UNIVERSIDAD DE GRANADA

MEMORIA PRÁCTICA 3 PARTE 2: PROBLEMA DEL VIAJANTE DE COMERCIO

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

Introducción

El problema del viajante de comercio (TSP en inglés), 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. Más formalmente, dado un grafo conexo y ponderado, debemos encontrar el ciclo hamiltoniano de mínimo peso de ese grafo.
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

La heurística del vecino más cercano es extremadamente simple. Se elige una ciudad inicial v_0 (de forma aleatoria o arbitraria) y se añade la ciudad más cercana. Este proceso se repite añadiendo la ciudad más cercana (no incluida en el circuito) 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 (ya que el camino óptimo comenzaba en ésta) y con los datos *pr2392* obtuvimos:

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

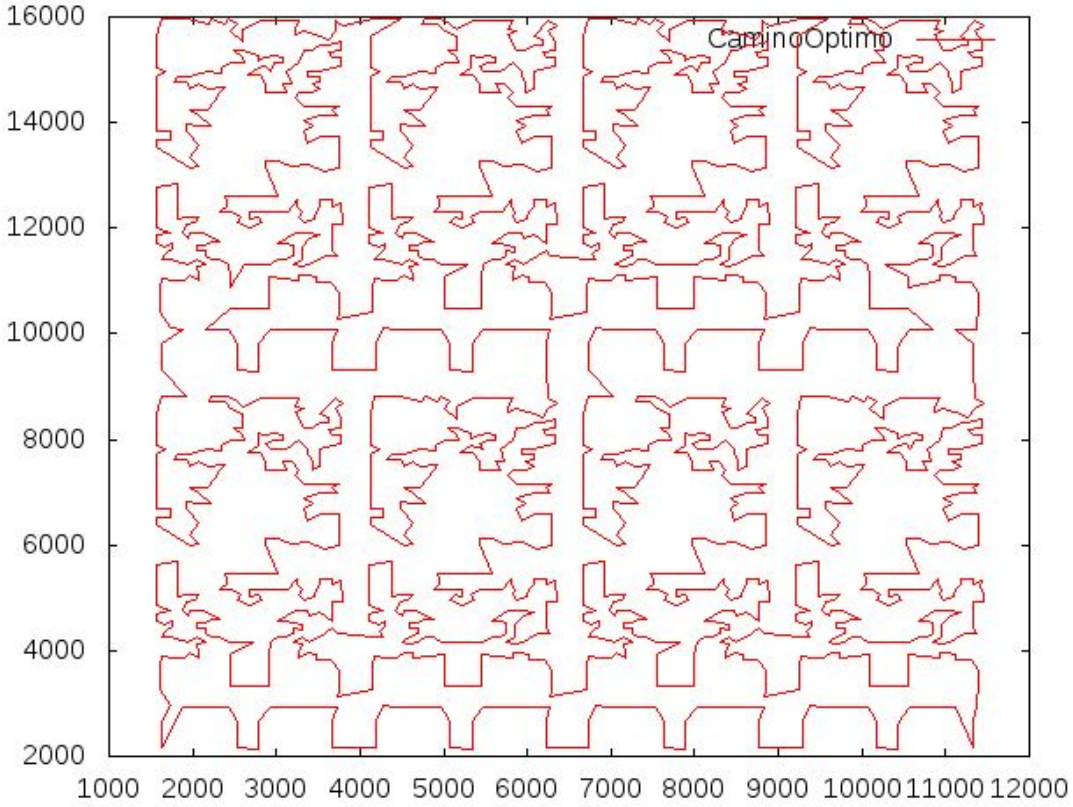
Vemos que el camino óptimo es alrededor de un 18% más eficiente que con esta heurística.
Para asegurarnos, comprobamos con otra serie de ciudades iniciales para ver el cambio de resultados.

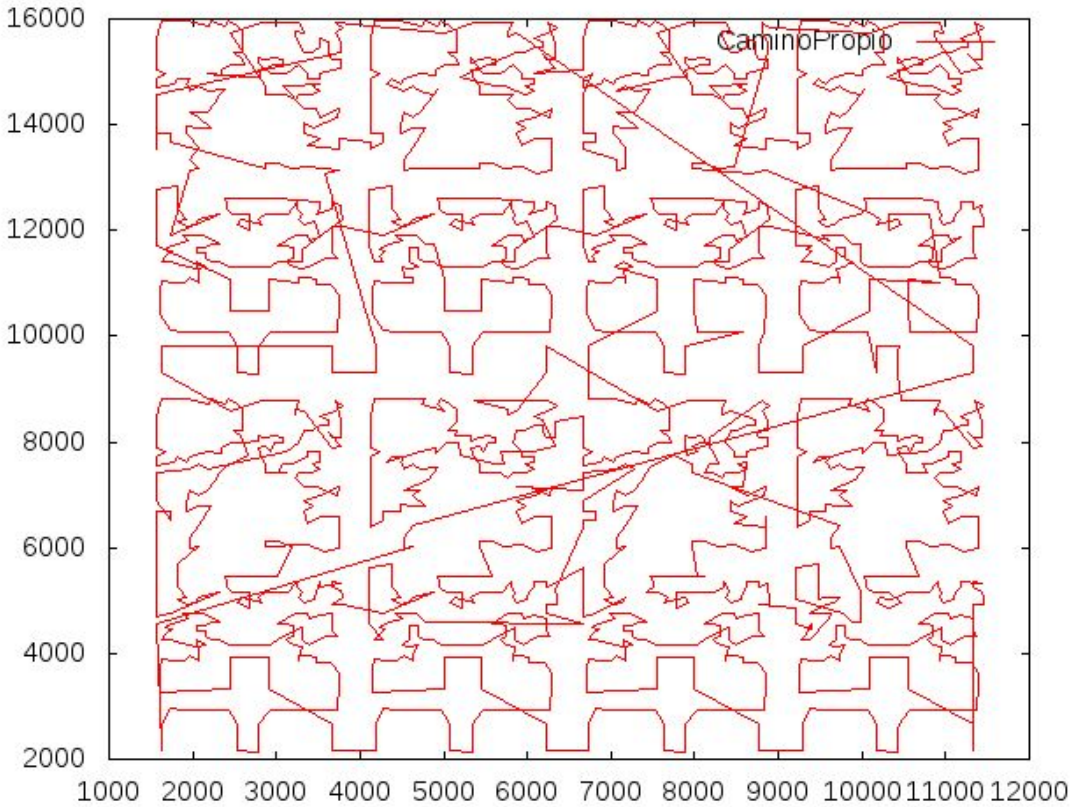
Comenzando por la ciudad: 150
La diferencia entre ambos algoritmos es: 96442
Comenzando por la ciudad: 606
La diferencia entre ambos algoritmos es: 101066
Comenzando por la ciudad: 1148
La diferencia entre ambos algoritmos es: 93085
Comenzando por la ciudad: 1057
La diferencia entre ambos algoritmos es: 97954
Comenzando por la ciudad: 675
La diferencia entre ambos algoritmos es: 100123
Comenzando por la ciudad: 1953
La diferencia entre ambos algoritmos es: 93258
Comenzando por la ciudad: 1548
La diferencia entre ambos algoritmos es: 100135
Comenzando por la ciudad: 923
La diferencia entre ambos algoritmos es: 96561
Comenzando por la ciudad: 1210
La diferencia entre ambos algoritmos es: 97235

Como se puede apreciar, dependiendo de la ciudad inicial la longitud del recorrido cambia. Esto sería suficiente para demostrar que no es óptimo. Además vemos que se produce una media de 90.000 “pasos” extra con esta heurística.

Para apreciar esta diferencia, mostramos el cambio entre el camino óptimo y el que produce esta heurística (comenzando en *ciudad 1*).

Camino óptimo y camino aplicando heurística:





Heurística de inserción más económica

La heurística de la inserción más económica es otra estrategia basada en técnicas greedy que intenta dar una solución aproximada al problema del TSP. Se comienza con un recorrido inicial que ya incluya algunas ciudades/nodos, y se inserta en el recorrido aquella ciudad/nodo 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 realizar la inserción.

Para resolver el problema anterior, 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 de entre todos los posibles. Este incremento se puede calcular de la siguiente forma (siendo *d* la distancia entre dos ciudades y *c* el coste):

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

Una vez habiendo resuelto la forma de determinar cuál es la mejor ciudad a insertar y en qué posición, vamos a proceder a explicar lo que hemos realizado a la hora de implementar el algoritmo.

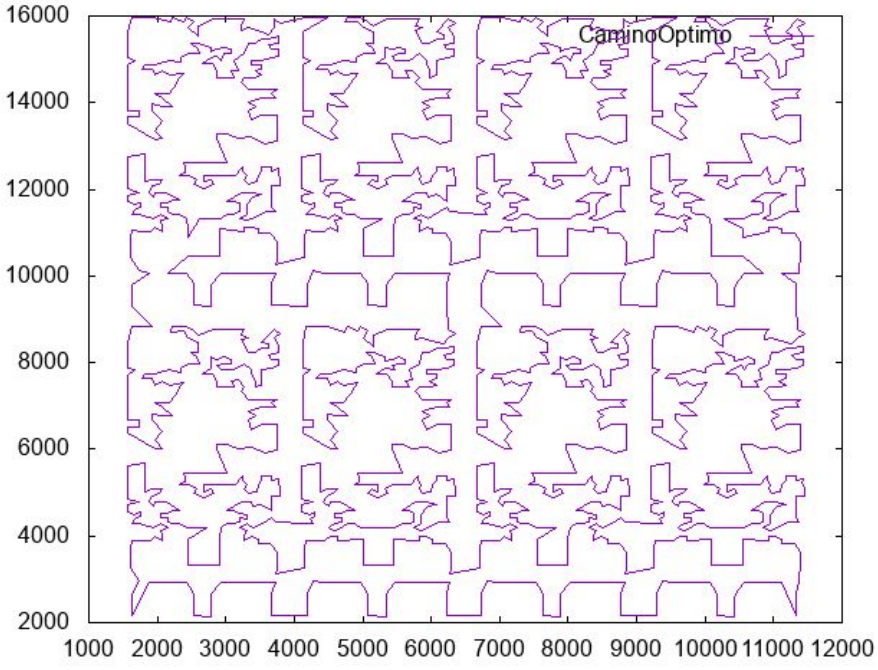
Para tener un recorrido inicial, se han escogido la ciudad más al norte, la ciudad más al oeste, y la ciudad más al este. Obtener estas ciudades es algo trivial y solo se va a dar una pincelada de lo que se ha hecho. Para el caso de la ciudad más al norte, se ha buscado aquella cuya coordenada Y sea la mayor. Para obtener la ciudad más al oeste, se ha buscado aquella cuya coordenada X sea la menor de todas, y para obtener la ciudad más al este, aquella cuya coordenada X sea la mayor de todas. Una vez obtenidas, se han insertado en *S* y se han borrado de *C*.

A continuación se recorre la lista de elementos de *S* y se seleccionan dos ciudades que aparezcan de forma consecutiva en la lista, ya que significa que hay camino entre ambas. Para cada par de ciudades conectadas, se recorren todos los elementos de *C* en busca de aquel cuya inserción entre esas dos ciudades provoque el menor incremento de coste. Cuando se encuentra, se guarda la ciudad de *C* y la posición en la que debe insertarse en la lista de ciudades solución, además de actualizar el costo mínimo que se tiene hasta el momento. Una vez terminados los dos recorridos anteriores, se inserta la ciudad *k* en la posición que le corresponde en la lista de elementos *S* y se elimina de *C*. Todo este proceso se repite mientras *C* contenga ciudades candidatas.

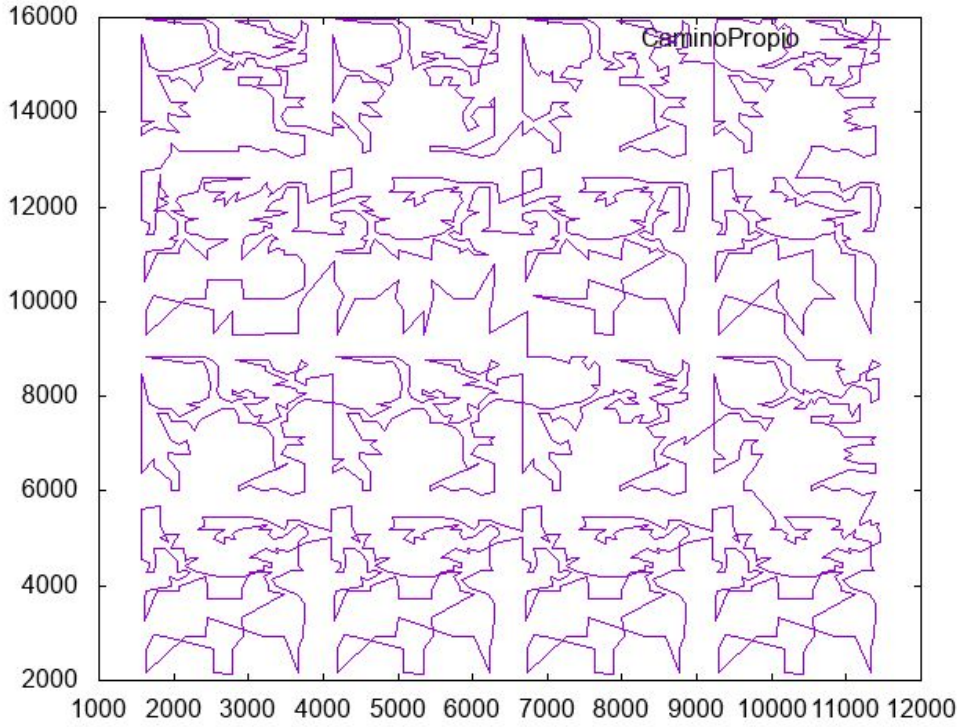
Seguidamente se ofrece un pseudocódigo para mostrar una implementación del procedimiento anteriormente descrito.

```
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);
```

Vamos a ver una comparativa entre el recorrido óptimo y el obtenido por este algoritmo. Para ello, usaremos los datos de *pr2392*. En el caso del recorrido óptimo, obtenemos la siguiente gráfica:



En el caso del algoritmo de inserción, obtenemos la siguiente gráfica:



Como se puede observar a simple vista, en la gráfica obtenida por el algoritmo de la inserción se producen muchos más cortes en el recorrido, mientras que el recorrido óptimo se ve más “natural”. No obstante, la segunda gráfica se parece bastante a la del recorrido óptimo, con lo cual la heurística podría ofrecer una solución “aceptable” dentro de lo que cabe.

No obstante, si comparamos los costes de los recorridos, en el caso del algoritmo óptimo obtenemos que el coste es de 378.063, mientras que el obtenido por la heurística de inserción es de 454.593. Calculando la diferencia entre los dos costes, obtenemos que ésta es 76.530, lo cual representa un incremento de coste respecto al del óptimo de en torno al 20%. Por tanto, aunque la solución al problema pueda parecer aceptable, no es la más óptima en este caso. No obstante, como se podrá comprobar más adelante, esta heurística puede llegar a ofrecer mejores resultados que la solución considerada como óptima.

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. Está claro que a pesar de que no será un algoritmo óptimo lo elegimos para poder analizar como un algoritmo evoluciona aplicando nuevas técnicas.

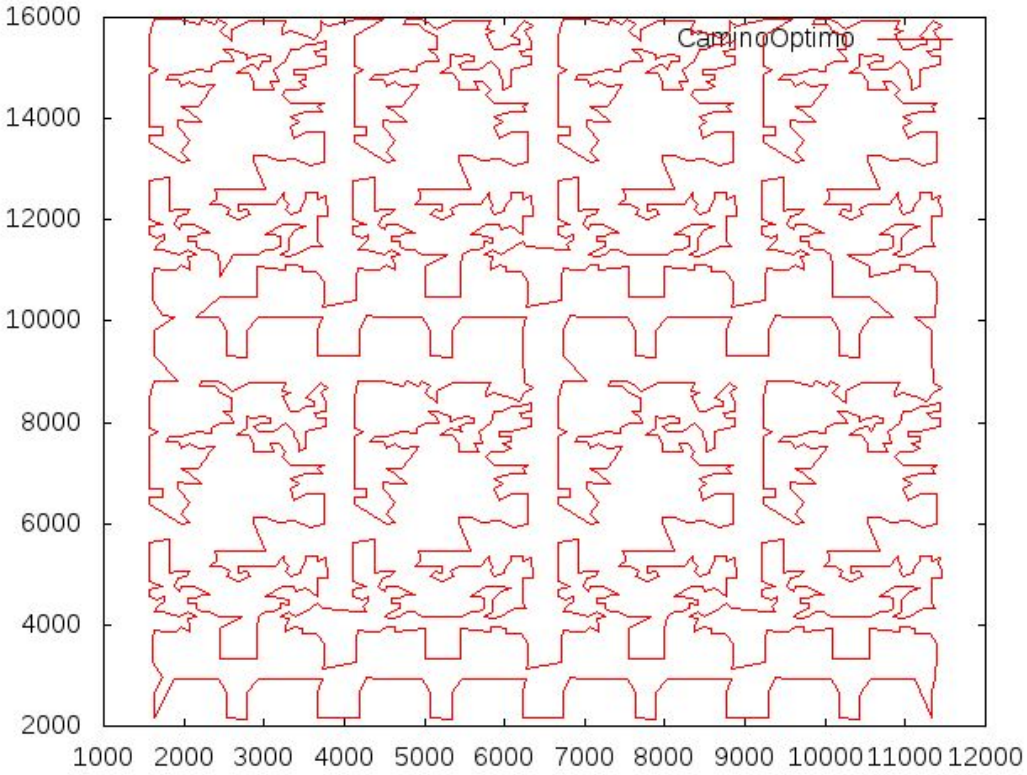
Hemos realizado una comparación entre esta heurística y la óptima. Como un primer acercamiento, situamos la *ciudad 1* como la inicial (ya que el camino óptimo comenzaba en esta) y con los datos *pr2392*. Obtuvimos:

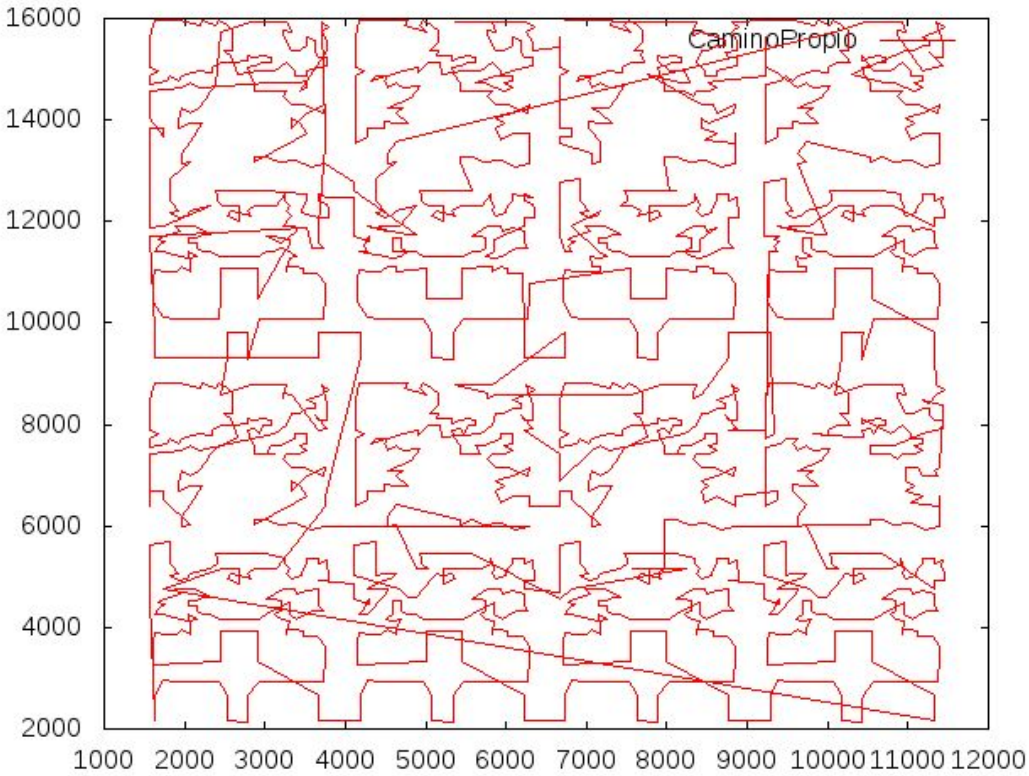
El recorrido total del algoritmo propio es: 465457
El recorrido total del algoritmo óptimo es: 378063
La diferencia entre ambos algoritmos es: 87394.2

Vemos que el camino óptimo es alrededor de un 17% más eficiente que con esta heurística. Al contrario de lo imaginado, el camino ha aumentado. Comprobamos con ciudades iniciales escogidas aleatoriamente:

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
Comenzando por la ciudad: 12
La diferencia entre ambos algoritmos es: 104410
Comenzando por la ciudad: 410
La diferencia entre ambos algoritmos es: 98984.3
Comenzando por la ciudad: 184
La diferencia entre ambos algoritmos es: 101942
Comenzando por la ciudad: 275
La diferencia entre ambos algoritmos es: 101609

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. Además, vemos que la longitud media de los caminos a aumentado significativamente.
Para apreciar esta diferencia, mostramos el cambio entre el camino óptimo y el que produce esta heurística (comenzando en *ciudad 1*).


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

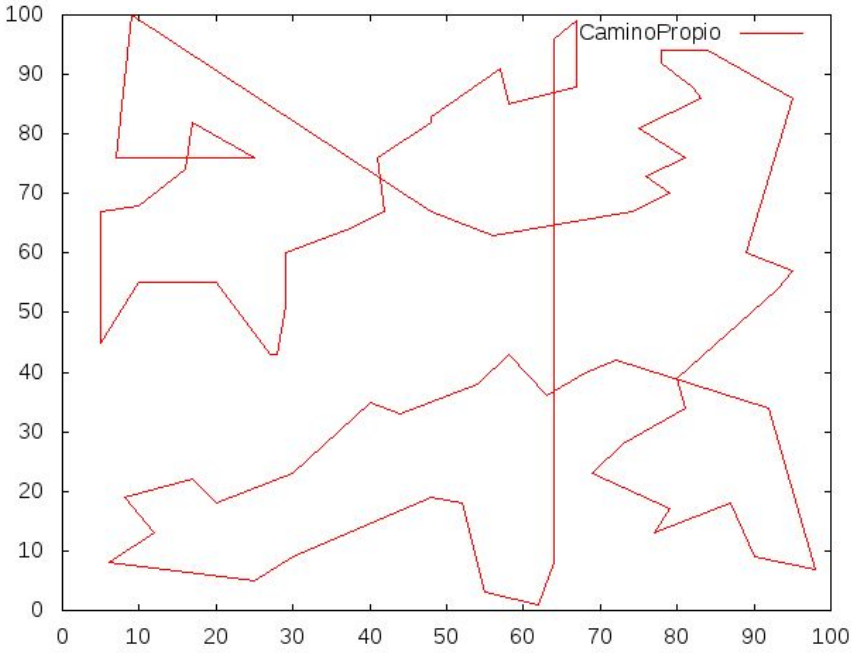
Como podemos ver, la representación gráfica del camino es muy parecida. Sin embargo, podemos decir que no siempre desarrollar un algoritmo es la mejor forma de mejorarlo. Aunque la modificación de la primera heurística permite que tengamos un mayor conjunto de candidatos entre los que seleccionar para añadir, también profundiza los puntos ciegos de la heurística. Esto hace que los errores que la heurística cometía anteriormente hayan sido duplicados. Por tanto, concluimos que antes del desarrollo de un algoritmo, se necesita hacer un estudio previo sobre las implicaciones que podrá tener y, si merece la pena, considerando las posibles ventajas y desventajas.

Tabla comparativa de los algoritmos:

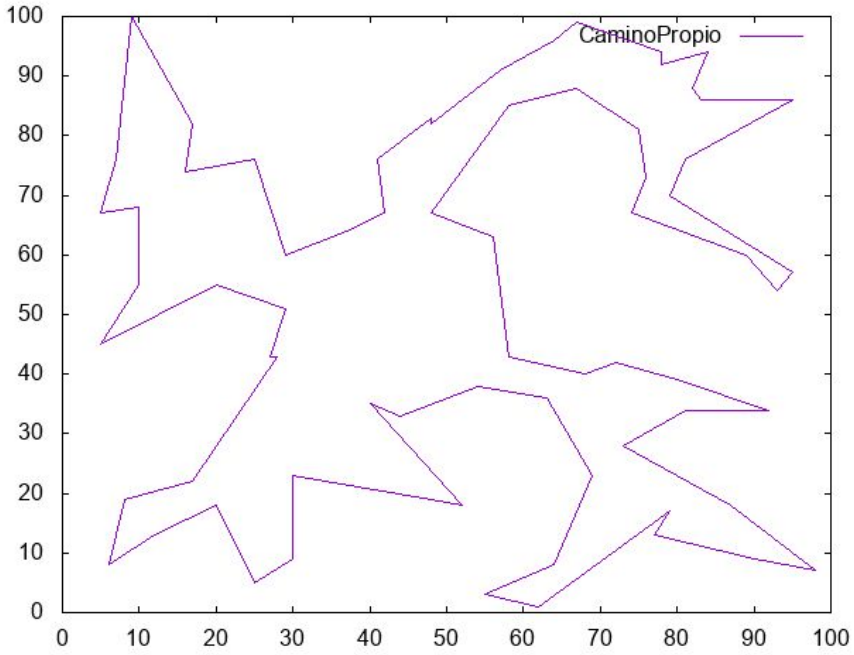
Mapa	Cercano	Inserción	heuristica propia
pr2392	Óptimo: 378063 Real: 461207 Difer: 83144.7	Óptimo: 378063 Real: 454593 Difer: 76530.3	Óptimo 378063 Real 465008 Difer 86945.6
pa561	Óptimo: 19330.8 Real: 19075 Difer: -255.766	Óptimo: 19330.8 Real: 16575.2 Difer: -2755.57	Óptimo 19330.8 Real 19053.9 Difer: -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

Imágenes comparando los algoritmos:

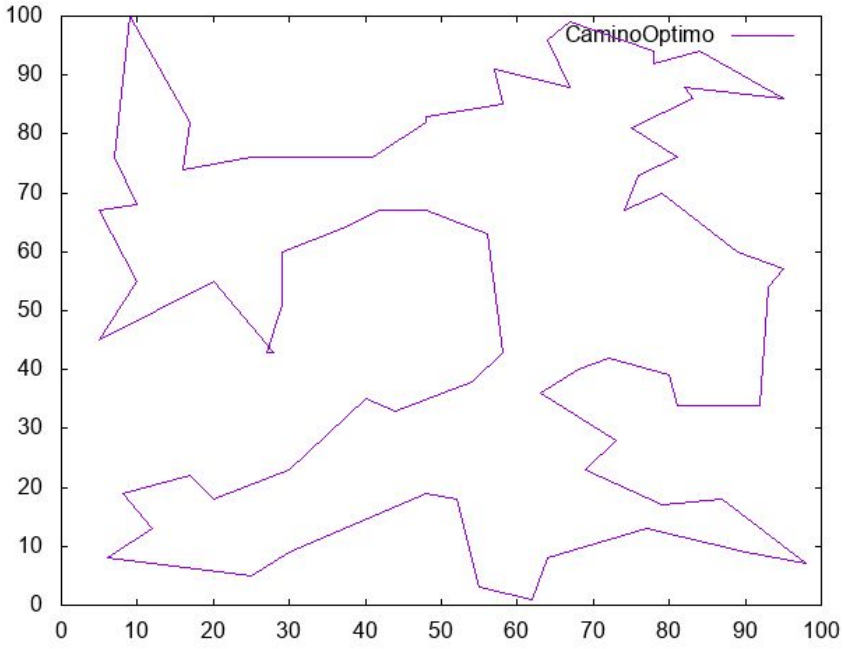
Circuito obtenido con la heurística del vecino más cercano para el mapa ST70



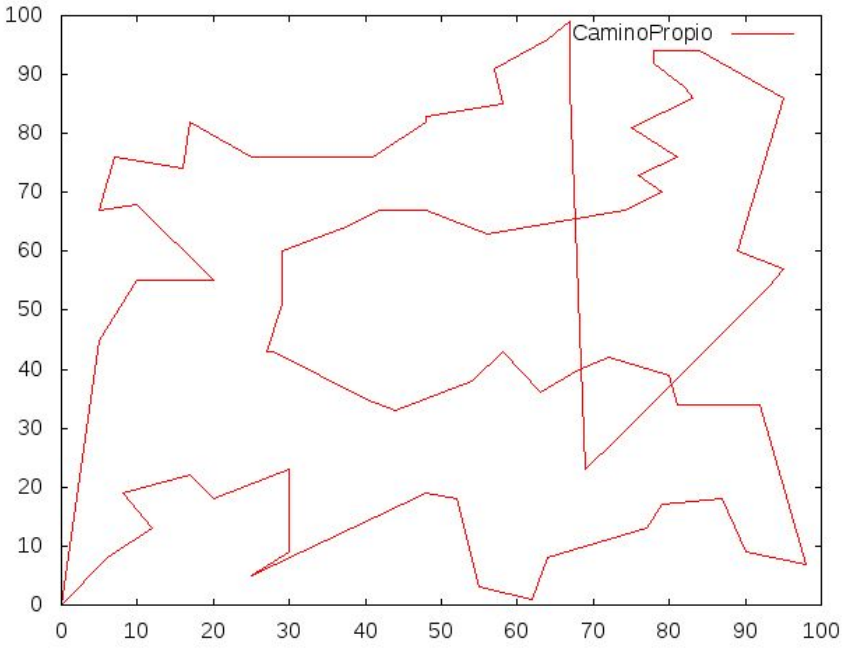
Circuito obtenido con la heurística de inserción de menor coste para el mapa ST70



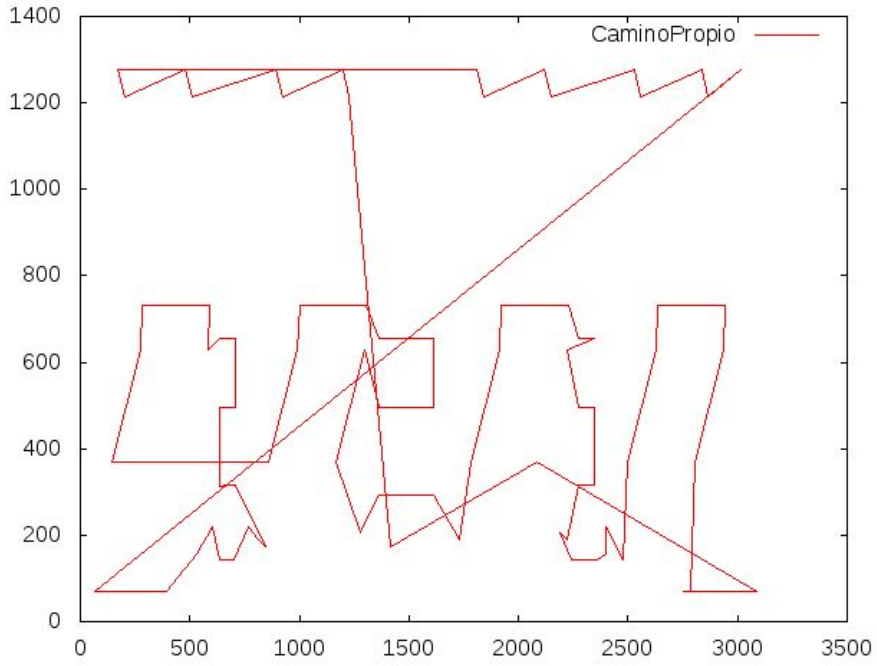
Circuito óptimo para el mapa ST70



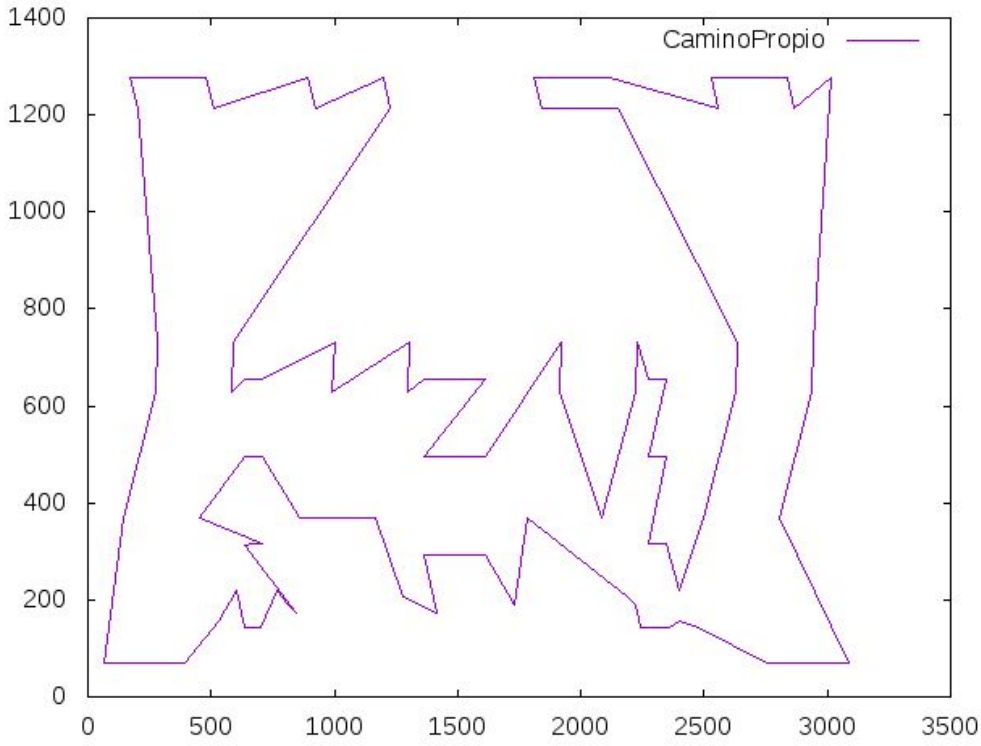
Circuito obtenido con la heurística del vecino más cercano doble para el mapa ST70



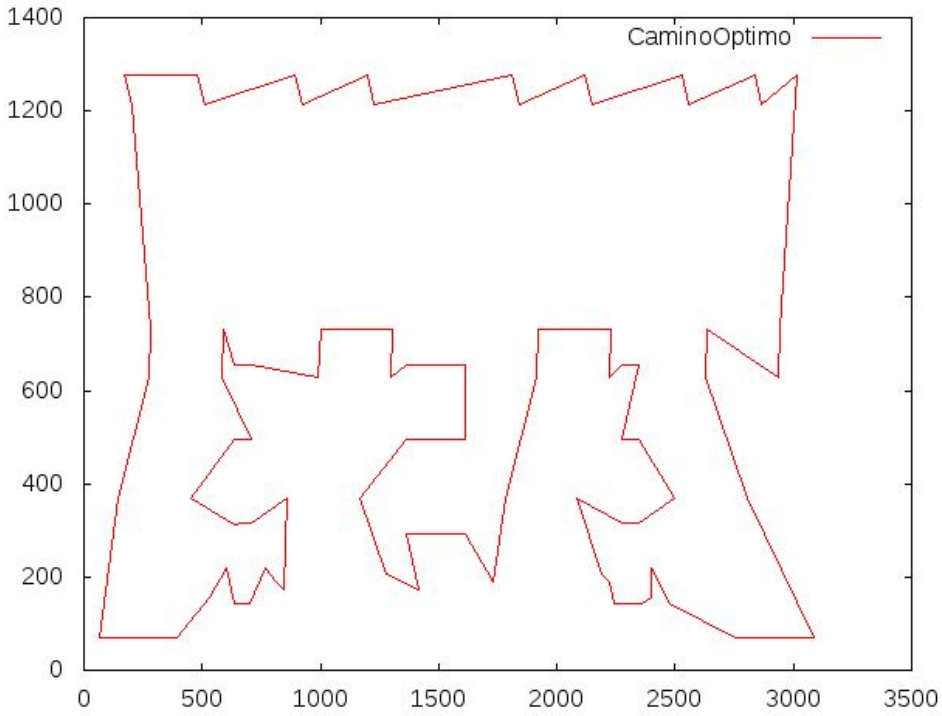
Circuito obtenido con la heurística del vecino más cercano para el mapa lin105



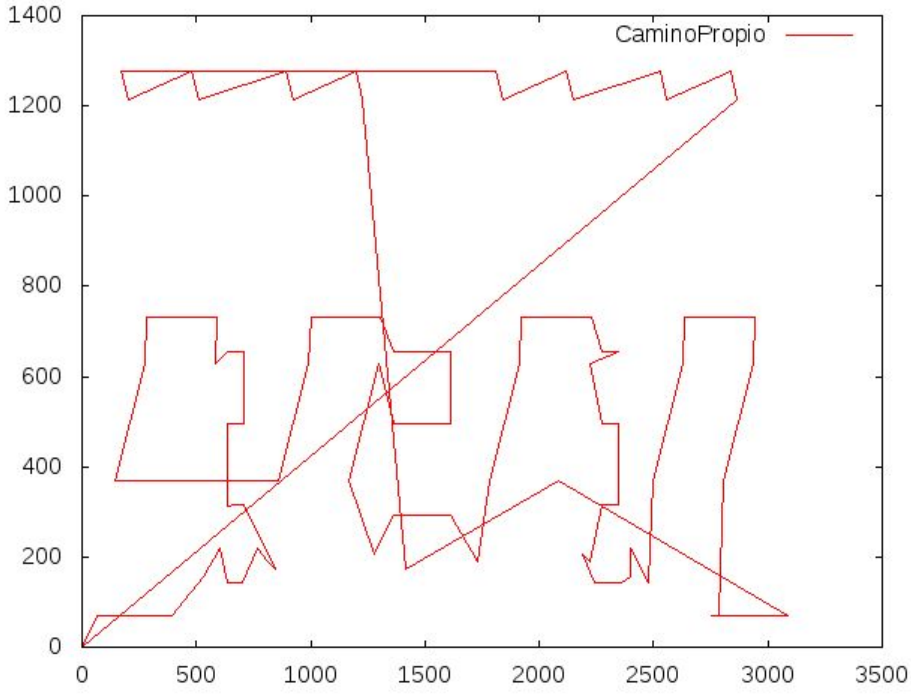
Circuito obtenido con la heurística de inserción de menor coste para el mapa lin105



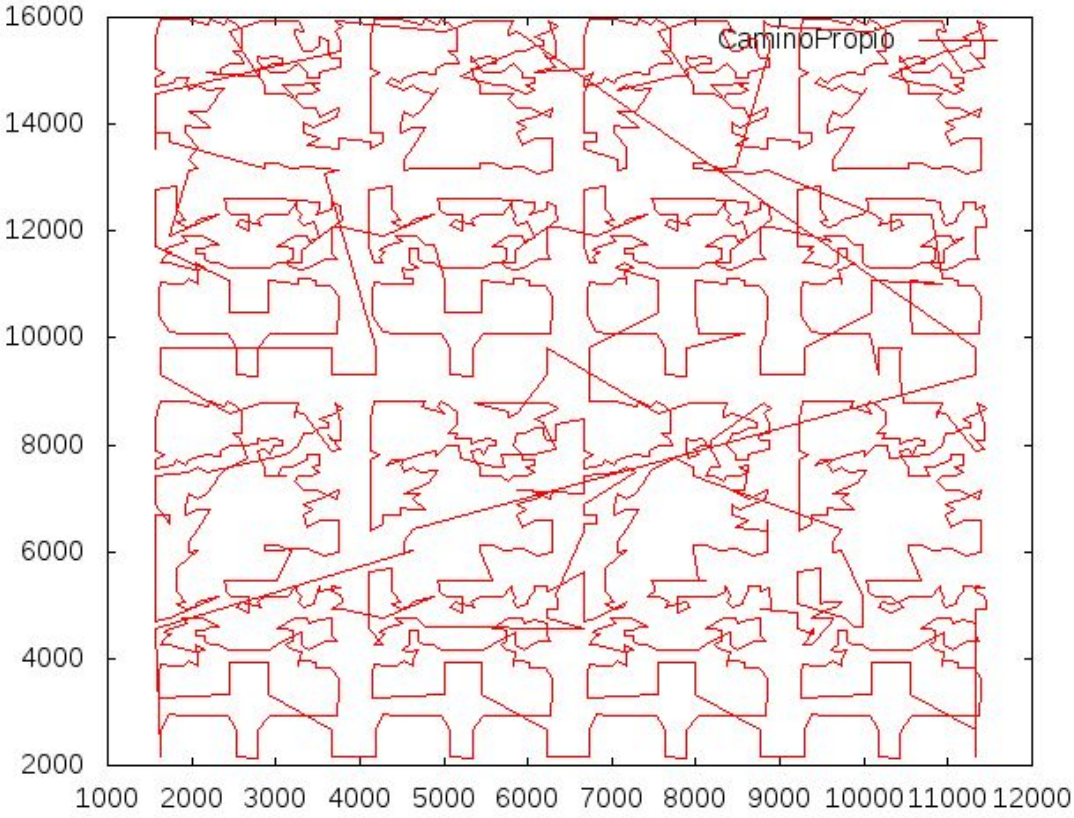
Circuito óptimo para el mapa lin105



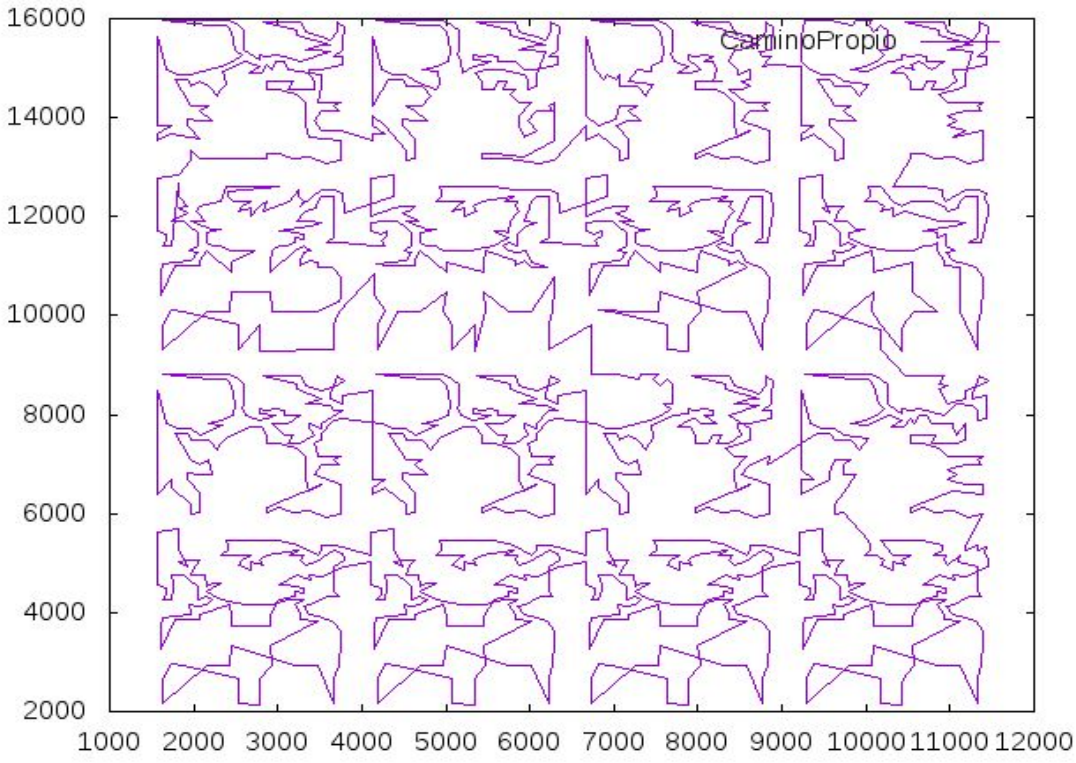
Circuito obtenido con la heurística del vecino más cercano doble para el mapa lin105



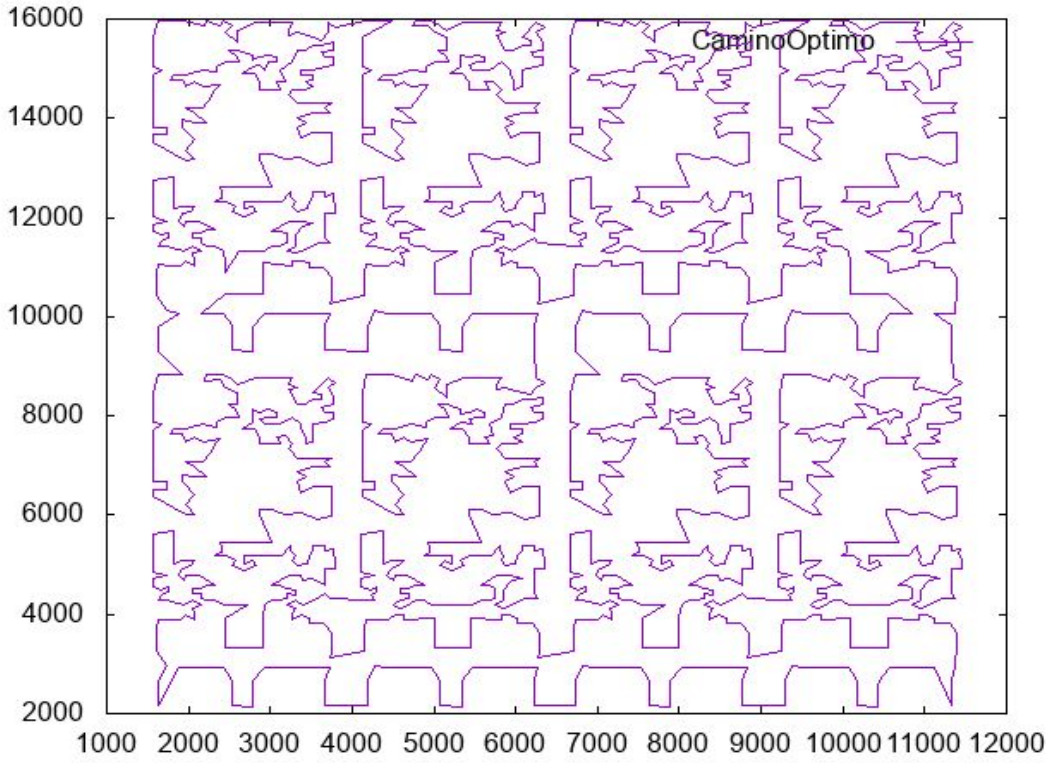
Circuito obtenido con la heurística del vecino más cercano para el mapa pr2392



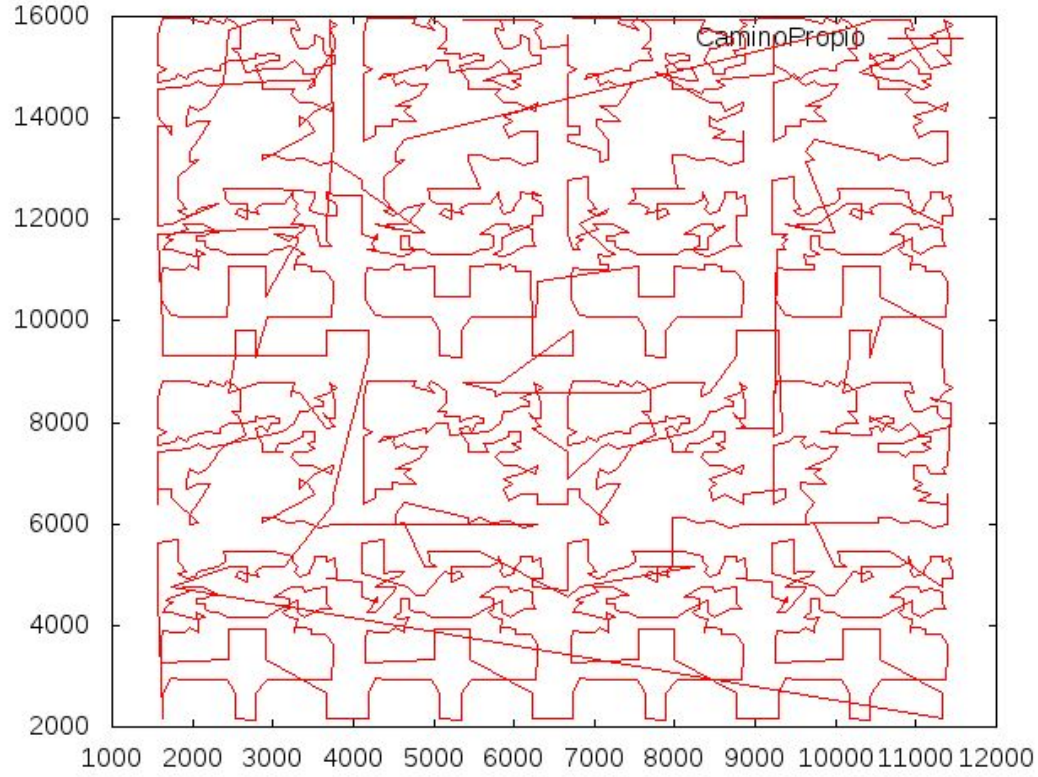
Circuito obtenido con la heurística de inserción de menor coste para el mapa pr2392



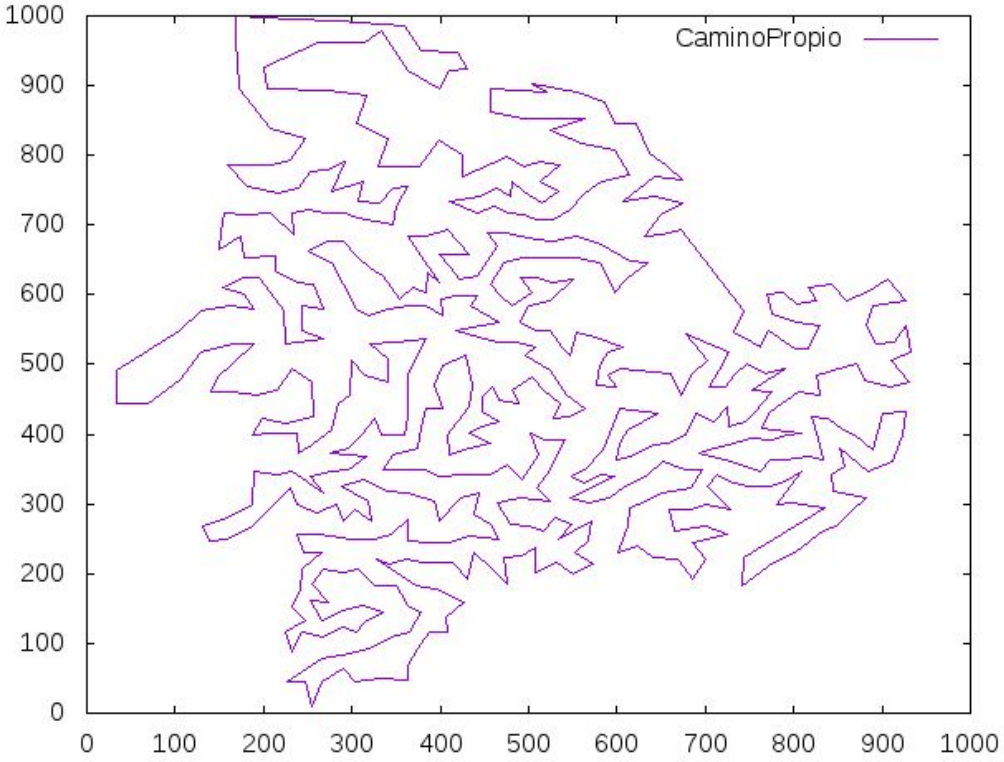
Circuito óptimo para el mapa pr2392



Circuito obtenido con la heurística del vecino más cercano doble para el mapa pr2392



Circuito obtenido con la heurística de inserción de menor coste para el mapa pa561



Circuito “óptimo” para el mapa pa561