VNiVERSiDAD D SALAMANCA

FACVLTAD DE CIENCIAS

TRABAJO FIN DE GRADO EN ESTADÍSTICA

Análisis del Problema del Viajante: Algoritmo del Vecino más Cercano (NN) versus Algoritmo Genético (GA)

Autor: Álvaro González Cáceres
Tutor: José-Ángel Domínguez Pérez

Curso Académico 2023 – 2024.

UNIVERSIDAD DE SALAMANCA

FACULTAD DE CIENCIAS

TRABAJO FIN DE GRADO EN ESTADÍSTICA

Análisis del Problema del Viajante: algoritmo del Vecino más Cercano (NN) versus Algoritmo Genético (GA)

Salamanca, a 29 de Julio de 2024
El Autor:

Firmado: Álvaro González Cáceres

UNIVERSIDAD DE SALAMANCA

FACULTAD DE CIENCIAS

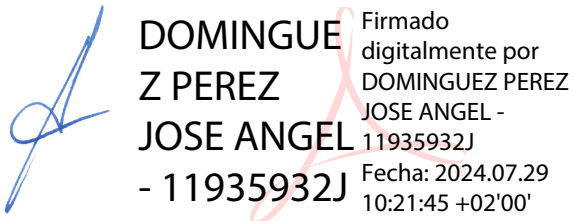
DOCUMENTO ACREDITATIVO DE PRESENTACIÓN DEL TRABAJO FIN DE GRADO EN ESTADÍSTICA CERTIFICADO DEL TUTOR

D. José-Ángel Domínguez Pérez, profesor del Departamento de Matemáticas de la Universidad de Salamanca,

HACE CONSTAR:

Que el trabajo titulado "Análisis del Problema del Viajante: algoritmo del Vecino más Cercano (NN) versus Algoritmo Genético (GA)" que se presenta ha sido realizado por Álvaro González Cáceres, con DNI 45133678L y constituye la memoria del trabajo realizado para la superación de la asignatura Trabajo Fin de Grado en Estadística de esta Universidad.

Salamanca, a 29 de Julio de 2024
El Tutor:



Firmado: José-Ángel Domínguez Pérez

RESUMEN

El presente trabajo analiza el Problema del Viajante (TSP, por sus siglas en inglés) a través del estudio de dos algoritmos diferentes para su resolución: el algoritmo heurístico del Vecino más Cercano (NN) y el algoritmo metaheurístico Genético (GA). Se desarrollan implementaciones de ambos algoritmos en el programa de RStudio y se aplican a tres problemas específicos: uno de elaboración propia (las nuevas 7 maravillas del mundo) y dos estudios previos (problemas de 15 y 20 ciudades). Se compararon los resultados obtenidos en términos de distancia mínima alcanzada y el tiempo de cómputo. Los resultados indican que el algoritmo NN, aunque es simple y rápido, es dependiente de la ciudad de inicio y de decisiones parciales, mientras el algoritmo genético GA, aunque es más complejo y tiene un mayor tiempo de cómputo, aborda soluciones globales y ofrece soluciones más optimizadas dependiendo de la configuración de sus operadores.

PALABRAS CLAVE

“Problema del Viajante (TSP)”, “Algoritmo del Vecino más Cercano (NN)”, “Algoritmo Genético (GA)”, “Optimización combinatoria”, “Heurística” y “Metaheurística”.

ABSTRACT

The present work analyses the Traveling Salesman Problem (TSP) through the study of two different algorithms for its resolution: the heuristic Nearest Neighbor Algorithm (NN) and the metaheuristic Genetic Algorithm (GA). Implementations for both algorithms were developed in RStudio and applied to three specific problems: one our own elaboration (the new 7 Wonders of the World) and two previous studies (problems of 15 and 20 cities). The obtained results were compared in terms of the minimum distance achieved and computation time. The results indicate that the NN algorithm, although simple and fast, is dependent on the starting city and partial decisions, while the GA, although more complex and having a longer computation time, deals with global solutions and offers more optimized solutions depending on the configuration of its operators.

KEYWORDS

"Traveling Salesman Problem (TSP)", "Nearest Neighbor Algorithm (NN)", "Genetic Algorithm (GA)", "Combinatorial Optimization", "Heuristic", "Metaheuristic".

ÍNDICE

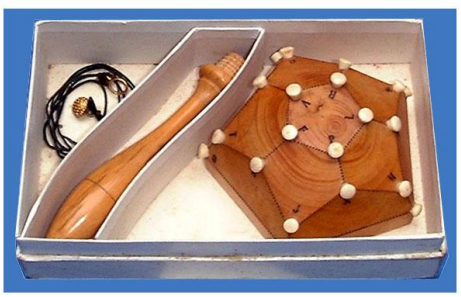
1. INTRODUCCIÓN

1.1. Síntesis del estado de la cuestión

El Problema del Viajante (TSP por sus siglas en inglés, Traveling Salesman Problem) es uno de los problemas más estudiados en el ámbito de optimización combinatoria. La cuestión que plantea es encontrar la "ruta óptima" que un viajante debe seguir para visitar un conjunto de ciudades conectadas entre sí por una red de carreteras, entendiendo como tal ruta óptima una selección de carreteras que, comenzando en una ciudad de origen, va pasando por todas y cada una de las ciudades una sola vez y regresa a esa ciudad de origen, todo ello minimizando la distancia total que suman las carreteras que componen la ruta.

El TSP tiene una clara modelización en términos de Teoría de Grafos, donde las ciudades a visitar son los vértices de un grafo y las carreteras que las conectan, con sus distancias, son las aristas con sus pesos. De este modo, la resolución del TSP puede abordarse mediante métodos exactos, como por ejemplo la enumeración exhaustica de todas las posibles rutas (fuerza bruta) para seleccionar la de menor distancia total, pero la complejidad factorial de estos métodos hace que cuando se aborda el problema sobre un número grande de ciudades, la resolución exacta es impracticable. La alternativa es acudir a métodos aproximados o heurísticos, que buscan rutas óptimas para sub-grafos locales (en el entorno de cada ciudad), o una combinación metaherústica de diversos métodos, que permitan alcanzar una solución que se aproxime lo más posible a la ruta óptima.

Encontrar estos métodos y hacerlos más eficientes, formulándolos en términos de algoritmos que sigan una serie de pasos preestablecidos hasta alcanzar la solución al TSP, ha venido impulsado el desarrollo de la Teoría de Grafos desde el siglo XIX, cuando en 1856 el matemático inglés Thomas Kirkman identificó las rutas óptimas con "ciclos hamiltonianos", denominados así en honor de Sir William Rowan Hamilton, quien desarrolló el "The Icosian Game", un juego de ingenio basado en encontrar esos ciclos /rutas óptimas entre las ciudades / vértices un dodecaedro regular, viajando a través de las carreteras / aristas.



Actualmente la búsqueda de algoritmos que resuelvan el TSP de manera más eficiente sigue siendo un prolífico ámbito de investigación, en el que se combinan diveras técnicas científicas.

1.2. Objetivos del trabajo

El presente trabajo tiene como objetivo general analizar y comparar dos algoritmos utilizados para resolver el TSP: un método heurístico constructivo, el algoritmo del Vecino más Cercano (NN) y un método metaheurístico de problación, el Algoritmo Genético (GA), para concluir las ventajas e inconvenientes de aplicación de uno u otro algoritmo.

Este objetivo se desglosa en los siguientes objetivos específicos:

- Estudiar y describir los pasos a ejecutar en ambos algoritmos, aplicarlos a problemas específicos tomados como ejemplo.
- Implementar los algoritmos NN y GA en un programa informático, para facilitar su aplicación.
- Evaluar y comparar la eficiencia de los algoritmos sobre problemas específicos, en términos de optimalidad de la distancia mínima alcanzada (eficacia) y tiempo de cómputo, como medida de su complejidad (eficiencia).
- Identificar los elementos que condicionan la eficacia y eficiencia de cada algoritmo en según qué tipo de problemas concretos.
- Comparar los resultados obtenidos con los reportados en estudios previos publicados en revista científicas.

1.3. Estructura del trabajo

El trabajo se organiza de la siguiente forma:

- Capítulo 2: Se presenta una descripción detallada del Problema del Viajante y se explican los métodos de resolución, tanto exactos como heurísticos y metaheurísticos.
- Capítulo 3: Se analizarán en profundidad los algoritmos del Vecino más cercano (NN) y Genético (GA), incluyendo su implementación y aplicación a ejemplos concretos.
- Capítulo 4: Se realiza una comparación de los resultados obtenidos con los algoritmos NN y GA frente a otros estudios sobre problemas similares.
- Capítulo 5: Se presentan las conclusiones del trabajo, destacando las principales aportaciones y resultados obtenidos.

Finalmente se incluye la bibliografía que se ha utilizado como referencia para el trabajo, así como los anexos con los programas informáticos elaborados.

2. ALGORITMOS DE OPTIMIZACIÓN APLICADOS AL PROBLEMA DEL VIAJANTE

2.1. El problema del viajante

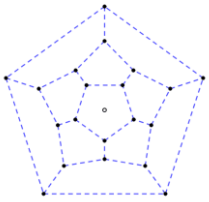
El problema del viajante, también conocido como TSP por sus siglas en inglés (Travel Salesman Problem), plantea encontrar el orden en que un viajero debe visitar un conjunto de ciudades conocidas, situadas unas de otras a unas determinadas distancias, para conseguir la "ruta más eficiente", lo que se concreta en: empezar y terminar en la misma ciudad, pasar por todas las ciudades una sola vez, y que la distancia total recorrida sea mínima.

En términos de Teoría de Grafos, donde las ciudades se representan como los vértices de un grafo y las distancias entre dos ciudades se representan como aristas de un determinado peso (la distancia) entre los vértices, el TSP consiste en encontrar un ciclo hamiltoniano de distancia mínima.

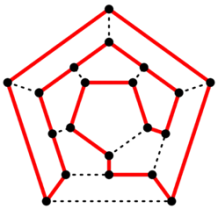
La resolución de este problema por "fuerza bruta", calculando los ciclos posibles y sus distancias, para elegir el ciclo de distancia mínima, para n ciudades supone calcular $(n-1)! = (n-1).(n-2) \dots 2 .1$ ciclos, teniendo en cuenta que el conjunto de todos los ciclos se corresponde con las permutaciones de las n ciudades, que son $n!$, y que en ese conjunto hay subconjuntos de n ciclos iguales (dado un ciclo, es igual a todos los ciclos generados empezando en cualquiera de sus vértices). Este cómputo supone una gran complejidad: por ejemplo, tomando las 52 ciudades capitales de provincia españolas, $51!$ es del orden 10^{67}, mucho mayor que el número de átomos sobre la Tierra, que es del orden de 10^{50}.

De ahí que el problema de encontrar técnicas exactas o aproximadas más eficientes para resolver el Problema del Viajante (TSP) haya constituido un reto para científicos de diversas disciplinas, impulsando el desarrollo de la teoría de grafos desde el siglo XIX.

En particular, en 1856, el matemático inglés Thomas Kirkman fue el primero en explorar los ciclos hamiltonianos en un contexto general, estableciendo condiciones para su existencia en grafos poliédricos. Sir William Rowan Hamilton descubrió en ese mismo año "The Icosian Calculus", un álgebra no conmutativa cuyos elementos representan acciones o posibles movimientos sobre los vértices de un dodecaedro regular. En este contexto, Hamilton planteó la pregunta de encontrar ciclos hamiltonianos dentro de ciertas restricciones. Hamilton desarrolló "The Icosian Game", un juego de ingenio que exigía a los jugadores encontrar un camino con características particulares en el grafo del dodecaedro regular. [VH, 2019]



Juego icosiano sin resolver.



Una solución al juego icosiano

Merrill Flood desempeñó un papel importante al popularizar el término TSP en la década de 1930 [ID, 2018], vinculándolo con la Teoría de los Juegos y estableciendo su relación con los problemas como la asignación y el transporte.

En el año 1954, George Dantzig, Ray Fulkerson y Selmer Johnson, estos formaban parte del equipo de matemáticos de la RAND Corporation, introdujeron uno de los primeros algoritmos para abordar el problema: el método de los planos cortantes. Este método se lleva a cabo en la programación lineal entera. [VH, 2019]

Después, los tres matemáticos: Cook en 1971, Karp en 1972 y Lewis en 1973, realizaron investigaciones sobre la complejidad de resolver una serie de problemas conocidos como NP-hard o non-deterministic polynomial time problems. Estos problemas son caracterizados por ser difíciles de resolver de forma computacional porque no se ha demostrado que exista un algoritmo con una complejidad polinomial que pueda garantizar una solución óptima. Después de sus investigaciones, se descubrió que estos problemas eran equivalentes, ya que un algoritmo polinomial que pudiese resolver uno de ellos también podría resolver los demás. Además, se demostró que algunos casos particulares de TSP eran NP-hard, como el problema de TSP euclídeo. [VH, 2019]

En las décadas siguientes, se lograron avances significativos tanto en los algoritmos que proporcionan una solución exacta como en los que facilitan una resolución aproximada o heurística al TSP, y en el año 1990, Vasek Chvátal y William J. Cook, con la colaboración de David Applegat y Robrt Bixby, lograron crear un software conocido como Concorde. Este programa fue capaz de resolver el problema más grande hasta ese momento, el cual estaba relacionado con la optimización de circuitos integrados. [MT, 2020]

La resolución eficiente del TSP como problema de optimización combinatoria tiene aplicaciones prácticas en campos como la logística, planificación de rutas y optimización de recursos en diversos contextos.

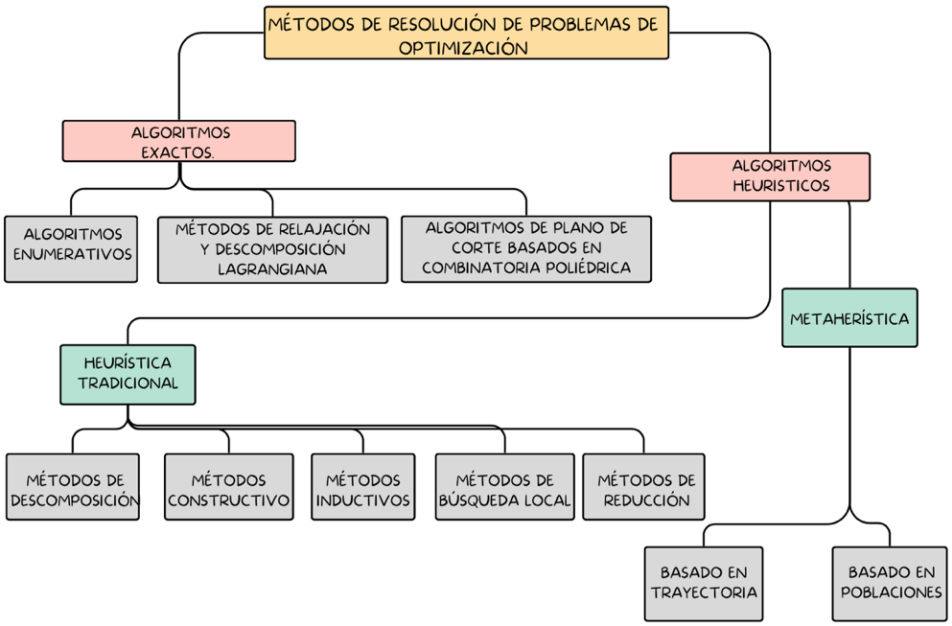
1.2. Métodos de resolución del problema

Los algoritmos exactos de resolución del TSP garantizan la obtención de la solución óptima, pero enfrentan desafíos en términos de complejidad computacional, necesitando grandes cantidades de tiempo (que puede resultar inabordable) para alcanzar esa solución, especialmente cuando las soluciones factibles no presenta propiedades convexas. Estos algoritmos abordarán estrategias como la enumeración, la relajación y descomposición lagrangiana y los planos de corte, que destacan por su aplicación en la resolución precisa de problemas de optimización combinatoria.

También tenemos los algoritmos heurísticos, estos son fundamentales en el ámbito de la investigación de operaciones, ofrecen soluciones efectivas, no son tan precisas para problemas considerados como complejos para obtener soluciones optimas exactas. Los algoritmos heurísticos se clasifican en tradicionales y metaheurísticos, resalta la capacidad para proporcionar soluciones cercanas a la óptima y mejorar la eficiencia de los algoritmos exactos en un tiempo accesible. A su vez, en los métodos heurísticos

tradicionales se establecen categorías como descomposición, constructivos, inductivos, búsqueda local, reducción, métodos constructivos y búsqueda local. Estos métodos tradicionales serán contrastados con los metaheurísticos, que combinan diversas estrategias heurísticas para abordar el problema del viajante de manera más eficaz.

Esta clasificación se resume en el siguiente gráfico [MB, 2015]:



2.2.1. Algoritmos exactos [H, 2000]

Para resolver problemas de optimización combinatoria es necesario encontrar la mejor solución posible entre el conjunto finito de opciones. En cambio, en la programación lineal, las soluciones factibles forman un conjunto convexo, en el caso combinatorio, esta región no tiene esa propiedad, por ello se complica la búsqueda de la solución óptima. En este entorno, los algoritmos exactos desempeñan un papel muy importante.

Los algoritmos exactos son métodos que garantizan encontrar la solución óptima a un problema dado. En la programación entera, las variables toman valores enteros, los algoritmos exactos se enfrentan a la tarea de explorar una red de puntos factibles o, en caso de variables enteras, conjuntos de semirrectas o segmentos de línea disjuntos. Por otro lado, en la programación lineal, la convexidad facilita la identificación de soluciones globales a partir de soluciones locales, en la programación entera, la presencia de múltiples óptimos locales requiere enfoques que demuestren la dominancia de una solución específica sobre todas las soluciones factibles, utilizando distintos argumentos de los derivados basados en cálculos de programación convexa.

Existen diferentes estrategias para tratar problemas de programación entera, actualmente, es común combinarlas en soluciones “hibridas” para aprovechar sus respectivas ventajas. En esta situación, son elementos muy importantes las técnicas enumerativas, los enfoques de relajación y descomposición, y los planos de corte. Las

técnicas enumerativas implican explorar exhaustivamente todas las soluciones posibles, por otro lado, los enfoques de relajación buscan soluciones aproximadas relajando restricciones. La descomposición involucra dividir el problema en subproblemas más manejables, y los planos de corte son estrategias para eliminar soluciones subóptimas.

La combinación efectiva de estas estrategias ha demostrado ser exitosa en la resolución de problemas que son extremadamente difíciles en el ámbito de la optimización combinatoria, para los casos complejos se proporcionan soluciones precisas y optimas. Este enfoque es esencial en situaciones donde la exactitud de la solución es fundamental y las soluciones aproximadas no son suficientes.

Algoritmos enumerativos [H, 2000]

Los algoritmos enumerativos, son estrategias fundamentales para tratar los problemas de optimización combinatoria, sobre todo en el enfoque de ramificación y poda. Este método implica explorar de forma completa todas las posibles combinaciones de soluciones y seleccionar la que optimiza la función objetivo. Sin embargo, la ejecución directa de esta enumeración puede ser computacionalmente costosa, especialmente cuando el número de elementos a considerar es grande, como en el caso de visitar ciudades en un problema de rutas.

En la ramificación y poda, se organiza un árbol de expansión donde cada nodo representa una parte de la solución hasta cierto punto. Por ejemplo, en el problema de rutas, cada nodo puede representar una etapa de la ruta visitada. Los nodos se pueden sondear si la solución al subproblema no es factible, satisface todas las restricciones de integridad o tiene un valor de función objetivo peor que una solución entera conocida. Los hijos de cada nodo son las posibles extensiones al agregar una nueva etapa al problema. Este proceso de expansión se realiza de manera parcial, y en cada paso, se calcula una cota para evaluar la calidad de la solución potencial.

La poda es una fase crítica en este método, donde se descartan ramas del árbol que no pueden producir soluciones óptimas. La cota se utiliza para determinar si la extensión propuesta puede superar la mejor solución encontrada hasta el momento. Si la cota indica que la rama no llegara a una solución mejor, se evita explorarla, esto reduce notablemente el tiempo de ejecución.

El algoritmo tiene tres etapas principales: la primera etapa es la selección del nodo entre el conjunto de nodos vivos (no descartados), la segunda etapa es la ramificación para generar nuevas extensiones (ramas) y por la última etapa es la poda para descartar soluciones no viables (que se alejan de la óptima). La ejecución sigue hasta que se acaba el conjunto de nodos vivos, lo que garantiza que la solución que queda sea la mejor posible dadas las restricciones del problema.

El enfoque de ramificación y poda es muy eficiente para resolver problemas combinatorios, aunque es compleja. Permite explorar el espacio de posibles soluciones

de manera estructurada, descartando opciones no viables y enfocándose en aquellas que tienen capacidad para ser óptima.

Métodos de relajación y descomposición lagrangiana [H, 2000]

La forma de reducir la restricción de la integralidad (exigencia de soluciones con valores enteros) no es la única para abordar problemas complejos. Una variante para resolver problemas de programación entera es emplear la denominada relajación lagrangiana, esta incorpora un conjunto de restricciones que son “complicadas” en la función objetivo convertida a esa forma lagrangiana. Este método consiste en la aproximación de la solución del problema difícil resolviendo un problema más simple, que se obtiene al eliminar una o varias restricciones del problema original e incorporarlas en la función objetivo como penalizaciones, donde el peso está dado por el denominado multiplicador de Lagrange o lagrangiano. Si lo llevamos a la práctica, este problema suele ser más manejable que el problema original.

Es esencial comprender la estructura del problema para aplicar la relajación lagrangiana, ya que requiere relajar restricciones que dificultan la solución. Un enfoque relacionado es la descomposición lagrangiana, que consiste en aislar conjuntos de restricciones para resolver problemas separados más simples. La introducción de variables de enlace, que conectan estos subconjuntos, aumenta la complejidad del problema. Todas las estrategias lagrangianas dependen de la estructura del problema, y no existe una teoría general subyacente desarrollada.

La mayoría de las estrategias basadas en lagrangiano se centran en estructuras de los datos del problema en filas especiales, mientras que otros problemas pueden tener una estructura de columnas especial. Uno de estos algoritmos, el de descomposición de Benders, corrige variables complicadas y resuelve el problema de manera iterativa, buscando un plano de corte basado en el dual asociado al problema. Este recorte se agrega a las desigualdades, y el proceso se repite.

En la actualidad, hay investigaciones en algoritmos para resolver problemas de programación semidefinida continua, que consideran una relajación semidefinida del problema de optimización combinatoria. Dado que los enfoques de descomposición proporcionan límites en la solución entera, se pueden incorporar en un algoritmo de ramificación y poda en lugar de la relajación de programación lineal más común. Sin embargo, estos algoritmos son específicos para problemas que aprovechan el "patrón de restricción" o la estructura especial del problema.

Algoritmos de plano de corte basados en combinatoria poliédrica [H, 2000]

Los algoritmos de planos de corte basados en combinatoria poliédrica han experimentado significativos avances computacionales en la optimización exacta. Estos avances han permitido resolver problemas de mayor tamaño y complejidad al aplicar la teoría poliédrica desarrollada en las últimas dos décadas a la resolución numérica de problemas.

La combinatoria poliédrica propone reemplazar el conjunto de restricciones de un problema de programación entera por una convexidad alternativa de los puntos factibles y las rayas extremas del problema, conv(S). Esta idea se basa en el teorema de Weyl, establecido en 1935, que define un poliedro convexo como la intersección de finitos semiespacios o como la envolvente convexa más el cono convexo de un número finito de vectores o puntos.

El algoritmo de "planos de corte" de Gomory para problemas de programación entera, desarrollado como una prueba constructiva del teorema de Weyl, converge a una solución óptima en un número finito de pasos. Sin embargo, la convergencia es extraordinariamente lenta debido a que los cortes algebraicos generados son "débiles" y pueden causar problemas de condicionamiento en la matriz de base.

La aplicación de técnicas de elevación ha permitido hacer válidos los cortes de Gomory en todo el árbol de enumeración. La generación de cortes basados en teoría poliédrica ha tenido éxito en problemas de tamaños antes considerados intratables.

En un enfoque general de planos de corte, se relajan las restricciones de integralidad en una primera etapa y se resuelve el programa lineal resultante sobre el conjunto relajado S'. Si el programa lineal es ilimitado o, lo es también el programa entero. Si la solución al programa lineal es entera, se ha resuelto el problema entero. En caso contrario, se aborda un problema de identificación de facetas para encontrar una desigualdad lineal que "corte" la solución fraccional, separando el punto fraccional del poliedro conv(S).

La teoría poliédrica busca identificar sistemas mínimos de desigualdades lineales que definen facetas del poliedro conv(S). Dado que, para la mayoría de los problemas interesantes de programación entera, el número mínimo de desigualdades necesarias es exponencial en el número de variables, surge la pregunta de si este enfoque es computacionalmente práctico. Sorprendentemente, la implementación de algoritmos de planos de corte basados en esta teoría ha tenido éxito en la resolución de problemas de tamaños considerados intratables.

La eficacia radica en la búsqueda de la optimalidad de un único punto extremo de conv(S). Así, se requiere solo una descripción parcial de F en el vecindario de la solución óptima. La generación de cortes fuertes se aborda como un problema de identificación de facetas, determinando los coeficientes de un hiperplano separador que maximice la distancia entre este y el punto fraccional.

El uso de cortes poliédricos, disyuntivos y técnicas de "*lift and project*" ha sido exitoso para obtener formulaciones más ajustadas en problemas de optimización combinatoria. La terminación de un algoritmo de planos de corte puede deberse a la obtención de una solución entera, la imposibilidad del programa lineal, la falta de identificación de cortes o la insuficiente mejora en la función objetivo.

Los planos de corte pueden emplearse como técnica de reformulación y, en particular, se destacan cuando se incorporan en un algoritmo de ramificación y acotación,

conocido como "*branch and cut*". La efectividad de dicho algoritmo depende de la fortaleza de los argumentos de acotación y de la generación de cortes, proporcionando una solución óptima cuando el límite inferior coincide con el límite superior.

2.2.2. Algoritmos heurísticos y metaheurísticos [H, 2000]

Para encontrar soluciones efectivas a problemas que se consideraban demasiado complejos para obtener soluciones óptimas, los analistas de investigación de operaciones han utilizado de manera frecuente los algoritmos heurísticos. Estos algoritmos han experimentado un gran cambio en los últimos años. Los algoritmos heurísticos se utilizan de forma frecuente en programas informáticos comerciales de optimización de rutas, no solo para proporcionar soluciones efectivas, sino también para mejorar la eficiencia de los algoritmos precisos al establecer límites tempranos efectivos en el procedimiento.

El inicio para la investigación en algoritmos heurísticos viene dado por los conceptos de la búsqueda local, en ellos se crea una solución viable y se mejora iterativamente mediante movimientos locales, que buscan posibles soluciones cercanas a la solución viable. Se propusieron algoritmos constructivos de este tipo, como los algoritmos de redondeo, para las soluciones de problemas de programación lineal. En la década de 1980 se hicieron populares los algoritmos que permiten movimientos que degradan la solución para evitar que la iteración se detenga en soluciones locales.

En este contexto se sitúan los algoritmos de recocido simulado, que se basan en las propiedades de la mecánica estadística, y permiten movimientos que degradan la solución, en cambio, a medida que el algoritmo avanza, la probabilidad de que se realicen los movimientos disminuye. Los algoritmos genéticos les ocurre algo parecido, se basan en propiedades de la mutación natural, y las redes neuronales modeladas a partir de la función cerebral. Estos algoritmos heurísticos, se usan para reconocer patrones, aprender respuestas y mejorar las soluciones a lo largo del tiempo.

Muchos de estos métodos fueron ampliados por Glover y Laguna en un enfoque conocido como "*tabu-search*", una meta-heurística que clasifica atributos deseados en un algoritmo. Incluye diversificación, reglas de aspiración y memoria a largo plazo. La aleatorización se puede incorporar fácilmente en este marco. La programación de restricciones, una metodología desarrollada en la comunidad de ciencias de la computación es otra forma de obtener soluciones efectivas. Para explorar el espacio de búsqueda de manera efectiva, se utiliza un lenguaje de programación como OPL (Open Programming Language).

Los algoritmos heurísticos que aprovechen la información de la relajación de programación lineal del problema se están desarrollando para abordar problemas generales de programación entera mixta. Un método sencillo es la "heurística de inmersión y x", con el cual fijamos algún subconjunto de variables enteras a valores enteros fijos, realizamos todas las fijaciones y procesamientos implícitos y resolvemos de nuevo el problema de programación lineal resultante. Este proceso no se detiene

hasta que el lenguaje de programación alcanza con una solución factible entera o porque no hay soluciones factibles para la relajación actual de lenguaje de programación. La enumeración de un subconjunto más pequeño de variables es un método alternativo que aumenta la probabilidad de encontrar una solución viable.

Estos enfoques heurísticos, basados en diversos conceptos, han demostrado ser valiosos para abordar problemas complejos de optimización combinatoria, proporcionando soluciones efectivas y mejorando la eficiencia de los algoritmos exactos. Existen una gran variedad de métodos heurísticos por lo que es muy difícil realizar una clasificación completa. Usaremos una clasificación formada por varias categorías no excluyentes [MB, 2015]:

- **Descomposición:** Divide el problema en subproblemas manejables, recordando que todas las divisiones forman parte del mismo problema.
- **Inductivos:** Este método generaliza soluciones simples al problema completo, utiliza propiedades validad en casos de fácil análisis.
- **Reducción:** Lo que hace es identificar y aplicar restricciones al problema, estas basadas en propiedades de buenas soluciones, simplificando el problema al acortar las soluciones posibles.
- **Búsqueda local:** Lo que hace es mejorar progresivamente la solución inicial, moviéndose hacia soluciones con mejor valor hasta que no se encuentra una mejora accesible.
- **Constructivos:** Uno de estos métodos será objeto de estudio en este trabajo. Consisten en construir paso a paso una solución añadiendo elementos en cada iteración, siendo métodos iterativos y deterministas que eligen la mejor opción en cada paso.

Estos métodos son los heurísticos tradicionales, y más recientemente han proliferado los metaheurísticos, que combinan métodos de búsqueda local con constructivos. En este caso, la clasificación contempla dos grupos diferenciados [MB, 2015]:

- **Trayectorias:** Se trata de métodos de búsqueda los que se añaden restricciones para evitar óptimos locales y buscar óptimos globales, partiendo de un punto inicial que se va actualizando y explorando sus vecinos, lo que genera una trayectoria en el espacio de soluciones.
- **Población:** Uno de estos métodos será objeto de estudio en este trabajo. Consisten en tratar el conjunto de soluciones como una población que va cambiando conforme a determinadas reglas de evolución.

Dentro del conjunto de algoritmos aplicados en este amplio espectro de métodos de resolución del TSP, nos centraremos a continuación en los dos métodos objeto de este trabajo: uno heurístico constructivo, el algoritmo del vecino más cercano (Nearest Neighbor, NN), y otro metaheurístico de población, el algoritmo genéritco (Genetic Algorithm, GA), tomando como referencia básica el análisis comparativo realizado por Arora, Argawal y Tanwar en 2016 [AAT, 2016]:

Heurística constructiva: algoritmo del Vecino más Cercano (NN)

El algoritmo del vecino más cercano (NN) fue introducido por Karl Menger en 1931, y destaca por su rapidez y simplicidad, que lo convierte en una de las primeras estrategias para resolver el problema del viajante [MT, 2020], sobre la que aún se siguen investigando varias mejoras y aplicaciones en diversos campos. Este algoritmo ha sido fundamental en la historia de la optimización combinatoria y sigue siendo una referencia para entender los principios básicos de los métodos heurísticos.

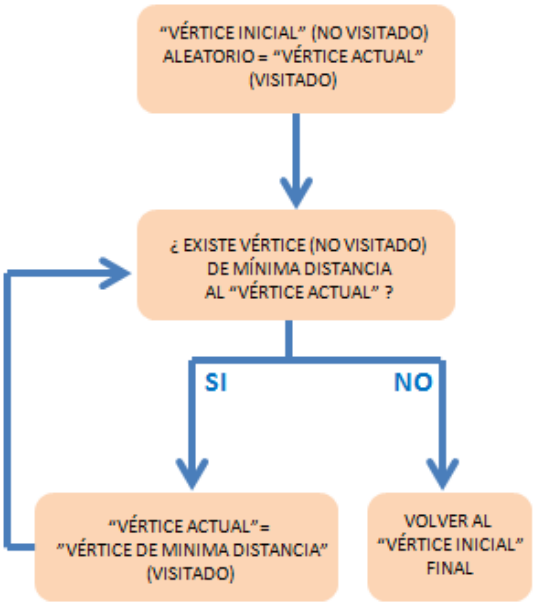
Posteriormente J.G.Skellam en 1952 estudio sistemáticamente este algoritmo para medir agrupaciones de datos, y P.J.Clark y F.C.Evans lo utilizaron en 1954 para medir las relaciones entre poblaciones, puesto que permite comparar distribuciones de distancias entre vecinos más cercanos en un conjunto de datos distribuidos aleatoriamente. Este enfoque ha permitido a los investigadores explorar patrones de dispersión y agrupación en distintos contextos geográficos y biológicos. [AB, 2017].

Descripción del algoritmo.

Identificadas las ciudades y distancias entre ellas con los vértices y aristas de un grafo, donde todos los vértices tienen inicialmente la condición de "vértices no visitados", se parte de un vértice aleatorio al que se asigna la condición de "vértice actual" y deja de estar entre los "vértices no visitados".

A continuación se busca entre los "vértices no visitado" conectados por aristas con el "vértice actual" aquel "vértice de mínima distancia".

Si existe, se le asigna la condición de "vértice actual", y deja de estar entre los "vértices no visitados", y se repite el procedimiento.



Si no existen "vértices no visitados", todos los vértices se han visitado al menos una vez, se vuelve al vértice de partida y el procedimiento termina.

Metaheurística de población: Algoritmo Genético (GA)

El algoritmo genético (GA) fue formalizado por Jon Holland en 1975, en un libro pionero donde comparaba la evolución natural de los seres vivos para adaptarse en condiciones óptimas a su entorno, con los métodos de solución de los problemas de

optimización en sistemas artificiales, como es el caso del problema del viajante [HJ,1975].

Realmente se trata de un método que permite desarrollar todo un conjunto de algoritmos evolutivos, que simulan los mecanismos genéticos por los cuales los cromosomas (conjuntos de genes) que determinan a cada individuo se combinan entre sí para dar lugar a nuevos individuos, de los que sobrevivirán aquellos que cuenten con mejores cromosomas (que los hagan más aptos para el medio natural en que viven):

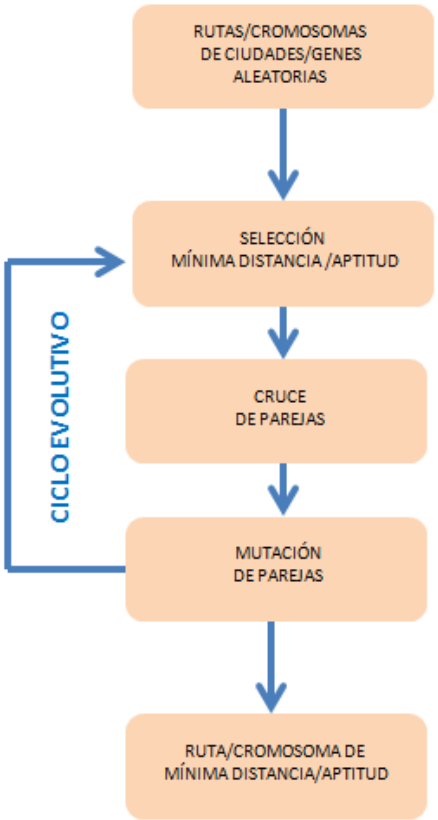
Esos mecanismos genéticos combinan:

1) Una "función de aptitud", que permite medir los cromosomas de un individuo y su mejor adaptación al medio.
2) Una "selección", que ordena los cromosomas de mejor a peor, según la función de aptitud.
3) Un "cruce" entre parejas cromosomas "padres", para dar lugar a parejas de cromosomas "hijos/sucesores" evolucionados.
4) Una "mutación" de cromosomas, que altera los genes que lo componen.
5) Una "evolución" donde se repiten ciclos de selección, cruce y mutación.

El algoritmo genético traslada estos mecanismos a un problema de optimización tomando como cromosomas un conjunto de posibles soluciones al problema (espacio de búsqueda) y como función de aptitud la función a optimizar, aplicando la selección, cruce y mutación para encontrar un conjunto de soluciones evolucionadas mejores.

Descripción del algoritmo.

El algoritmo genético aplicado al problema del viajante identifica las ciudades a visitar como genes, las posibles rutas solución que pasan por ellas como cromosomas, y la distancia total de cada una de esas rutas/cromosomas como la función de aptitud [GP, 2013]



Como punto de partida se toma un conjunto de rutas/cromosomas aleatorias iniciales, un número par (para luego hacer los cruces por parejas), que conforman el espacio de búsqueda, sobre los que se evalúa la distancia/aptitud, haciendo una selección que las ordena de menor a mayor.

A continuación, se van tomando parejas de rutas/cromosomas padres, que se cruzan para dar lugar a nuevas parejas de rutas / cromosomas hijos/sucesores que los sustituyen. Para ello se utiliza un determinado "operador de cruce".

Se pasa luego a introducir mutaciones en las rutas/cromosomas, alterando alguno de los genes/ciudades. Para ello se utiliza un determinado “operador de mutación”.

Se alcanza así un nuevo conjunto de rutas/cromosomas, evolución de los iniciales, sobre los que repite el ciclo evolutivo (selección, cruce, mutación), y así sucesivamente hasta que han realizado un determinado número de ciclos evolutivos.

Sobre el último conjunto de rutas/cromosomas se evalúa la distancia/aptitud, eligiendo como solución la ruta/cromosoma de menor valor.

Operadores de cruce [AMI, 2008]

El desarrollo de algoritmos genéticos ha dado lugar a diversos operadores de cruce y mutación que han sido experimentados con éxito. Se recogen a continuación los más utilizados:

- Correspondencia Parcial (PMX). Golberg-Lingle (1985). Se intercambia un sector parcial de los genes/ciudades de la pareja de cromosomas/rutas, que se hace corresponder con el sector análogo del otro miembro de la pareja.
- Ciclos (CX). Oliver-colaoradores (1987). Se van eligiendo genes/ciudades de cada pareja para construir la nueva pareja, alternando del primero al último, evitando repeticiones y construyendo ciclos.
- Orden uno (OX1). Davies (1985). Se cambia un sector parcial de genes/ciudades de uno de los miembros de la pareja, preservando el orden relativo de los genes/ciudades el otro miembro.
- Orden dos (OX2). Sysweda (1991). Es una modificación del OX1, que selecciona al azar varias posiciones de genes/ciudades de un miembro para imponer ese orden al otro, siguiendo modelos de los problemas de “secuenciación de tareas”.
- Posición (POS). Sysweda (1991). Es otra modificación del OX2, donde se impone al otro miembro las posiciones en los elementos seleccionados.
- Combinación de arcos (ER). Whitley-Starkwether (1990), Whitley-colaboradores (1991). Conecta los arcos que comienzan y terminan en los genes/ciudades de cada miembro de la pareja.
- Combinación del voto (VR). Mühlenbein (1989). Considera parejas múltiples, para analizar la frecuencia de las rutas entre los mismos genes/ciudades, tomando las más repetidas/votadas y haciendo mutaciones en el resto.
- Alternancia de posiciones (AP). Larrañaga-colaboradres (1999). Selecciona alternativamente genes/ciudades de cada miembro de la pareja.

Operadores de mutación [AMI, 2008]

También en el caso de la mutación se han desarrollado diversos operadores que han tenido éxito en su aplicación práctica:

- Desplazamiento (DM). Michalewizc (1997). Se selecciona aleatoriamente un sector de genes/ciudades que se extrae de cada cromosoma/ruta para insertarlo en un lugar aleatorio.
- Cambios (EM). Banzhaf (1990). Se seleccionan aleatoriamente dos genes/ciudades de cada cromosoma/ruta, que se intercambian.
- Inserción (ISM). Michalewizc (1992), Fogel (1993). Se selecciona aleatoriamente un gen/ciudad que se extrae de cada cromosoma/ruta para insertarlo en un lugar aleatorio.
- Inversión simple (SIM). Holland (1975), Grefenstette (1985). Se seleccionan aleatoriamente dos lugares de cada cromosoma/ruta, y se invierten el orden de genes/ciudades en el sector de comprendido entre esos lugares.
- Inversión (IVM). Foguel (1998, 1993). Se selecciona aleatoriamente un sector de genes/ciudades que se extrae de cada cromosoma/ruta para invertir el orden de sus genes/ciudades e insertar esa inversión en un lugar aleatorio.
- Cambio (SM). Syswerda (1991). Se selecciona aleatoriamente un sector de genes/ciudades, en el que se altera el orden de cada cromosoma/ruta.

3. ANÁLISIS DE LOS ALGORITMOS NN Y GA

3.1. Algoritmo del Vecino más Cercano (NN)

- Datos de entrada:
  - Vértices $V = \{v_1, v_2, \dots, v_n\}$.
  - Distancias $D = \{d_{ij}\}$ asociadas a las aristas $A = \{(v_i, v_j)\}, i, j = 1, \dots, n$. Calculadas directamente, o a partir de las coordenadas planas de los vértices $v_i = (x_i, y_i)$, utilizando la distancia euclídea:

$$d_{ij} = \sqrt{(x_i - x_j)^2 - (y_i - y_j)^2}.$$

- Inicialización de variables:
  - Vértices no visitados $V_{NV} = V$
  - Vértices visitados $V_V = \{\emptyset\}$.
  - Ciclo hamiltoniano buscado $C = \{\emptyset\}$.

- Paso 0:
  - Elección aleatoria del vértice inicial v_I.
  - Se designa como vértice actual $v_A = v_I$.

- Paso 1:
  - Se actualiza $V_{NV} = V_{NV} - \{v_A\}$.
  - Se actualiza $V_V = V_V \cup \{v_A\}$.

- Paso 2:
  - Mientras $V_{NV} \neq \{\emptyset\}$, búsqueda del vértice $v_{MD} \in V_{NV}$, conectado por una arista de mínima distancia con el vértice actual v_A.
    - Si existe v_{MD} , entonces:
      - Se incorpora la arista (v_A, v_{MD}) al ciclo hamiltoniano buscado, $C = C \cup \{(v_A, v_{MD})\}$.
      - Se designa como vértice actual $v_A = v_{MD}$
      - Se vuelve al paso 1.
    - Si no existe v_{MD}, entonces:
      - Se incorpora la arista (v_A, v_I) al ciclo hamiltoniano buscado, $C = C \cup \{(v_A, v_I)\}$.
      - Se vuelve al paso 1.

- Paso 3:
  - El algoritmo finaliza cuando $V_{NV} = \{\emptyset\}$, se han visitado todos los vértices, y C es el ciclo hamiltoniano buscado.
  - Si durante la construcción de C se completa un ciclo y aún $V_{NV} \neq \{\emptyset\}$, se tiene un ciclo hamiltoniano parcial, volviendo al paso 0 se pude completar otro ciclo parcial y uniendo ambos tener el C buscado [HW, 2004].

Esta última propiedad garantiza una cierta flexibilidad a la hora de construcción del recorrido y permite ajustar la búsqueda local para mejorar los resultados en algunos casos.

De este modo al final del algoritmo se obtiene un recorrido que contiene todos los vértices como visitados, y que es un recorrido mínimo, cuya distancia es la suma de las distancias del ciclo hamiltoniano final C. Pero puede que no sea el óptimo, pues debido a su naturaleza "codiciosa", este algoritmo pasa por alto algunas rutas más cortas (desde vértices intermedios del recorrido parcial y construyendo otros recorridos parciales) que pueden detectarse fácilmente con la inteligencia humana (por ensayo y error). Es posible, por tanto, que el algoritmo NN no encuentre un recorrido mínimo factible óptimo aunque éste exista [AB 2017].

3.2. Algoritmo Genético (GA)

- Datos de entrada:
  - Vértices $V = \{v_1, v_2, \dots, v_n\}$.
  - Distancias $D = \{d_{ij}\}$ asociadas a las aristas $A = \{(v_i, v_j)\}, i, j = 1, \dots, n$.
    Calculadas directamente, o a partir de las coordenadas planas de los vértices $v_i = (x_i, y_i)$, utilizando la distancia euclídea:
    $$d_{ij} = \sqrt{(x_i - x_j)^2 - (y_i - y_j)^2}.$$
- Inicialización de variables:
  - Número de rutas/cromosomas: $2r$.
  - Contador de ciclos evolutivos: $i = 1,2, \dots, s$.
  - Función distancia/aptitud aplicada a una ruta:
    $$d(\{(v_{j_a}, v_{kj_b})\}) = \sum d_{j_a j_b}$$
- Paso 0:
  - Construcción de $2r$ rutas/cromosomas aleatorias
    $R_k = \{(v_{kj_1}, v_{kj_2}), (v_{kj_2}, v_{kj_3}), \dots, (v_{kj_{(n-1)}}, v_{kj_n}), (v_{kj_n}, v_{kj_1})\}, k = 1,2, \dots, 2r.$
- Paso i:
  - $i.1$: Selección de rutas/cromosomas, ordenándolos mínima distancia.
    - Se calcula $d(R_k), k = 1,2, \dots, 2r$.
    - Se ordenan de menor a mayor, $R_{j_1}, R_{j_2}, \dots, R_{j_{2r}}$, para que actúen como padres $P_k = R_{j_k}$
  - $i.2$: Cruce de parejas de rutas/cromosomas.
    - Se toman como padres cada pareja P_k, P_{k+1} resultado de la selección.

- Se aplica un operador de cruce (algoritmo auxiliar) para obtener una nueva pareja de hijos/sucesores H_k, H_{k+1} que sustituye a la pareja de padres tomada.

  - $i.3$: Mutación de rutas/cromosomas.
    - Se toma cada ruta/cromosoma H_k resultado del cruce.
    - Se aplica un operador de mutación (algoritmo auxiliar) para obtener una nueva ruta/cromosoma R_k que sustituye a la ruta/cromosoma H_k tomada.

- Paso $i+1$: Sobre el conjunto de rutas/cromosomas R_k obtenidos tras el paso i, se vuelve a repetir paso i, hasta que $i = s$.

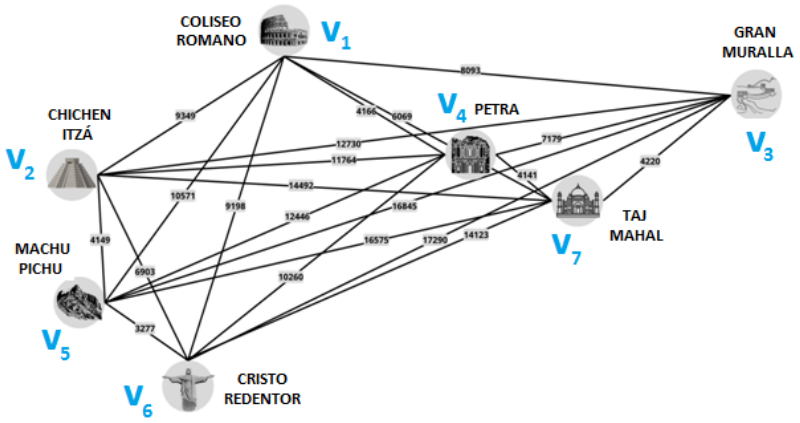
- Paso $s+1$:
  - Se calcula $d(R_k), k = 1,2, \dots, 2r$.
  - Se ordenan de menor a mayor, $R_{k_1}, R_{k_2}, \dots, R_{k_{2r}}$.
  - Se tomo como solución óptima R_{k_1}, la ruta/cromosoma de menor distancia.

3.3. Estudio de un ejemplo

Para estudiar estos algoritmos, se procede a realizar una programación propia en Rstudio, tomando un ejemplo del mundo real, también de elaboración propia: una persona quiere visitar las siete maravillas del mundo moderno (elegidas en el año 2000 a través de una encuesta desplegada por "The New 7 Wonders Foundation"), que están situadas en siete lugares del mundo, todos ellos conectados entre sí por rutas aéreas con una determinada distancia kilométrica

El grafo de este problema tiene 7 vértices (maravillas) y 23 aristas (rutas aéreas):

THE NEW 7 WONDERS	Coliseo Romano ITALIA	Chichen Itzá MÉXICO	Gran Muralla CHINA	Petra JORDANIA	Machu Pichu PERÚ	Cristo Redentor BRASIL	Taj Mahal INDIA
Coliseo Romano ITALIA	0	9349	8093	4166	10571	9198	6069
Chichen Itzá MÉXICO	9349	0	12730	11764	4149	6903	14492
Gran Muralla CHINA	8093	12730	0	7179	16845	17290	4220
Petra JORDANIA	4166	11764	7179	0	12446	10260	4141
Machu Pichu PERÚ	10571	4149	16845	12446	0	3277	16575
Cristo Redentor BRASIL	9198	6903	17290	10260	3277	0	14123
Taj Mahal INDIA	6069	14492	4220	4141	16575	14123	0



En este caso el método de fuerza bruta requerirá calcular $(7-1)! = 6! = 720$ rutas y ordenarlas según sus distancias, de menor a mayor, un cómputo asequible que da como resultado:

Orden	Rutas	Distancias
1	$v_1 \rightarrow v_4 \rightarrow v_7 \rightarrow v_3 \rightarrow v_2 \rightarrow v_5 \rightarrow v_6 \rightarrow v_1$	41881
2	$v_1 \rightarrow v_6 \rightarrow v_5 \rightarrow v_2 \rightarrow v_3 \rightarrow v_7 \rightarrow v_4 \rightarrow v_1$	41881
3	$v_1 \rightarrow v_2 \rightarrow v_5 \rightarrow v_6 \rightarrow v_4 \rightarrow v_7 \rightarrow v_3 \rightarrow v_1$	43489
4	$v_1 \rightarrow v_3 \rightarrow v_7 \rightarrow v_4 \rightarrow v_6 \rightarrow v_5 \rightarrow v_2 \rightarrow v_1$	43489
5	$v_1 \rightarrow v_2 \rightarrow v_5 \rightarrow v_6 \rightarrow v_4 \rightarrow v_3 \rightarrow v_7 \rightarrow v_1$	44503
…	…	…
716	$v_1 \rightarrow v_5 \rightarrow v_3 \rightarrow v_6 \rightarrow v_7 \rightarrow v_2 \rightarrow v_4 \rightarrow v_1$	89251
717	$v_1 \rightarrow v_2 \rightarrow v_4 \rightarrow v_5 \rightarrow v_7 \rightarrow v_6 \rightarrow v_3 \rightarrow v_1$	89640
718	$v_1 \rightarrow v_3 \rightarrow v_6 \rightarrow v_7 \rightarrow v_5 \rightarrow v_4 \rightarrow v_2 \rightarrow v_1$	89640
719	$v_1 \rightarrow v_4 \rightarrow v_2 \rightarrow v_7 \rightarrow v_5 \rightarrow v_3 \rightarrow v_6 \rightarrow v_1$	90330
720	$v_1 \rightarrow v_6 \rightarrow v_3 \rightarrow v_5 \rightarrow v_7 \rightarrow v_2 \rightarrow v_4 \rightarrow v_1$	90330

Por tanto, en este ejemplo existen dos posibles rutas óptimas, con la misma distancia mínima de 41.881 kilómetros.

A continuación, se procede a aplicar los algoritmos NN y GA a este mismo ejemplo, para comparar los resultados que se alcanzan.

3.3.1. Aplicación del algoritmo NN

- Inicialización de variables:
  - Vértices no visitados $V_{NV} = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$.
  - Vértices visitados $V_V = \{\emptyset\}$.
  - Ciclo hamiltoniano buscado $C = \{\emptyset\}$.
- Paso 0:
  - Elección aleatoria del vértice inicial $v_I = v_1$ = Coliseo Romano.
  - Se designa como vértice actual $v_A = v_I$.

- Paso 1:
  - Se actualiza $V_{NV} = \{v_2, v_3, v_4, v_5, v_6, v_7\} \neq \{\emptyset\}$.
  - Se actualiza $V_V = \{v_1\}$.

- Paso 2:
  - Como $V_{NV} \neq \{\emptyset\}$, se busca del vértice $v_{MD} \in V_{NV}$, conectado por una arista de mínima distancia con el vértice actual $v_A = v_I =$ Coliseo Romano
    - Ese vértice existe: $v_{MD} = v_4 =$ Petra, donde la arista $(v_A, v_{MD}) = (v_1, v_4)$ tiene distancia 4166.
      - Se incorpora a C esta arista, $C = \{(v_1, v_4)\}$.
      - Se designa como vértice actual $v_A = v_{MD} = v_4 =$ Petra.
      - Se vuelve al paso 1.

- Paso 1:
  - Se actualiza $V_{NV} = \{v_2, v_3, v_5, v_6, v_7\} \neq \{\emptyset\}$.
  - Se actualiza $V_V = \{v_1, v_4\}$.

- Paso 2:
  - Como $V_{NV} \neq \{\emptyset\}$, se busca del vértice $v_{MD} \in V_{NV}$, conectado por una arista de mínima distancia con el vértice $v_A = v_4 =$ Petra
    - Ese vértice existe: $v_{MD} = v_7 =$ Taj Mahal, donde la arista $(v_A, v_{MD}) = (v_4, v_7)$ tiene distancia 4141.
      - Se incorpora a C esta arista, $C = \{(v_1, v_4), (v_4, v_7)\}$.
      - Se designa como vértice actual $v_A = v_{MD} = v_7 =$ Taj Mahal.
      - Se vuelve al paso 1.

- Paso 1:
  - Se actualiza $V_{NV} = \{v_2, v_3, v_5, v_6\} \neq \{\emptyset\}$.
  - Se actualiza $V_V = \{v_1, v_4, v_7\}$.

- Paso 2:
  - Como $V_{NV} \neq \{\emptyset\}$, se busca del vértice $v_{MD} \in V_{NV}$, conectado por una arista de mínima distancia con el vértice $v_A = v_7 =$ Taj Majal
    - Ese vértice existe: $v_{MD} = v_3 =$ Gran Muralla, donde la arista $(v_A, v_{MD}) = (v_7, v_3)$ tiene distancia 4220.
      - Se incorpora a C esta arista, $C = \{(v_1, v_4), (v_4, v_7), (v_7, v_3)\}$.
      - Se designa como vértice actual $v_A = v_{MD} = v_3 =$ Gran Muralla.
      - Se vuelve al paso 1.

- Paso 1:
  - Se actualiza $V_{NV} = \{v_2, v_5, v_6\} \neq \{\emptyset\}$.
  - Se actualiza $V_V = \{v_1, v_3, v_4, v_7\}$.

- Paso 2:

- Como $V_{NV} \neq \{\emptyset\}$, se busca del vértice $v_{MD} \in V_{NV}$, conectado por una arista de mínima distancia con el vértice $v_A = v_3 =$ Gran Muralla.
  - Ese vértice existe: $v_{MD} = v_2 =$ Chichen Itzá, donde la arista $(v_A, v_{MD}) = (v_3, v_2)$ tiene distancia 12730.
    - Se incorpora a C esta arista, $C = \{(v_1, v_4), (v_4, v_7), (v_7, v_3), (v_3, v_2)\}$.
    - Se designa como vértice actual $v_A = v_{MD} = v_2 =$ Chichen Itzá.
    - Se vuelve al paso 1.

- Paso 1:
  - Se actualiza $V_{NV} = \{v_5, v_6\} \neq \{\emptyset\}$.
  - Se actualiza $V_V = \{v_1, v_2, v_3, v_4, v_7\}$.

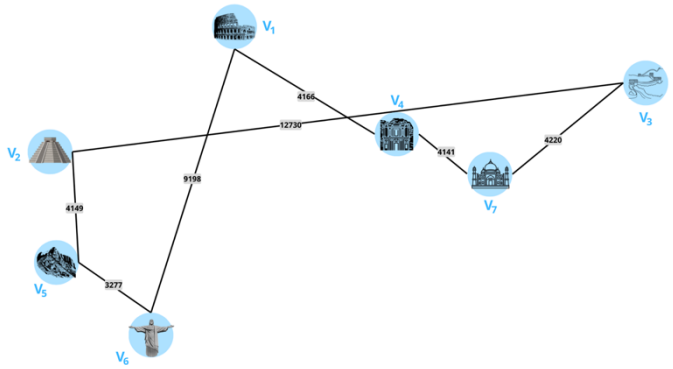
- Paso 2:
  - Como $V_{NV} \neq \{\emptyset\}$, se busca del vértice $v_{MD} \in V_{NV}$, conectado por una arista de mínima distancia con el vértice $v_A = v_2 =$ Chichen Itzá.
    - Ese vértice existe: $v_{MD} = v_5 =$ Machu Pichu, donde la arista $(v_A, v_{MD}) = (v_2, v_5)$ tiene distancia 4149.
      - Se incorpora a C esta arista, $C = \{(v_1, v_4), (v_4, v_7), (v_7, v_3), (v_3, v_2), (v_2, v_5)\}$.
      - Se designa como vértice actual $v_A = v_{MD} = v_5 =$ Machu Pichu.
      - Se vuelve al paso 1.
- Paso 1:
  - Se actualiza $V_{NV} = \{v_6\} \neq \{\emptyset\}$.
  - Se actualiza $V_V = \{v_1, v_2, v_3, v_4, v_5, v_7\}$.

- Paso 2:
  - Como $V_{NV} \neq \{\emptyset\}$, se busca del vértice $v_{MD} \in V_{NV}$, conectado por una arista de mínima distancia con el vértice $v_A = v_5 =$ Machu Pichu.
    - Ese vértice existe: $v_{MD} = v_6 =$ Cristo Redentor, donde la arista $(v_A, v_{MD}) = (v_5, v_6)$ tiene distancia 3277.
      - Se incorpora a C esta arista, $C = \{(v_1, v_4), (v_4, v_7), (v_7, v_3), (v_3, v_2), (v_2, v_5), (v_5, v_6)\}$.
      - Se designa como vértice actual $v_A = v_{MD} = v_6 =$ Cristo Redentor.
      - Se vuelve al paso 1.
- Paso 1:
  - Se actualiza $V_{NV} = \{\emptyset\}$.
  - Se actualiza $V_V = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$.

- Paso 2:
  - Como $V_{NV} = \{\emptyset\}$, no se pueden buscar vértices $v_{MD} \in V_{NV}$.
    - No existe $v_{MD} \in V_{NV}$, entonces:
      - Se incorpora la arista $(v_A, v_I) = (v_6, v_1)$ al ciclo hamiltoniano, $C = \{(v_1, v_4), (v_4, v_7), (v_7, v_3), (v_3, v_2), (v_2, v_5), (v_5, v_6), (v_6, v_1)\}$.

- Se va al paso 3.

- Paso 3:
  - El algoritmo finaliza: C es el ciclo hamiltoniano buscado, y la distancia mínima es 4166 + 4141 + 4220 + 12730 + 4149 + 3277 + 9198 = 41.881 km.



Repitiendo el ejemplo, tomando otros vértices iniciales, se obtienen diferentes soluciones, que se resumen en el siguiente cuadro:

Vértice inicial	Ruta	Distancia
V_1	$V_1 \to V_4 \to V_7 \to V_3 \to V_2 \to V_5 \to V_6 \to V_1$	41881
V_2	$V_2 \to V_5 \to V_6 \to V_1 \to V_4 \to V_7 \to V_3 \to V_2$	41881
V_3	$V_3 \to V_7 \to V_4 \to V_1 \to V_6 \to V_5 \to V_2 \to V_3$	41881
V_4	$V_4 \to V_7 \to V_3 \to V_1 \to V_6 \to V_5 \to V_2 \to V_4$	44842
V_5	$V_5 \to V_6 \to V_2 \to V_1 \to V_4 \to V_7 \to V_3 \to V_5$	48901
V_6	$V_6 \to V_5 \to V_2 \to V_1 \to V_4 \to V_7 \to V_3 \to V_6$	46592
V_7	$V_7 \to V_4 \to V_1 \to V_3 \to V_2 \to V_5 \to V_6 \to V_7$	50679

Obsérvese que la ruta que comienza en V_3 es la misma que la que comienza en V_1, recorriendo los vértices en sentido contrario. Por tanto este algoritmo nos da dos rutas óptimas, las mismas que las obtenidas por fuerza bruta.

Análisis del algoritmo.

La solución alcanzada en el algoritmo del vecino más próximo no tiene porqué dar la distancia global óptima, por varios motivos:

- En cada paso se toma la distancia local más corte en ese paso, sin considerar el impacto en la distancia total, que podría no ser más corta.
- Las ciudades con mayores distancias quedarán para el final del recorrido, incrementándose entonces la distancia total.
- El regreso al punto de partida puede suponer una gran distancia, como se observa en este ejemplo.

Para encontrar una solución realmente óptima, es necesario aplicar algoritmos más complejos que consideren todas las posibles rutas y seleccionen la más corta en términos globales y no locales.

3.3.2. Aplicación del algoritmo GA

Elección de operadores: cruce OX, mutación EM.

Como operador de cruce se adopta el OX2, cuya aplicación a cada pareja de rutas/cromosomas padres P_k, P_{k+1} para obtener la pareja de hijos H_k, H_{k+1} consiste en las siguientes operaciones:

- Selección aleatoria de posiciones de genes/ciudades a intercambiar entre las parejas P_k, P_{k+1} : posiciones 3, 4 y 5
- Para obtener H_k :
  - Selección de genes/ciudades en esas posiciones en P_{k+1}.
  - Eliminación de esos genes/ciudades en P_k para dar lugar al H_k inicial.
  - Para alcanzar el H_k final, completar los huecos del H_k inicial con los genes/ciudades eliminados, en el orden en que aparecen en P_{k+1}
- Para obtener H_{k+1} :
  - Se repiten las mismas operaciones anteriores sobre la pareja intercambiada de orden, P_{k+1} , P_k.

Como operador de mutación se adopta el EM, cuya aplicación a cada ruta/cromosoma R_k consiste en las siguientes operaciones

- Intercambio de los genes/ciudades en las posiciones 3 y 6.

Los pasos del algoritmo son entonces los siguientes.

- Inicialización de variables:
  - Número de rutas/cromosomas: $2r = 2\ x\ 3 = 6$.
  - Contador de ciclos evolutivos: $i = 1,2, \dots ,100$
  - Función distancia/aptitud aplicada a una ruta: $d(\{(v_{j_a}, v_{kj_b})\}) = \sum d_{j_a j_b}$
- Paso 0:
  - Construcción de 6 rutas/cromosomas aleatorias

Rutas / Cromosomas de partida	Genes / Ciudades
R_1	$v_7 \rightarrow v_3 \rightarrow v_6 \rightarrow v_2 \rightarrow v_4 \rightarrow v_5 \rightarrow v_1 \rightarrow v_7$
R_2	$v_5 \rightarrow v_4 \rightarrow v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow v_6 \rightarrow v_7 \rightarrow v_5$
R_3	$v_3 \rightarrow v_7 \rightarrow v_1 \rightarrow v_4 \rightarrow v_5 \rightarrow v_6 \rightarrow v_2 \rightarrow v_3$
R_4	$v_6 \rightarrow v_1 \rightarrow v_3 \rightarrow v_5 \rightarrow v_7 \rightarrow v_2 \rightarrow v_4 \rightarrow v_6$
R_5	$v_2 \rightarrow v_5 \rightarrow v_1 \rightarrow v_6 \rightarrow v_7 \rightarrow v_4 \rightarrow v_3 \rightarrow v_2$
R_6	$v_4 \rightarrow v_5 \rightarrow v_6 \rightarrow v_3 \rightarrow v_2 \rightarrow v_1 \rightarrow v_7 \rightarrow v_4$

- Paso $i = 1$
  - 1.1: Selección de rutas/cromosomas, ordenándolos mínima distancia.

Rutas / Cromosomas de partida	Genes / Ciudades	Distancias
R_1	$v_7 \rightarrow v_3 \rightarrow v_6 \rightarrow v_2 \rightarrow v_4 \rightarrow v_5 \rightarrow v_1 \rightarrow v_7$	69263
R_2	$v_5 \rightarrow v_4 \rightarrow v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow v_6 \rightarrow v_7 \rightarrow v_5$	86679
R_3	$v_3 \rightarrow v_7 \rightarrow v_1 \rightarrow v_4 \rightarrow v_5 \rightarrow v_6 \rightarrow v_2 \rightarrow v_3$	49811
R_4	$v_6 \rightarrow v_1 \rightarrow v_3 \rightarrow v_5 \rightarrow v_7 \rightarrow v_2 \rightarrow v_4 \rightarrow v_6$	87227
R_5	$v_2 \rightarrow v_5 \rightarrow v_1 \rightarrow v_6 \rightarrow v_7 \rightarrow v_4 \rightarrow v_3 \rightarrow v_2$	62091
R_6	$v_4 \rightarrow v_5 \rightarrow v_6 \rightarrow v_3 \rightarrow v_2 \rightarrow v_1 \rightarrow v_7 \rightarrow v_4$	65302

Rutas / Cromosomas reordenadas	Genes / Ciudades	Distancias
$P_1 = R_3$	$v_3 \rightarrow v_7 \rightarrow v_1 \rightarrow v_4 \rightarrow v_5 \rightarrow v_6 \rightarrow v_2 \rightarrow v_3$	49811
$P_2 = R_5$	$v_2 \rightarrow v_5 \rightarrow v_1 \rightarrow v_6 \rightarrow v_7 \rightarrow v_4 \rightarrow v_3 \rightarrow v_2$	62091
$P_3 = R_6$	$v_4 \rightarrow v_5 \rightarrow v_6 \rightarrow v_3 \rightarrow v_2 \rightarrow v_1 \rightarrow v_7 \rightarrow v_4$	65302
$P_4 = R_1$	$v_7 \rightarrow v_3 \rightarrow v_6 \rightarrow v_2 \rightarrow v_4 \rightarrow v_5 \rightarrow v_1 \rightarrow v_7$	69263
$P_5 = R_2$	$v_5 \rightarrow v_4 \rightarrow v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow v_6 \rightarrow v_7 \rightarrow v_5$	86679
$P_6 = R_4$	$v_6 \rightarrow v_1 \rightarrow v_3 \rightarrow v_5 \rightarrow v_7 \rightarrow v_2 \rightarrow v_4 \rightarrow v_6$	87227

- 1.2: Cruce de parejas de rutas/cromosomas.

  Operador OX2 sobre la pareja P_1, P_2

  - Para obtener H_1 :
  - Selección de genes/ciudades en posiciones 3, 4 y 5 en P_2 : v_1, v_6, v_7
  - Eliminación de esos genes/ciudades en P_1 para dar lugar al H_1 inicial.

    $$v_3 \rightarrow \emptyset \rightarrow \emptyset \rightarrow v_4 \rightarrow v_5 \rightarrow \emptyset \rightarrow v_2 \rightarrow v_3$$

  - Para alcanzar H_1 final, completar huecos en H_1 inicial con los genes/ciudades eliminados, en el orden en que aparecen en P_2

    $$v_3 \rightarrow v_1 \rightarrow v_6 \rightarrow v_4 \rightarrow v_5 \rightarrow v_7 \rightarrow v_2 \rightarrow v_3$$

  - Para obtener H_2 :
  - Selección de genes/ciudades en posiciones 3, 4 y 5 en P_1 : v_1, v_4, v_5
  - Eliminación de esos genes/ciudades en P_2 para dar lugar al H_2 inicial.

    $$v_2 \rightarrow \emptyset \rightarrow \emptyset \rightarrow v_6 \rightarrow v_7 \rightarrow \emptyset \rightarrow v_3 \rightarrow v_2$$

  - Para alcanzar H_2 final, completar huecos en H_2 inicial con los genes/ciudades eliminados, en el orden en que aparecen en P_1

    $$v_2 \rightarrow v_1 \rightarrow v_4 \rightarrow v_6 \rightarrow v_7 \rightarrow v_5 \rightarrow v_3 \rightarrow v_2$$

  Operador OX2 sobre la pareja P_3, P_4

  - Para obtener H_3 :

- Selección de genes/ciudades en posiciones 3, 4 y 5 en P_4 : v_6, v_2, v_4
- Eliminación de esos genes/ciudades en P_3 para dar lugar al H_3 inicial.

$$\emptyset \rightarrow v_5 \rightarrow \emptyset \rightarrow v_3 \rightarrow \emptyset \rightarrow v_1 \rightarrow v_7 \rightarrow \emptyset$$

- Para alcanzar H_3 final, completar huecos en H_3 inicial con los genes/ciudades eliminados, en el orden en que aparecen en P_4

$$v_6 \rightarrow v_5 \rightarrow v_2 \rightarrow v_3 \rightarrow v_4 \rightarrow v_1 \rightarrow v_7 \rightarrow v_6$$

- Para obtener H_4 :
- Selección de genes/ciudades en posiciones 3, 4 y 5 en P_3 : v_6, v_3, v_2
- Eliminación de esos genes/ciudades en P_4 para dar lugar al H_4 inicial.

$$v_7 \rightarrow \emptyset \rightarrow \emptyset \rightarrow \emptyset \rightarrow v_4 \rightarrow v_5 \rightarrow v_1 \rightarrow v_7$$

- Para alcanzar H_4 final, completar huecos en H_4 inicial con los genes/ciudades eliminados, en el orden en que aparecen en P_3

$$v_7 \rightarrow v_6 \rightarrow v_3 \rightarrow v_2 \rightarrow v_4 \rightarrow v_5 \rightarrow v_1 \rightarrow v_7$$

Operador OX2 sobre la pareja P_5, P_6

- Para obtener H_5 :
- Selección de genes/ciudades en posiciones 3, 4 y 5 en P_6 : v_3, v_5, v_7
- Eliminación de esos genes/ciudades en P_5 para dar lugar al H_5 inicial.

$$\emptyset \rightarrow v_4 \rightarrow v_1 \rightarrow v_2 \rightarrow \emptyset \rightarrow v_6 \rightarrow \emptyset \rightarrow \emptyset$$

- Para alcanzar H_5 final, completar huecos en H_5 inicial con los genes/ciudades eliminados, en el orden en que aparecen en P_6

$$v_3 \rightarrow v_4 \rightarrow v_1 \rightarrow v_2 \rightarrow v_5 \rightarrow v_6 \rightarrow v_7 \rightarrow v_3$$

- Para obtener H_6 :
- Selección de genes/ciudades en posiciones 3, 4 y 5 en P_5 : v_1, v_2, v_3
- Eliminación de esos genes/ciudades en P_6 para dar lugar al H_6 inicial.

$$v_6 \rightarrow \emptyset \rightarrow \emptyset \rightarrow v_5 \rightarrow v_7 \rightarrow \emptyset \rightarrow v_4 \rightarrow v_6$$

- Para alcanzar H_6 final, completar huecos en H_6 inicial con los genes/ciudades eliminados, en el orden en que aparecen en P_5

$$v_6 \rightarrow v_1 \rightarrow v_2 \rightarrow v_5 \rightarrow v_7 \rightarrow v_3 \rightarrow v_4 \rightarrow v_6$$

Resultado del cruce

Rutas / Cromosomas Hijos	Genes / Ciudades
H_1	$v_3 \rightarrow v_1 \rightarrow v_6 \rightarrow v_4 \rightarrow v_5 \rightarrow v_7 \rightarrow v_2 \rightarrow v_3$
H_2	$v_2 \rightarrow v_1 \rightarrow v_4 \rightarrow v_6 \rightarrow v_7 \rightarrow v_5 \rightarrow v_3 \rightarrow v_2$
H_3	$v_6 \rightarrow v_5 \rightarrow v_2 \rightarrow v_3 \rightarrow v_4 \rightarrow v_1 \rightarrow v_7 \rightarrow v_6$
H_4	$v_7 \rightarrow v_6 \rightarrow v_3 \rightarrow v_2 \rightarrow v_4 \rightarrow v_5 \rightarrow v_1 \rightarrow v_7$
H_5	$v_3 \rightarrow v_4 \rightarrow v_1 \rightarrow v_2 \rightarrow v_5 \rightarrow v_6 \rightarrow v_7 \rightarrow v_3$
H_6	$v_6 \rightarrow v_1 \rightarrow v_2 \rightarrow v_5 \rightarrow v_7 \rightarrow v_3 \rightarrow v_4 \rightarrow v_6$

- 1.3: Mutación de rutas/cromosomas.

  Operador EM sobre cada H_k , intercambiando los genes/ciudades en las posiciones 3 y 6.

  Resultado de la Mutación

Rutas / Cromosomas Mutados	Genes / Ciudades
R_1	$v_3 \rightarrow v_1 \rightarrow v_7 \rightarrow v_4 \rightarrow v_5 \rightarrow v_6 \rightarrow v_2 \rightarrow v_3$
R_2	$v_2 \rightarrow v_1 \rightarrow v_5 \rightarrow v_6 \rightarrow v_7 \rightarrow v_4 \rightarrow v_3 \rightarrow v_2$
R_3	$v_6 \rightarrow v_5 \rightarrow v_1 \rightarrow v_3 \rightarrow v_4 \rightarrow v_2 \rightarrow v_7 \rightarrow v_6$
R_4	$v_7 \rightarrow v_6 \rightarrow v_5 \rightarrow v_2 \rightarrow v_4 \rightarrow v_3 \rightarrow v_1 \rightarrow v_7$
R_5	$v_3 \rightarrow v_4 \rightarrow v_6 \rightarrow v_2 \rightarrow v_5 \rightarrow v_1 \rightarrow v_7 \rightarrow v_3$
R_6	$v_6 \rightarrow v_1 \rightarrow v_3 \rightarrow v_5 \rightarrow v_7 \rightarrow v_2 \rightarrow v_4 \rightarrow v_6$

Sobre estas rutas obtenidas en el paso $i = 1$, se repiten los mismos cálculos en el paso $i = 2$ y se continua sucesivamente.

- Paso $i = 2$
  - 2.1: Selección de rutas/cromosomas, ordenándolos mínima distancia.

Rutas / Cromosomas de partida	Genes / Ciudades	Distancias
R_1	$v_3 \rightarrow v_1 \rightarrow v_7 \rightarrow v_4 \rightarrow v_5 \rightarrow v_6 \rightarrow v_2 \rightarrow v_3$	53659
R_2	$v_2 \rightarrow v_1 \rightarrow v_5 \rightarrow v_6 \rightarrow v_7 \rightarrow v_4 \rightarrow v_3 \rightarrow v_2$	61370
R_3	$v_6 \rightarrow v_5 \rightarrow v_1 \rightarrow v_3 \rightarrow v_4 \rightarrow v_2 \rightarrow v_7 \rightarrow v_6$	69499
R_4	$v_7 \rightarrow v_6 \rightarrow v_5 \rightarrow v_2 \rightarrow v_4 \rightarrow v_3 \rightarrow v_1 \rightarrow v_7$	54654
R_5	$v_3 \rightarrow v_4 \rightarrow v_6 \rightarrow v_2 \rightarrow v_5 \rightarrow v_1 \rightarrow v_7 \rightarrow v_3$	49351
R_6	$v_6 \rightarrow v_1 \rightarrow v_3 \rightarrow v_5 \rightarrow v_7 \rightarrow v_2 \rightarrow v_4 \rightarrow v_6$	87227

Rutas / Cromosomas reordenadas	Genes / Ciudades	Distancias
$P_1 = R_5$	$v_3 \rightarrow v_4 \rightarrow v_6 \rightarrow v_2 \rightarrow v_5 \rightarrow v_1 \rightarrow v_7 \rightarrow v_3$	49351
$P_2 = R_1$	$v_3 \rightarrow v_1 \rightarrow v_7 \rightarrow v_4 \rightarrow v_5 \rightarrow v_6 \rightarrow v_2 \rightarrow v_3$	53659
$P_3 = R_4$	$v_7 \rightarrow v_6 \rightarrow v_5 \rightarrow v_2 \rightarrow v_4 \rightarrow v_3 \rightarrow v_1 \rightarrow v_7$	54654
$P_4 = R_2$	$v_2 \rightarrow v_1 \rightarrow v_5 \rightarrow v_6 \rightarrow v_7 \rightarrow v_4 \rightarrow v_3 \rightarrow v_2$	61370

$P_5 = R_3$	$v_6 \rightarrow v_5 \rightarrow v_1 \rightarrow v_3 \rightarrow v_4 \rightarrow v_2 \rightarrow v_7 \rightarrow v_6$	69499
$P_6 = R_6$	$v_6 \rightarrow v_1 \rightarrow v_3 \rightarrow v_5 \rightarrow v_7 \rightarrow v_2 \rightarrow v_4 \rightarrow v_6$	87227

- 2.2: Cruce de parejas de rutas/cromosomas.

  Resultado del cruce

Rutas / Cromosomas Hijos	Genes / Ciudades
H_2	$v_3 \rightarrow v_1 \rightarrow v_7 \rightarrow v_4 \rightarrow v_6 \rightarrow v_2 \rightarrow v_5 \rightarrow v_3$
H_3	$v_5 \rightarrow v_6 \rightarrow v_7 \rightarrow v_2 \rightarrow v_4 \rightarrow v_3 \rightarrow v_1 \rightarrow v_5$
H_4	$v_5 \rightarrow v_1 \rightarrow v_2 \rightarrow v_6 \rightarrow v_7 \rightarrow v_4 \rightarrow v_3 \rightarrow v_5$
H_5	$v_6 \rightarrow v_3 \rightarrow v_1 \rightarrow v_5 \rightarrow v_4 \rightarrow v_2 \rightarrow v_7 \rightarrow v_6$
H_6	$v_6 \rightarrow v_1 \rightarrow v_3 \rightarrow v_5 \rightarrow v_7 \rightarrow v_2 \rightarrow v_4 \rightarrow v_6$

- 2.3: Mutación de rutas/cromosomas.

  Resultado de la Mutación

R_1	$v_3 \rightarrow v_7 \rightarrow v_1 \rightarrow v_2 \rightarrow v_4 \rightarrow v_6 \rightarrow v_5 \rightarrow v_3$
R_2	$v_3 \rightarrow v_1 \rightarrow v_2 \rightarrow v_4 \rightarrow v_6 \rightarrow v_7 \rightarrow v_5 \rightarrow v_3$
R_3	$v_5 \rightarrow v_6 \rightarrow v_3 \rightarrow v_2 \rightarrow v_4 \rightarrow v_7 \rightarrow v_1 \rightarrow v_5$
R_4	$v_5 \rightarrow v_1 \rightarrow v_4 \rightarrow v_6 \rightarrow v_7 \rightarrow v_2 \rightarrow v_3 \rightarrow v_5$
R_5	$v_6 \rightarrow v_3 \rightarrow v_2 \rightarrow v_5 \rightarrow v_4 \rightarrow v_1 \rightarrow v_7 \rightarrow v_6$
R_6	$v_6 \rightarrow v_1 \rightarrow v_2 \rightarrow v_5 \rightarrow v_7 \rightarrow v_3 \rightarrow v_4 \rightarrow v_6$

- Paso $i = 3$
  - 3.1: Selección de rutas/cromosomas, ordenándolos mínima distancia.

Rutas / Cromosomas mutados	Genes / Ciudades	Distancias
R_1	$v_3 \rightarrow v_7 \rightarrow v_1 \rightarrow v_2 \rightarrow v_4 \rightarrow v_6 \rightarrow v_5 \rightarrow v_3$	61784
R_2	$v_3 \rightarrow v_1 \rightarrow v_2 \rightarrow v_4 \rightarrow v_6 \rightarrow v_7 \rightarrow v_5 \rightarrow v_3$	87009
R_3	$v_5 \rightarrow v_6 \rightarrow v_3 \rightarrow v_2 \rightarrow v_4 \rightarrow v_7 \rightarrow v_1 \rightarrow v_5$	65842
R_4	$v_5 \rightarrow v_1 \rightarrow v_4 \rightarrow v_6 \rightarrow v_7 \rightarrow v_2 \rightarrow v_3 \rightarrow v_5$	83187
R_5	$v_6 \rightarrow v_3 \rightarrow v_2 \rightarrow v_5 \rightarrow v_4 \rightarrow v_1 \rightarrow v_7 \rightarrow v_6$	70973
R_6	$v_6 \rightarrow v_1 \rightarrow v_2 \rightarrow v_5 \rightarrow v_7 \rightarrow v_3 \rightarrow v_4 \rightarrow v_6$	60930

Rutas / Cromosomas reordenadas	Genes / Ciudades	Distancias
$P_1 = R_6$	$v_6 \rightarrow v_1 \rightarrow v_2 \rightarrow v_5 \rightarrow v_7 \rightarrow v_3 \rightarrow v_4 \rightarrow v_6$	60930
$P_2 = R_1$	$v_3 \rightarrow v_7 \rightarrow v_1 \rightarrow v_2 \rightarrow v_4 \rightarrow v_6 \rightarrow v_5 \rightarrow v_3$	61784
$P_3 = R_3$	$v_5 \rightarrow v_6 \rightarrow v_3 \rightarrow v_2 \rightarrow v_4 \rightarrow v_7 \rightarrow v_1 \rightarrow v_5$	65842
$P_4 = R_5$	$v_6 \rightarrow v_3 \rightarrow v_2 \rightarrow v_5 \rightarrow v_4 \rightarrow v_1 \rightarrow v_7 \rightarrow v_6$	70973
$P_5 = R_4$	$v_5 \rightarrow v_1 \rightarrow v_4 \rightarrow v_6 \rightarrow v_7 \rightarrow v_2 \rightarrow v_3 \rightarrow v_5$	83187
$P_6 = R_2$	$v_3 \rightarrow v_1 \rightarrow v_2 \rightarrow v_4 \rightarrow v_6 \rightarrow v_7 \rightarrow v_5 \rightarrow v_3$	87009

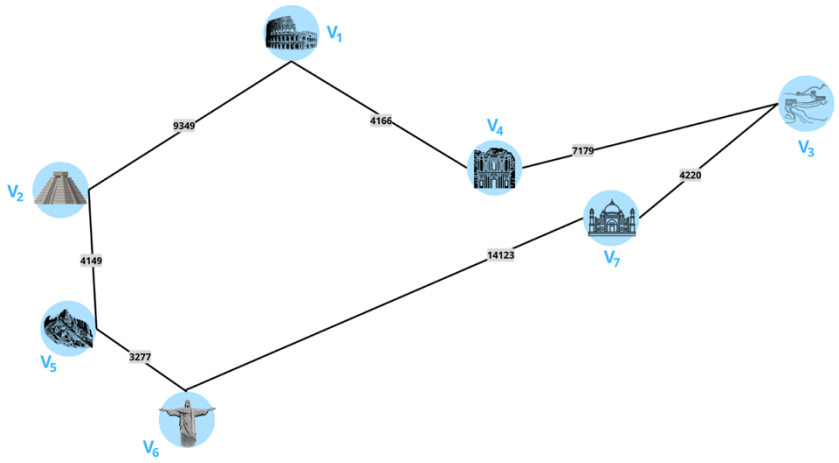
:

Se presentan a continuación de manera resumida los resultados alcanzados tras llegar al paso $i = 48$, incluyendo el tiempo de cómputo del algoritmo

Paso i	Genes / Ciudades de distancia mínima	Distancia mínima	Tiempo de cómputo
1	$v_3 \to v_4 \to v_6 \to v_2 \to v_5 \to v_1 \to v_7 \to v_3$	49351	0
2	$v_6 \to v_1 \to v_2 \to v_5 \to v_7 \to v_3 \to v_4 \to v_6$	60930	0
3	$v_3 \to v_2 \to v_6 \to v_5 \to v_4 \to v_1 \to v_7 \to v_3$	49811	0,009
4	$v_3 \to v_1 \to v_2 \to v_6 \to v_5 \to v_7 \to v_4 \to v_3$	55517	0,01
5	$v_2 \to v_6 \to v_5 \to v_1 \to v_7 \to v_3 \to v_4 \to v_2$	49983	0,012
6	$v_3 \to v_5 \to v_2 \to v_6 \to v_7 \to v_1 \to v_4 \to v_3$	59434	0,012
7	$v_7 \to v_1 \to v_6 \to v_5 \to v_4 \to v_2 \to v_3 \to v_7$	59704	0,012
8	$v_2 \to v_1 \to v_6 \to v_5 \to v_7 \to v_3 \to v_4 \to v_2$	61562	0,012
9	$v_7 \to v_1 \to v_5 \to v_6 \to v_4 \to v_2 \to v_3 \to v_7$	58891	0,012
10	$v_3 \to v_2 \to v_1 \to v_5 \to v_6 \to v_7 \to v_4 \to v_3$	61370	0,014
11	$v_3 \to v_1 \to v_6 \to v_5 \to v_4 \to v_2 \to v_7 \to v_3$	63490	0,014
12	$v_3 \to v_6 \to v_2 \to v_5 \to v_7 \to v_1 \to v_4 \to v_3$	62331	0,014
13	$v_2 \to v_6 \to v_1 \to v_5 \to v_7 \to v_3 \to v_4 \to v_2$	66410	0,016
14	$v_2 \to v_1 \to v_5 \to v_6 \to v_4 \to v_3 \to v_7 \to v_2$	59348	0,016
15	$v_2 \to v_1 \to v_5 \to v_6 \to v_4 \to v_3 \to v_7 \to v_2$	59348	0,016
16	$v_2 \to v_5 \to v_3 \to v_7 \to v_6 \to v_1 \to v_4 \to v_2$	64465	0,016
17	$v_3 \to v_2 \to v_1 \to v_5 \to v_6 \to v_7 \to v_4 \to v_3$	61370	0,018
18	$v_3 \to v_2 \to v_7 \to v_6 \to v_5 \to v_1 \to v_4 \to v_3$	66538	0,019
19	$v_2 \to v_1 \to v_5 \to v_6 \to v_7 \to v_3 \to v_4 \to v_2$	60483	0,018
20	$v_2 \to v_5 \to v_6 \to v_3 \to v_7 \to v_1 \to v_4 \to v_2$	50935	0,018
21	$v_2 \to v_1 \to v_5 \to v_6 \to v_3 \to v_7 \to v_4 \to v_2$	60612	0,019
22	$v_5 \to v_2 \to v_6 \to v_3 \to v_7 \to v_1 \to v_4 \to v_5$	55243	0,02
23	$v_2 \to v_1 \to v_6 \to v_5 \to v_3 \to v_7 \to v_4 \to v_2$	58794	0,019
24	$v_2 \to v_1 \to v_5 \to v_6 \to v_3 \to v_7 \to v_4 \to v_2$	60612	0,018
25	$v_1 \to v_2 \to v_3 \to v_7 \to v_5 \to v_6 \to v_4 \to v_1$	60577	0,019
26	$v_1 \to v_2 \to v_3 \to v_7 \to v_6 \to v_5 \to v_4 \to v_1$	60311	0,019
27	$v_1 \to v_2 \to v_6 \to v_5 \to v_7 \to v_3 \to v_4 \to v_1$	51669	0,021
28	$v_1 \to v_2 \to v_6 \to v_5 \to v_3 \to v_7 \to v_4 \to v_1$	48901	0,022
29	$v_1 \to v_2 \to v_5 \to v_6 \to v_3 \to v_7 \to v_4 \to v_1$	46592	0,025
30	$v_1 \to v_2 \to v_5 \to v_6 \to v_7 \to v_3 \to v_4 \to v_1$	46463	0,022
31	$v_2 \to v_1 \to v_3 \to v_6 \to v_5 \to v_7 \to v_4 \to v_2$	70489	0,024
32	$v_1 \to v_2 \to v_6 \to v_5 \to v_7 \to v_3 \to v_4 \to v_1$	51669	0,024
33	$v_1 \to v_2 \to v_6 \to v_5 \to v_3 \to v_7 \to v_4 \to v_1$	48901	0,023
34	$v_1 \to v_2 \to v_3 \to v_7 \to v_5 \to v_6 \to v_4 \to v_1$	60577	0,026
35	$v_1 \to v_2 \to v_6 \to v_5 \to v_7 \to v_3 \to v_4 \to v_1$	51669	0,023
36	$v_1 \to v_2 \to v_7 \to v_5 \to v_6 \to v_3 \to v_4 \to v_1$	72328	0,024
37	$v_2 \to v_1 \to v_5 \to v_6 \to v_3 \to v_7 \to v_4 \to v_2$	60612	0,024
38	$v_1 \to v_2 \to v_5 \to v_6 \to v_3 \to v_7 \to v_4 \to v_1$	46592	0,026
39	$v_1 \to v_2 \to v_3 \to v_7 \to v_5 \to v_6 \to v_4 \to v_1$	60577	0,027

40	$v_1 \rightarrow v_2 \rightarrow v_5 \rightarrow v_6 \rightarrow v_7 \rightarrow v_3 \rightarrow v_4 \rightarrow v_1$	46463	0,028
41	$v_1 \rightarrow v_2 \rightarrow v_6 \rightarrow v_3 \rightarrow v_7 \rightarrow v_5 \rightarrow v_4 \rightarrow v_1$	70949	0,026
42	$v_2 \rightarrow v_1 \rightarrow v_5 \rightarrow v_6 \rightarrow v_7 \rightarrow v_3 \rightarrow v_4 \rightarrow v_2$	60483	0,026
43	$v_1 \rightarrow v_2 \rightarrow v_5 \rightarrow v_6 \rightarrow v_3 \rightarrow v_7 \rightarrow v_4 \rightarrow v_1$	46592	0,026
44	$v_1 \rightarrow v_2 \rightarrow v_5 \rightarrow v_6 \rightarrow v_7 \rightarrow v_3 \rightarrow v_4 \rightarrow v_1$	46463	0,026
45	$v_1 \rightarrow v_2 \rightarrow v_6 \rightarrow v_3 \rightarrow v_7 \rightarrow v_5 \rightarrow v_4 \rightarrow v_1$	70949	0,029
46	$v_2 \rightarrow v_1 \rightarrow v_5 \rightarrow v_6 \rightarrow v_7 \rightarrow v_3 \rightarrow v_4 \rightarrow v_2$	60483	0,027
47	$v_1 \rightarrow v_2 \rightarrow v_5 \rightarrow v_6 \rightarrow v_3 \rightarrow v_7 \rightarrow v_4 \rightarrow v_1$	46592	0,032
48	$v_1 \rightarrow v_2 \rightarrow v_5 \rightarrow v_6 \rightarrow v_7 \rightarrow v_3 \rightarrow v_4 \rightarrow v_1$	46463	0,031

Se aprecian oscilaciones de aumento y disminución de la distancia mínima alcanzada en cada paso. A partir del paso $i = 42$, el algoritmo entra en un bucle, en el que se repiten 4 distancias iguales, por tanto, se adopta ese bucle como criterio parada en el paso $i = 41$, donde se sitúa la última distancia mínima nueva alcanzada, con la ruta:



Así, la mejor ruta obtenida con este algoritmo y con estos operadores de cruce y mutación es la alcanzada en el paso $i = 30$, con una distancia mínima de 46.463 kilómetros, lejos de la ruta óptima obtenida por fuerza bruta y por el algoritmo del vecino más cercano, cuya distancia mínima era 41.881 kilómetros.

Elección de operadores: cruce PMX, mutación DM.

Como operador de cruce se adopta el PMX, cuya aplicación a cada pareja de rutas/cromosomas padres P_k, P_{k+1} para obtener la pareja de hijos H_k, H_{k+1} consiste en las siguientes operaciones:

- Selección aleatoria de posiciones de genes/ciudades a intercambiar entre parejas P_k y P_{k+1}: posiciones 3,4 y 5.
- Para obtener H_k :
  - Selección de genes/ciudades en esas posiciones en P_{k+1}.
  - Crear una correspondencia de los genes/ciudades seleccionados en esas posiciones entre P_k y P_{k+1}.

- Transferir los genes/ciudades seleccionados en esas posiciones de P_{k+1} a H_k en las mismas posiciones.
- Mantener los genes/ciudades que no están en las posiciones seleccionadas de P_k en H_k, pero ajustando las posiciones de acuerdo a la correspondencia parcial para evitar duplicaciones.

- Para obtener H_{k+1} :
  - Se repiten las mismas operaciones anteriores sobre la pareja intercambiada de orden, P_{k+1} , P_k.

Como operador de mutación se adopta el DM, cuya aplicación a cada ruta/cromosoma R_k consiste en las siguientes operaciones:

- Identificación de los genes/ciudades ubicados en las posiciones 2,3 y 4 del en H_k.
- Extracción de estos genes/ciudades de en H_k.
- Reubicación de los genes/ciudades extraídos en la posición 4 del en H_k.
- Los genes/ciudades restantes en H_k se desplazan adecuadamente para acomodar los genes/ciudades insertados.

Los pasos del algoritmo son entonces los siguientes.

- Inicialización de variables:
  - Número de rutas/cromosomas: $2r = 2\ x\ 3 = 6$.
  - Contador de ciclos evolutivos: $i = 1,2, \ldots ,100$
  - Función distancia/aptitud aplicada a una ruta: $d(\{(v_{j_a}, v_{kj_b})\}) = \sum d_{j_a j_b}$

- Paso 0:
  - Construcción de 6 rutas/cromosomas aleatorias

Rutas / Cromosomas de partida	Genes / Ciudades
R_1	$v_7 \rightarrow v_1 \rightarrow v_4 \rightarrow v_6 \rightarrow v_2 \rightarrow v_5 \rightarrow v_3 \rightarrow v_7$
R_2	$v_3 \rightarrow v_7 \rightarrow v_5 \rightarrow v_4 \rightarrow v_6 \rightarrow v_2 \rightarrow v_1 \rightarrow v_3$
R_3	$v_6 \rightarrow v_5 \rightarrow v_7 \rightarrow v_3 \rightarrow v_1 \rightarrow v_2 \rightarrow v_4 \rightarrow v_6$
R_4	$v_3 \rightarrow v_5 \rightarrow v_7 \rightarrow v_2 \rightarrow v_1 \rightarrow v_4 \rightarrow v_6 \rightarrow v_3$
R_5	$v_5 \rightarrow v_2 \rightarrow v_7 \rightarrow v_3 \rightarrow v_6 \rightarrow v_4 \rightarrow v_1 \rightarrow v_5$
R_6	$v_4 \rightarrow v_5 \rightarrow v_7 \rightarrow v_6 \rightarrow v_2 \rightarrow v_1 \rightarrow v_3 \rightarrow v_4$

- Paso $i = 1$
  - 1.1: Selección de rutas/cromosomas, ordenándolos mínima distancia.

Rutas / Cromosomas de partida	Genes / Ciudades	Distancias
R_1	$v_7 \rightarrow v_1 \rightarrow v_4 \rightarrow v_6 \rightarrow v_2 \rightarrow v_5 \rightarrow v_3 \rightarrow v_7$	52612

R_2	$v_3 \rightarrow v_7 \rightarrow v_5 \rightarrow v_4 \rightarrow v_6 \rightarrow v_2 \rightarrow v_1 \rightarrow v_3$	67846
R_3	$v_6 \rightarrow v_5 \rightarrow v_7 \rightarrow v_3 \rightarrow v_1 \rightarrow v_2 \rightarrow v_4 \rightarrow v_6$	63538
R_4	$v_3 \rightarrow v_5 \rightarrow v_7 \rightarrow v_2 \rightarrow v_1 \rightarrow v_4 \rightarrow v_6 \rightarrow v_3$	88977
R_5	$v_5 \rightarrow v_2 \rightarrow v_7 \rightarrow v_3 \rightarrow v_6 \rightarrow v_4 \rightarrow v_1 \rightarrow v_5$	65148
R_6	$v_4 \rightarrow v_5 \rightarrow v_7 \rightarrow v_6 \rightarrow v_2 \rightarrow v_1 \rightarrow v_3 \rightarrow v_4$	74668

Rutas / Cromosomas reordenadas	Genes / Ciudades	Distancias
$P_1 = R_1$	$v_7 \rightarrow v_1 \rightarrow v_4 \rightarrow v_6 \rightarrow v_2 \rightarrow v_5 \rightarrow v_3 \rightarrow v_7$	52612
$P_2 = R_3$	$v_6 \rightarrow v_5 \rightarrow v_7 \rightarrow v_3 \rightarrow v_1 \rightarrow v_2 \rightarrow v_4 \rightarrow v_6$	63538
$P_3 = R_5$	$v_5 \rightarrow v_2 \rightarrow v_7 \rightarrow v_3 \rightarrow v_6 \rightarrow v_4 \rightarrow v_1 \rightarrow v_5$	65148
$P_4 = R_2$	$v_3 \rightarrow v_7 \rightarrow v_5 \rightarrow v_4 \rightarrow v_6 \rightarrow v_2 \rightarrow v_1 \rightarrow v_3$	67846
$P_5 = R_6$	$v_4 \rightarrow v_5 \rightarrow v_7 \rightarrow v_6 \rightarrow v_2 \rightarrow v_1 \rightarrow v_3 \rightarrow v_4$	74668
$P_6 = R_4$	$v_3 \rightarrow v_5 \rightarrow v_7 \rightarrow v_2 \rightarrow v_1 \rightarrow v_4 \rightarrow v_6 \rightarrow v_3$	88977

- 1.2: Cruce de parejas de rutas/cromosomas.

  Operador PMX sobre la pareja P_1, P_2

  - Para obtener H_1 :

  - Selección de genes/ciudades en posiciones 3, 4 y 5 en P_2 : v_7, v_3, v_1

  - Añadir esos genes/ciudades en P_1, sin el resto de genes/ciudades para dar lugar al H_1 inicial.

    $$\emptyset \rightarrow \emptyset \rightarrow v_7 \rightarrow v_3 \rightarrow v_1 \rightarrow \emptyset \rightarrow \emptyset \rightarrow \emptyset$$

  - Para alcanzar H_1 final, completar huecos en H_1 inicial con los genes/ciudades que faltan en el orden en que aparecen en P_1, pero si teniendo en cuanta los cambios de: $v_7 \rightarrow v_4$, $v_3 \rightarrow v_6$ y $v_1 \rightarrow v_2$.

    $$v_4 \rightarrow v_2 \rightarrow v_7 \rightarrow v_3 \rightarrow v_1 \rightarrow v_5 \rightarrow v_6 \rightarrow v_4$$

  - Para obtener H_2 :

  - Selección de genes/ciudades en posiciones 3, 4 y 5 en P_1 : v_4, v_6, v_2

  - Añadir esos genes/ciudades en P_2, sin el resto de genes/ciudades para dar lugar al H_2 inicial.

    $$\emptyset \rightarrow \emptyset \rightarrow v_4 \rightarrow v_6 \rightarrow v_2 \rightarrow \emptyset \rightarrow \emptyset \rightarrow \emptyset$$

  - Para alcanzar H_2 final, completar huecos en H_2 inicial con los genes/ciudades que faltan en el orden en que aparecen en P_2, pero si teniendo en cuanta los cambios de: $v_4 \rightarrow v_7$, $v_6 \rightarrow v_3$ y $v_2 \rightarrow v_1$.

    $$v_3 \rightarrow v_5 \rightarrow v_4 \rightarrow v_6 \rightarrow v_2 \rightarrow v_1 \rightarrow v_7 \rightarrow v_3$$

  Operador PMX sobre la pareja P_3, P_4

  - Para obtener H_3 :

  - Selección de genes/ciudades en posiciones 3, 4 y 5 en P_4 : v_5, v_4, v_6

- Añadir esos genes/ciudades en P_3, sin el resto de genes/ciudades para dar lugar al H_3 inicial.

$$\emptyset \rightarrow \emptyset \rightarrow v_5 \rightarrow v_4 \rightarrow v_6 \rightarrow \emptyset \rightarrow \emptyset \rightarrow \emptyset$$

- Para alcanzar H_3 final, completar huecos en H_3 inicial con los genes/ciudades que faltan en el orden en que aparecen en P_3, pero si teniendo en cuanta los cambios de: $v_5 \rightarrow v_7$, $v_4 \rightarrow v_3$ y $v_6 \rightarrow v_6$.

$$v_7 \rightarrow v_2 \rightarrow v_5 \rightarrow v_4 \rightarrow v_6 \rightarrow v_3 \rightarrow v_1 \rightarrow v_7$$

- Para obtener H_4 :

- Selección de genes/ciudades en posiciones 3, 4 y 5 en P_3 : v_7, v_3, v_6

- Añadir esos genes/ciudades en P_4, sin el resto de genes/ciudades para dar lugar al H_4 inicial.

$$\emptyset \rightarrow \emptyset \rightarrow v_7 \rightarrow v_3 \rightarrow v_6 \rightarrow \emptyset \rightarrow \emptyset \rightarrow \emptyset$$

- Para alcanzar H_4 final, completar huecos en H_4 inicial con los genes/ciudades que faltan en el orden en que aparecen en P_4, pero si teniendo en cuanta los cambios de: $v_7 \rightarrow v_5$, $v_3 \rightarrow v_4$ y $v_6 \rightarrow v_6$.

$$v_4 \rightarrow v_5 \rightarrow v_7 \rightarrow v_3 \rightarrow v_6 \rightarrow v_2 \rightarrow v_1 \rightarrow v_4$$

Operador PMX sobre la pareja P_5, P_6

- Para obtener H_5 :

- Selección de genes/ciudades en posiciones 3, 4 y 5 en P_6 : v_7, v_2, v_1

- Añadir esos genes/ciudades en P_5, sin el resto de genes/ciudades para dar lugar al H_5 inicial.

$$\emptyset \rightarrow \emptyset \rightarrow v_7 \rightarrow v_2 \rightarrow v_1 \rightarrow \emptyset \rightarrow \emptyset \rightarrow \emptyset$$

- Para alcanzar H_5 final, completar huecos en H_5 inicial con los genes/ciudades que faltan en el orden en que aparecen en P_5, pero si teniendo en cuanta los cambios de: $v_7 \rightarrow v_7$, $v_2 \rightarrow v_6$ y $v_1 \rightarrow v_2$.

$$v_4 \rightarrow v_5 \rightarrow v_7 \rightarrow v_2 \rightarrow v_1 \rightarrow v_6 \rightarrow v_3 \rightarrow v_4$$

- Para obtener H_6 :

- Selección de genes/ciudades en posiciones 3, 4 y 5 en P_5 : v_7, v_6, v_2

- Añadir esos genes/ciudades en P_6, sin el resto de genes/ciudades para dar lugar al H_6 inicial.

$$\emptyset \rightarrow \emptyset \rightarrow v_7 \rightarrow v_6 \rightarrow v_2 \rightarrow \emptyset \rightarrow \emptyset \rightarrow \emptyset$$

- Para alcanzar H_6 final, completar huecos en H_6 inicial con los genes/ciudades que faltan en el orden en que aparecen en P_6, pero si teniendo en cuanta los cambios de: $v_7 \rightarrow v_7$, $v_6 \rightarrow v_2$ y $v_2 \rightarrow v_1$.

$$v_3 \rightarrow v_5 \rightarrow v_7 \rightarrow v_6 \rightarrow v_2 \rightarrow v_4 \rightarrow v_1 \rightarrow v_3$$

Resultado del cruce:

Rutas / Cromosomas Hijos	Genes / Ciudades
H_1	$v_4 \rightarrow v_2 \rightarrow v_7 \rightarrow v_3 \rightarrow v_1 \rightarrow v_5 \rightarrow v_6 \rightarrow v_4$
H_2	$v_3 \rightarrow v_5 \rightarrow v_4 \rightarrow v_6 \rightarrow v_2 \rightarrow v_1 \rightarrow v_7 \rightarrow v_3$
H_3	$v_7 \rightarrow v_2 \rightarrow v_5 \rightarrow v_4 \rightarrow v_6 \rightarrow v_3 \rightarrow v_1 \rightarrow v_7$
H_4	$v_4 \rightarrow v_5 \rightarrow v_7 \rightarrow v_3 \rightarrow v_6 \rightarrow v_2 \rightarrow v_1 \rightarrow v_4$
H_5	$v_4 \rightarrow v_5 \rightarrow v_7 \rightarrow v_2 \rightarrow v_1 \rightarrow v_6 \rightarrow v_3 \rightarrow v_4$
H_6	$v_3 \rightarrow v_5 \rightarrow v_7 \rightarrow v_6 \rightarrow v_2 \rightarrow v_4 \rightarrow v_1 \rightarrow v_3$

- 1.3: Mutación de rutas/cromosomas.

  Operador DM sobre cada H_k, extrayendo los genes/ciudades en las posiciones 2,3 y 4 y reubicarlos después en la posición 4.

  Resultado de la Mutación

Rutas / Cromosomas Mutados	Genes / Ciudades	Distancia
R_1	$v_4 \rightarrow v_1 \rightarrow v_5 \rightarrow v_6 \rightarrow v_2 \rightarrow v_7 \rightarrow v_3 \rightarrow v_4$	50808
R_2	$v_3 \rightarrow v_2 \rightarrow v_1 \rightarrow v_7 \rightarrow v_5 \rightarrow v_4 \rightarrow v_6 \rightarrow v_3$	84718
R_3	$v_7 \rightarrow v_6 \rightarrow v_3 \rightarrow v_1 \rightarrow v_2 \rightarrow v_5 \rightarrow v_4 \rightarrow v_7$	69591
R_4	$v_4 \rightarrow v_6 \rightarrow v_2 \rightarrow v_1 \rightarrow v_5 \rightarrow v_7 \rightarrow v_3 \rightarrow v_4$	65057
R_5	$v_4 \rightarrow v_1 \rightarrow v_6 \rightarrow v_3 \rightarrow v_5 \rightarrow v_7 \rightarrow v_2 \rightarrow v_4$	90330
R_6	$v_3 \rightarrow v_2 \rightarrow v_4 \rightarrow v_1 \rightarrow v_5 \rightarrow v_7 \rightarrow v_6 \rightarrow v_3$	87219

A continuación sobre estas rutas se repiten los mismos cálculos, continuando sucesivamente con los pasos $i = 2,3, \ldots$ La siguiente tabla resume los resultados alcanzados hasta el paso $i = 17$,

Paso i	Genes / Ciudades de distancia mínima	Distancia mínima	Tiempo de cómputo
1	$v_4 \rightarrow v_1 \rightarrow v_5 \rightarrow v_6 \rightarrow v_2 \rightarrow v_7 \rightarrow v_3 \rightarrow v_4$	50808	0
2	$v_3 \rightarrow v_5 \rightarrow v_2 \rightarrow v_4 \rightarrow v_6 \rightarrow v_1 \rightarrow v_7 \rightarrow v_3$	62505	0,002
3	$v_4 \rightarrow v_1 \rightarrow v_6 \rightarrow v_2 \rightarrow v_5 \rightarrow v_7 \rightarrow v_3 \rightarrow v_4$	52390	0,003
4	$v_4 \rightarrow v_3 \rightarrow v_6 \rightarrow v_5 \rightarrow v_2 \rightarrow v_7 \rightarrow v_1 \rightarrow v_4$	56622	0,007
5	$v_4 \rightarrow v_3 \rightarrow v_7 \rightarrow v_1 \rightarrow v_5 \rightarrow v_6 \rightarrow v_2 \rightarrow v_4$	49983	0,007
6	$v_3 \rightarrow v_5 \rightarrow v_2 \rightarrow v_6 \rightarrow v_7 \rightarrow v_1 \rightarrow v_4 \rightarrow v_3$	63564	0,01
7	$v_3 \rightarrow v_2 \rightarrow v_7 \rightarrow v_1 \rightarrow v_5 \rightarrow v_6 \rightarrow v_4 \rightarrow v_3$	64578	0,007
8	$v_6 \rightarrow v_2 \rightarrow v_1 \rightarrow v_3 \rightarrow v_7 \rightarrow v_5 \rightarrow v_4 \rightarrow v_6$	67846	0,007
9	$v_4 \rightarrow v_7 \rightarrow v_5 \rightarrow v_6 \rightarrow v_2 \rightarrow v_1 \rightarrow v_3 \rightarrow v_4$	55517	0,008
10	$v_4 \rightarrow v_3 \rightarrow v_1 \rightarrow v_6 \rightarrow v_2 \rightarrow v_5 \rightarrow v_7 \rightarrow v_4$	56238	0,01
11	$v_7 \rightarrow v_3 \rightarrow v_5 \rightarrow v_6 \rightarrow v_1 \rightarrow v_2 \rightarrow v_4 \rightarrow v_7$	58794	0,01
12	$v_5 \rightarrow v_2 \rightarrow v_6 \rightarrow v_7 \rightarrow v_4 \rightarrow v_1 \rightarrow v_3 \rightarrow v_5$	58420	0,014
13	$v_1 \rightarrow v_7 \rightarrow v_3 \rightarrow v_6 \rightarrow v_5 \rightarrow v_2 \rightarrow v_4 \rightarrow v_1$	50935	0,01
14	$v_6 \rightarrow v_5 \rightarrow v_2 \rightarrow v_1 \rightarrow v_4 \rightarrow v_3 \rightarrow v_7 \rightarrow v_6$	46463	0,01
15	$v_3 \rightarrow v_4 \rightarrow v_6 \rightarrow v_1 \rightarrow v_5 \rightarrow v_2 \rightarrow v_7 \rightarrow v_3$	60069	0,01

16	$v_2 \to v_5 \to v_1 \to v_3 \to v_7 \to v_4 \to v_6 \to v_2$	48337	0,011
17	$v_3 \to v_2 \to v_5 \to v_6 \to v_1 \to v_4 \to v_7 \to v_3$	41881	0,012

Se aprecian también con estos operadores aumentos y disminuciones de la distancia mínima alcanzada, y al llegar al paso $i = 17$ aparece ya una ruta óptima que coincide con la obtenida por fuerza bruta y por el algoritmo del vecino más cercano, con distancia mínima 41.881 kilómetros.

Análisis del algoritmo.

La solución alcanzada en el algoritmo genético no tiene porqué dar la distancia global óptima, por varios motivos:

- En cada paso la distancia total alcanzada puede aumentar o disminuir respecto al paso anterior.
- El algoritmo puede entrar en un bucle donde se repiten resultados de pasos anteriores.
- Los resultados dependen de los operadores de cruce y mutación elegidos

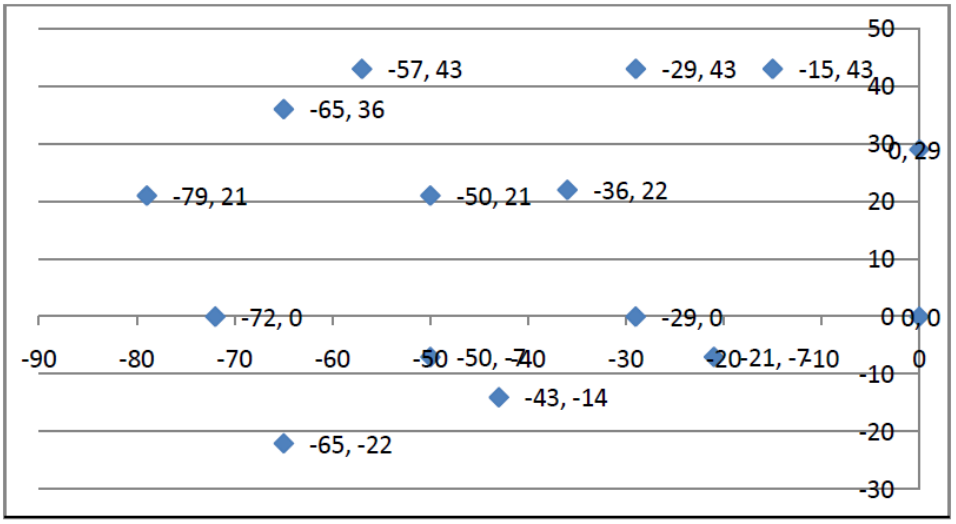
Para encontrar una solución realmente óptima, es necesario que esos operadores de cruce y mutación sean los adecuados para el problema en cuestión, al igual que ocurre en la genética de cada especie de seres vivos, donde sus procesos evolutivos son específicos de cada especie.

4. COMPARACIÓN CON OTROS ESTUDIOS

Para esta comparación, además de los algoritmos propios programados en RStudio, se utilizará adicionalmente el "GA package" desarrollado en RStudio por Luca Scrucca [SL-2013], en su actualización de enero de 2024, con sus opciones por defecto: número de rutas iniciales según número de vértices/ciudades (generadas aleatoriamente para cada prueba realizada), operador de cruce "gabin_spCrossover" (single point) y operador de mutación "gabin_raMutation" (random aleatory).

4.1. Problema de las 15 ciudades **[GP, 2013]**

El problema de las 15 ciudades aparece en números estudios científicos sobre el TSP. Los datos de entrada se toman en particular del estudio realizado por Gupta y Panwar en 2013.

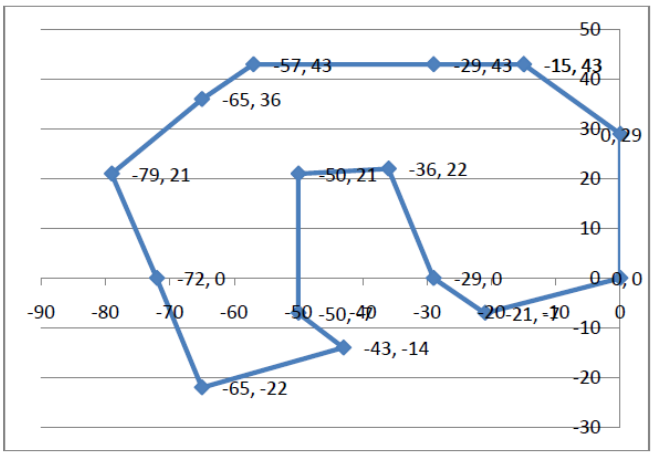


Coordenadas planas de los vértices de las 15 ciudades

City	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	29	82	46	68	52	72	42	51	55	29	74	23	72	46
2		0	55	46	42	43	43	23	23	31	41	51	11	52	21
3			0	68	46	55	23	43	41	29	79	21	64	31	51
4				0	82	15	72	31	62	42	21	51	51	43	64
5					0	74	23	52	21	46	82	58	46	65	23
6						0	61	23	55	31	33	37	51	29	59
7							0	42	23	31	77	37	51	46	33
8								0	33	15	37	33	33	31	37
9									0	29	62	46	29	51	11
10										0	51	21	41	23	37
11											0	65	42	59	61
12												0	61	11	55
13													0	62	23
14														0	59
15															0

Distancias entre las 15 ciudades, utilizando la distancia euclídea

Para este problema la mejor distancia óptima conocida es de 291, utilizando el algoritmo GA con operador de cruce PMX de 2 genes/ciudades y mutación EM de 2 genes/ciudades.



A continuación, aplicaremos los algoritmos NN con diferentes vértices/ciudades de inicio y GA con operador de cruce PMX y mutación DM de diversos números (de 3 a 6) genes/ciudades.

4.1.1. Resolución con el algoritmo NN

Aplicando el algoritmo del vecino más cercano (NN), y tomando como vértices iniciales cada una de las ciudades, se alcanzan los siguientes resultados:

Vértice inicial	Ruta de ciudades de distancia mínima	Distancia mínima	Tiempo de cómputo
1	1 → 13 → 2 → 15 → 9 → 5 → 7 → 3 → 12 → 10 → 8 → 6 → 4 → 11 → 14 → 1	388	0,001
2	2 → 13 → 14 → 10 → 8 → 6 → 4 → 11 → 1 → 15 → 9 → 5 → 7 → 3 → 12 → 2	352	0,001
3	3 → 12 → 10 → 8 → 2 → 13 → 14 → 6 → 4 → 11 → 1 → 15 → 9 → 5 → 7 → 3	328	0,001
4	4 → 6 → 8 → 10 → 12 → 3 → 7 → 5 → 9 → 15 → 2 → 13 → 14 → 11 → 1 → 4	358	0,001
5	5 → 7 → 3 → 12 → 10 → 8 → 2 → 13 → 14 → 6 → 4 → 11 → 1 → 15 → 9 → 5	328	0,001
6	6 → 4 → 11 → 1 → 13 → 2 → 15 → 9 → 5 → 7 → 3 → 12 → 10 → 8 → 14 → 6	329	0,001
7	7 → 3 → 12 → 10 → 8 → 2 → 13 → 14 → 6 → 4 → 11 → 1 → 15 → 9 → 5 → 7	328	0,001
8	8 → 10 → 12 → 3 → 7 → 5 → 9 → 15 → 2 → 13 → 14 → 6 → 4 → 11 → 1 → 8	322	0,001
9	9 → 15 → 2 → 13 → 14 → 10 → 8 → 6 → 4 → 11 → 1 → 5 → 7 → 3 → 12 → 9	367	0,001
10	10 → 8 → 2 → 13 → 14 → 12 → 3 → 7 → 5 → 9 → 15 → 4 → 6 → 11 → 1 → 10	345	0,001
11	11 → 4 → 6 → 8 → 10 → 12 → 3 → 7 → 5 → 9 → 15 → 2 → 13 → 14 → 1 → 11	346	0,001
12	12 → 3 → 7 → 5 → 9 → 15 → 2 → 13 → 14 → 10 → 8 → 6 → 4 → 11 → 1 → 12	350	0,001
13	13 → 2 → 15 → 9 → 5 → 7 → 3 → 12 → 10 → 8 → 6 → 4 → 11 → 1 → 14 → 13	346	0,001
14	14 → 13 → 2 → 15 → 9 → 5 → 7 → 3 → 12 → 10 → 8 → 6 → 4 → 11 → 1 → 14	346	0,001
15	15 → 9 → 2 → 13 → 14 → 10 → 8 → 6 → 4 → 11 → 1 → 5 → 7 → 3 → 12 → 15	385	0,001

La mejor solución es la que se inicia en el vértice/ciudad 8, con una distancia de 322, lejos de la mejor conocida de 291.

4.1.2. Resolución con el algoritmo GA

En la aplicación del algoritmo genético GA, con el operador de cruce PMX y mutación DM, se realizan los cálculos a partir de diferentes rutas aleatorias iniciales, diferentes posiciones de diferentes genes/ciudades a intercambiar tanto en el operador de cruce como en el operador de mutación. Tras repetir el bucle de selección, cruce y mutación durante $i = 30.000$ pasos / ciclos evolutivos, los mejores resultados en cada una de las pruebas son los siguientes.

Prueba	Paso i	Genes / Ciudades de distancia mínima	Distancia mínima	Tiempo de cómputo
A	8543	$13 \to 2 \to 15 \to 5 \to 9 \to 14 \to 12 \to 8 \to 7 \to 3 \to 10 \to 6 \to 4 \to 11 \to 1 \to 13$	398	25,342
B	9599	$15 \to 9 \to 5 \to 7 \to 3 \to 10 \to 12 \to 14 \to 13 \to 8 \to 11 \to 4 \to 6 \to 1 \to 2 \to 15$	376	32,964
C	20668	$11 \to 4 \to 6 \to 8 \to 15 \to 9 \to 2 \to 5 \to 7 \to 12 \to 3 \to 10 \to 14 \to 13 \to 1 \to 11$	374	72,373
D	21653	$4 \to 15 \to 9 \to 10 \to 8 \to 12 \to 3 \to 7 \to 5 \to 2 \to 6 \to 14 \to 13 \to 1 \to 11 \to 4$	382	76,462
E	18760	$1 \to 11 \to 4 \to 6 \to 10 \to 3 \to 12 \to 7 \to 9 \to 5 \to 15 \to 2 \to 13 \to 14 \to 8 \to 1$	374	36,476
F	1255	$9 \to 5 \to 15 \to 2 \to 13 \to 1 \to 11 \to 6 \to 8 \to 10 \to 4 \to 12 \to 14 \to 3 \to 7 \to 9$	394	2,041
G	3582	$14 \to 13 \to 2 \to 1 \to 9 \to 15 \to 4 \to 11 \to 6 \to 8 \to 10 \to 5 \to 7 \to 12 \to 3 \to 14$	392	5,52
H	14884	$15 \to 5 \to 2 \to 8 \to 10 \to 12 \to 3 \to 7 \to 9 \to 13 \to 14 \to 6 \to 4 \to 11 \to 1 \to 15$	371	25,952
I	17510	$8 \to 14 \to 13 \to 2 \to 9 \to 10 \to 6 \to 4 \to 11 \to 1 \to 15 \to 5 \to 7 \to 3 \to 12 \to 8$	376	32,319
J	23779	$8 \to 14 \to 13 \to 2 \to 9 \to 10 \to 12 \to 3 \to 7 \to 5 \to 15 \to 4 \to 11 \to 1 \to 6$	370	45,085

La mejor solución es la obtenida en la prueba J, con una distancia de 370, lejos tanto de la alcanzada con el algoritmo NN (322) como de la mejor conocida (291).

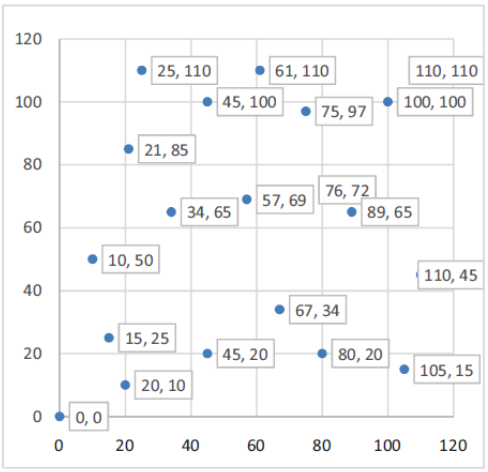
Como alternativa, podemos contrastar estos resultados con los que obtiene el GA package de RStudio en sus opciones por defecto (en este caso, 50 rutas iniciales en cada prueba):

Prueba	Paso i	Ruta de genes / ciudades de distancia mínima	Distancia mínima	Tiempo de cómputo
α	441	$9 \to 15 \to 2 \to 13 \to 1 \to 11 \to 4 \to 6 \to 8 \to 10 \to 14 \to 12 \to 3 \to 7 \to 5 \to 9$	291	2,097

Así, con esta configuración del algoritmo genético la mejor solución obtenida de distancia mínima 291 no sólo mejora la alcanzada con el algoritmo genético, sino que coincide con la mejor conocida.

4.2. Problema de las 20 ciudades **[JSMC, 2019]**

El problema de las 20 ciudades es también un referente en los estudios sobre el TSP. Para tomar los datos de entrada en este caso se acude al estudio realizado por Junejal Saraswat, Singh, Sharma, Majumdar, Chowdhary en 2019.

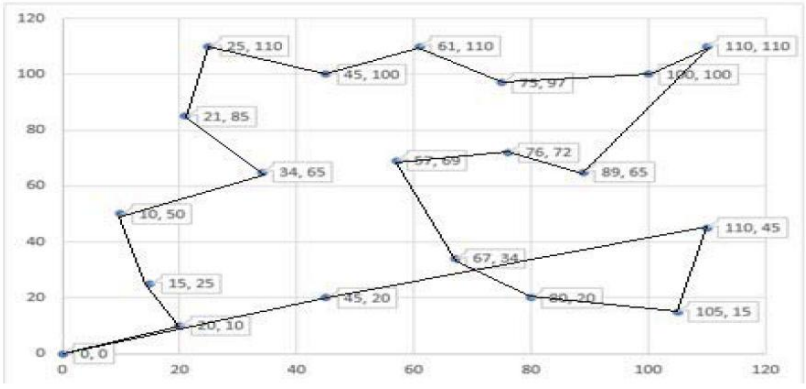


Coordenadas planas de los vértices de las 20 ciudades

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P 10	P 11	P 12	P 13	P 14	P 15	P 16	P 17	P 18	P 19	P 20
P1	0	22	49	29	51	82	89	75	105	73	110	88	126	110	123	141	113	119	156	106
P2		0	27	16	41	61	70	53	84	57	88	75	108	93	103	120	100	97	135	85
P3			0	30	46	35	50	26	61	46	63	69	91	80	83	97	92	70	111	60
P4				0	25	65	61	53	77	44	84	60	97	81	94	113	86	97	127	91
P5					0	76	51	59	70	28	80	37	79	61	80	103	62	100	117	101
P6						0	54	19	52	64	46	88	92	87	77	82	105	39	95	25
P7							0	36	10	23	32	39	41	33	33	53	52	58	67	72
P8								0	39	45	38	69	76	70	64	74	87	44	87	42
P9									0	43	15	57	41	42	25	37	64	43	51	64
P10										0	55	24	52	37	52	75	46	79	88	91
P11											0	71	53	56	35	37	78	29	50	52
P12												0	47	28	55	80	25	98	92	109
P13													0	19	19	40	36	81	49	105
P14														0	30	55	92	85	66	104
P15															0	25	52	63	37	87
P16																0	76	56	14	85
P17																	0	107	85	125
P18																		0	65	30
P19																			0	95
P20																				0

Distancias entre las 20 ciudades, utilizando la distancia euclídea

Para este problema la mejor distancia óptima conocida es de 552, utilizando el algoritmo GA con operador de cruce PMX de m genes/ciudades y mutación EM de 2 genes/ciudades.



A continuación, aplicaremos los algoritmos NN con diferentes vértices/ciudades de inicio y GA con operador de cruce PMX y mutación DM de diversos números (de 4 a 9) genes/ciudades.

4.2.1. Resolución con el algoritmo NN

Aplicando el algoritmo del vecino más cercano (NN), y tomando como vértices inicial cada una de las ciudades, se alcanzan los siguientes resultados:

Vértice inicial	Ruta de ciudades de distancia mínima	Distancia mínima	Tiempo de cómputo
1	$1 \to 2 \to 4 \to 5 \to 10 \to 7 \to 9 \to 11 \to 18 \to 20 \to 6 \to 8 \to 3 \to 12 \to 17 \to 13 \to 14 \to 15 \to 16 \to 19 \to 1$	642	0,001
2	$2 \to 4 \to 5 \to 10 \to 7 \to 9 \to 11 \to 18 \to 20 \to 6 \to 8 \to 3 \to 1 \to 12 \to 17 \to 13 \to 14 \to 15 \to 16 \to 19 \to 2$	667	0,001
3	$3 \to 8 \to 6 \to 20 \to 18 \to 11 \to 9 \to 7 \to 10 \to 12 \to 17 \to 13 \to 14 \to 15 \to 16 \to 19 \to 5 \to 4 \to 2 \to 1 \to 3$	579	0,001
4	$4 \to 2 \to 1 \to 3 \to 8 \to 6 \to 20 \to 18 \to 11 \to 9 \to 7 \to 10 \to 12 \to 17 \to 13 \to 14 \to 15 \to 16 \to 19 \to 5 \to 4$	579	0,001
5	$5 \to 4 \to 2 \to 1 \to 3 \to 8 \to 6 \to 20 \to 18 \to 11 \to 9 \to 7 \to 10 \to 12 \to 17 \to 13 \to 14 \to 15 \to 16 \to 19 \to 5$	579	0,001
6	$6 \to 8 \to 3 \to 2 \to 4 \to 5 \to 10 \to 7 \to 9 \to 11 \to 18 \to 20 \to 16 \to 19 \to 15 \to 13 \to 14 \to 12 \to 17 \to 1 \to 6$	670	0,001
7	$7 \to 9 \to 11 \to 18 \to 20 \to 6 \to 8 \to 3 \to 2 \to 4 \to 5 \to 10 \to 12 \to 17 \to 13 \to 14 \to 15 \to 16 \to 19 \to 1 \to 7$	668	0,001
8	$8 \to 6 \to 20 \to 18 \to 11 \to 9 \to 7 \to 10 \to 12 \to 17 \to 13 \to 14 \to 15 \to 16 \to 19 \to 3 \to 2 \to 4 \to 5 \to 1 \to 8$	629	0,001
9	$9 \to 7 \to 10 \to 12 \to 17 \to 13 \to 14 \to 15 \to 16 \to 19 \to 11 \to 18 \to 20 \to 6 \to 8 \to 3 \to 2 \to 4 \to 5 \to 1 \to 9$	609	0,001
10	$10 \to 7 \to 9 \to 11 \to 18 \to 20 \to 6 \to 8 \to 3 \to 2 \to 4 \to 5 \to 12 \to 17 \to 13 \to 14 \to 15 \to 16 \to 19 \to 1 \to 10$	660	0,001
11	$11 \to 9 \to 7 \to 10 \to 12 \to 17 \to 13 \to 14 \to 15 \to 16 \to 19 \to 18 \to 20 \to 6 \to 8 \to 3 \to 2 \to 4 \to 5 \to 1 \to 11$	615	0,001
12	$12 \to 10 \to 7 \to 9 \to 11 \to 18 \to 20 \to 6 \to 8 \to 3 \to 2 \to 4 \to 5 \to 1 \to 14 \to 13 \to 15 \to 16 \to 19 \to 17 \to 12$	617	0,001
13	$13 \to 14 \to 12 \to 10 \to 7 \to 9 \to 11 \to 18 \to 20 \to 6 \to 8 \to 3 \to 2 \to 4 \to 5 \to 1 \to 17 \to 15 \to 16 \to 19 \to 13$	620	0,001
14	$14 \to 13 \to 15 \to 9 \to 7 \to 10 \to 12 \to 17 \to 5 \to 4 \to 2 \to 1 \to 3 \to 8 \to 6 \to 20 \to 18 \to 11 \to 16 \to 19 \to 14$	565	0,001
15	$15 \to 13 \to 14 \to 12 \to 10 \to 7 \to 9 \to 11 \to 18 \to 20 \to 6 \to 8 \to 3 \to 2 \to 4 \to 5 \to 1 \to 17 \to 16 \to 19 \to 15$	626	0,001
16	$16 \to 19 \to 15 \to 13 \to 14 \to 12 \to 10 \to 7 \to 9 \to 11 \to 18 \to 20 \to 6 \to 8 \to 3 \to 2 \to 4 \to 5 \to 1 \to 17 \to 16$	626	0,001
17	$17 \to 12 \to 10 \to 7 \to 9 \to 11 \to 18 \to 20 \to 6 \to 8 \to 3 \to 2 \to 4 \to 5 \to 1 \to 14 \to 13 \to 15 \to 16 \to 19 \to 17$	617	0,001
18	$18 \to 11 \to 9 \to 7 \to 10 \to 12 \to 17 \to 13 \to 14 \to 15 \to 16 \to 19 \to 8 \to 6 \to 20 \to 3 \to 2 \to 4 \to 5 \to 1 \to 18$	679	0,001
19	$19 \to 16 \to 15 \to 13 \to 14 \to 12 \to 10 \to 7 \to 9 \to 11 \to 18 \to 20 \to 6 \to 8 \to 3 \to 2 \to 4 \to 5 \to 1 \to 17 \to 19$	623	0,001
20	$20 \to 6 \to 8 \to 3 \to 2 \to 4 \to 5 \to 10 \to 7 \to 9 \to 11 \to 18 \to 16 \to 19 \to 15 \to 13 \to 14 \to 12 \to 17 \to 1 \to 20$	660	0,001

La mejor solución es la que se inicia en el vértice/ciudad 14, con una distancia de 565, lejos de la mejor conocida de 552.

4.2.2. Resolución con el algoritmo GA

También aquí se aplicará el algoritmo genético GA, con el operador de cruce PMX y mutación DM, realizando los cálculos tomando diferentes rutas aleatorias iniciales, diferentes posiciones de diferentes genes/ciudades a intercambiar en los operadores, repitiendo el bucle durante $i = 30.000$ pasos / ciclos evolutivos. A continuación, se presentan los mejores resultados en cada una de las pruebas.

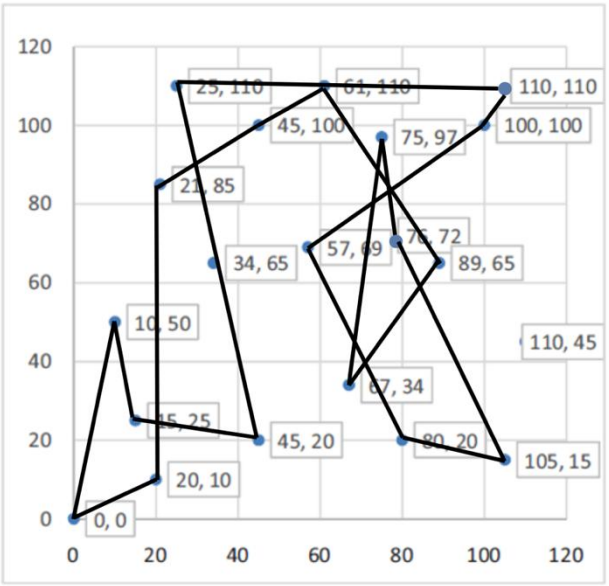
Prueba	Paso i	Genes / Ciudades de distancia mínima	Distancia mínima	Tiempo de cómputo
A	17407	9 → 7 → 1 → 3 → 2 → 4 → 5 → 10 → 12 → 17 → 15 → 14 → 13 → 6 → 11 → 19 → 16 → 18 → 20 → 8 → 9	763	32,394
B	12216	5 → 3 → 4 → 2 → 8 → 20 → 6 → 1 → 10 → 11 → 16 → 9 → 7 → 14 → 13 → 15 → 18 → 19 → 17 → 1 → 5	852	27,207
C	7358	16 → 15 → 14 → 13 → 17 → 12 → 2 → 10 → 5 → 4 → 3 → 1 → 9 → 19 → 7 → 18 → 20 → 6 → 8 → 11 → 16	829	16,087
D	9029	14 → 10 → 16 → 19 → 13 → 12 → 5 → 2 → 4 → 1 → 3 → 6 → 7 → 9 → 11 → 18 → 20 → 8 → 17 → 15 → 14	778	17,369
E	7354	12 → 1 → 3 → 5 → 11 → 16 → 19 → 7 → 8 → 18 → 20 → 6 → 2 → 4 → 10 → 9 → 14 → 15 → 13 → 17 → 12	832	15,331
F	19338	6 → 20 → 18 → 16 → 19 → 13 → 17 → 15 → 11 → 9 → 8 → 5 → 1 → 2 → 12 → 4 → 10 → 14 → 7 → 3 → 6	817	45,309
G	22677	18 → 16 → 19 → 11 → 14 → 12 → 4 → 5 → 1 → 2 → 6 → 20 → 8 → 3 → 10 → 7 → 13 → 17 → 15 → 9 → 18	782	49,735
H	9018	18 → 6 → 9 → 7 → 14 → 12 → 10 → 17 → 13 → 11 → 15 → 16 → 19 → 20 → 8 → 1 → 4 → 5 → 2 → 3 → 18	799	19,479
I	22314	7 → 14 → 12 → 5 → 10 → 17 → 9 → 13 → 15 → 11 → 16 → 19 → 18 → 6 → 8 → 2 → 1 → 4 → 3 → 20 → 7	771	53,487
J	21168	14 → 13 → 16 → 19 → 11 → 6 → 20 → 7 → 9 → 18 → 8 → 3 → 4 → 1 → 5 → 10 → 2 → 15 → 17 → 12 → 14	792	49,479

La mejor solución es la obtenida en la prueba A, con una distancia de 763, lejos tanto de la alcanzada con el algoritmo NN (565) como de la mejor conocida (552).

Se pasa ahora a realizar los cómputos con GA package de Rstudio en sus opciones por defecto (en este caso toma 100 rutas iniciales en cada prueba).

Prueba	Paso i	Genes / Ciudades de distancia mínima	Distancia mínima	Tiempo de cómputo
α	500	3 → 2 → 1 → 4 → 5 → 10 → 12 → 17 → 13 → 14 → 7 → 9 → 15 → 19 → 16 → 11 → 18 → 20 → 6 → 8 → 3	520	5,192
β	484	17 → 13 → 19 → 16 → 15 → 14 → 7 → 9 → 11 → 18 → 20 → 6 → 8 → 3 → 2 → 1 → 4 → 5 → 10 → 12 → 17	521	4,862
χ	282	5 → 10 → 12 → 17 → 19 → 16 → 15 → 13 → 14 → 7 → 9 → 11 → 18 → 20 → 6 → 8 → 3 → 2 → 1 → 4 → 5	529	2,707
δ	500	2 → 3 → 8 → 6 → 20 → 18 → 11 → 16 → 19 → 9 → 7 → 14 → 15 → 13 → 17 → 12 → 10 → 5 → 4 → 1 → 2	539	5,333
ε	445	9 → 7 → 16 → 19 → 15 → 13 → 14 → 12 → 17 → 10 → 5 → 4 → 1 → 2 → 3 → 8 → 6 → 20 → 18 → 11 → 9	526	4,002

Se concluye así una configuración del algoritmo GA con una ruta óptima de 520, mejorando las encontradas anteriormente.



Ruta óptima

5. CONCLUSIONES

Tras estudiar la resolución del problema del viajante (TSP) con el algoritmo heurístico constructivo del vecino más cercano (NN) y el algoritmo metaheurístico de población genético (GA), realizar su programación en Rstudio y aplicarlos con diferentes configuraciones a tres problemas concretos, uno de construcción propia (7 nuevas maravillas del mundo) y dos tomados de otros estudios científicos (problemas de las 15 y de las 20 ciudades), la tabla resumen de resultados obtenidos es la siguiente:

PROBLEMA	Distancia mínima de referencia	ALGORITMO	Configuración Operadores	Distancia mínima alcanzada	Tiempo de cómputo
7 maravillas	41.881	NN	Inicio 1	41.881	0,000
			Inicio 2	41.881	0,000
		GA	OX, EM	46.463	0,026
			PMX, DM	41.881	0,012
15 ciudades	291	NN	Inicio 8	322	0,001
		GA	PMX, DM	370	45,085
			GA package	291	2,097
20 ciudades	552	NN	Inicio 14	565	0,001
		GA	PMX, DM	763	32,394
			GA package	520	5,192
			GA package	521	4,862
			GA package	529	2,707
			GA package	539	5,333
			GA package	526	4,002

Estos resultados muestran las ventajas e inconvenientes de cada uno de estos dos algoritmos:

- El algoritmo del vecino más cercano tiene poca complejidad, como muestran sus tiempos de cómputo, frente al algoritmo genético cuya complejidad es mucho mayor (del orden de 45.000 veces más en el caso de las 15 ciudades).
- El algoritmo del vecino más cercano está fuertemente condicionado por la ciudad de inicio elegida.
- El algoritmo genético está fuertemente condicionado tanto por los operadores de cruce y mutación elegidos, como por la elección de los genes/ciudades a cruzar y mutar (con un mismo operador) y de los cromosomas/rutas iniciales:
    - El número de pasos (ciclos genéticos) para alcanzar la solución óptima depende de los operadores y genes/ciudades elegidos.
    - La solución óptima alcanzada puede también variar sustancialmente.

En definitiva, cuanto se trata de encontrar la solución a un TSP específico, la elección del algoritmo a utilizar depende del balance entre la necesidad de una resolución

rápida y la calidad de la solución requerida. Para problemas que tengan una menor escala o cuando se necesita una solución inmediata, el algoritmo NN es adecuado, sin embargo, en situaciones donde la precisión de la solución es crítica y se dispone de tiempo de cómputo, el algoritmo GA ofrece soluciones más eficaces, que a su vez dependen tanto de los operadores concretos elegidos como de las rutas iniciales elegidas para desarrollar el algoritmo.

BIBLIOGRAFÍA.

[AAT, 2016] Arora, K., Agarwal, S., Tanwar, R. "*Solving TSP using Genetic Algorithm and Nearest Neighbour Algorithm and their Comparison*". International Journal of Scientific & Engineering Research, 7 (2016), páginas 1014-1018.

[AB, 2017] Alsalibi, B., Babaeianjelodar, M., Venkat, I. (2017). A Comparative Study between the Nearest Neighbor and Genetic Algorithms: A revisit to the Traveling Salesman Problem. International Journal of Computer Science and Electronics Engineering (IJCSEE) Volume 1, Issue 1 (2013), páginas 34-38.

[AMI, 2008] Moujahid, A., Inza, I., Larrañaga, P. "*Tema 2. Algoritmos Genéticos*" en "*Apuntes de Métodos Matemáticos en Ciencias de la Computación*", Departamento de Ciencias de la Computación e Inteligencia Artificial, Universidad del País Vasco,(2007).

[GD, 2022] González Domínguez, Gabriel: "*Problemas de flujo multiservicio: resolución del problema de trazado de canalizaciones en el interior de un buque*". Trabajo Fin de Máster, Depósito de Investigación de la Universidad de Sevilla, "idUS" (2022).

[GP, 2013] Gupta, S., Panwar, P. (2013*). "Solving Travelling Salesman Problem Using Genetic Algorithm".* International Journal of Advanced Researcha in Computer Science and Software Technology, 3-6 (2013), páginas 376-380.

[HJ, 1975] Holland, J. H. "*Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence.* U Michigan Press (1975).

[HK, 2000] Hoffman, Karla Leigh, "*Combinatorial optimization: Current successes and directions for the future*", en Journal of Computational and Applied Mathematics 124 (2000), páginas 341–360.

[HW, 2004] Hurkens C. A. J., Woeginger, G. J. (2004). "*On the nearest neighbor rule for the traveling salesman problem*". Operations Research Letters, *32* (2004), páginas 1–4.

[ID, 2018] Infantes Durán, Miguel: "*El problema del Viajante (TSP)"*.Trabajo Fin de Grado, Depósito de Investigación de la Universidad de Sevilla, "idUS" (2018).

[JSMC, 2019] Sahib Singh Juneja1, Pavi Saraswat, Kshitij Singh, Jatin Sharma, Rana Majumdar, Sunil Chowdhary: "Travelling Salesman Problem Optimization Using Genetic Algoritm", Proceeding of the Amity International Conference on Artificial Intelligence (AICAI). Institute of Electrical and Electronics Engineers (IEEE). (2019), pp. 264-268.

[MB, 2015] Morán Bermúdez, Noemí: "*Desarrollo y aplicación del algoritmo PSO al problema del ATSP*". Trabajo Fin de Grado, Repositorio Documental de la Universidad de Valladolid, "UVaDoc", (2015).

[MT, 2020] Montoya Torres, Laura: "*Una aplicación del problema del viajante de comercio a la distribución del dinero en efectivo en la región de Murcia*". Trabajo Fin de Grado. Repositorio Digital de la Universidad Politécnica de Cartagena (2020).

[RO, 2010] Rodríguez Ortiz, Carlos: "*Algoritmos Heurísticos y metaheurísticos para el problema de localización de regeneradores*". Trabajo Fin de Carrera, Repositorio Institucional de la Universidad Rey Juan Carlos, "BURJC-Digital", (2010).

[SL 2013] Scrucca, Luca "*GA: A Package for Genetic Algorithms in R*." *Journal of Statistical Software*, 53-4 (1013), páginas 1–37.

[SM, 2021] Sebastián Martínez-Cava, C. "*El Problema del Viajante, heurísticas basadas en algoritmos genéticos"*. Repositorio Digital de la Universidad Complutense, "Docta" (2021).

[VH, 2019] Velasco Heras, J. M. "*Reconstrucción de trayectorias de aeronaves usando heurísticas de mejora para resolver una versión del problema del viajante (TSP)"* Repositorio Documental de la Universidad de Valladolid, "UVaDoc" (2019).

ANEXOS

Ficheros de programación en Rstudio de los algoritmos utilizados

- Problema de las 7 nuevas maravillas del mundo:

  - Algoritmo fuerza bruta:
    Método de fuerza bruta para las 7 maravillas.R

  - Algoritmo genético con operadores OX2 y EM:
    GA, cruce OX2 y mutación EM para las 7 maravillas.R

  - Algoritmo genético con operadores PMX y DM:
    GA, cruce PMX y mutación DM para las 7 maravillas .R

- Problema de las 15 ciudades:

  - Algoritmo vecino más cercano:
    Vecino más cercano para 15 ciudades.R

  - Algoritmo genético con operadores PMX y DM:
    GA, cruce PMX y mutación DM para 15 ciudades .R

  - Algoritmo genético con GA package:
    Paquete GA, 15 ciudades.R

- Problema de las 20 ciudades:

  - Algoritmo vecino más cercano:
    Vecino más cercano para 20 ciudades .R

  - Algoritmo genético con operadores PMX y DM:
    GA, cruce PMX y mutación DM para 20 ciudades.R

  - Algoritmo genético con GA package:
    Paquete GA, 20 ciudades.R

VNiVERSiDAD
Đ SALAMANCA