Práctica 5: Resolución del problema del viajante mediante algoritmos genéticos

Autor: Lucas Fortun Iñurrieta

NIA: 140006

Asignatura: Computación

Índice

Introducción y objetivos

En esta práctica se aborda el Problema del Viajante. El problema consiste en determinar la ruta más eficiente que permita visitar todas las ciudades de un conjunto dado exactamente una vez y regresar al punto de partida, minimizando el costo total del recorrido. En este caso, tenemos un conjunto de ciudades, el objetivo es que el viajante, saliendo de pamplona, vuelva a pamplona recorriendo todas las ciudades 1 vez. La motivación de esto es encontrar la ruta en la que se recorren menos km (coste) y que cumpla los requisitos anteriores. Para ello utilizaremos los algoritmos genéticos. Los Algoritmos Genéticos son técnicas de optimización que simulan la evolución natural en el mundo, pero aplicado a la informática.

Uno de los objetivos principales de esta práctica es entender el funcionamiento de los algoritmos genéticos, y como beneficiarnos de ellos para encontrar soluciones optimas a problemas, en este caso, el Problema del Viajante.

A lo largo de la práctica, llevare a cabo experimentos variando diferentes aspectos de los algoritmos genéticos, como los métodos de selección y cruzamiento de los progenitores, así como la mutación de la descendencia. También se modificará la configuración de los parámetros específicos. La idea es analizar las distintas combinaciones de estos métodos para para extraer conclusiones sobre la eficacia y el rendimiento de los algoritmos genéticos en este problema.

Algoritmos genéticos

Como he comentado antes los Algoritmos Genéticos son técnicas de optimización y búsqueda inspiradas en los principios de la evolución natural y la genética. Estos algoritmos comienzan con una población aleatoria de soluciones para un problema determinado. Estas soluciones compiten y se reproducen, transmitiendo sus características útiles a las nuevas generaciones. Con el tiempo, las soluciones mejores sobreviven y se mejoran. Este proceso se repite hasta encontrar una solución óptima.

El funcionamiento básico de un algoritmo genético sigue una serie de pasos:

1. **Inicialización**: Se crea una población inicial de posibles soluciones al azar o mediante un método heurístico.
2. **Evaluación de la aptitud (fitness):** Cada individuo de la población se evalúa, mediante una función de coste, en función de su adecuación a resolver el problema. Se asigna un valor de aptitud que representa qué tan buena es cada solución.
3. **Selección**: Se seleccionan los individuos más aptos (mejores soluciones) para reproducirse y formar la siguiente generación. Esto se realiza mediante diferentes técnicas de selección, como la ruleta, el torneo o la selección por ranking, donde se priorizan los individuos con mejor aptitud.
4. **Reproducción**: Los individuos seleccionados se cruzan (recombinan) para generar descendencia. Esto se realiza mediante operadores de cruzamiento que combinan las características de dos soluciones parentales para producir nuevos individuos.
5. **Mutación**: Ocasionalmente, se introduce una pequeña variación aleatoria en los individuos de la población para aumentar la diversidad genética. Esto ayuda la variabilidad genética de las soluciones.
6. **Reemplazo**: La nueva población (generación) reemplaza a la anterior, y el proceso de evaluación, selección, reproducción y mutación se repite durante un número fijo de iteraciones o hasta que se cumpla un criterio de terminación. No toda la nueva generación reemplazara a la anterior, solo aquellos con mejor fitness.

Como he comentado, el ciclo de selección, reproducción y reemplazo se repite iterativamente hasta que se alcanza una solución satisfactoria o se alcanza un límite de iteraciones predefinido.

Planteamiento para la resolución del problema

Codificación de los cromosomas

En el contexto del problema del viajante los cromosomas representan las posibles rutas que el viajante puede seguir para visitar todas las ciudades exactamente una vez y regresar al punto de partida. En este caso, cada cromosoma es una lista de índices que representan el orden en el que se visitan las ciudades.

Por ejemplo, si tenemos el cromosoma [0, 3, 5, 2, ..., 0] indicaría que la ruta comienza en la ciudad con índice 0 en la matriz de ciudades, luego visita la ciudad 3, luego la ciudad 5, la ciudad 2 y así sucesivamente, hasta regresar nuevamente a la ciudad 0 para completar el ciclo cerrado.

Por tanto, debido a la naturaleza del problema, la codificación de los cromosomas es por permutación, por ello usare métodos de cruzamiento y mutación para las codificaciones por permutaciones.

Es importante tener en cuenta que la primera y última ciudad de cada cromosoma siempre es la ciudad inicial (Pamplona para este caso) para garantizar que la ruta forme un ciclo cerrado. En el código, he añadido que en cualquier momento se pueda cambiar la ciudad inicial por otra, solo habría que cambiar el parámetro ciudad inicial que recibe el algoritmo genético.

Función de coste

La función de coste es fundamental en los algoritmos genéticos ya que da un valor a cada individuo según su acercamiento a la solución óptima. En el caso del problema del viajante, evalúa una ruta calculando la suma de las distancias entre cada par de ciudades en la ruta, más la distancia de regreso a la ciudad inicial.

En este caso, utilizo la función def calcular_costo que suma las distancias utilizando la matriz de distancias proporcionada M. Esta matriz almacena las distancias entre todas las ciudades del problema. Por ejemplo, M[i][j] representa la distancia entre la ciudad i y la ciudad j.

La función de coste se utiliza para evaluar la calidad de una solución, es decir, cuanto menor sea el costo de una ruta, mejor será la solución. Como he comentado en la introducción, durante el proceso de optimización mediante el algoritmo genético, el objetivo es encontrar la ruta con el menor costo posible.

La función de costes elegida ha sido esta, ya que como queremos minimizar el número de km recorrido para salir de una ciudad y volver a ella pasando 1 vez por todas ciudades, nos permite saber los km recorridos de cada posible solución, y así poder elegir la más optima.

Experimentación

Funciones y sus variantes

Para llevar a cabo la resolución del problema de la manera más optima he implementado diversas funciones para el algoritmo genético, con el objetivo de combinarlas entre ellas y encontrar aquella configuración que resuelva el problema de la manera más optima (menor costo). Como he comentado antes, los métodos de cruzamiento y mutación se basan en la codificación por permutacion.

- **Método de selección de progenitores:** Para la selección de progenitores he implementado 2 variaciones de la función:

    - **Método de la ruleta:** Este método de selección de progenitores asigna probabilidades de selección a cada individuo de la población en función de su aptitud relativa. Cuanto mayor sea la aptitud de un individuo, mayor será la probabilidad de que sea seleccionado como progenitor. La probabilidad de escoger el cromosoma i-ésimo es igual al valor de la función de ajuste en el cromosoma i-ésimo dividido por la suma de los valores de ajuste de todos los cromosomas de la población. Un progenitor puede salir más de una vez.

    - **Método del torneo:** En este método, se eligen aleatoriamente k individuos de la población (llamados participantes, todos con la misma probabilidad de ser escogidos) y se selecciona el mejor de entre ellos como progenitor. Este proceso se repite varias veces para seleccionar múltiples progenitores. Si el método es con remplazo, el individuo escogido para un torneo puede volver a ser escogido para dicho torneo. En caso de que sea sin remplazo, el individuo escogido para un torneo no puede volver a ser escogido para dicho torneo. Los experimentos los hago con remplazo, sin embargo, esto se puede cambiar en cualquier momento, ya que es un parámetro de entrada de la función.

- **Método de cruzamiento:** Para la selección el cruzamiento de individuos he implementado 2 variaciones de la función. En todos ellos se generan dos descendientes a partir de dos padres:

    - **Cruzamiento parcialmente mapeado (PMX):** En este método, se seleccionan aleatoriamente dos puntos de cruzamiento en los cromosomas de los padres. Luego, se copia el segmento entre estos puntos de un padre al primer descendiente. Los elementos restantes del segundo padre se asignan al primer descendiente siguiendo un proceso de mapeo. Aunque no preserva toda la información de los progenitores, conserva ciertas adyacencias importantes. Sin embargo, pueden perderse algunas relaciones entre elementos durante el proceso.

    - **Cruzamiento basado en orden:** Aunque su inicio es similar al del PMX, sigue un camino diferente para transmitir información sobre el orden relativo de los elementos del segundo progenitor. El proceso consiste en seleccionar dos puntos de cruzamiento aleatorios y copiar el segmento entre ellos del primer progenitor al primer descendiente. Luego, a partir del segundo punto de cruzamiento, se copian en orden los valores del segundo progenitor que aún no han aparecido. Este proceso se repite con los papeles de los progenitores invertidos para generar el segundo descendiente.

- **Mutaciones:** Para la generación de mutaciones en los descendientes he implementado los siguientes operadores:

    - **Mutación por intercambio:** Este operador implica intercambiar dos posiciones en el cromosoma de forma aleatoria.

    - **Mutación por inserción:** Este operador, seleccione dos genes al azar y uno de ellos se mueve para insertarlo junto al otro, desplazando los genes resultantes para permitir la inserción.

Descripción experimentos

A grandes rasgos, los experimentos que he implementado han sido 16. Estos 16 experimentos podrían combinarse en 2 macro experimentos.

Estos dos experimentos consisten en probar todas las posibles combinaciones de las funciones de cruzamiento, selección de progenitores y variaciones de k en el método de selección de progenitores por torneo que he implementado para el algoritmo genético. Cuando digo que estos 16 experimentos podrían combinarse en 2 macro experimentos, me refiero a que he realizado los mismo 8 experimentos uno para el algoritmo con mutaciones y otro sin mutaciones, de esta manera podremos observar los resultados de todas las posibles combinaciones de las funciones que ya he nombrado, tanto para el algoritmo con mutaciones y sin ellas. Esto nos permitirá encontrar la mejor configuración para el algoritmo y poder así encontrar la solución más eficiente para el problema del viajante. Quiero destacar que todas las pruebas se han realizado para un tamaño de población de 100 individuos y durante 1000 generaciones.

Siendo más concretos, las pruebas realizadas han sido las siguientes:

Algoritmo genético sin mutaciones:

- **Prueba 1**
  - Función de cruce: Cruzamiento parcialmente mapeado (PMX)
  - Función de selección de progenitores: Método de la ruleta

- **Prueba 2**
  - Función de cruce: Cruzamiento parcialmente mapeado (PMX)
  - Función de selección de progenitores: Método del torneo
  - Numero de contrincantes(k): 2

- **Prueba 3**
  - Función de cruce: Cruzamiento parcialmente mapeado (PMX)
  - Función de selección de progenitores: Método del torneo
  - Numero de contrincantes(k): 5

- **Prueba 4**
  - Función de cruce: Cruzamiento parcialmente mapeado (PMX)
  - Función de selección de progenitores: Método del torneo
  - Numero de contrincantes(k): 10

- **Prueba 5**
  - Función de cruce: Cruzamiento basado en orden
  - Función de selección de progenitores: Método de la ruleta

- **Prueba 6**
  - Función de cruce: Cruzamiento basado en orden
  - Función de selección de progenitores: Método del torneo
  - Numero de contrincantes(k): 2

- **Prueba 7**
  - Función de cruce: Cruzamiento basado en orden
  - Función de selección de progenitores: Método del torneo
  - Numero de contrincantes(k): 5

- **Prueba 8**
  - Función de cruce: Cruzamiento basado en orden
  - Función de selección de progenitores: Método del torneo
  - Numero de contrincantes(k): 10

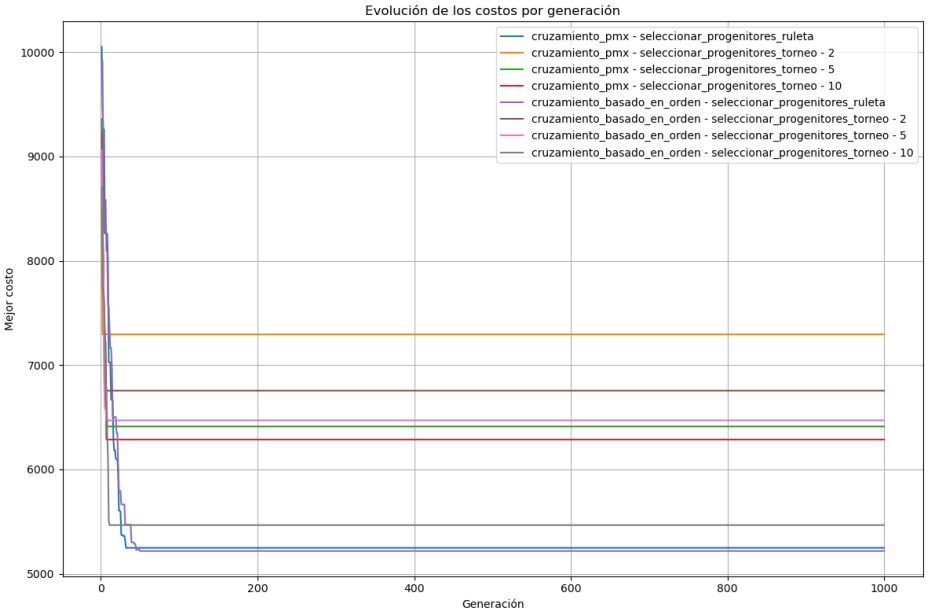
Algoritmo genético con mutaciones:

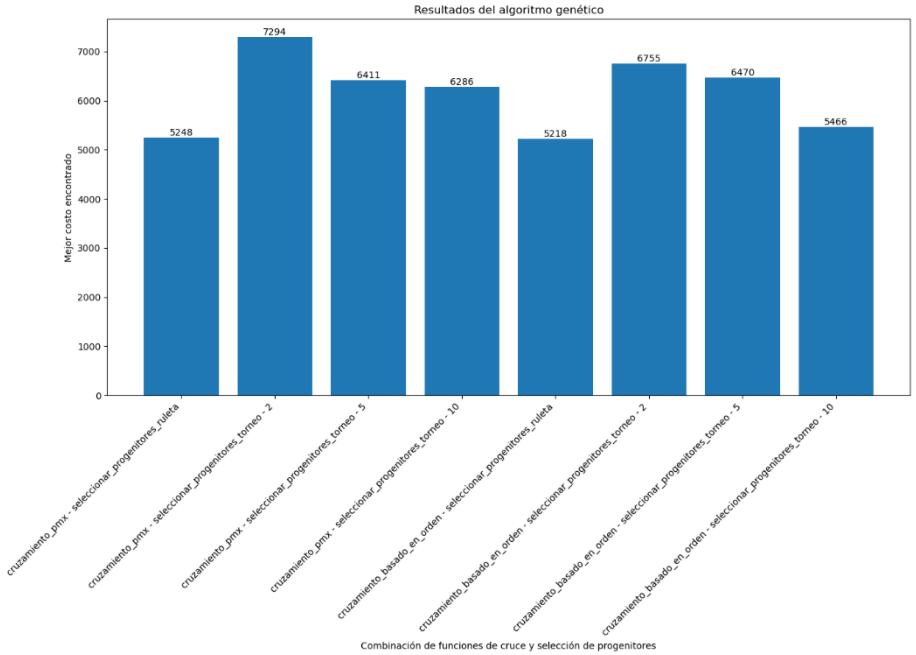
Para el algoritmo con mutaciones, las pruebas se mantienen iguales a las anteriores, pero ahora se incorporan mutaciones a la descendencia. Como los métodos de cruzamiento que he implementado generan dos descendientes, he codificado dos operadores de mutación, uno para cada uno. En esta etapa, obtendremos otras 8 pruebas, aplicando mutaciones a la descendencia. Un hijo experimentará una mutación por inserción, y el otro por intercambio. Esto permite aumentar la variabilidad genética y evitar estancarse en una descendencia que no varía y sigue reproduciéndose. Estas mutaciones proporcionan una mayor exploración del espacio de búsqueda, lo que puede ayudar al algoritmo genético a encontrar soluciones óptimas y evitar quedar atrapado en óptimos locales.

Resultados obtenidos

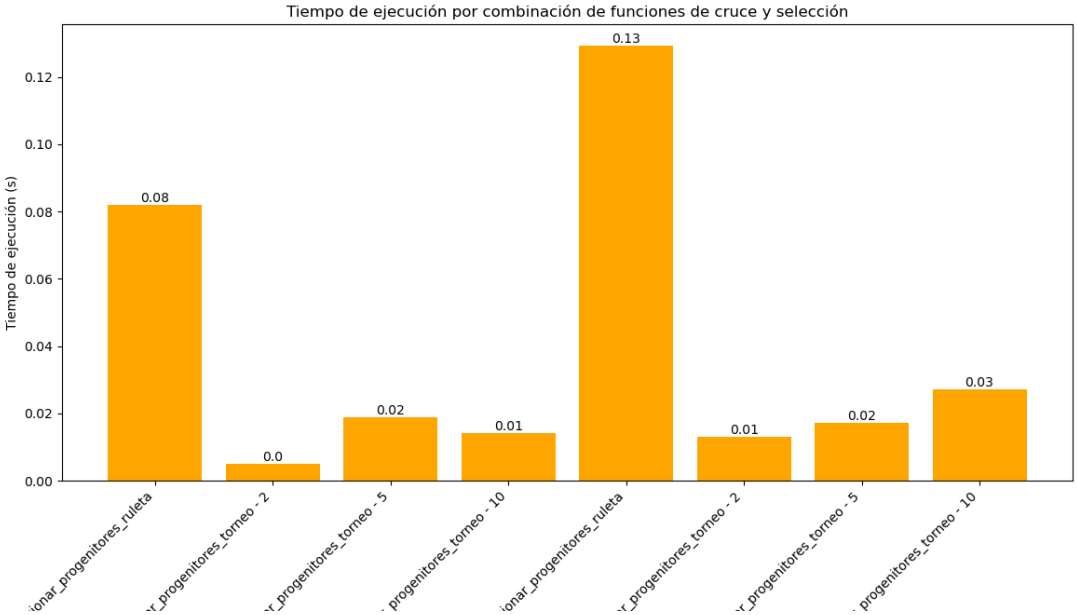
Los resultados obtenidos han sido:

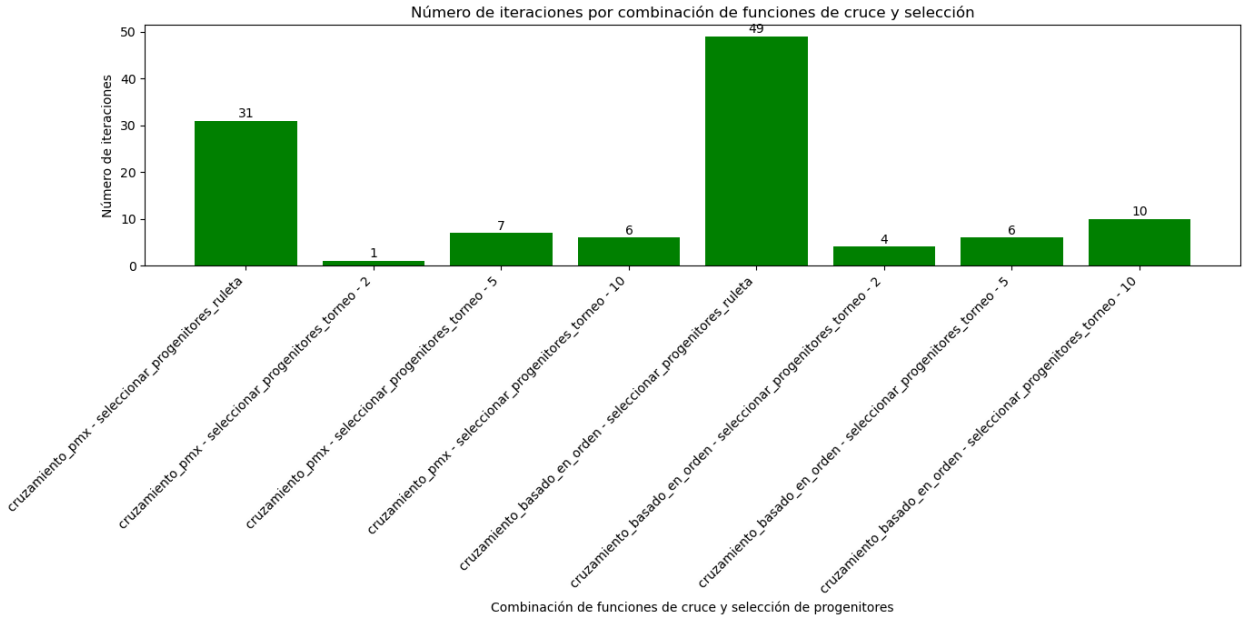
- **Algoritmo sin mutaciones**
  - **Evolución del coste durante las generaciones y coste final de cada experimento**



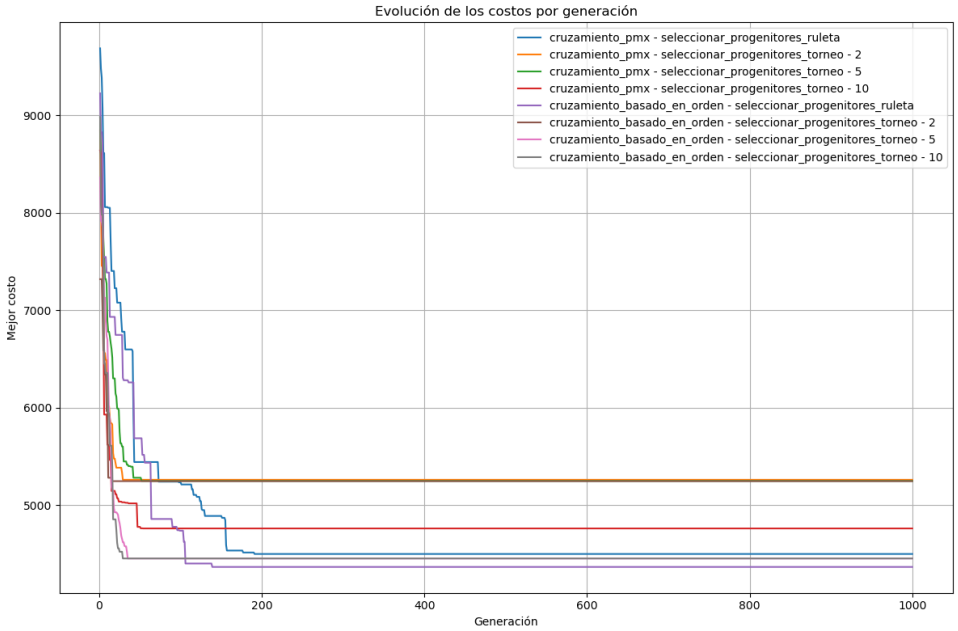


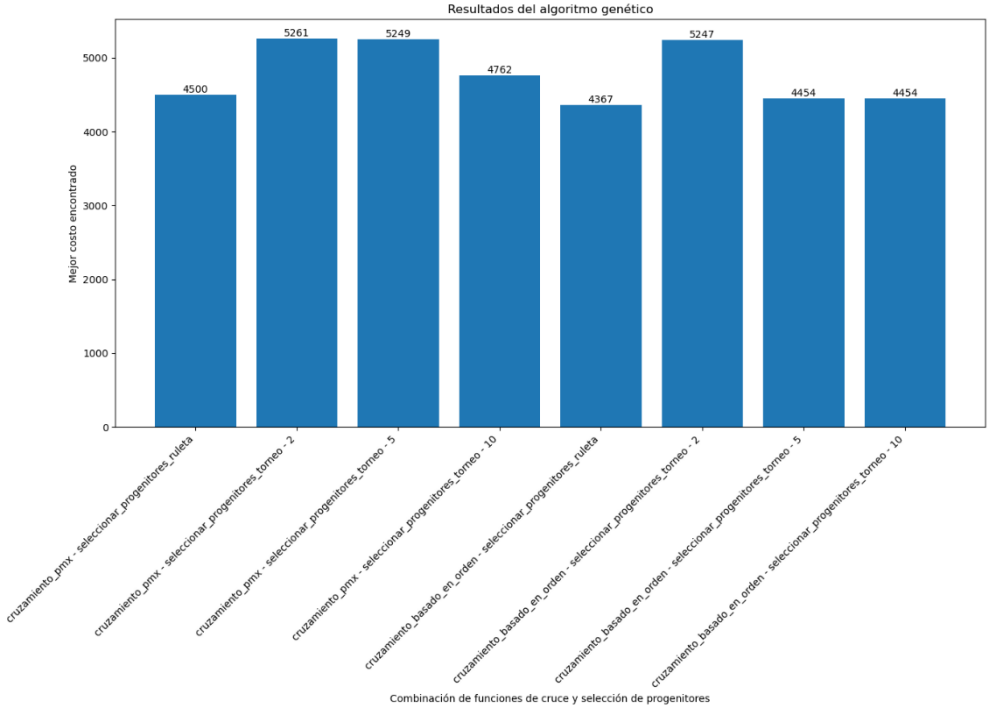
- **Coste temporal y de iteraciones**



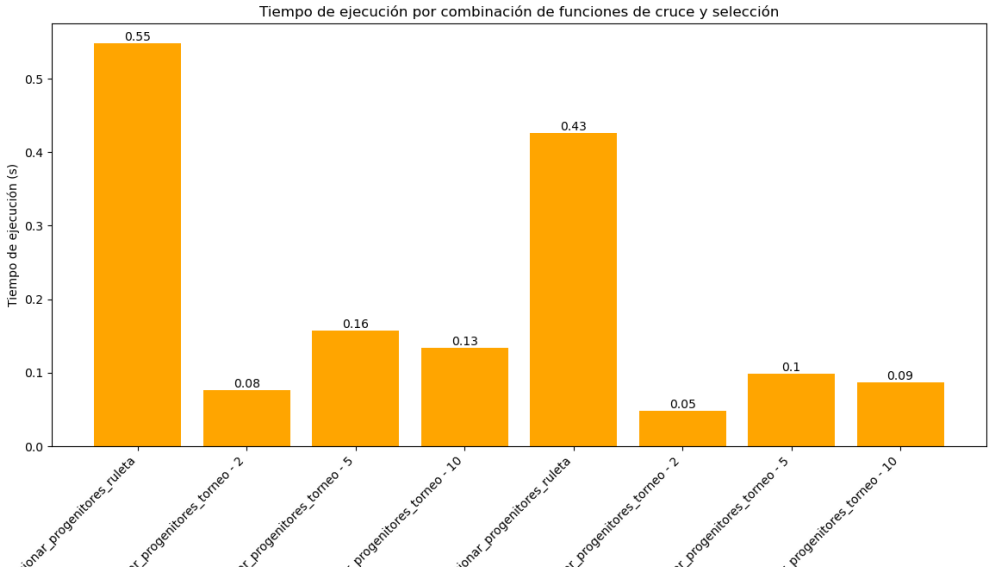


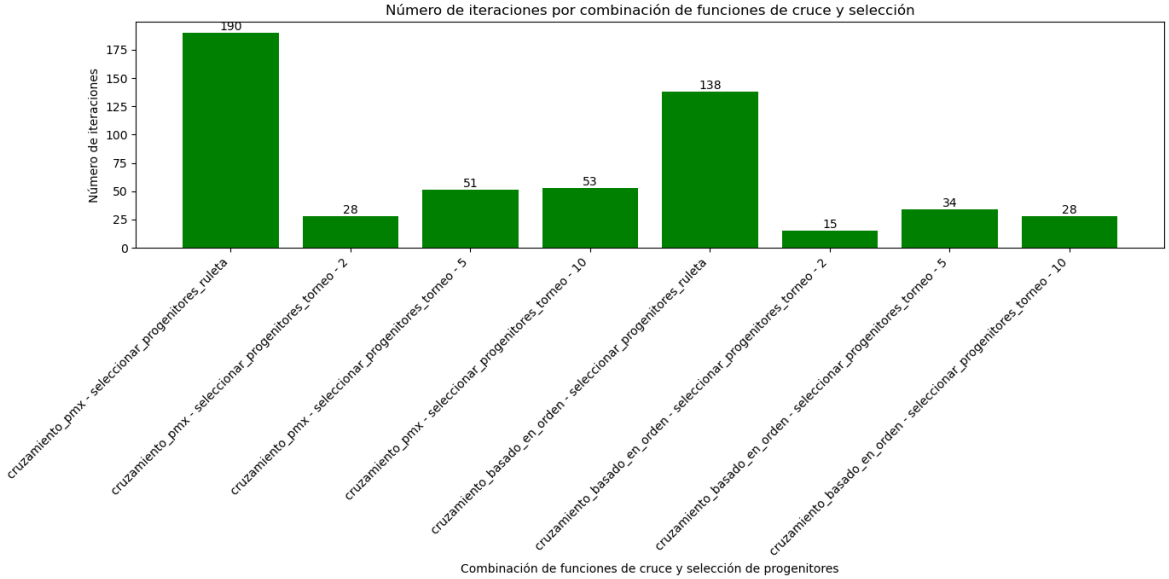
- **Algoritmo con mutaciones**
    - **Evolución del coste durante las generaciones y coste final de cada experimento**





- **Coste temporal y de iteraciones de cada experimento**





Interpretación y análisis detallado de los resultados

- **Análisis Detallado de los Resultados de Cada Experimento para el algoritmo sin mutaciones**

1. **Cruzamiento PMX y Selección por Ruleta**

    - **Mejor ruta:** ['Pamplona', 'Donostia', 'Madrid', 'Toledo', 'Valencia', 'Girona', 'Barcelona', 'Tarragona', 'Zaragoza', 'Alicante', 'Murcia', 'Cordoba', 'Cadiz', 'Sevilla', 'Huelva', 'Caceres', 'Leon', 'Coruna', 'Oviedo', 'Bilbao', 'Pamplona']
    - **Costo mejor resultado:** 5248
    - **Iteración de mejor resultado:** 31
    - **Tiempo en que se encontró el mejor resultado:** 0.082 segundos
    - **Análisis:** La selección por ruleta favorece la diversidad genética al permitir que incluso los individuos con menor aptitud tengan una probabilidad de ser seleccionados, evitando la convergencia prematura y permitiendo una exploración más amplia del espacio de soluciones. El cruzamiento PMX es efectivo para mantener partes significativas de los padres en los hijos, lo que ayuda a preservar buenas subsecuencias de ciudades.

2. **Cruzamiento PMX y Selección por torneo (k = 2)**

    - **Mejor ruta:** ['Pamplona', 'Oviedo', 'Coruna', 'Sevilla', 'Cadiz', 'Huelva', 'Toledo', 'Leon', 'Valencia', 'Zaragoza', 'Barcelona', 'Tarragona', 'Girona', 'Murcia', 'Alicante', 'Cordoba', 'Caceres', 'Madrid', 'Donostia', 'Bilbao', 'Pamplona']
    - **Costo mejor resultado:** 7294
    - **Iteración de mejor resultado:** 1
    - **Tiempo en que se encontró el mejor resultado:** 0.004 segundos
    - **Análisis:** La selección por torneo con k=2 y cruzamiento PMX favorece una mayor variabilidad genética. Al usar un tamaño de torneo pequeño, se reduce la probabilidad de que el cromosoma con mejor costo gane repetidamente. Esto permite que otros cromosomas también tengan oportunidades, incrementando la diversidad genética y evitando el estancamiento en mínimos locales. Aunque puede llevar a soluciones de mayor costo inicial y requiere menos tiempo de ejecución, la variabilidad genética promovida es crucial para explorar un mayor espacio de soluciones.

3. **Cruzamiento PMX y Selección por torneo (k = 5)**

    - **Mejor ruta:** ['Pamplona', 'Donostia', 'Toledo', 'Zaragoza', 'Bilbao', 'Caceres', 'Huelva', 'Cadiz', 'Cordoba', 'Sevilla', 'Oviedo', 'Leon', 'Coruna', 'Madrid', 'Murcia', 'Alicante', 'Valencia', 'Girona', 'Barcelona', 'Tarragona', 'Pamplona']
    - **Costo mejor resultado:** 6411
    - **Iteración de mejor resultado:** 7
    - **Tiempo en que se encontró el mejor resultado:** 0.018 segundos
    - **Análisis:** A medida que se aumenta k, se disminuye la variabilidad genética, pero el resultado más óptimo tiene una mayor probabilidad de ser seleccionado. Esto conduce a una población con menor costo en general y una mayor probabilidad de alcanzar un resultado óptimo.

4. **Cruzamiento PMX y Selección por torneo (k = 10)**

    - **Mejor ruta:** ['Pamplona', 'Donostia', 'Bilbao', 'Coruna', 'Leon', 'Toledo', 'Caceres', 'Cordoba', 'Alicante', 'Murcia', 'Sevilla', 'Cadiz', 'Huelva', 'Oviedo', 'Madrid', 'Valencia', 'Tarragona', 'Girona', 'Barcelona', 'Zaragoza', 'Pamplona']
    - **Costo mejor resultado:** 6286
    - **Iteración de mejor resultado:** 6
    - **Tiempo en que se encontró el mejor resultado:** 0.014 segundos
    - **Análisis:** En este caso, la ruta óptima obtenida presenta similitudes con el caso anterior, pero con una menor variabilidad genética. De esto podemos destacar que el resultado alcanzado con k=10 es superior al obtenido con k=2 y k=5. Esto sugiere que la variabilidad genética tiene un menor impacto en el resultado que la frecuencia con la que el mejor cromosoma se presenta en la nueva generación (porque participa en más torneos y los gana). Esto puede parecer contradictorio ya que una mayor variabilidad genética permite explorar más en profundidad las posibles soluciones.

5. **Cruzamiento Basado en Orden y Selección por Ruleta**

    - **Mejor ruta:** ['Pamplona', 'Zaragoza', 'Girona', 'Barcelona', 'Tarragona', 'Valencia', 'Caceres', 'Cordoba', 'Sevilla', 'Cadiz', 'Huelva', 'Alicante', 'Murcia', 'Toledo', 'Madrid', 'Leon', 'Coruna', 'Oviedo', 'Bilbao', 'Donostia', 'Pamplona']
    - **Costo mejor resultado:** 5218
    - **Iteración de mejor resultado:** 49
    - **Tiempo en que se encontró el mejor resultado:** 0.129 segundos
    - **Análisis:** El cruzamiento basado en orden (OX) es eficaz en preservar el orden relativo de las ciudades, lo que es crucial en el TSP. Combinado con la selección por ruleta, que mantiene la diversidad genética, se logra encontrar una solución con buen costo. Sin embargo, este método requiere muchas iteraciones y más tiempo debido a su naturaleza exploratoria, similar al primer experimento, pero con una efectividad ligeramente mejor en términos de costo.

6. **Cruzamiento Basado en Orden y Selección por Torneo (k=2)**

    - **Mejor ruta:** ['Pamplona', 'Valencia', 'Murcia', 'Alicante', 'Tarragona', 'Cordoba', 'Huelva', 'Sevilla', 'Cadiz', 'Madrid', 'Bilbao', 'Oviedo', 'Leon', 'Coruna', 'Caceres', 'Toledo', 'Zaragoza', 'Barcelona', 'Girona', 'Donostia', 'Pamplona']
    - **Costo mejor resultado:** 6755
    - **Iteración de mejor resultado:** 4
    - **Tiempo en que se encontró el mejor resultado:** 0.012 segundos
    - **Análisis:** Este método muestra una mejora significativa en el costo frente al cruzamiento PMX, indicando que la preservación del orden tiene un impacto considerable en la calidad de las soluciones.

7. **Cruzamiento Basado en Orden y Selección por Torneo (k=5)**

    - **Mejor ruta:** ['Pamplona', 'Oviedo', 'Coruna', 'Caceres', 'Huelva', 'Cadiz', 'Cordoba', 'Sevilla', 'Leon', 'Zaragoza', 'Tarragona', 'Murcia', 'Alicante', 'Valencia', 'Girona', 'Barcelona', 'Toledo', 'Madrid', 'Bilbao', 'Donostia', 'Pamplona']
    - **Costo mejor resultado:** 6470
    - **Iteración de mejor resultado:** 6
    - **Tiempo en que se encontró el mejor resultado:** 0.017 segundos
    - **Análisis:** Al aumentar k y reducir la variabilidad genética, se observa una mejora en los resultados. Esto destaca la importancia de priorizar los cromosomas con mejores costos en la selección.

8. **Cruzamiento Basado en Orden y Selección por Torneo (k=10)**

    - **Mejor ruta:** ['Pamplona', 'Toledo', 'Caceres', 'Cordoba', 'Huelva', 'Sevilla', 'Cadiz', 'Alicante', 'Murcia', 'Valencia', 'Tarragona', 'Barcelona', 'Girona', 'Zaragoza', 'Donostia', 'Bilbao', 'Oviedo', 'Coruna', 'Leon', 'Madrid', 'Pamplona']
    - **Costo mejor resultado:** 5466
    - **Iteración de mejor resultado:** 10
    - **Tiempo en que se encontró el mejor resultado:** 0.027 segundos
    - **Análisis:** Como en los casos anteriores, un mayor k mejora los resultados. Esto muestra que priorizar los mejores cromosomas y reducir la variabilidad genética puede llevar a mejores soluciones, aunque con un menor grado de exploración.

- **Análisis Detallado de los Resultados de Cada Experimento para el algoritmo con mutaciones**

  9. **Cruzamiento PMX y Selección por Ruleta**

     - **Mejor ruta:** ['Pamplona', 'Donostia', 'Zaragoza', 'Barcelona', 'Girona', 'Tarragona', 'Valencia', 'Alicante', 'Murcia', 'Cordoba', 'Sevilla', 'Cadiz', 'Huelva', 'Caceres', 'Toledo', 'Madrid', 'Leon', 'Coruna', 'Oviedo', 'Bilbao', 'Pamplona']
     - **Costo mejor resultado:** 4500
     - **Iteración de mejor resultado:** 190
     - **Tiempo en que se encontró el mejor resultado:** 0.547 segundos
     - **Análisis:** La inclusión de mutaciones con la selección por ruleta y cruzamiento PMX permite una exploración exhaustiva del espacio de soluciones, lo que lleva a una solución de bajo costo. La selección por ruleta mantiene una alta diversidad genética, y las mutaciones introducen variabilidad adicional, evitando el estancamiento en óptimos locales. Sin embargo, esto requiere más iteraciones y tiempo debido a la mayor cantidad de operaciones genéticas realizadas.

  10. **Cruzamiento PMX y Selección por torneo (k = 2)**

     - **Mejor ruta:** ['Pamplona', 'Bilbao', 'Oviedo', 'Coruna', 'Leon', 'Toledo', 'Cordoba', 'Sevilla', 'Cadiz', 'Huelva', 'Caceres', 'Madrid', 'Zaragoza', 'Barcelona', 'Girona', 'Tarragona', 'Valencia', 'Alicante', 'Murcia', 'Donostia', 'Pamplona']
     - **Costo mejor resultado:** 5261
     - **Iteración de mejor resultado:** 28
     - **Tiempo en que se encontró el mejor resultado:** 0.075 segundos
     - **Análisis:** Observamos que esta configuración experimenta una gran mejora en el resultado al añadir la variabilidad genética de las mutaciones.

  11. **Cruzamiento PMX y Selección por torneo (k = 5)**

     - **Mejor ruta:** ['Pamplona', 'Donostia', 'Bilbao', 'Oviedo', 'Coruna', 'Leon', 'Madrid', 'Toledo', 'Murcia', 'Alicante', 'Valencia', 'Tarragona', 'Girona', 'Barcelona', 'Zaragoza', 'Cordoba', 'Cadiz', 'Sevilla', 'Huelva', 'Caceres', 'Pamplona']
     - **Costo mejor resultado:** 5249
     - **Iteración de mejor resultado:** 51
     - **Tiempo en que se encontró el mejor resultado:** 0.157 segundos
     - **Análisis:** Al igual que el algoritmo sin mutaciones, al aumentar la k y priorizar el mejor cromosoma frente a la mayor variabilidad genética a la hora de seleccionar progenitores mejora el costo.

12. **Cruzamiento PMX y Selección por torneo (k = 10)**

- **Mejor ruta:** ['Pamplona', 'Donostia', 'Bilbao', 'Zaragoza', 'Girona', 'Barcelona', 'Tarragona', 'Valencia', 'Alicante', 'Murcia', 'Madrid', 'Toledo', 'Cordoba', 'Cadiz', 'Sevilla', 'Huelva', 'Caceres', 'Leon', 'Coruna', 'Oviedo', 'Pamplona']
- **Costo mejor resultado:** 4762
- **Iteración de mejor resultado:** 53
- **Tiempo en que se encontró el mejor resultado:** 0.133 segundos
- **Análisis:** Con este resultado podemos concluir que al igual que el algoritmo sin mutaciones, el aumento de k a la hora de seleccionar progenitores mediante el método del torneo es directamente proporcional a un mejor resultado. Por tanto, si priorizamos los mejores cromosomas aumentando la k y añadimos la mejora es muy considerable.

13. **Cruzamiento Basado en Orden y Selección por Ruleta**

- **Mejor ruta:** ['Pamplona', 'Zaragoza', 'Girona', 'Barcelona', 'Tarragona', 'Valencia', 'Alicante', 'Murcia', 'Cordoba', 'Sevilla', 'Cadiz', 'Huelva', 'Caceres', 'Toledo', 'Madrid', 'Leon', 'Coruna', 'Oviedo', 'Bilbao', 'Donostia', 'Pamplona']
- **Costo mejor resultado:** 4367
- **Iteración de mejor resultado:** 138
- **Tiempo en que se encontró el mejor resultado:** 0.426 segundos
- **Análisis:** El cruzamiento basado en orden (OX) combinado con la selección por ruleta y mutaciones resulta en la mejor solución de costo más bajo. La alta diversidad genética y la preservación del orden relativo de las ciudades son beneficiosas para el TSP. Sin embargo, esto requiere un mayor número de iteraciones y tiempo debido a la cantidad significativa de exploración y operaciones de mutación.

14. **Cruzamiento Basado en Orden y Selección por Torneo (k=2)**

- **Mejor ruta:** ['Pamplona', 'Donostia', 'Bilbao', 'Oviedo', 'Coruna', 'Leon', 'Madrid', 'Murcia', 'Alicante', 'Valencia', 'Girona', 'Barcelona', 'Tarragona', 'Cordoba', 'Sevilla', 'Cadiz', 'Huelva', 'Caceres', 'Toledo', 'Zaragoza', 'Pamplona']
- **Costo mejor resultado:** 5247
- **Iteración de mejor resultado:** 15
- **Tiempo en que se encontró el mejor resultado:** 0.048 segundos
- **Análisis:** Aquí podemos observar, al igual que para el algoritmo sin mutaciones, que priorizar el orden permite alcanzar un mejor resultado, además variabilidad genética de las mutaciones mejora aún más este.

15. **Cruzamiento Basado en Orden y Selección por Torneo (k=5)**

- **Mejor ruta:** ['Pamplona', 'Zaragoza', 'Girona', 'Barcelona', 'Tarragona', 'Valencia', 'Alicante', 'Murcia', 'Madrid', 'Toledo', 'Cordoba', 'Cadiz', 'Sevilla', 'Huelva', 'Caceres', 'Leon', 'Coruna', 'Oviedo', 'Bilbao', 'Donostia', 'Pamplona']
- **Costo mejor resultado:** 4454
- **Iteración de mejor resultado:** 34
- **Tiempo en que se encontró el mejor resultado:** 0.098 segundos
- **Análisis:** Como hasta ahora, el aumento de k y la aplicación de mutaciones sigue mejorando el resultado.

16. **Cruzamiento Basado en Orden y Selección por Torneo (k=10)**

- **Mejor ruta:** ['Pamplona', 'Donostia', 'Bilbao', 'Oviedo', 'Coruna', 'Leon', 'Caceres', 'Huelva', 'Sevilla', 'Cadiz', 'Cordoba', 'Toledo', 'Madrid', 'Murcia', 'Alicante', 'Valencia', 'Tarragona', 'Barcelona', 'Girona', 'Zaragoza', 'Pamplona']
- **Costo mejor resultado:** 4454
- **Iteración de mejor resultado:** 28
- **Tiempo en que se encontró el mejor resultado:** 0.086 segundos
- **Análisis:** Finalmente con esta última prueba podemos observar el gran impacto de priorizar el orden, los cromosomas con mejor coste en la selección (k grande) así como la variabilidad genética de las mutaciones. Sin embargo, aunque este resultado es muy prometedor, el mejor obtenido ha sido para el algoritmo con mutaciones con configuración cruzamiento basado en orden y selección por ruleta, priorizando, igual que este, el orden y la variabilidad genética de las mutaciones, sin embargo al utilizar el método de la ruleta, en vez del torneo, está dando mayor posibilidad a los cromosomas no prometedores de poder crear descendencia y así aumentando aún más la variabilidad y con ello la exploración del espacio de resultados, a diferencia de la selección por torneo, ya que según aumenta k, aumenta la probabilidad de que cromosomas no prometedores nunca creen descendencia reduciendo así la variabilidad genética y pudiendo estancar el resultado en mínimos locales.

Conclusiones

- **Conclusiones respecto a los experimentos**

  Hemos obtenido diversos resultados para cada experimento. En el caso del algoritmo sin mutaciones, la selección por ruleta combinada con el cruzamiento PMX favorece la diversidad genética, permitiendo evitar la convergencia prematura y explorar una mayor variedad de soluciones. Esto resulta en soluciones de bajo costo, aunque a resulta en un mayor número de iteraciones y tiempo de ejecución. En contraste, la selección por torneo, especialmente con un mayor tamaño de torneo (k), incrementa la presión selectiva, acelerando la convergencia, pero reduciendo la diversidad genética y, en consecuencia, aumentando el riesgo de estancamiento en óptimos locales.

  Por otro lado, el algoritmo con mutaciones demuestra ser más robusto y eficiente en la búsqueda de soluciones óptimas. La inclusión de mutaciones introduce una variabilidad adicional que es crucial para evitar el estancamiento en óptimos locales. En particular, la combinación de selección por ruleta y cruzamiento PMX con mutaciones permite una exploración más exhaustiva del espacio de soluciones, logrando soluciones de menor costo, aunque requiera más iteraciones y tiempo. La selección por torneo, con un tamaño de torneo mayor (como k=10), junto con mutaciones, mantiene un buen equilibrio entre presión selectiva y diversidad genética, resultando en soluciones de bajo costo con tiempos de ejecución razonables. Sin embargo, una presión selectiva excesivamente más alta puede llevar a una rápida convergencia hacia soluciones subóptimas debido a la reducción de la diversidad genética, aunque las mutaciones ayudan a reducir parcialmente este efecto.

  En general, el algoritmo con mutaciones supera al algoritmo sin mutaciones en términos de la calidad de las soluciones obtenidas, a pesar de requerir más tiempo e iteraciones. La capacidad de las mutaciones para introducir y mantener la diversidad genética es fundamental para evitar el estancamiento y explorar de manera más efectiva el espacio de soluciones. La selección por ruleta y un tamaño de torneo moderado (k=10) han dado buenos resultados, sin embargo, hemos visto que priorizar el orden relativo de las ciudades, así como priorizar la variabilidad genética no solo en la descendencia (mutaciones) sino también a la hora de seleccionar los cromosomas que van a crear descendencia (permitiendo que incluso los candidatos menos óptimos se reproduzcan aumentando, así, la variabilidad genética).

- **Conclusiones respecto a la solución del problema**

  Una vez analizados todos los experimentos, tanto para el algoritmo con y si mutaciones, observamos que la mejor configuración para este problema en concreto es el algoritmo genético con el método Cruzamiento Basado en Orden y el método de Selección de progenitores por Ruleta, ya que con esta configuración hemos podido alcanzar el mejor resultado, y no solo eso, sino que hemos alcanzado el mínimo global del problema.

  Por tanto, partiendo desde Pamplona, la ruta más corta que nos permite visitar cada ciudad una vez y volver a Pamplona es: Pamplona, luego Zaragoza, Girona, Barcelona, Tarragona, Valencia, Alicante, Murcia, Córdoba, Sevilla, Cádiz, Huelva, Cáceres, Toledo, Madrid, León, Coruña, Oviedo, Bilbao, Donostia y finalmente volver a Pamplona con un total de 4367 kilómetros recorridos.