**Trabajo Práctico Nº 3**

**Algoritmos Genéticos**

**2020**

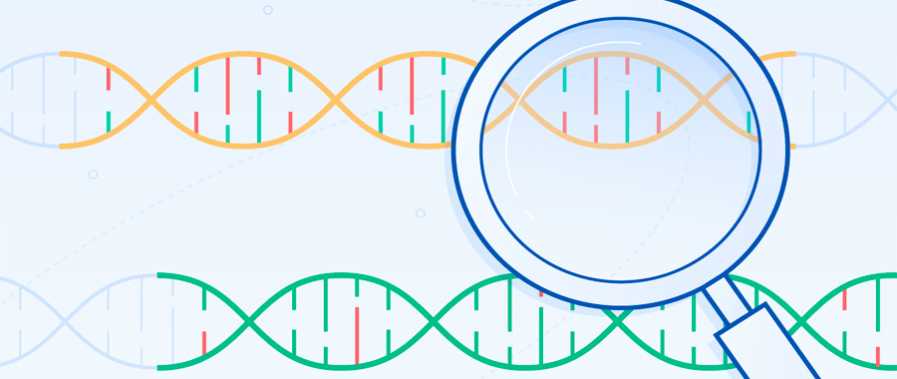
**Integrantes**

**Luciano Vannelli 45744 - lucianovannelli@gmail.com**

**Manuel Bahamonde - 45699 - manuelbahamonde6@gmail.com**

**Franco Ceschan - 42737 - francoceschan@gmail.com**

**Milena Lebedinsky - 43982 - milenalebedin@gmail.com**

****

[Enunciado](#_jt0p0iy940bv)

[1. Resolución exhaustiva](#_slyap09iao8y)

[2. Desarrollo del programa](#_58dagrnlqqwu)

[Herramientas de programación utilizadas](#_vh5ejrke9par)

[Forma de trabajo en equipo](#_2byklmycujk3)

[El programa](#_ljsp436zxgpu)

[Código del programa](#_on1yf9e3tdyl)

[Salida](#_p2sggt9w4g3k)

[Aportes prácticos del TSP](#_li08f8cs2izy)

[Conclusión](#_61xwg6ibto5e)

# Enunciado

El problema del Viajante

El problema del viajante (también conocido como problema del viajante de comercio o por sus siglas en inglés: TSP (Traveling Salesman Problem), es uno de los problemas más famosos (y quizás el mejor estudiado) en el campo de la optimización combinatoria computacional.

A pesar de la aparente sencillez de su planteamiento, el TSP es uno de los más complejos de resolver .

Definición: Sean N ciudades de un territorio. La distancia entre cada ciudad viene dada por la [matriz D:](https://frro.cvg.utn.edu.ar/course/Tabla%20de%20Distancias.docx) NxN, donde d[x,y] representa la distancia que hay entre la ciudad X y la ciudad Y.

El objetivo es encontrar una ruta que, comenzando y terminando en una ciudad concreta, pase una sola vez por cada una de las ciudades y minimice la distancia recorrida por el viajante.

Ejercicios:

1.Hallar la ruta de distancia mínima que logre unir todas las capitales de provincias de la República Argentina, utilizando un método exhaustivo. ¿Puede resolver el problema? Justificar de manera teórica.

2.Realizar un programa que cuente con un menú con las siguientes opciones:

a)Permitir ingresar una provincia y hallar la ruta de distancia mínima que logre unir todas las capitales de provincias de la República Argentina partiendo de dicha capital utilizando la siguiente heurística: “Desde cada ciudad ir a la ciudad más cercana no visitada.” Recordar regresar siempre a la ciudad de partida. Presentar un mapa de la República con el recorrido indicado. Además indicar la ciudad de partida, el recorrido completo y la longitud del trayecto. El programa deberá permitir seleccionar la capital que el usuario desee ingresar como inicio del recorrido.

b)Encontrar el recorrido mínimo para visitar todas las capitales de las provincias de la República Argentina siguiendo la heurística mencionada en el punto a. Deberá mostrar como salida el recorrido y la longitud del trayecto.

c)Hallar la ruta de distancia mínima que logre unir todas las capitales de provincias de la República Argentina, utilizando un algoritmo genético.

Recomendaciones para el algoritmo:

N = 50 Número de cromosomas de las poblaciones.

M = 200 Cantidad de ciclos.

Cromosomas: permutaciones de 23 números naturales del 1 al 23 donde cada gen es una ciudad.

Las frecuencias de crossover y de mutación quedan a criterio del grupo.

Se deberá usar crossover cíclico.

Comparar los resultados obtenidos entre la resolución a través de heurísticas y con algoritmos genéticos a través de una conclusión que deberá anexarse al informe.

Agregar en el informe un apartado final denominado «Aportes Prácticos del TSP» donde se expliquen algunas aplicaciones en las que actualmente se use el problema del viajante. Tomar por lo menos dos y explicarlas.

# 1. Resolución exhaustiva

Para encontrar el recorrido óptimo mediante una búsqueda exhaustiva, se deben primero hallar las combinaciones de todos los recorridos posibles. Dada la ciudad inicial, se tienen 23 posibilidades para la ciudad siguiente. Seleccionada ésta, quedan 22 posibilidades, y así sucesivamente. De esta manera, por combinatoria, la cantidad de recorridos posibles es de: 23\*22\*21\*...\*2\*1 = 23!.

Si bien podría resolverse este problema para una menor cantidad de ciudades, la cantidad de pasos a realizar para 24 ciudades es de 2.585201674 \* 10²², por lo que el tiempo de cómputo requerido no permite encontrar un resultado de forma exhaustiva.

# 2. Desarrollo del programa

**Clase Poblacion:** guarda en sus atributos el cromosoma con menor distancia del ciclo, la suma de las distancias de cada cromosoma y todos los cromosomas de esa población. Además, tiene diversos métodos que hacen funcionar al algoritmo, tales como ***seleccionRuleta,*** ***aplicarCrossoverMutacion***, etc.

**Clase Cromosoma:** tiene los datos de cada recorrido, como sus “bits” (cada una de las ciudades del recorrido), la distancia del recorrido y el valor fitness. Tambien tiene metodos que permiten calcular el fitness y la distancia

## Herramientas de programación utilizadas

**Lenguaje de programación** : Python

**Librerías utilizadas:**

* **random:** utilizado generar numeros random
* **numpy, pandas, xlsxwriter:** utilizadas para trabajar con el archivo de excel que tiene las distintas distancias entre las ciudades
* **attrgetter:** utilizado para obtener los cromosomas con mayor fitness (en elitismo)
* **pygame:** utilizada para mostrar el mapa con el recorrido.
* **tkinter:** utilizada para hacer las ventanas del menú.

## Forma de trabajo en equipo

Nos manejamos con aplicaciones de mensajería y chat de voz para la comunicación, con aplicaciones de almacenamiento en la nube y versionado de código para compartir el código del TP y con una extensión del IDE Visual Studio Code para editar todos al mismo tiempo el archivo

* **Visual Studio Code:** el IDE con el que escribimos el algoritmo
* **Extensión Live Share:** la extensión del IDE que permitió que todos podamos editar y ver el código en tiempo real
* **Repositorios en Google Drive y Github:** lo utilizamos para guardar el código del algoritmo actualizado
* **Grupo de Whatsapp:** utilizado para coordinar reuniones y notificar cambios y modificaciones realizadas en el código.
* **Discord:** lo utilizamos para discutir el código que escribíamos por voz.

# 

# 

# 

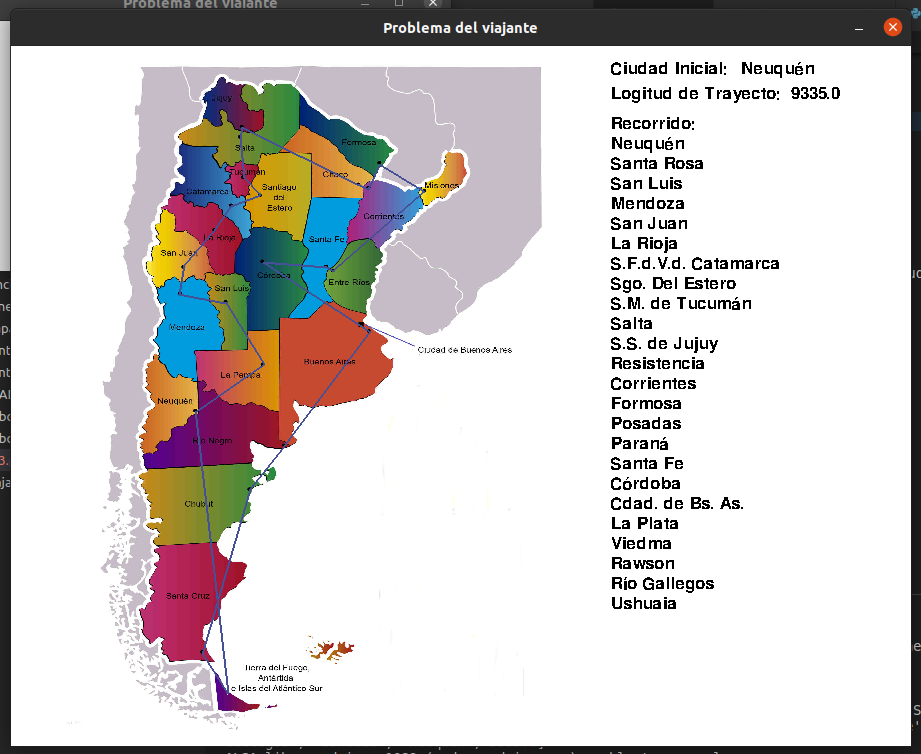
# 

# 

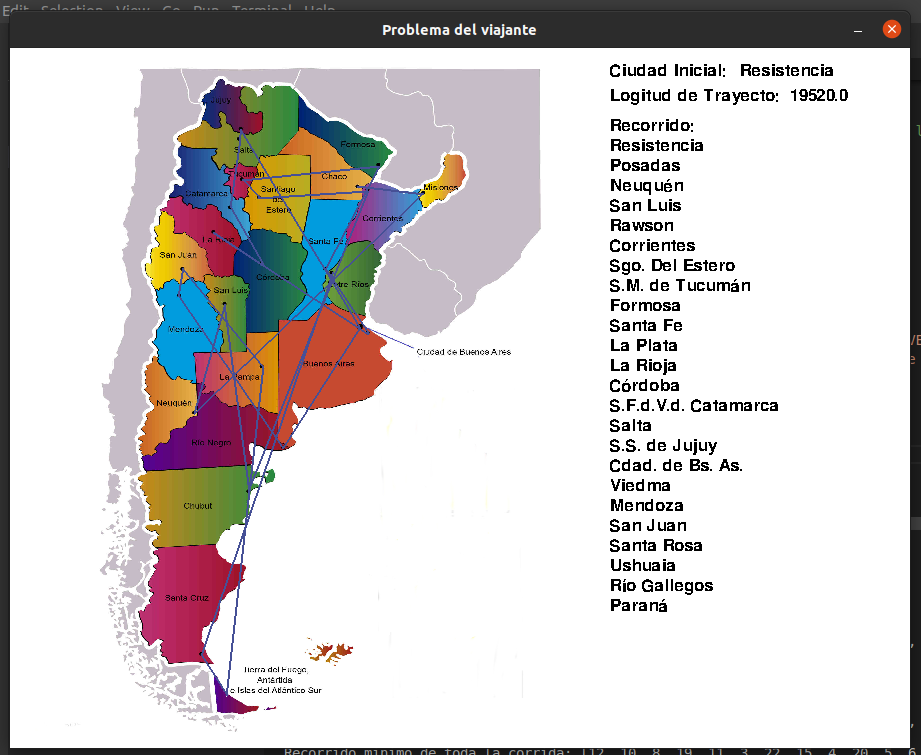
# El programa

## Salida

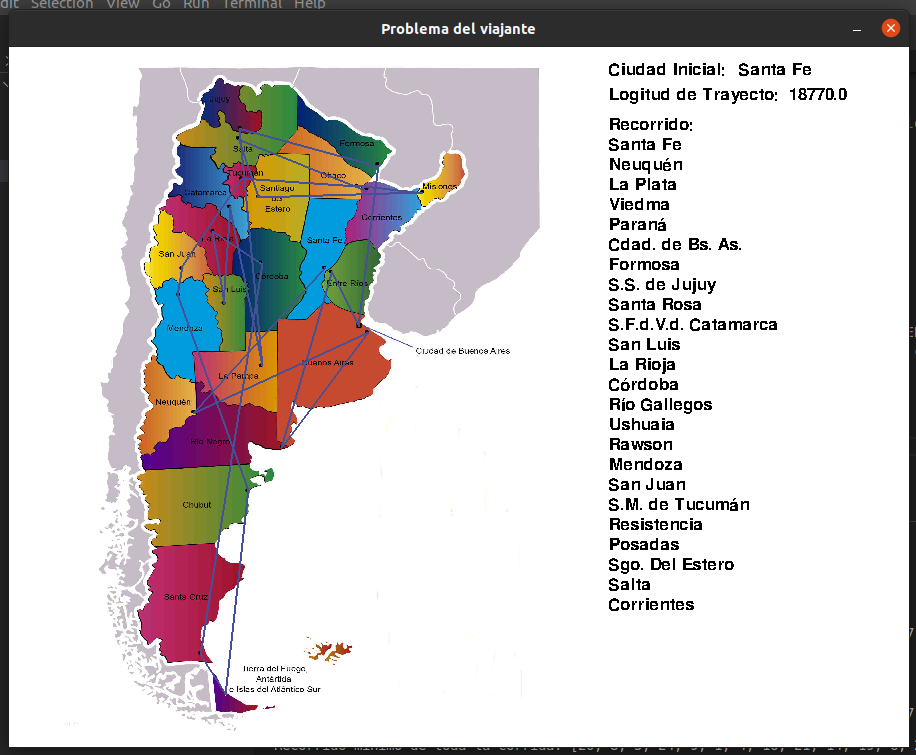
**Método Heurístico**

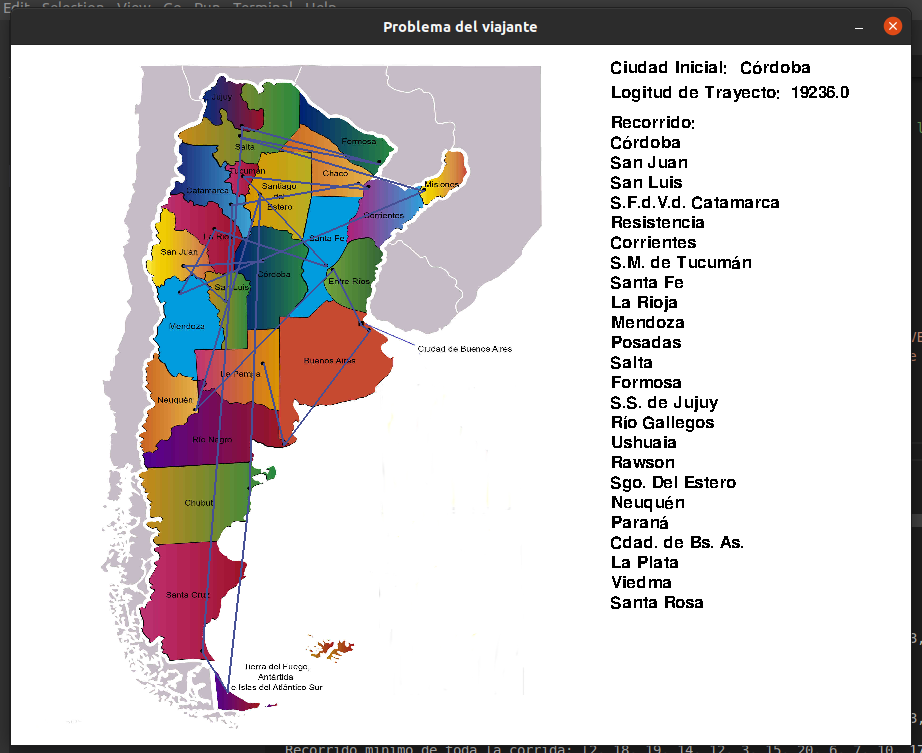
****

**100 Ciclos**



**200 Ciclos**

****

**300 Ciclos**

# Aportes prácticos del TSP

* Planificación de recorridos en diversas aplicaciones (transporte interurbano, turismo, viajes de negocios, etc.)

Se puede extender la cantidad de aplicaciones del problema del viajante si en lugar de tratar con distancias entre ciudades, se trata con diversos “puntos” u “objetivos” y una función de costo asociada a pares. Por ejemplo:

**Aplicación en la medicina**

Si se requiere que un médico visite regularmente una serie de hospitales. La cantidad de pacientes que deberá ver en un hospital incrementará si transcurre mucho tiempo desde la última vez que lo visitó. A su vez, este incremento será mayor si los pacientes presentan riesgo o afecciones crónicas en una determinada unidad. De esta manera, se puede establecer un costo en términos de tiempo que requiere ver una cantidad de pacientes en un determinado hospital y luego viajar a otro. Se puede establecer una mejor ruta de visitas para aprovechar los recursos humanos en la medicina si se minimiza el costo.

**Aplicación en el transporte**

En lugar de utilizar ciudades, una empresa dedicada al servicio de transporte público puede tomar como referencia las zonas más concurridas de una ciudad y buscar los recorridos más óptimos que puedan unirlas para definir el trayecto a seguir de sus unidades, optimizando de esta forma combustible y tiempo.

# 

# Conclusión

Al usar heurística se puede encontrar una solución aceptablemente buena en un tiempo razonable. Los métodos de búsqueda heurística disponen de información sobre la proximidad de cada estado a un estado objetivo, lo que permite explorar en primer lugar los caminos más prometedores.

Al utilizar algoritmos genéticos, el tiempo de ejecución aumentó considerablemente con el número de cíclos empleados.

Se observaron resultados peores y una rápida convergencia en un mínimo local para 200 ciclos. Al aumentar el número de cíclos el tiempo de ejecución superó el minuto y no se alcanzaron grandes mejoras.

Se notó una mejora al aumentar significativamente la probabilidad de mutación junto con un aumento razonable del número de cíclos considerando el costo del tiempo de cómputo, teniendo como desventaja la probabilidad de mutación en la última iteración alterando el resultado.

Se puede concluir que, para este caso particular, se cuenta con una buena heurística que permite llegar a un resultado aceptable del problema en un tiempo mucho más corto y con menor costo de implementación.