Los nombres de los archivos de datos correspondientes se obtienen añadiendo el sufijo ".tsp" al nombre del problema. También se proporcionan algunos recorridos óptimos. Los archivos correspondientes tienen nombres con el sufijo ".opt.tour".

Realizaremos el algoritmo genético para el TSP, con la ruta más corta generada por el algoritmo, mostrar en una gráfica los puntos de cada una de las coordenadas x, y, para el trazo de la ruta óptima para la visualización del recorrido.

Se realizará el algoritmo genético con base en la fase que contiene el algoritmo genético, selección, cruza, mutación, para obtener la ruta óptima de las formas posible que puede tener la serie de puntos que se encuentran en los archivos TSPLIB, realizar las pruebas de selección de padres tomando la mitad de la población y evaluar el resultado.

Evaluar la población con el método de ruleta para elegir los padres para generar la población, para generar los hijos se tomara el método de cruza para generarlos ya generarlos se utiliza el método de mutación para tener mayor diversidad en las seleccione y el algoritmo pueda tener mejores resultados evaluados en la función objetivo que es la mejor distancia evaluada en cada generación teniendo en cuenta que se tomara una generación de 1000 propuesta, y con una población de 50 individuos para que las interacciones puedan ser más rápidas ya que el procesamiento de cómputo disminuirá

Al generar los puntos de coordenadas del archivo seleccionado del repositorio TSP se evaluará con rutas con el algoritmo, para encontrar una ruta aproximada igual o mejor indicada en el repositorio.

El algoritmo genético diseñad debe ser capaz de leer una instancia nueva que se quiera resolver desde un fichero externo en formato TSPLIB. Este formato es el utilizado en el repositorio de instancias de problemas TSP con ciudades reales de todo el mundo.

Se deberán realizar varios experimentos con una instancia de dicho repositorio de la que se conozca la solución óptima, tratando de determinar un tamaño de población inicial y una tasa de reproducción adecuados que garanticen una elevada probabilidad de encontrar ese óptimo.

Links para obtener la ruta optima especificada en la ruta optima mostradas en el repositorio.

- softlib.rice.edu/pub/tsplib/tsp/?C
- <https://gitlab.crans.org/alopez/projet-optimisation/-/tree/master/problemes>

Se implementará el algoritmo del TSP y se añadirán los nodos dinámicos para evaluar de nuevo la ruta generada por el algoritmo genético, almacenando la ruta última óptima que se encontró antes de que se añaden los nodos, para obtener el TSP dinámico se inserta puntos para que se genere la nueva ruta.

Se realizan experimentos con los archivos de TSPLIB para evaluar el algoritmo.

Teniendo la ruta óptima trazaremos la gráfica los puntos (coordenadas) y crear el entorno de aplicación para su observación.

Se realiza una prueba con 15 coordenada para evaluar el algoritmo genético y se generar 33 veces para ver si el resultado se acerca al óptimo propuesto, en el segundo experimento se tomará el ch150 para generar y evaluar la ruta óptima y comprar la ruta con el rango de Kendall para conocer el coeficiente de correlación y tener una evaluación óptima de la ruta generada por el algoritmo genético.

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 [28].

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 óptima durante su trayecto, introducirlo a la población y obtener más rápido una manera más eficiente el resultado.

4.2 Diseño

La utilización de los archivos de TSPLIB [29] 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 óptima 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.

4.3 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.

4.4 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.

4.5 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.

4.6 Analisis de información

La prueba de correlación thau de Kendall se utiliza para medir el grado y el sentido de la relación que hay entre dos variables medidas por los menos en un nivel ordinal. Es una buena alternativa al coeficiente de correlación de Pearson cuando las variables se encuentran en un nivel de medición de intervalos, pero no se puede cumplir satisfactoriamente con los requisitos básicos para el uso de una prueba paramétrica.

La distancia Kendall-Tau es la que se utiliza en esta memoria para medir la distancia entre dos permutaciones. La distancia Kendall-Tau mide el número de intercambios de elementos necesarios, en adelante desordenes, para convertir una permutación en otra, por lo que a mayor distancia mayor será la diferencia entre dos permutaciones.

La distancia Kendall-Tau entre dos permutaciones π y ρ viene dada por:

$$d(\pi, \rho) = \left| \{(i, j): i < j, ((\pi(i) < \pi(j) \wedge \rho(i) > \rho(j)) \vee (\pi(i) > \pi(j) \wedge \rho(i) < \rho(j)))\} \right| \quad (20)$$

Donde, $\pi(i)$ y $\rho(i)$ es la posición del elemento i en π y ρ respectivamente. Su valor máximo es $n(n - 1)/2$, que es el número máximo de desórdenes para n elementos. Esta métrica es una de las más utilizadas para la comparación de permutaciones.

Para contrastar el resultado entre la ruta optima de TSPLIB y la generada por el algoritmo SGA propuesto, se utilizó la Prueba de Kendall [30], la cual se plantea a continuación.

Hipótesis nula.

H₀: 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

$z = 2.5936$, $p\text{-value} = 0.009498$

$0.009498 < 0.05$ (se acepta la hipótesis alternativa)

$\tau = 0.1423525$

Capítulo V

Experimentación

5.1 Resultados de la experimentación del algoritmo genético con elitismo en la solución al problema de agente viajero dinámico (DTSP por sus siglas en inglés)

Realizaremos una prueba con 15 coordenadas, para probar el algoritmo genético para resolver el problema del agente viajero dinámico, para ver la eficiencia y verificar si muestra una ruta optima en el resultado con 500 generaciones ver Figura 5.3.

```
{ "x": 37.43935167, "y": 541.2090699 },  
{ "x": 612.1759509, "y": 494.3166877 },  
{ "x": 38.13123382, "y": 353.1484582 },  
{ "x": 53.44180811, "y": 131.4849014 },  
{ "x": 143.0606355, "y": 631.7200954 },  
{ "x": 689.9451267, "y": 468.5354999 },  
{ "x": 112.7478816, "y": 529.4177578 },  
{ "x": 141.4875865, "y": 504.8184856 },  
{ "x": 661.0513902, "y": 445.9375182 },  
{ "x": 98.78990366, "y": 384.5926031 },  
{ "x": 697.3881697, "y": 180.3962284 },  
{ "x": 536.489419, "y": 287.2279085 },  
{ "x": 192.4067321, "y": 20.43940593 },  
{ "x": 282.7865259, "y": 229.8001556 },  
{ "x": 240.8251726, "y": 281.5141437 },  
{ "x": 246.9281323, "y": 322.4613321 },
```

Figura 5.1: Coordenadas de prueba (elaboración propia).

Se obtiene la tabla de matriz de costos la cual representa una instancia de 15 puntos, se indica la distancia de los puntos por renglón y columna para sumar las distancias entre los puntos y tener la distancia final evaluada del recorrido.

			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
Punto	Suma	x	y	37.439	612	38	53.441808	143.06064	689.94513	112.7479	141.48750	661.05139	98.789004	697.38817	536.4894	192.40673	282.7865	240.82517
1	5537.64	37.4	541	0	576.6	188.1	410.0365506	139.0974527	656.5403523	76.22604359	110.2284437	630.8475994	168.2040662	752.1425058	559.961963	543.3377696	396.4476609	329.859435
2	5999.25	612	494	576.6464	0	591.1	666.2062349	488.823972	81.93115616	500.6600455	470.8055056	68.7702887	524.9805789	325.2801516	220.4863116	633.06072	422.4528247	428.0027141
3	5074.02	38.1	353	188.0619	591.1	0	222.1916877	297.6782431	661.9482762	191.411886	183.5383689	629.7930858	68.3242806	681.5152532	502.6991144	366.737294	273.9909044	214.9797847
4	6074.86	53.4	131	410.0366	666.2	222.2	0	508.1995509	720.2357812	402.3279366	383.5753175	684.156307	257.1380144	645.8012345	507.5341157	177.8829732	245.5293345	240.0443685
5	5679.62	143	632	139.0975	488.8	297.7	508.1995509	0	570.7117126	106.6987878	126.911359	550.2995438	251.0615366	714.8232286	522.9350577	613.2692056	425.514938	363.5960854
6	6609.03	690	469	656.5404	81.93	661.9	720.2357812	570.7117126	0	580.399267	549.6563731	36.68128659	597.0853437	288.2353876	237.5312546	669.5778995	471.9880198	486.5035671
7	4903.35	113	529	76.22604	500.7	191.4	402.3279366	106.6987878	580.399267	0	37.82981404	554.6221127	145.4962219	680.8966841	488.0705007	515.1742376	344.5052222	279.034038
8	4648.27	141	505	110.2284	470.8	183.5	383.5753175	126.911359	549.6563731	37.82981404	0	522.8895815	127.5827376	643.6421827	450.9679666	487.0481005	309.1932601	244.4029155
9	6233.33	661	446	630.8476	68.77	629.8	684.156307	550.2995438	36.68128659	554.6221127	522.8895815	0	565.9980709	268.0159289	201.7533764	632.990094	435.6600361	451.2484017
10	4578.28	98.8	385	168.2041	523	68.32	257.1380144	251.0615366	397.0854347	145.4962219	127.5827376	565.9980709	0	632.4682154	448.3979811	375.9942368	240.4484535	175.4969698
11	7340.83	697	180	752.1425	525.3	681.5	685.8012345	714.8232286	288.2353876	680.8966841	643.6421827	268.0159289	632.4682154	0	193.1357448	529.7097671	417.5347543	467.6565637
12	5324.71	536	287	559.962	220.5	602.7	507.5341157	522.9350771	237.5312546	488.0705007	450.9679666	201.7533764	448.3979811	193.1357446	0	435.3952334	260.121337	395.719451
13	6473.74	192	20.4	543.3378	633.1	366.7	177.8829732	613.2692056	669.5778995	515.1742376	487.0481005	632.990094	375.9942368	529.7097671	435.3952334	0	278.0360293	265.5265789
14	4542.02	283	230	396.4477	422.5	274	249.5293345	425.514938	471.9880198	344.5052222	309.1932601	435.6600361	240.4484535	417.5347543	260.121337	228.0360292	0	66.59648441
15	4308.64	241	282	329.8594	428	215	240.0443685	363.5960854	486.5035671	279.034038	244.4029155	451.2484017	175.4969698	467.6265637	295.719451	265.5265789	66.59648441	0

Tabla 5.1 Simétrica de distancias de 15 puntos (elaboración propia).

Se evalúa la ruta generada por el algoritmo genético

Ruta generada

1,7,8,10,3,4,13,14,15,12,11,6,9,2,5,1

1	7	76.2260436
7	8	76.2260436
8	10	76.2260436
10	3	68.3242891
3	4	222.191688
4	13	177.882973
13	14	228.036029
14	15	66.5964844
15	12	295.719451
12	11	193.135745
11	6	288.235388
6	9	36.6812866
9	2	68.7702887
2	5	488.823972
5	1	139.097453
		2508

Tabla 5.2 Suma de distancias entre cada nodo, ruta (elaboración propia).

Grafica generada

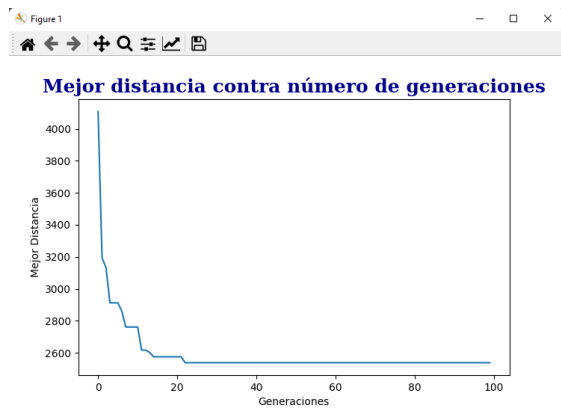


Figura 4.2: Mejor distancia contra el número de generaciones (elaboración propia).

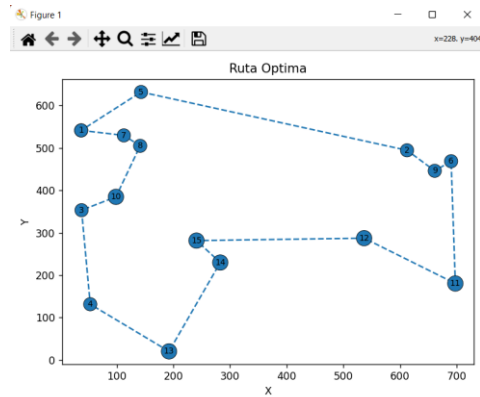


Figura 5.3: Grafica puntos a recorrer (elaboración propia).

5.1.1 Evaluación con la librería TSPLIB

Se utilizo el archivo CH150.lib

Repositorios

- <https://gitlab.crans.org/alopez/projet-optimisation/-/tree/master/problemes>
- softlib.rice.edu/pub/tsplib/tsp/

se encuentran las librerías de TSPLIB y la ruta óptima para cada archivo

Ejecutando el algoritmo genético pata los puntos de la librería CH150.lib

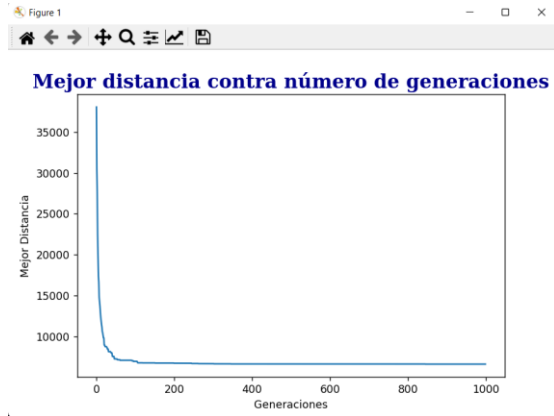


Figura 5.4: Mejor distancia contra número de generaciones (elaboración propia).

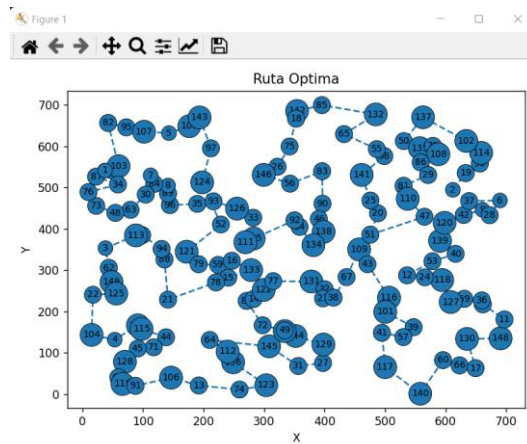


Figura 5.5: Ruta optima (elaboración propia).

```
[1, 98, 103, 82, 95, 107, 5, 100, 143, 97, 124, 52, 121, 79, 59, 16, 133, 15, 78, 21, 88, 94, 10, 113, 3, 62, 149, 125, 22, 104, 4, 150, 115, 44, 71, 45, 128, 68, 119, 91, 106, 13, 74, 123, 136, 112, 64, 145, 31, 27, 129, 144, 147, 49, 72, 80, 14, 122, 77, 131, 32, 23, 38, 67, 109, 43, 116, 101, 39, 57, 41, 117, 140, 60, 66, 17, 130, 148, 11, 61, 36, 69, 127, 118, 24, 12, 53, 40, 139, 120, 42, 9, 28, 6, 37, 2, 19, 99, 114, 102, 137, 50, 135, 70, 108, 86, 29, 81, 110, 47, 51, 20, 25, 141, 58, 55, 65, 132, 85, 142, 18, 75, 26, 146, 56, 83, 90, 46, 138, 134, 54, 92, 105, 111, 33, 126, 93, 35, 96, 89, 8, 84, 7, 30, 63, 48, 73, 76, 34, 87, 1]
```

*Figura 5.6: Ruta optima generada por el Algoritmo Genético con elitismo
(elaboración propia).*

Fitness: 6625

1,98,103,82,95,107,5,100,143,97,146,26,75,18,142,85,65,132,137,50,55,58,141,8
3,56,90,46,

92,54,138,134,131,32,23,38,67,43,109,51,20,25,110,81,29,86,135,70,108,102,114
,99,19,2,

37,6,28,9,42,120,47,139,40,53,118,24,12,116,101,41,57,39,127,69,36,61,11,148,1
30,17,66

60,140,117,129,27,31,123,74,13,106,91,119,68,128,45,71,44,64,112,136,145,144,
49,147,72,80,14,122,77,133,15,78,21,150,115,4,104,22,125,149,62,3,113,10,94,8
8,121,79,59,16,111,

105,33,126,52,93,124,35,96,89,8,7,84,30,63,48,73,76,34,87,1

Fitness 6528

5.2 Resumen y conclusiones del estado del arte

Con base en los resultados obtenidos en este trabajo de la generación del algoritmo genético se puede comprobar que es posible tener una ruta óptima generada al visitar todos los puntos y regresar al origen

Este reporte es un caso de estudio del TSP, propuesto el algoritmo genético para resolver el problema del agente viajero, el algoritmo genético nos ofrece soluciones adecuadas en cuanto a las rutas óptimas, si son demasiados puntos lo resuelve pero con un costo computacional en tiempo pero nos muestra la ruta óptima, el lenguaje utilizado para elaborar el algoritmo genético es Python que nos generó mejor facilidad en su programación así como cambiar funciones y parámetros para poder programar el algoritmo, así incluir las fases del algoritmo genético, la eficiencia del algoritmo genético se muestra entre menos puntos es más rápido pero nos regresa un resultado óptimo que es lo que se busca.

La solución generada por el algoritmo es la serie de puntos a moverse para poder llegar al destino pasando por todos los nodos, con una menor población inicial nos da la optimización en sí, donde se muestra la eficiencia del algoritmo en la generación de la ruta, si se aumenta el tamaño de la población inicial aumenta el tiempo de cómputo en el procesamiento de información.

Los algoritmos genéticos tradicionales son de gran ayuda para problemas de búsqueda y optimización, sin embargo, estos algoritmos se vuelven insuficientes para problemas grandes, y su tiempo de ejecución se vuelve excesivo.

Con respecto al primer experimento realizado, se concluye que la técnica de selección de ruleta por costo brinda mejores resultados. Para la gran mayoría de problemas se alcanzaron buenas soluciones con el mecanismo de ruleta por costo, excepto en el primer problema, con 5 ciudades, en la cual las dos versiones lograron encontrar la mejor solución reportada.

Observando los resultados registrados del experimento 2, se concluye que de las distintas técnicas de mutación que se implementaron, solo dos de éstas presentan buenos resultados. Es decir, las técnicas de mutación por intercambio y revuelto (scrambled). Inclusive, para el caso de 150 ciudades se obtiene un resultado mayor al reportado en la literatura; la mejor solución presentada en los benchmark es de

6420 mientras que el obtenido con la implementación de mutación por desplazamiento es de 6640.

En este reporte se explora la estrategia orientada a mejorar la operación de entrega de carga completa punto a punto. La estrategia empleada se enfocó en la construcción de rutas que generan una secuencia continua en las entregas programadas para cada vehículo buscando reducir su tiempo de inactividad. La mayoría de los trabajos revisados para esta disertación enfocan su estrategia de optimización en la reducción de distancias o en la reducción de costos de las rutas. Lo anterior abre un espacio de oportunidad interesante para explorar la contribución de los algoritmos genéticos en este tipo de problemas.

Para futuros experimentos se realizarán con más archivos de TSPLIB, para realizar pruebas del funcionamiento del algoritmo genético y la adaptación y modificación del código para el DTSP.

5.3 Simulación DTSP

La simulación se realizó en código Python con las librerías Turtle [31], donde podemos tener un punto rojo que es nodo en movimiento completamente al azar, y a su vez se generan los puntos a partir de las librerías de TSPLIB o se generan de manera aleatoria indicando los puntos que uno quiera.

En el experimento se utilizó las coordenadas de la librería de TSPLIB que incluye el archivo ch150.tsp donde se encuentran las 150 coordenadas donde cada punto se muestra un x, y, se grafica en pantalla con la posición inicial 0,0 en la parte inferior izquierda, y se toma al azar el punto rojo donde se coloca en la posición y se marca un radio donde el radio toma los puntos donde el algoritmo genético con elitismo trabajo y genera la ruta optima ver figura 5.9.

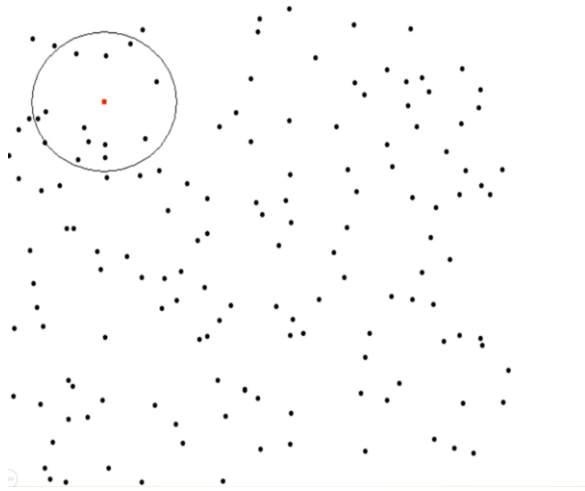


Figura 5.7: Primer movimiento (elaboración propia).

Podemos observar la ruta generada por el algoritmo genético con elitismo la ruta optima que se encuentra los puntos dentro del radio y el punto rojo representa el agente donde el inicio y fin del recorrido ver figura 3.10.

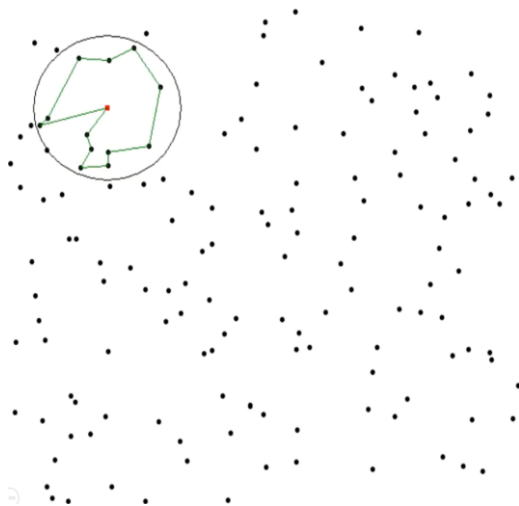


Figura 5.8: Se genera la ruta en el primer movimiento (elaboración propia).

En el segundo movimiento se mueve al azar el punto rojo y a partir de ahí se traza un radio definido por el usuario y los puntos que están dentro del radio son los puntos que trabajara el algoritmo genético con elitismo para generar la ruta optima ver figura 3.11.

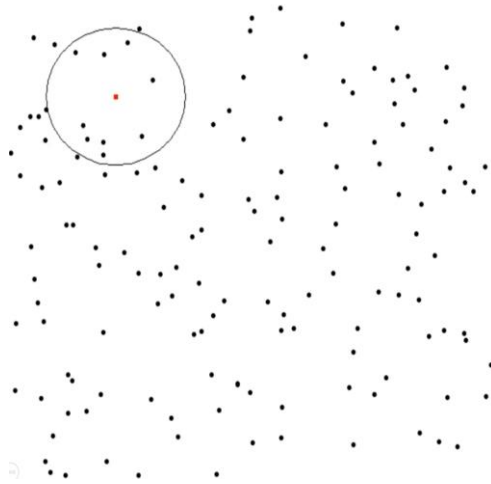


Figura 5.9: Segundo movimiento (elaboración propia).

En el segundo movimiento podemos observar en la figura 3.12 que el algoritmo genético con elitismo trabajo para darnos la ruta optima de los puntos que están dentro del radio que inicialmente se tiene y con programación se grafica la ruta y podemos verla en mod gráfico.

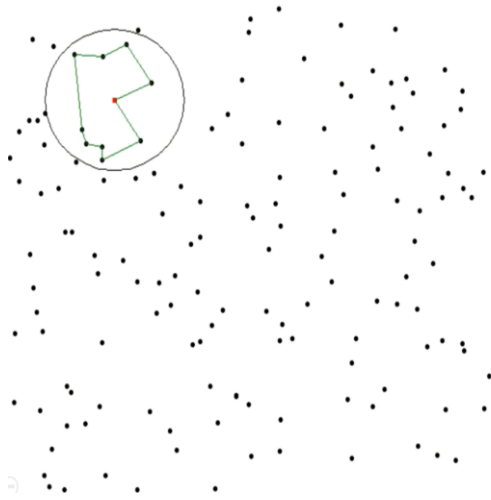


Figura 5.10: Ruta generada en el segundo movimiento (elaboración propia).

En el tercer movimiento el punto rojo se mueve al azar para indicar el punto inicial y a partir del radio que define el usuario se marca un limite y los puntos que estén dentro del radio el algoritmo genético trabaja para darnos la ruta optima ver figura 5.13.

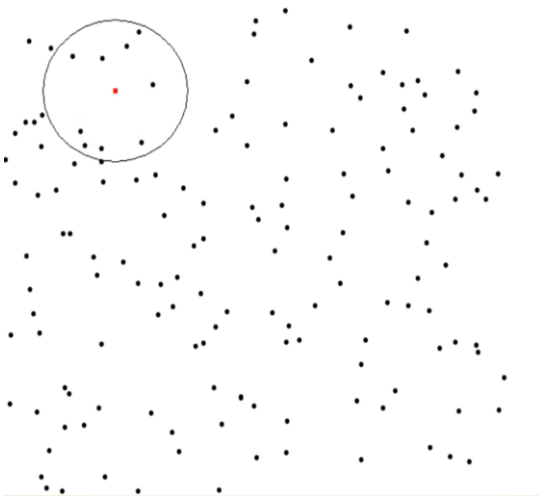


Figura 5.11: Tercer movimiento (elaboración propia).

Al terminar de procesar la información de los puntos el algoritmo genético con el algoritmo genético con elitismo nos da la ruta optima y por medio de traficación con Python se muestra la ruta optima ver figura 3.14

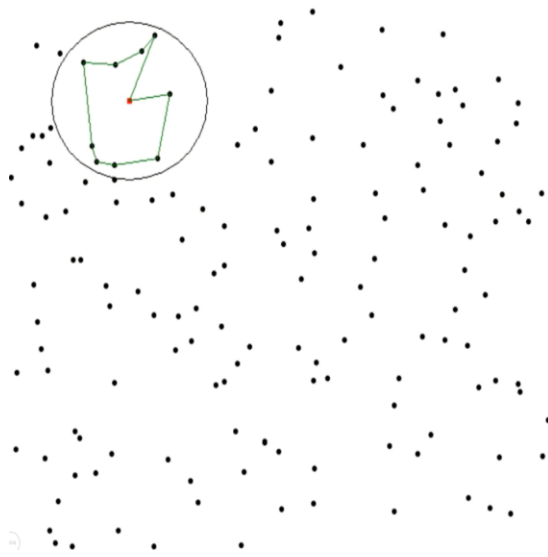


Figura 5.12: Ruta generada en el tercer movimiento (elaboración propia).

Archivo utilizado CH150.TSP, radio 100, 500 generaciones y una pausa de 10 segundos para que el algoritmo genético nos de la ruta optima en cada movimiento del agente en este caso se muestra con el punto rojo al azar.

Se realizo una muestra de 6 puntos de coordenadas de GPS en la aplicación en Lenguaje Flutter para la generación de la ruta con la API de Google Maps en su trazado en la ruta de la vida real (Ver figura 5.15).

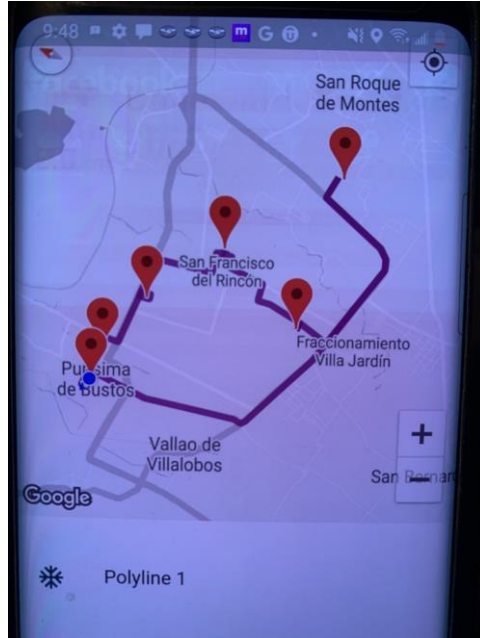


Figura 5.13: Aplicación en dispositivo móvil Flutter (elaboración propia).

Capítulo VI

Resultados

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

6.1 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

6.2 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

Capítulo VII

Conclusiones

En el presente trabajo se presenta el problema del agente viajero en el contexto de e-commerce. Este problema es importante porque nos permite identificar los puntos de venta de un producto que estamos buscando para indicarnos la ruta más corta para poderlo adquirir, ya que el agente se mueve los puntos de venta aparecerán dependiendo de la ruta donde se encuentre y se presenta el problema del agente viajero dinámico.

Se propuso el algoritmo genético con elitismo (SAG) para el TSP y el DTSP.

De la revisión del algoritmo genético podemos concluir que se encuentra la ruta más corta indicándonos los puntos donde se pueden visitar.

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.

Se realizo una revisión exhaustiva del estado del arte para el Problema del Agente Viajero (TSP). Es un de los problemas más importantes de la optimización combinatoria, tanto por su complejidad computacional, como sus múltiples aplicaciones. Actualmente hay muchas revistas científicas para resolver el TSP, GRASP, Simulated annealing, Tabú search, etc.

Se realizaron los resultados con el Algoritmo genético con elitismo para resolver el TSP y el DTSP, la estructura se puede plantear como la función objetivo de reducir el mínimo coste de la ruta y con una población inicial.

Se realizo el algoritmo genético en Python para resolver el TSP, se realizaron pruebas con las instancias de la librería TSPLIB, alcanzado algunos óptimos. En cuanto al DTSP se añadieron nodos y eliminación de nodos para reutilizar el algoritmo genético.

Se realizo una simulación en Python donde se colocaron punto al azar o puntos de TSPLIB indicándole un radio fijo y un punto en movimiento al azar para observar la rapidez del algoritmo y el trazo de rutas en movimiento.

Se realizo una App en Flutter indicando las coordenadas y haciendo un trazo de los puntos con las API de Google para dibujar el trazo en la vida real de la ruta optima.

Apéndice A

Publicación académica producida

Como producto de esta investigación, el siguiente artículo fue publicado en 5 Congreso Estudiantil de Inteligencia Artificial Aplicada a la ingeniería y Tecnología (CEIAAIT), realizado del 9 a 11 de noviembre del 2022 En la Universidad Nacional Autónoma de México Facultad de Estudios Superiores Cuautitlán Departamento de ingeniería. Folio: 202200422.

TSP dinámico para la toma de decisiones en escenarios de ecommerce.

TSP dinámico para la toma de decisiones en escenarios de e-commerce.

José Adrián Galván García
Departamento de Estudios de
Posgrado e Investigación
Tecnológico Nacional de México,
Instituto Tecnológico de León
León, Guanajuato
M93240319@leon.tecnm.mx

Dr. Juan Martín Carpio Valadez
Departamento de Estudios de
Posgrado e Investigación
Tecnológico Nacional de México,
Instituto Tecnológico de León
León, Guanajuato
juanmartin.carpio@leon.tecnm.mx

Dr. Zamudio Rodríguez Víctor
M.
Departamento de Estudios de
Posgrado e Investigación
Tecnológico Nacional de México,
Instituto Tecnológico de León
León, Guanajuato
vic.zamudio@leon.tecnm.mx

Dr. Carlos Lino Ramírez
Departamento de Estudios de
Posgrado e Investigación
Tecnológico Nacional de México,
Instituto Tecnológico de León
León, Guanajuato
carloslino@leon.tecnm.mx

Dr. Héctor José Puga Soberanes
Departamento de Estudios de
Posgrado e Investigación
Tecnológico Nacional de México,
Instituto Tecnológico de León
León, Guanajuato
pugahector@yahoo.com

Resumen— El problema del vendedor ambulante dinámico (DTSP) en un ambiente experimental es estocástico y dinámico. La capacidad de cambio requiere que el algoritmo 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 el control de una convergencia estable del algoritmo al óptimo y proporciona un buen mecanismo para evitar el estancamiento.

En este trabajo se hace un análisis del algoritmo genético y herramientas para la resolución del problema TSP dinámico, incluyendo las librerías de TSPLIB que indica cual es la ruta óptima para poderla evaluar los algoritmos propuesto como el genético, como estrategia para la solución del problema.

Palabras claves: TSP, DTSP, optimización

I. INTRODUCCION

El problema del agente viajero (TSP) es uno de los problemas más intensamente estudiados en la optimización combinatoria. Dado un conjunto de objetivos, la tarea consiste en determinar un recorrido más corto a cada objetivo de forma precisa y se reencuentra en el inicio. Si la asistencia entre dos objetivos cualesquiera es igual a la distancia euclidiana, el problema se denomina problema del vendedor ambulante euclidiano (ETSP). El problema del vendedor viajero asimétrico (ATSP) es un problema donde la distancia entre dos blancos no es simétrica, pero depende de la dirección del trasversal. Otras variaciones del TSP incluyen el Problema del vendedor ambulante con los vecindarios (TSPN), donde cada objetivo del recorrido puede moverse en una región determinada, el Problema del vendedor ambulante de cuello de botella (BTSP), donde debe ser minimizada la mayor distancia entre los dos objetivos en los itinerarios, y otros.

Los problemas de planificación de rutas enumerados anteriormente tienen como objetivo encontrar los mejores recorridos posibles sin tener en cuenta las características del vehículo. Sin embargo, cuando se trabaja con vehículos del mundo real, hay que tener en cuenta las restricciones cinemáticas, como el radio de giro mínimo. 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. 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. El problema del vendedor ambulante para un vehículo se llama generalmente Problema del vendedor viajero (DTSP) o TSP Curvatura.

El DTSP ha atraído considerable atención debido a muchas aplicaciones civiles y militares. Una configuración típica es monitorear una colección de objetivos distribuidos espacialmente por un vehículo aéreo no tripulado (UAV). Esto podría referirse al control del tráfico en lugares específicos, la recopilación de inteligencia y el reconocimiento de objetivos 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 objetivo y verificación de su destrucción), entre otros. Para otras aplicaciones [8]. 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 de calidad, es aquí cuando los algoritmos heurísticos toman relevancia.

II. METODOLOGIA

Algoritmo genético es utilizar la tecnología de búsqueda de población para resolver la población como un conjunto de problemas y generar una nueva generación de población mediante la aplicación de una serie de operaciones genéticas como la selección, el cruce y la mutación de factores ambientales genéticos biológicos similares a la población actual. Y optimizar gradualmente la población a un estado que contenga la solución óptima aproximada.

El TSP Dinámico (DTSP) fue introducido en 1998 por Psaraftis, 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. Algunos investigadores han trabajado el DTSP con optimización con colonia de hormigas (ACO, Ant Colony Optimization). El DTSP tratado desde un enfoque de optimización de colonia de hormigas tiene problemas con la evaporación de feromonas, incluso en el autor propone una variante del algoritmo ACO con una función de evaporación de feromonas diferente a la tradicional para poder adaptar este algoritmo a DTSP.

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

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

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 la ciudad (nodo) c_i a la ciudad c_j , y t es el tiempo real. En esta definición, el número de ciudades $n(t)$ y el costo de la matriz son tiempos dependientes. El problema dinámico del agente viajero es encontrar el mínimo costo de la ruta que contiene todos los $n(t)$ nodos.

Po dados todos $n(t)$ ($P_1, P_2, \dots, P_{n(t)}$) puntos, y la correspondencia de costo matriz $D = \{d_{ij}(t)\}, i, j = 1, 2, \dots, n(t)$, encontrar un mínimo costo de la ruta que contiene todos los $n(t)$ puntos, donde t se encuentra en el momento del tiempo t se encuentra por la distancia entre el punto del objetivo P_i y el punto objetivo P_j , y $d_{ij}(t) = d_{ji}(t)$.

$$\text{Min}(d(T(t))) = \min \left(\sum_{i=1}^{n(t)} d_{r_i, r_{i+1}}(t) \right) \quad (1)$$

Donde $T \in \{1, 2, \dots, n(t)\}$ if $i \neq j$, then $T_i \neq T_j$, $T_{n(t)+1} = T$
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.

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

$k = 0, 1, 2, \dots, m - 1$, con la ventana de tiempo $[t_k, t_{k+1}]$, donde $\{t_k\}_{k=0}^m$ es una secuencia de puntos de muestreo de tiempo del mundo real.

III. EVALUACION

Los algoritmos de AG son potentes herramientas para resolver problemas de TSP, y se tomara como base para resolver el problema del DTSP con la estrategia de almacenar la última ruta óptima durante su trayecto introducirlo a la población y no forzar el algoritmo y obtener rápido el resultado.

Se utilizo el Algoritmo Genético con elitismo (SGA).

Algoritmo Pseudocódigo de un AG

```

1: /* Parametros de un AG */
2: t=0 // número de generación
3: P (0)
4: Evaluar P (0)
5: While not (Condición de terminación) do
6:   P'(t) = seleccionar(P(t))
7:   P'(t) = aplicar operadores de cruce(P(t))
8:   P(t+1) = reemplazar (P(t), P'(t))
9:   Evaluar P(t+1)
10:  t=t+1
11:  return Mejor Solución Encontrada
12: end while

```

A. Resultados

Nuestro algoritmo en pruebas en TSPLIB. (CPU: Intel core i7 RAM:16GB). La productividad fue evaluada en una

simulación en código Python 3.10.6, librería Turtle (gráficos) para evaluar los experimentos. Los experimentos son evaluados en el algoritmo genético para resolver el TSP dinámico.

Población 50, Probabilidad de cruce 0.9, Probabilidad de mutación 0.5, 2000 generaciones.

TABLE I. EVALUACIÓN DEL TSP, EXPERIMENTOS 33 VECES Y EL RESULTADO ES EL SIGUIENTE

TSPLIB	Evaluación TSP				Tiempo por Con cord
	Distancia Óptima	Distancia óptima (AG)	Diferencia distancia óptima y AG	Generación	
CH150	6528	6528	0	1491	42.7644
Kroc100	21282	21282	0	1963	43.0102
Kroc100	20749	20753	4	2000	45.9550
Lin105	14379	14392	13	2000	43.2640

TABLE II. EVALUACIÓN DEL TSP DINAMICO, EXPERIMENTO DEL RESULTADO ES EL SIGUIENTE

TSPLIB	Evaluación TSP Dinámico			
	Ch150	Kroc100	Kroc100	Lin105
Tiempo de inserción	155.0285	91.1992	93.2149	99.5681
Tiempo de borrado	38.5920	21.9458	20.9421	21.9743

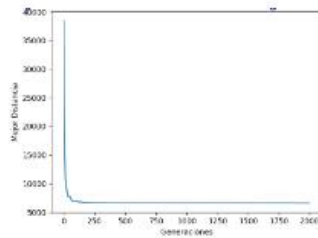


Fig. 1. Mejor distancia contra el número de generaciones

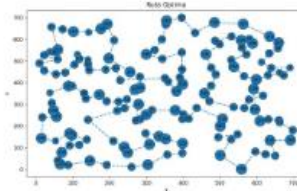


Fig. 2. Tspib ch150.tsp, representa la ruta optima

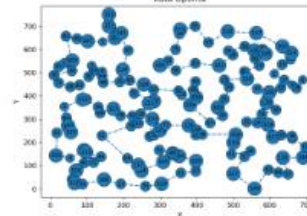


Fig. 3. Añadiendo 2 puntos P1(160,700) y P2(160,750)

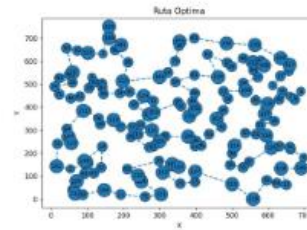


Fig. 4. Borrando los 2 primeros puntos P1(37.43935,541.20906) y P2(612.17595,494.31668)

IV. DISCUSION

Con la experimentación del algoritmo genético al finalizar las generaciones nos muestra la ruta optima y almacenando la ruta optima al momento de añadir o quitar elemento podemos definir que realizamos un menor esfuerzo en obtener la ruta porque tenemos ya una distancia menor que es lo que queremos encontrar , podemos obtener los puntos de las tiendas con su menor costo de los productos que estamos buscando, significa que si estamos buscando productos con distintos precios el precio más bajo nos representaría una

tienda y el recorrido nos mostrará el algoritmo genético con la ruta especificada ya que el comprador puede moverse las tiendas se presentan en un radio y se añaden los puntos en movimiento y se eliminan, se realizó una simulación el Python un radio fijo para tener controlado los puntos y movimientos al azar y se obtiene una respuesta optima de la ruta en movimiento, para que nos represente ya en la simulación se le especifica el tiempo de paro de las interacciones de las generaciones para que nos muestre la ruta optima a visitar las tiendas.

V. CONCLUSIONES Y/O PROYECTOS FUTUROS

En el presente trabajo se presenta el problema del agente viajero en el contexto de e-commerce. Este problema es importante porque nos permite identificar los puntos de venta de un producto que estamos buscando para indicarnos la ruta más corta para poderlo adquirir, ya que el agente se mueve los puntos de venta aparecerán dependiendo de la ruta donde se encuentre y se presenta el problema del agente viajero dinámico.

Se propuso el algoritmo genético con elitismo (SAG) para el TSP y el DTSP.

De la revisión del algoritmo genético podemos concluir que se encuentra la ruta más corta indicándonos los puntos donde se pueden visitar.

VI. 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 al Departamento de Posgrado e Investigación (DEPI) del Instituto Tecnológico de León donde se desarrolló este trabajo.

REFERENCIAS

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

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

[3] Chura, H. E. T., Delgado, C. A. S., Gonzales, E. E. A., & Espinoza, E. F. (2015). Aplicación del algoritmo de colonia de hormigas al problema del agente viajero. *Ciencia & Desarrollo*, (20), 98-102.

[4] Guntisch, M., & Middendorf, M. (2001, April). Pheromone modification strategies for ant algorithms applied to dynamic TSP. In *Workshops on applications of evolutionary computation* (pp. 213-222). Springer, Berlin, Heidelberg.

[5] Guntisch, M., Middendorf, M., & Schmeck, H. (2001, July). An ant colony optimization approach to dynamic TSP. In *Proceedings of the*

3rd annual conference on genetic and evolutionary computation (pp. 860-867).

[6] Restrepo, J. H., & Sánchez, J. J. (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*, 10(26), 121-126.

[7] Riaño, E. R., Toro, G. M. M., & Rico-Bautista, D. (2018). Árbol de caminos mínimos: enrutamiento, algoritmos aproximados y complejidad. *REVISTA COLOMBIANA DE TECNOLOGÍAS DE AVANZADA (RCTA)*, 1(31), 11-21.

[8] S. J. Russell and P. Norvig, *Inteligencia artificial. Un enfoque moderno*. Madrid:Pearson, 2 ed., 2004.

[9] S. Dokania, S. Bagga and R. Sharma, "Opportunistic Self Organizing Migrating Algorithm for real-time Dynamic Traveling Salesman Problem," 2017 51st Annual Conference on Information Sciences and Systems (CISS), 2017, pp. 1-6, doi: 10.1109/CISS.2017.7926065.

[10] Smart Delivery systems .Solving complex vehicle Routing Problems, intelligent Data centric System, Series Fatos Xhafá, Edit by Jakub Nalepa, Ed. Elsevier

[11] Talbi, E.-G. (2009). *Metaheuristics: From design to implementation*. Hoboken, NJ: John Wiley & Sons. Chura, H. E. T., Delgado, C. A. S., Gonzales, E. E. A., & Espinoza, E. F. (2015). Aplicación del algoritmo de colonia de hormigas al problema del agente viajero. *Ciencia & Desarrollo*, (20), 98-102.

[12] Riaño, E. R., Toro, G. M. M., & Rico-Bautista, D. (2018). Árbol de caminos mínimos: enrutamiento, algoritmos aproximados y complejidad. *REVISTA COLOMBIANA DE TECNOLOGÍAS DE AVANZADA (RCTA)*, 1(31), 11-21.

[13] Restrepo, J. H., & Sánchez, J. J. (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*, 10(26), 121-126.

Constancia

UNAM CUAUTITLÁN

Universidad Nacional Autónoma de México
Facultad de Estudios Superiores Cuautitlán
Departamento de Ingeniería

CEIAAIT

Otorgan la presente

Constancia

A: José Adrián Galván

Por su participación como ponente del tema

***TSP dinámico para la toma de decisiones
en escenarios de ecommerce***

En el **5º Congreso Estudiantil de Inteligencia Artificial
Aplicada a la Ingeniería y Tecnología (CEIAAIT)**,
realizado del 9 al 11 de noviembre de 2022.

"POR MI RAZA HABLARÁ EL ESPÍRITU"
Cuautitlán Izcalli, Estado de México, noviembre de 2022.



Folio: 2022/00422

Dr. David Quintanar Guerrero
Director



Asunto: Carta de aceptación

**GALVÁN GARCÍA, JOSÉ ADRIÁN;
JUAN MARTÍN CARPIO VALADEZ;
ZAMUDIO RODRÍGUEZ VÍCTOR M.;
CARLOS LINO RAMÍREZ;
HÉCTOR JOSÉ PUGA SOBERANES
P R E S E N T E S**

Por este conducto informamos que el Comité de Arbitraje **aceptó su artículo** titulado **“Estudio de Optimización del TSP Dinámico para la Planeación de Rutas de Entrega de Mercancía”** para su publicación en la Revista Electrónica de Divulgación de la Investigación del SABES con ISSN 2007 - 3542, en su número 24, edición diciembre de 2022.

Sin otro particular, reciba un cordial y afectuoso saludo.

ATENTAMENTE

“PASIÓN POR TRASCENDER”

MTRA. GEORGINA VALDERÓN SIERRA
DIRECTORA ACADÉMICA

C.c.p. Archivo académico

**SISTEMA AVANZADO DE BACHILLERATO
Y EDUCACIÓN SUPERIOR EN EL ESTADO DE GUANAJUATO**
Blvd. Guanajuato #1615 | León, Gto. México | C.P. 37234 | Tel. (477) 788 5500 y 01 800 00 SABES
www.sabes.edu.mx #sabes.guanajuato SABES_GTO @sabes.gto

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

*Tecnológico Nacional de México, Instituto Tecnológico de León,
Departamento de Estudios de Posgrado e Investigación, León, Gto, México*

Recibido: 10/Octubre/2022
Aceptado: 2/Enero/2023

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 algoritmo Genético y herramientas para la resolución del problema TSP dinámico, incluyendo las librerías de TSPLIB. La librería TSPLIB consiste de un conjunto de instancias de las cuales se conoce la ruta óptima y se utilizan para calibrar el algoritmo Genético básico 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 autoría 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 el 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:

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

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.

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

$k = 0, 1, 2, \dots, m - 1$, 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 óptima 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
1:  * Parámetros de un AG *
2:  t=0 // número de generación
3:  P (0)
4:  Evaluar P (0)
5:  While not (Condición de terminación) do
6:    P'(t) = seleccionar(P(t))
7:    P''(t) = aplicar operadores de cruce(P'(t))
8:    P(t+1) = reemplazar (P(t), P''(t))
9:    Evaluar P(t+1)
10:   t=t+1
11:  return Mejor Solución Encontrada
12: 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 óptima 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 utilizó 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 óptima,

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 óptima que están fuera del nuevo vecindario de selección de puntos.

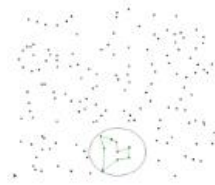


Figura 1: Fuente: Elaboración propia código Python, 900x900, puntos 150, Ch150.lib, radio 100

2.4 TRATAMIENTO

Se programo un algoritmo SGA utilizando el lenguaje Python 3.10.6. Se leen los archivos TSPLIB o se generan 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

$z = 2.5936$, $p\text{-value} = 0.009498$

$0.009498 < 0.05$ (se acepta la hipótesis alternativa)

$\tau = 0.1423525$

3. RESULTADOS.

Los parámetros que se establecieron para el entrenamiento del algoritmo SGA fueron: Población inicial 50, Probabilidad de cruce 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 artículo)	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

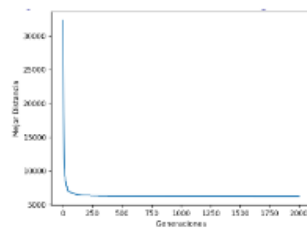


Figura 1: Fuente: Elaboración propia código Python, Mejor distancia contra el número de generaciones

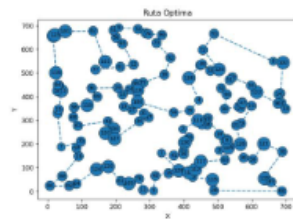


Figura 2: Fuente: Elaboración propia código Python, TSPLIB CH130.TSP, representa la ruta optima

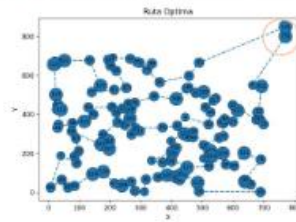


Figura 3: Fuente: Elaboración propia código Python, Añadiendo 2 puntos P1(770,800) y P2(770,850)

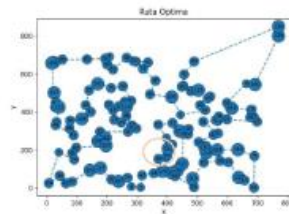


Figura 4: Fuente: Elaboración propia código Python, borrando los 2 primeros puntos

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

4. DISCUSIÓN

Se trabajó con el problema del agente viajero con el contexto de entrega de mercancías. El problema nos permitió 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 óptima. 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 óptima indicándonos los puntos donde pueden entregar la mercancía.

Se logró un desempeño en la generación de la ruta óptima 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 óptima. 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 óptima 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 Paramétrica Aplicada a las ciencias de la conducta*. Mexico: Trillas.
- Adrián Quispe Andía, K. M. (2019). *Estadística no paramétrica 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 ingeniería y tecnología (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*. Elsevier.
- 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*. Elsevier.
- Nalepa, J. (2019). *Smart Delivery Systems , Solving complex vehicle Routing Problems, intelligent Data centric System*. Elsevier.
- 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 annealing 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,.

Referencias

- [1] U. d. (. f. Cádiz, «knuth.uca.es,» TSP, 26 06 2018. [En línea]. Available: <https://knuth.uca.es/moodle/mod/page/view.php?id=4131>. [Último acceso: 15 12 2022].
- [2] S. G. Ugás, «Warketing,» 1 Abril 2014. [En línea]. Available: <https://warketing.cl/articulos/aplicacion-de-e-commerce-en-las-empresas/>. [Último acceso: 15 12 2022].
- [3] A. I. B. Boubeta, «Distribución logística y comercial,» de *La logica de la empresa*, España, Ideas propias VIGO, 2007, p. 77.
- [4] J. K. L. A. H. G. R. K. D. B. S. E. L. Lawler, «The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization,» de *A guided tour of combinatorial optimization*, California, wiley, 1985, p. 476.
- [5] J. J. P. F. abraham Duarte Muñoz, *Metahuríticas*, España: DYKINSON, 2007.
- [6] R. E. Armenta, «Matemáticas discretas,» Mexico, Alfaomega, 2010, p. 492.
- [7] A. M. L. G. M. A. M. VIEITES RODRÍGUEZ, «Teoría de grafos. Ejercicios y problemas resueltos.,» Paraninfo, p. 148.
- [8] H. R. Eduardo Moreno, *Grafos: fundamentos y algoritmos*, Chile: JCSAEZ, 2011.
- [9] D. Jungnickel, *GRaph, Networks and Algorithms*, German: Springer, 2000.
- [10] B. Heinold, «A Simple Introduction to Graph Theory,» 2022, p. 135.
- [11] J. A. J. Murillo, *MATemáticas para la computación*, México: Alafaomega, 2009.
- [12] D. R. Cachimuel-Iza, «Proceso de diseño y planificación de rutas de transporte para mejorar los tiempos de entrega,» *Polo del conocimiento*, nº DOI: 10.23857/pc.v7i4.3806, p. 19, Abril 2022.

- [13] J. Sáez Aguado, «Problemas de rutas de vehículos: modelos, aplicaciones logísticas y métodos de resolución,» Universidad de Valladolid, Valladolid, 2015.
- [14] C. D. A. R. Wallace, «Selección natural: tres fragmentos para la historia,» España, CSIC, 2009.
- [15] J. F. Nonidez, «La herencia mendeliana: introducción al estudio de la genética,» La herencia mendeliana: introducción al estudio de la genética, Junta para Ampliación de Estudios e Investigaciones Científicas, 1922, 2008, p. 271.
- [16] A. S. Fraser, «Simulation of Genetic Systems by Automatic Digital Computers I.,» *Introduction. Australian Journal of Biological Sciences*, vol. 10, p. 484–491, 1957.
- [17] A. S. Fraser, «Simulation of Genetic Systems by Automatic Digital Computers II. Effects of Linkage on Rates of Advance Under Selection,» *Australian Journal of*, vol. 10, p. 492–499, 1957.
- [18] A. S. Fraser, «Simulation of Genetic Systems by Automatic Digital Computers VI.,» *Australian Journal of Biological Sciences*, vol. 10, p. 150–162, 1960.
- [19] C. R. D. Serna, *Nuevas Técnicas metaheurísticas: Aplicación al transporte escolar*, Universidad de Burgos: Universidad de Burgos, 2002.
- [20] P. J. (1996), «The Vehicle Routing Problem with Time Windows—Part II: Genetic Search.,» *The Vehicle Routing Problem with Time Windows—Part II: Genetic Search.*, nº <http://dx.doi.org/10.1287/ijoc.8.2.165>, p. 8, 1996.
- [21] I. Oliver, D. Smith y J. Holland, «study of permutation crossover operators on the traveling salesman problem,» p. 6, 1987.
- [22] D. Oliva, «Modelo de filtros IIR usando un algoritmo inspirado en el electromagnetismo,» *ResearchGate*, nº DOI: 10.1016/S1405-7743(13)72231-5, p. 15, 2013.
- [23] M. M. M. & S. H. Guntsch, «An ant colony optimization approach to dynamic TSP,» *In Proceedings of the 3rd annual conference on genetic and evolutionary computation*, pp. 860–867, 2001, July.
- [24] A. E. O. Olorunda, «Measuring exploration/exploitation in particle swarms using swarm diversity,» *Computer Science, Geology*, nº DOI:10.1109/CEC.2008.4630938, 2008.
- [25] G. D. T.-D. W. P. A. Whigham, *Simulated Evolution and Learning 6th International Conference*, Hefei, China, : Springer, October 15–18, 2006.
- [26] R. H. Ballou, *Logística empresarial: control y planificación*, Díaz de Santos, 1991.

- [27] G. B. D. a. J. H. Ramser, "The Truck Dispatching Problem" in *plactics*, New York: McGraw-Hill, 1964, p. 2.
- [28] H. E. T. D. C. A. S. G. E. E. A. & E. E. F. Chura, «Aplicación del algoritmo de colonia de hormigas al problema del agente viajero.,» *Ciencia & Desarrollo*, pp. 98-102, (2015).
- [29] U. Heidelberg, «<http://comopt.ifi.uni-heidelberg.de/>,» [En línea]. Available: <http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp/>.
- [30] S. S. A. Jay, *Estadística No Paramétrica Aplicada a las ciencias de la conducta*, Mexico: Trillas, 1998.
- [31] «turtle — Gráficos con Turtle — documentación de Python - 3.9.14. (n.d.),» [En línea]. Available: <https://docs.python.org/es/3.9/library/turtle.html>. [Último acceso: 12 4 2022].
- [32] R. v. Alejandro, *Grafos: software para la construcción, edición y análisis de grafos*, Bubok Publishing S.L, 2017.
- [33] R. Pérez Aguila, *Una introducción a las matemáticas discretas y teoría de grafos*, El Cid Editor, 2013.
- [34] L. Babel, *New heuristic algorithms for the Dubins traveling salesman problem.*, *Journal of Heuristics*, 1-28., 2020.
- [35] J. Nalepa, *Smart Delivery Systems , Solving complex vehicle Routing Problems*, intelligent Data centric System, Elsevier, 2019.
- [36] J. H. S. J. J. Restrepo, «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*, pp. 121-126, 2004.
- [37] E. R. T. G. M. M. & R.-B. Riaño, «Árbol de caminos mínimos: enrutamiento, algoritmos aproximados y complejidad.,» *REVISTA COLOMBIANA DE TECNOLOGIAS DE AVANZADA*, pp. 11-21, 2018.
- [38] S. B. a. R. S. S. Dokania, «Opportunistic Self Organizing Migrating Algorithm for real-time Dynamic Traveling Salesman Problem,» *2017 51st Annual Conference on Information Sciences and Systems (CISS)*, pp. pp. 1-6, 2017.
- [39] M. & M. M. Guntsch, «Pheromone modification strategies for ant algorithms applied to dynamic TSP. In *Workshops on applications of evolutionary computation,*» *Springer*, pp. pp. 213-222, 2001.

- [40] A. A. N. C. J. G. B. P. E. Otto, Optimization approaches for civil applications of unmanned aerial vehicles (UAVs) or aerial drones: a survey, <https://doi.org/10.1002/net.21818>, 2018.
- [41] S. LaValle, Planning Algorithms. Cambridge University Press, Cambridge, 2006.
- [42] W. A. S. , N. L. G. C. María Alejandra Enciso Caicedo, «MODELO DE RUTEO DE VEHÍCULOS COMO ALTERNATIVA DE TRANSPORTE, ESTUDIO DE CASO: UMNG SEDE CAMPUS,» *Revista Politécnica*, p. 12, 2018.
- [43] A. & Z. & D. Baykasoglu, «Agent-based solution approaches for dynamic traveling salesman problem: Resolving or adapting existing solutions to new conditions?,» *Journal of Multiple-Valued Logic and Soft Computing.*, nº 359-378, p. 31, 2018.
- [44] U. & S. Ł. Boryczka, «Diversification and entropy improvement on the DPSO algorithm for DTSP. In Asian Conference on Intelligent Information and Database Systems,» *Springer*, pp. 337-347, 2015, March.
- [45] D. F. A. M. M. S. C. Archetti, «Dynamic traveling salesman problem with stochastic release dates,» *European Journal of Operational Research*, vol. 280, nº ISSN 0377-2217, pp. 832-844, 2020.
- [46] H. E. T. D. C. A. S. G. E. E. A. & E. E. F. Chura, «Aplicación del algoritmo de colonia de hormigas al problema del agente viajero.,» *Ciencia & Desarrollo*, vol. 20, pp. 98-102, 2015.
- [47] R. E. B. V. C. a. W. J. C. David L. Applegate, «The travelling salesman problem,» *Princeton Series in Applied Mathematics*, vol. 17.
- [48] M. & M. M. Guntsch, Pheromone modification strategies for ant algorithms applied to dynamic TSP In Workshops on applications of evolutionary computation, Berlin: Springer, 2001, April, pp. 213-222.
- [49] A. Z. B. M. Y. L. a. Z. K. Lishan Kang, «Benchmarking algorithms for dynamic travelling salesman problems,» *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753)*, vol. 2, nº Doi: 0.1109/CEC.2004.1331045, pp. 1286-1292, 2004.
- [50] S. LaValle, Planning Algorithms., Cambridge: Cambridge University Press, 2006.
- [51] A. A. Otto, «Optimization approaches for civil applications of unmanned aerial vehicles (UAVs) or aerial drones,» *survey*, nº <https://doi.org/10.1002/net.21818>., 2018.
- [52] J. H. & S. J. J. Restrepo, «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 technical*, vol. 10, pp. 121-126, 2004.

- [53] E. R. T. G. M. M. & R.-B. D. Riaño, «Árbol de caminos mínimos: enrutamiento, algoritmos aproximados y complejidad,» *REVISTA COLOMBIANA DE TECNOLOGIAS DE AVANZADA (RCTA)*, vol. 1, nº 31, pp. 11-21, 2018.
- [54] S. B. a. R. S. S. Dokania, «Opportunistic Self Organizing Migrating Algorithm for real-time Dynamic Traveling Salesman Problem,» *2017 51st Annual Conference on Information Sciences and Systems (CISS)*, nº Doi: 10.1109/CISS.2017.7926065, pp. 1-6, 2017.
- [55] S. B. a. R. S. S. Dokania, «Opportunistic Self Organizing Migrating Algorithm for real-time Dynamic Traveling Salesman Problem,» *2017 51st Annual Conference on Information Sciences and Systems (CISS)*, nº 10.1109/CISS.2017.7926065, pp. 1-6, 2017.
- [56] N. J. W. & S. C. H. E. T. D. C. A. S. G. E. E. A. & E. Hoboken, *Metaheuristics: From design to implementation*, Talbi, 2015.
- [57] W. cook, *In Pursuit of the Traveling Salesman: Mathematics at the Limits of Computation* 1st Edition, Princeton University: Princeton University Press, 16 Enero 2012.
- [58] J. S. A.E. Eiben, *Introduction to Evolutionary Computing*, Natural Computing Series, Springer, Second Edition.
- [59] t. ones, *Crossover, macromutation, and population-based search*, San Mateo: Morgan Kaufmann, 1995.
- [60] H. R. E. Moreno, *Grafos: fundamentos y algoritmos*, Chile: JCSAEZ, 2011.
- [61] D. Jungnickel, *GGraph, Networks and Algorithms*, German: Springer, 2000.

