Práctica 3 Parte 2

PROBLEMA DEL VIAJANTE DE COMERCIO

Grupo: 2º A1
Mena Barrera, Miguel Ángel
Nikolov Vasilev, Vladislav
Sánchez Guerrero, José María
Vallecillos Ruiz, Fernando

Introducción

El problema del viajante de comercio se define como: dada una lista de ciudades y las distancias entre cada par de ellas, un viajante debe recorrer todas las ciudades exactamente una vez y volver al punto de partida, de forma que la distancia recorrida sea mínima.

En esta práctica realizaremos un análisis y comparación de tres diferentes heurísticas para resolver este problema:

- Heurística del vecino más cercano.
- Heurística de inserción.
- Heurística propia.

Heurística del vecino más cercano

Se elige una ciudad inicial v_0 y se añade la ciudad más cercana. Este proceso se repite añadiendo la ciudad más cercana a la última añadida hasta que no queden más ciudades.

Solo con un primer acercamiento se podría deducir que esta heurística, aunque rápida y sencilla, no va a proporcionar un camino óptimo.

A continuación, el pseudocódigo de la implementación de este algoritmo:

```
posCiudadCercana(ciudades, pos)
    pos_mejor := -1;
    mejor_distancia := inf;

    for it := 0 to candidatos.size()  do
        distancia := CalcularDistancia( ciudades[it],
ciudades[pos]);

        if ( distancia < mejor_distancia)  then
            pos_cercana := it;
            mejor_distancia := distancia;

    return pos_cercana;
main()
    orden := int<0..N>;
    aux, it_pos := pos_inicio;

    for i := 0 to ciudades.size()-1  do
        it_pos = posCiudadCercana(ciudades, it_pos);
        orden.push_back( it_pos );
```

Hemos realizado una comparación entre esta heurística y la óptima. Como un primer acercamiento, situamos la *ciudad 1* como la inicial. Obtuvimos:

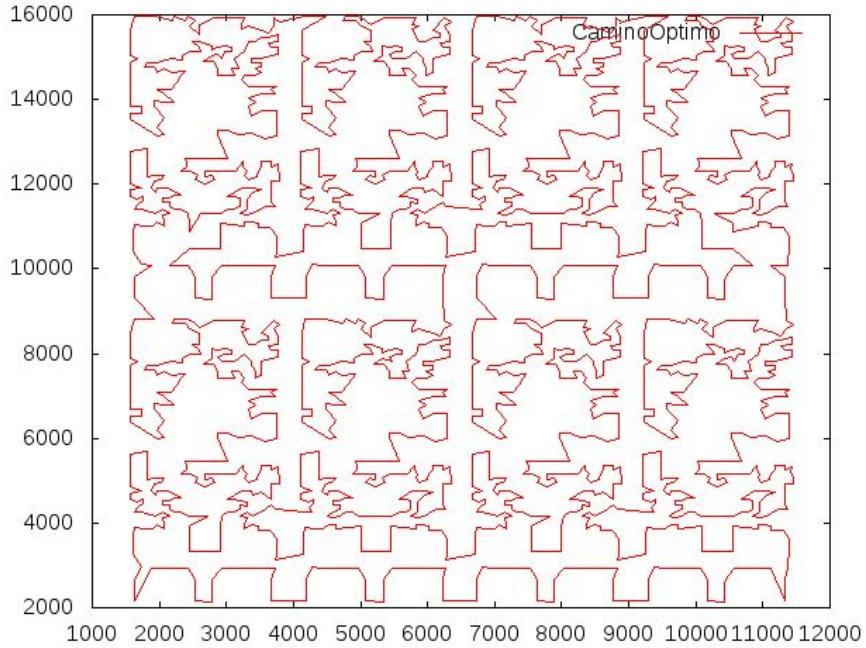
El recorrido total del algoritmo propio es: 461207
El recorrido total del algoritmo óptimo es: 378063
La diferencia entre ambos algoritmos es: 83144

Vemos que el camino óptimo es alrededor de un 18% más eficiente que con esta heurística.
Para asegurarnos, comprobamos con otra serie de ciudades iniciales para ver el cambio de resultados.

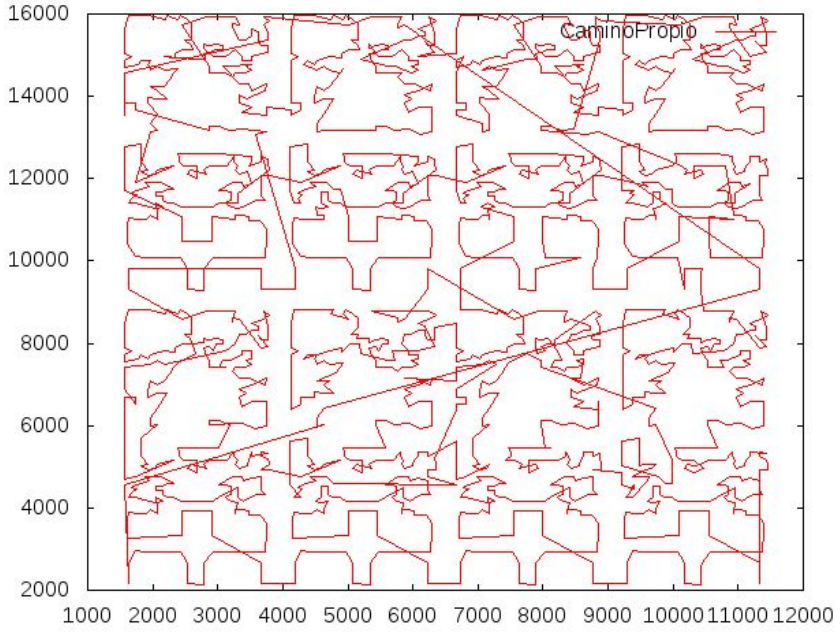
Comenzando por la ciudad: 150
La diferencia entre ambos algoritmos es: 96442
Comenzando por la ciudad: 606
La diferencia entre ambos algoritmos es: 101066
Comenzando por la ciudad: 1148
La diferencia entre ambos algoritmos es: 93085

Como se puede apreciar, dependiendo de la ciudad inicial la longitud del recorrido cambia. Esto sería suficiente para demostrar que no es óptimo.

Camino óptimo



Camino aplicando heurística



Heurística con inserción mas economica

La heurística de la inserción más económica es otra estrategia que intenta dar una solución aproximada al problema del TSP. Se comienza con un recorrido inicial que ya incluya algunas ciudades, y se inserta en el recorrido aquella ciudad que provoque el menor incremento del coste total del recorrido.

El problema viene a la hora de determinar cuál es la mejor ciudad a insertar y la mejor posición donde insertarla.

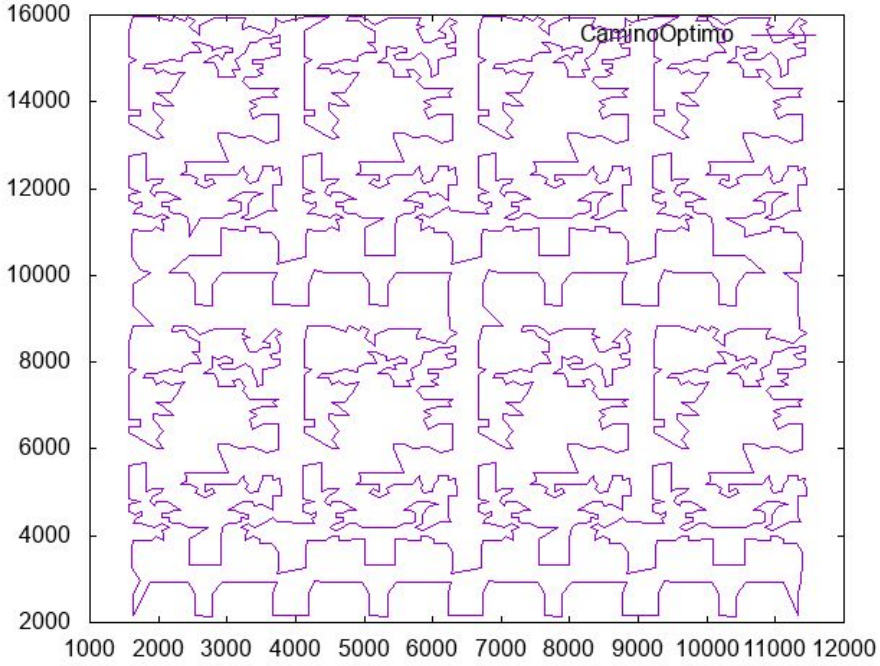
Vamos a partir de un conjunto S de ciudades que formen un **recorrido parcial** y un conjunto C de **ciudades candidatas**. El problema se reduce a encontrar dos ciudades $i, j \in S$ entre las que exista un camino, y una ciudad $k \in C$ tal que al insertar k entre i, j, el coste total del nuevo recorrido sea el mínimo. Este incremento se puede calcular de la siguiente forma:

$$\Delta c = d_{ik} + d_{kj} - d_{ij}$$

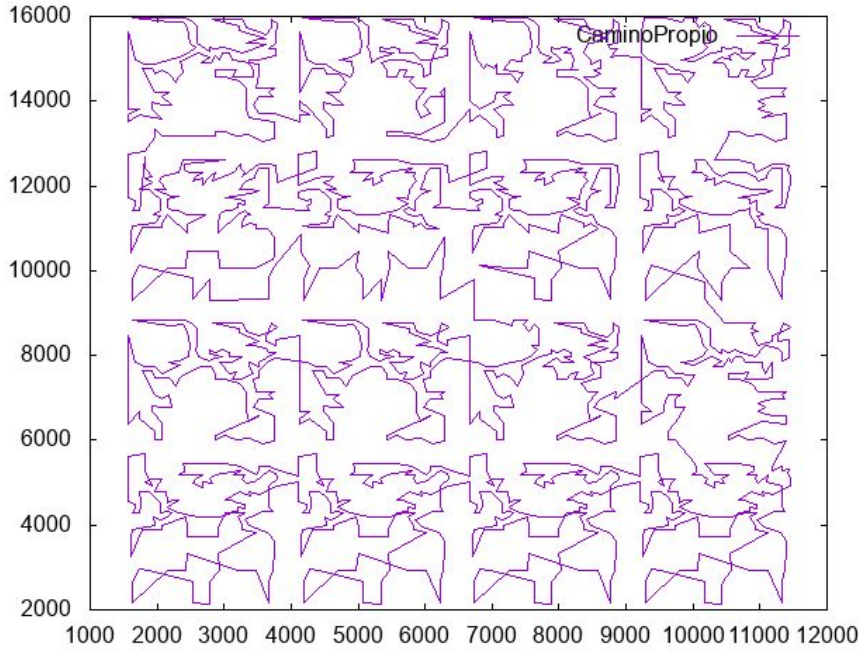
Seguidamente se ofrece un pseudocódigo para demostrar una implementación del procedimiento implementado:

```
insercionTSP(candidatos, solucion)
  // se insertan las primeras ciudades
  solucion := ciudadNorte, ciudadEste, ciudadOeste;

  while (!candidatos.empty())
    coste = inf;

    for i := 0 to solucion.size() - 1 do
      nodoActual := solucion[i];
      nodoSiguiente := solucion[(i + 1) % solucion.size()];

      for j := 0 to candidatos.size() - 1 do
        costeAux := distancia(nodoActual, j) + distancia(j, nodoSiguiente)
                              - distancia(nodoActual, nodoSiguiente);

        if (costeAux < coste) then
          coste := costeAux;
          ciudad := j;
          posicion := nodoSiguiente;

    solucion.insert_before(posicion, ciudad);
    candidatos.erase(ciudad);
```

Camino óptimo



Camino aplicando heurística (inserción)



Heurística de cercanía doble

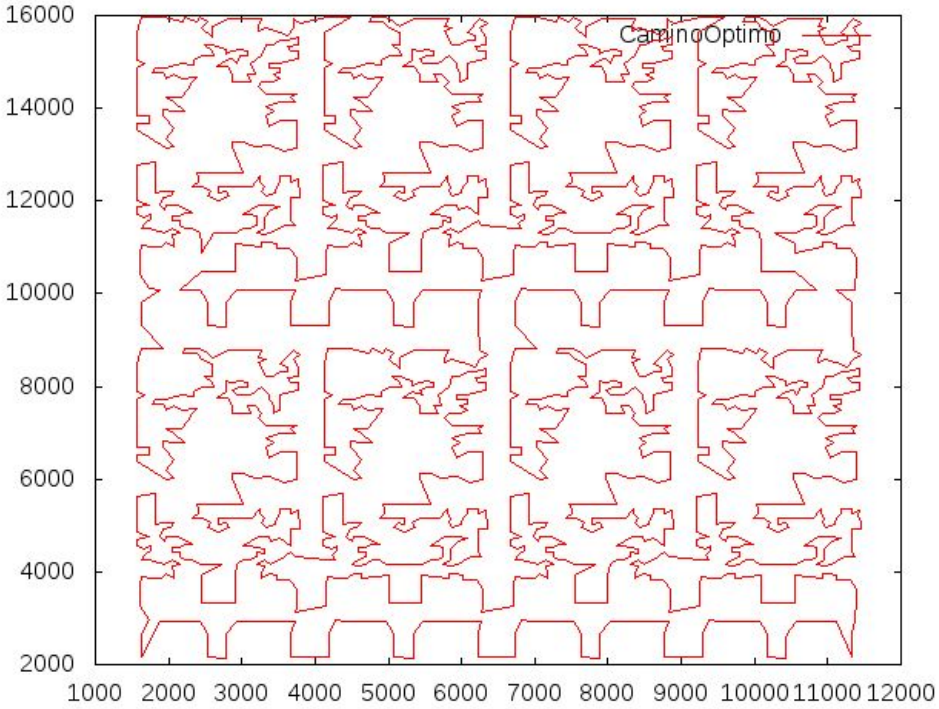
Esta estrategia está basada en la heurística de cercanía vista anteriormente. Sin embargo, vamos a tener en cuenta las posibles distancias por ambos extremos de nuestro camino. Es claro que no será un algoritmo óptimo sin embargo, lo elegimos para poder analizar como un algoritmo evoluciona aplicando nuevas técnicas.

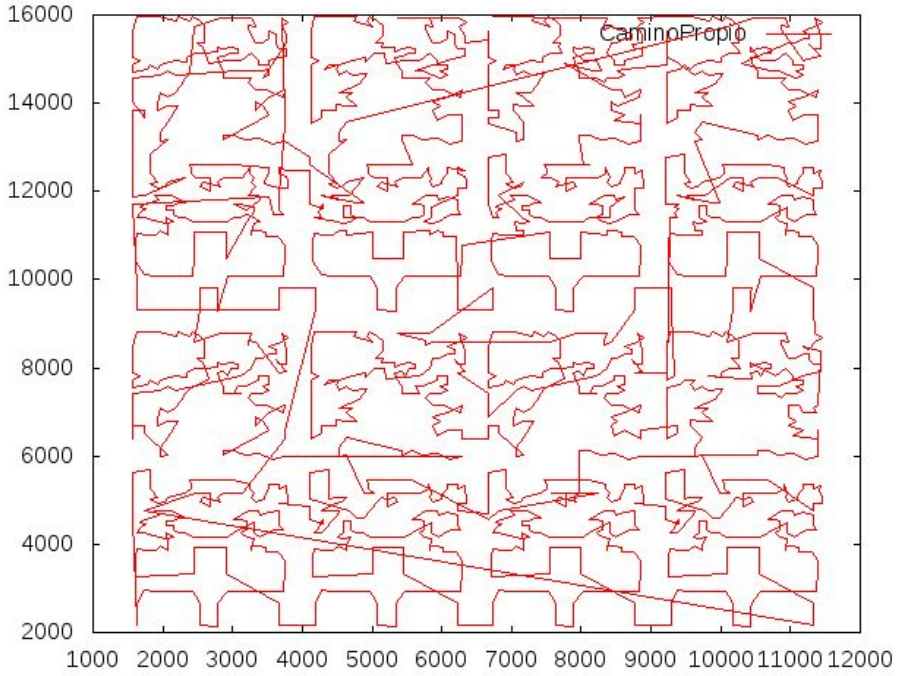
El recorrido total del algoritmo propio es: 465457
El recorrido total del algoritmo óptimo es: 378063
La diferencia entre ambos algoritmos es: 87394.2

Vemos que el camino óptimo es alrededor de un 17% más eficiente que con esta heurística.

Comenzando por la ciudad: 450
La diferencia entre ambos algoritmos es: 100080
Comenzando por la ciudad: 196
La diferencia entre ambos algoritmos es: 100798
Comenzando por la ciudad: 259
La diferencia entre ambos algoritmos es: 102120

Como se puede apreciar, al igual que en el primer algoritmo, dependiendo de la ciudad inicial la longitud del recorrido cambia. Esto sería suficiente para demostrar que no es óptimo.

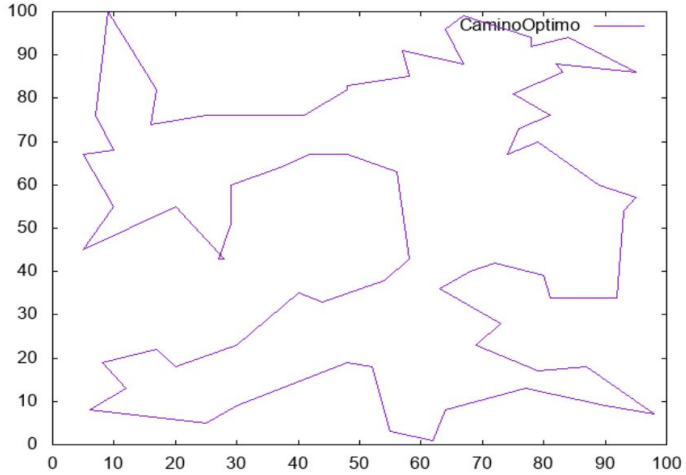

CaminoOptimo
16000
14000
12000
10000
8000
6000
4000
2000
1000 2000 3000 4000 5000 6000 7000 8000 9000 10000 11000 12000


CaminoPropio
16000
14000
12000
10000
8000
6000
4000
2000
1000 2000 3000 4000 5000 6000 7000 8000 9000 10000 11000 12000

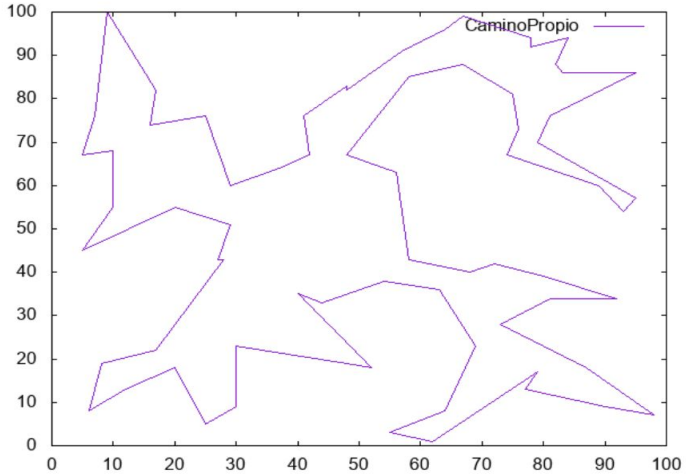
Comparativa

Mapa	Cercano	Inserción	Heurística propia
pr2392	Óptimo: 378063 Real: 461207 Difer: 83144.7	Óptimo: 378063 Real: 454593 Difer: 76530.3	Óptimo 378063 Real 465008 Difer 86945.6
pa561	Óptimo: 19330.8 Real: 19075 Difer: -255.766	Óptimo: 19330.8 Real: 16575.2 Difer: -2755.57	Óptimo 19330.8 Real 19053.9 Difer: -276.855
tsp225	Óptimo: 3859 Real: 4829 Difer: 969.997	Óptimo: 3859 Real: 4509.49 Difer: 650.494	Óptimo 3859 Real 4786.29 Difer: 927.286
lin105	Óptimo: 14383 Real: 20362.8 Difer: 5979.76	Óptimo: 14383 Real: 16285.1 Difer: 1902.07	Óptimo 14383 Real 20217.7 Difer 5834.72
st70	Óptimo: 678.597 Real: 805.531 Difer: 126.934	Óptimo: 678.597 Real: 761.881 Difer: 83.2838	Óptimo 678.597 Real 745.536 Difer 66.9388
ulysses16	Óptimo: 74.1087 Real: 104.735 Difer: 30.6262	Óptimo: 74.1087 Real: 78.6968 Difer: 4.58805	Óptimo 74.1087 Real 122.881 Difer 48.7718

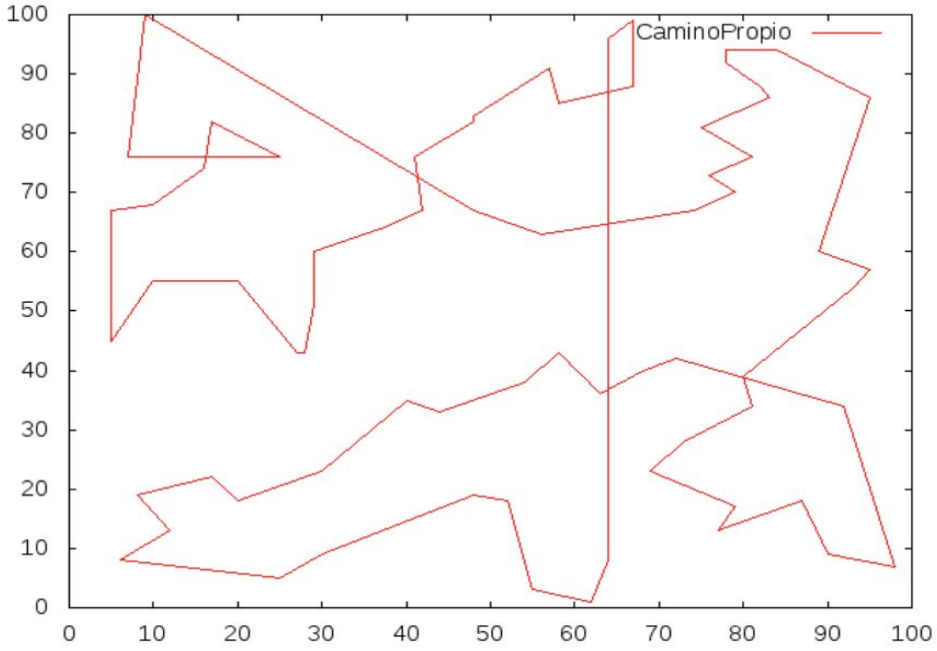
Circuito óptimo para el mapa ST70



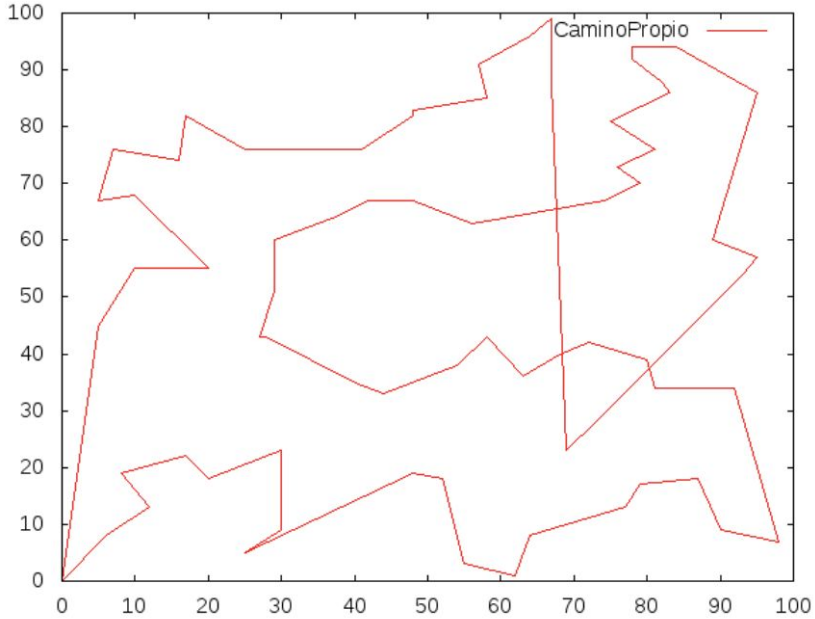
Circuito obtenido con la heurística de inserción de menor coste para el mapa ST70



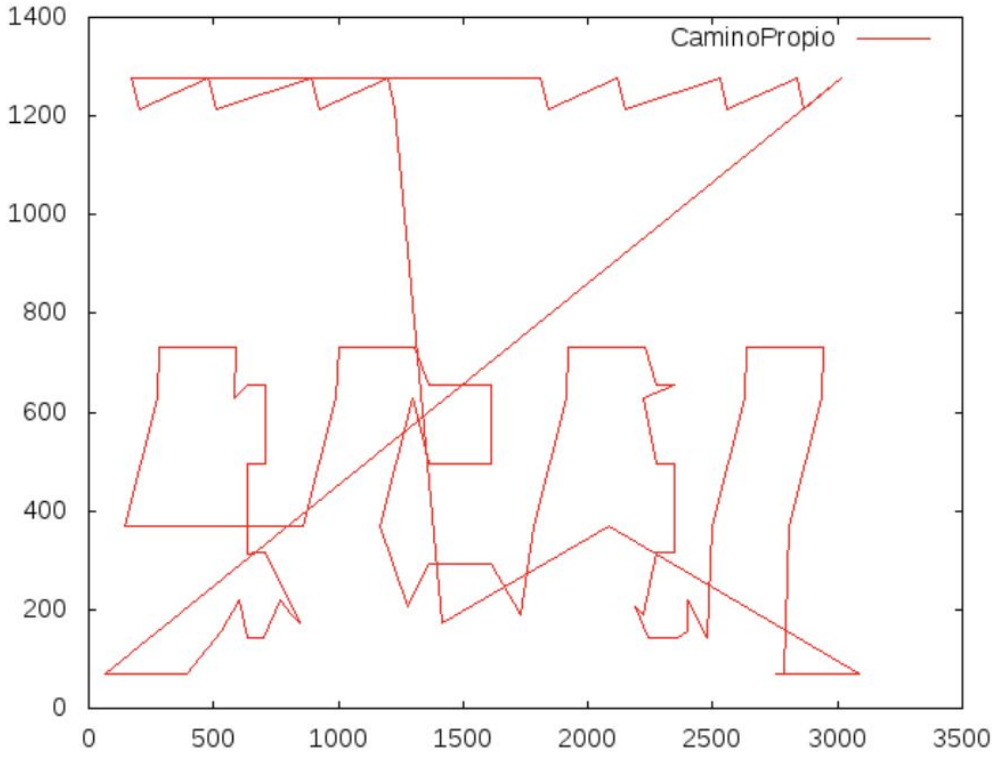
Circuito obtenido con la heurística del vecino más cercano para el mapa ST70



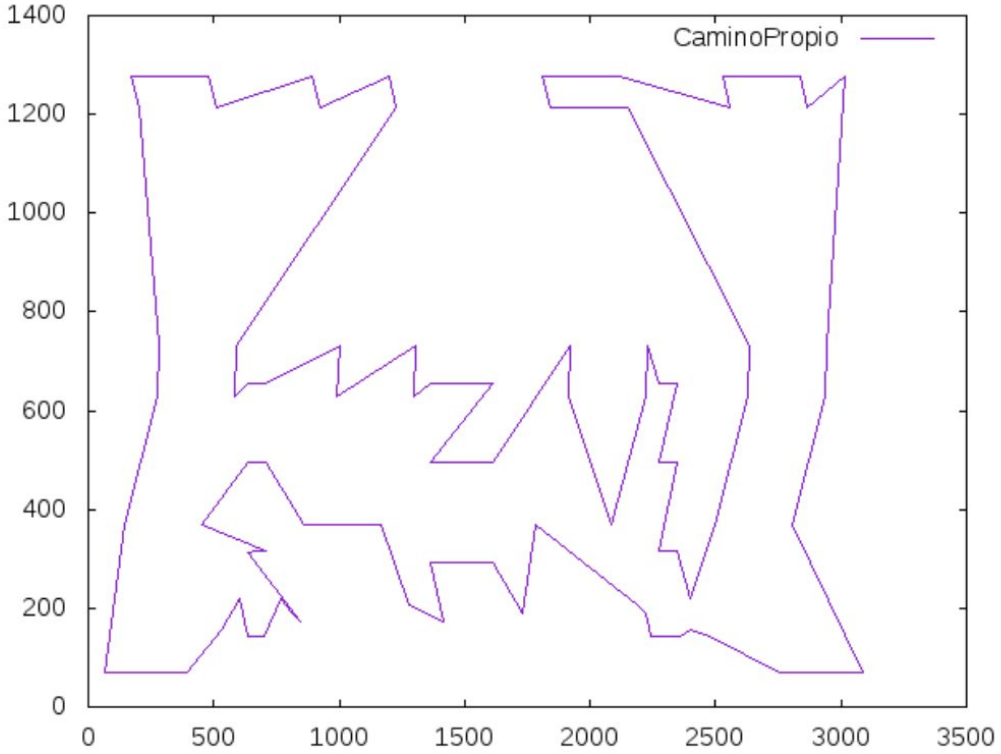
Circuito obtenido con la heurística del vecino más cercano doble para el mapa ST70



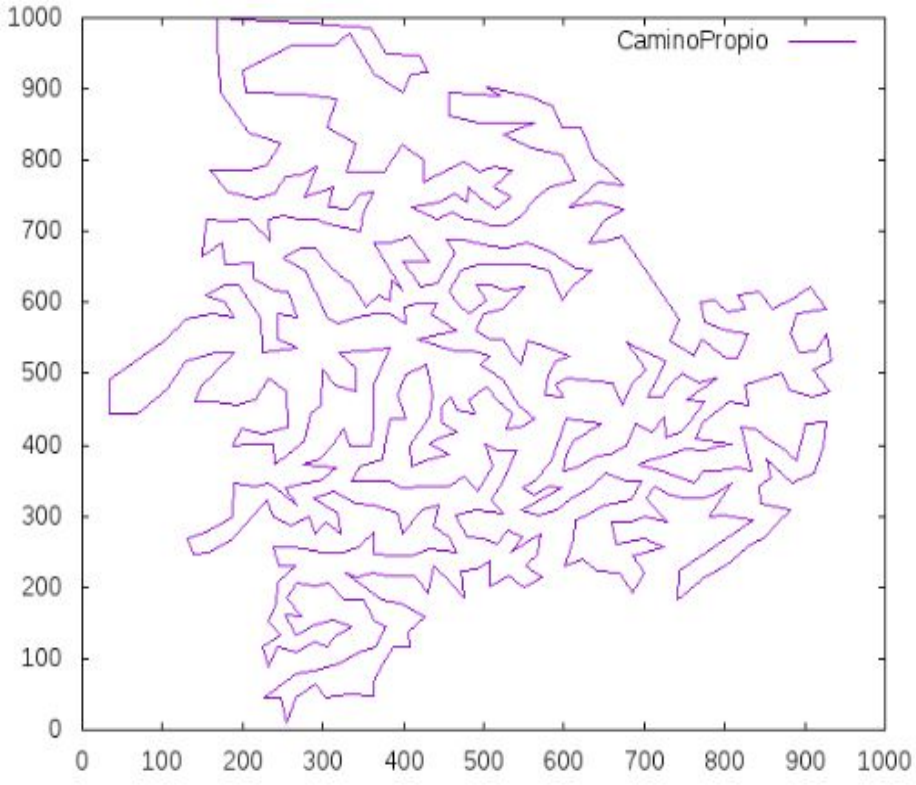
Circuito obtenido con la heurística del vecino más cercano para el mapa LIN105



Circuito obtenido con la heurística de inserción de menor coste para el mapa LIN105



Inserción pa561



Óptimo pa561