Tarea 2: Desarrollar algoritmo genético para el problema del vendedor viajero

Ignacio Haeussler

24 de noviembre de 2018

1. Problema: TSP

El problema a resolver elegido es el problema del vendedor viajero (TSP por sus siglas en inglés), en el que un vendedor debe viajar por n ciudades una única vez, volviendo a la inicial finalmente, recorriendo la menor cantidad de distancia posible. El problema es modelado por n puntos en el plano cartesiano, y los recorridos pueden ser modelados por permutaciones de los índices de las n ciudades.

Se propone resolver el problema mediante un algoritmo genético, en el que los individuos corresponden a distintas permutaciones, inicialmente aleatorias, de los índices de las ciudades, una operaciones de mutación a un intercambio entre 2 índices aleatorios en una permutación, y la reproducción a una aplicación de crossover ordenado entre dos permutaciones. Adicionalmente, la selección de miembros utilizada consistió en seleccionar una fracción con mayor aptitud, por ejemplo la mitad, el tercio, el cuarto o el quinto de la población con mayor aptitud.

Los tests realizados consistieron en generar 100 archivos de 25 pares de números enteros, para luego aplicar algoritmos genéticos con distintos parámetros u otros algoritmos que resolvieran TSP. El primer test obtiene la distancia total promedio para distintos tamaños de población, fijando como valores iniciales las épocas, tasa de mutación y fracción de selección (150, 0.5 y 1/2 respectivamente). El segundo elige la mejor población y la fija junto a los otros parámetros, excepto la fracción de selección. Finalmente, el tercer experimento compara el mejor obtenido del experimento anterior con los algoritmos aproximados Convex Hull y Minimum Spanning Tree.

2. Sistema operativo, lenguaje de programación y librerías utilizadas

El sistema operativo utilizado fue ubuntu 18.04, el lenguaje fue C++14 y para compilar fue creado una archivo Makefile, el cual una vez estando en el directorio raíz del proyecto, puede ser ejecutado utilizando el comando make. Adicionalmente, éste asume un ambiente linux para crear carpetas asociadas a la creación de los ejecutables.

Por otro lado, la única librería utilizada fue OpenCV 3.4 para realizar gráficos en C++. Para su instalación en Ubuntu 18.04 las instrucciones en el siguiente link fueron seguidas: https://www.pyimagesearch.com/2018/05/28/ubuntu-18-04-how-to-install-opencv/, las que además permiten en ubuntu 18.04 compilar código C++14.

3. Correr implementación

Una vez estando en la raíz del código y habiendo compilado satisfactoriamente, el siguiente comando ejecuta el programa de la tarea, el cual consiste en el despliegue de gráficos que entregan los resultados asociados a los distintos experimentos:

```
./build/Arrays/task2
```

Por otro lado, la implementación de los algoritmos genéticos están en los archivos src/GP.h, src/GP.cpp y src/GP/Arrays/testArrays.cpp. El ejecutable asociado a los test estaría en build/GP/Arrays/testArrays.

4. Resultados de experimentos

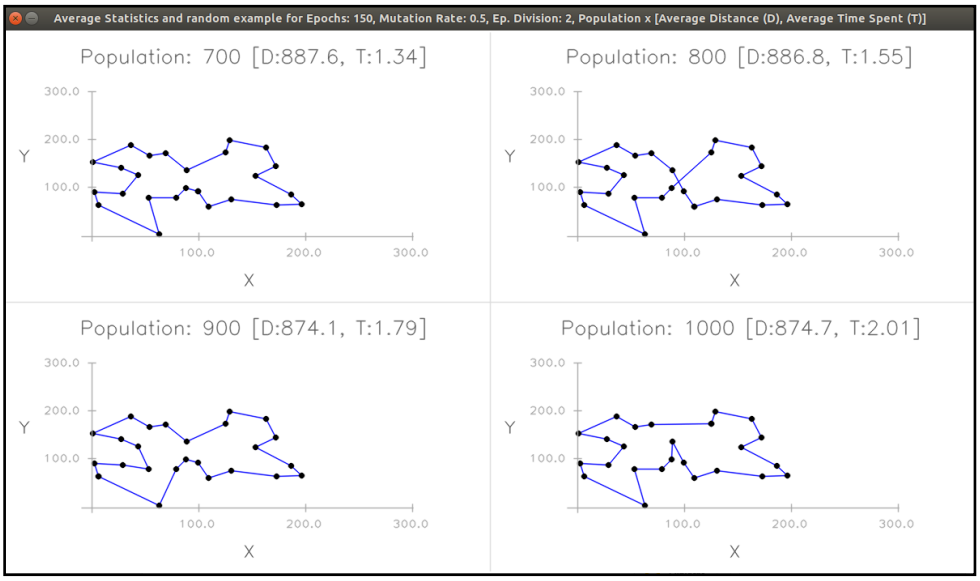

Figura 1: Distancia total promedio, tiempo promedio y ejemplo aleatorio de miembro más apto de algoritmo genético, para 150 épocas, tasa de mutación 0.5 y selección de mitad más apta, para distintos tamaños de población

Se puede apreciar en la Figura 1 que para 900 miembros, el algoritmo obtiene el mejor resultado promedio para los experimentos realizados. La tendencia parece indicar que a una mayor cantidad de población, mejores resultados promedio tienden a obtenerse, con un tope máximo a esta mejora, a la vez que un aumento en el tiempo requerido promedio.

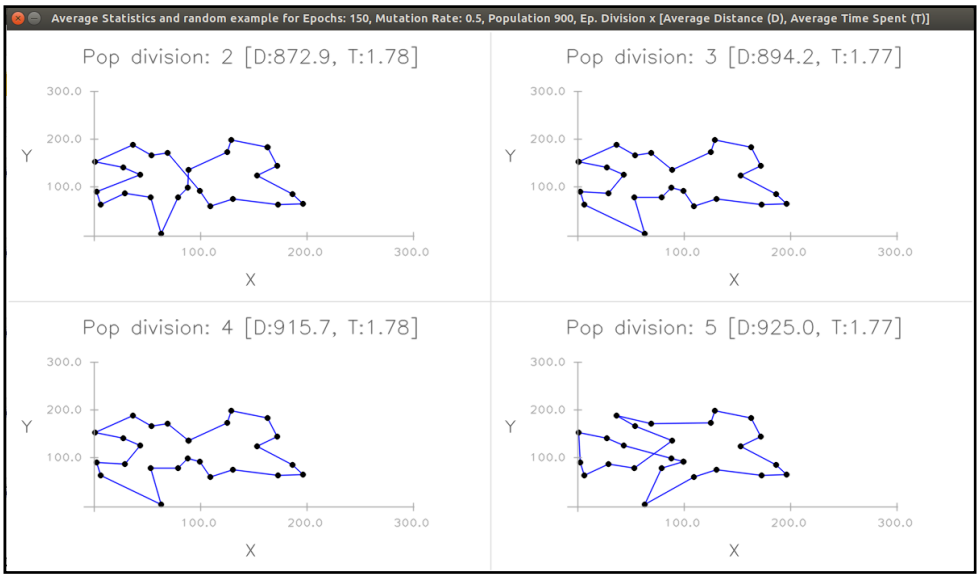

Figura 2: Distancia total promedio, tiempo promedio y ejemplo aleatorio de miembro más apto de algoritmo genético, para 150 épocas, tasa de mutación 0.5 y población de 900 individuos, para distintos niveles de selección (2:mitad más apta, 3:tercio más apto, 4:cuarto más apto y 5:quinto más apto).

En la Figura 2 puede observarse que seleccionar la mitad con mayor aptitud tiende a ser mejor que seleccionar fracciones más pequeñas. Esto puede estar asociado a una disminución perjudicial inicial de la variabilidad de las soluciones encontradas.

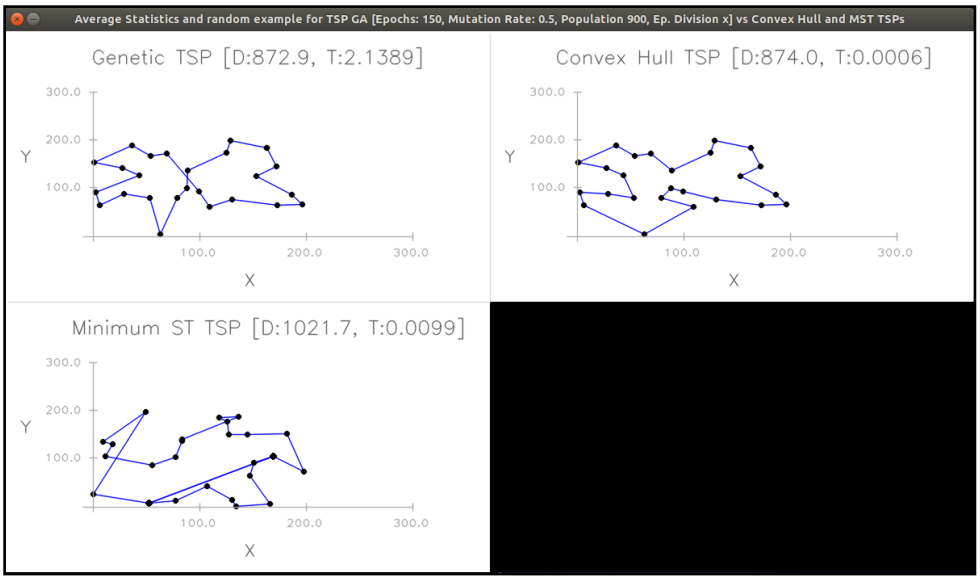

Figura 3: Distancia total promedio, tiempo promedio y ejemplo aleatorio de miembro más apto de algoritmo genético, para 150 épocas, tasa de mutación 0.5, nivel de selección 2 y población de 900 individuos, comparado con algoritmos Convex Hull y Minimum Spanning Tree

Finalmente, en la figura 3 se observa que el algoritmo Genético obtenido supera en promedio a los otros algoritmos probados (Convex Hull y Minumum Spanning Tree), a pesar de ser considerablemente más lento. Esto insinúa la hipótesis de que en general los algoritmos genéticos serían inferiores a algoritmos que incorporen una mayor parte de la estructura del dominio del problema que se esté enfrentando.