Universidad de Granada

Algorítmica

Algoritmos Greedy, parte 2

Laura Calle Caraballo
Cristina María Garrido López
Germán González Almagro
Javier León Palomares
Antonio Manuel Milán Jiménez

2 de mayo de 2016

Índice

1. Introducción.

El objetivo de esta práctica es el estudio de los algoritmos de tipo *Greedy* aplicados particularmente al problema del viajante de comercio. Para ello, hemos implementado tres variantes según el criterio utilizado a la hora de construir soluciones y las hemos comparado según sus tiempos de ejecución y su grado de desviación respecto a las soluciones óptimas.

2. Descripción del problema.

El problema del viajante de comercio, también llamado *TSP* por sus siglas en inglés, es conocido desde el siglo XIX y fue planteado en su forma general en la década de 1930; desde entonces, es uno de los problemas más estudiados en optimización. Su complejidad hace computacionalmente muy costosa una solución mediante fuerza bruta. Por ello, se han ido creando una serie de métodos que obtienen resultados válidos según la situación: soluciones exactas para dimensiones pequeñas, aproximadas para dimensiones más grandes o particularizaciones del problema donde se pueda emplear una aproximación mejor.

La pregunta que busca responder este problema es la siguiente: *"Considerando un conjunto de ciudades y las distancias entre ellas dos a dos, ¿cuál es el camino más corto que pasa por todas ellas una única vez y retorna al origen?"*

En términos de grafos esto significa que, a partir de un grafo ponderado, conexo y no dirigido, debemos encontrar el circuito hamiltoniano de menor peso.

3. Resolución.

Se han diseñado tres algoritmos *Greedy* para encontrar soluciones al problema en diferentes instancias de prueba; la diferencia que existe entre ellos es la estrategia de elección de la siguiente ciudad a incluir en el camino (Función Selección): el vecino más cercano, la inserción más económica o la inserción radial.

Los elementos comunes a los tres algoritmos son los siguientes:

- Conjunto de Candidatos: ciudades de la instancia a evaluar.
- Conjunto de Seleccionados: ciudades ya incluidas en el circuito.
- Función Solución: todas las ciudades forman parte del circuito.
- Función de Factibilidad: se puede o no completar el circuito a partir de las ciudades ya elegidas.
- Función Objetivo: el circuito y su coste.

4. Estrategia del vecino más cercano ($O(n^3)$).

La estrategia del vecino más cercano es la forma más intuitiva de buscar el recorrido más corto: a partir de una ciudad, buscamos la más cercana que esté conectada a ella sin formar ya parte del circuito; a continuación, nos situamos sobre esta nueva ciudad y repetimos el proceso hasta que todas las ciudades hayan sido visitadas, construyendo un circuito hamiltoniano de forma secuencial.

Adicionalmente, comenzaremos el algoritmo desde todas las ciudades y nos quedaremos con el circuito más corto de todos los obtenidos.

La correspondiente Función Selección consiste en escoger la ciudad cuya distancia a la última añadida al circuito parcial sea mínima.

4.1. Pseudocódigo.

A continuación se muestra el pseudocódigo del algoritmo:

```
function Greedy(conjuntoCiudades);
costeMejorCircuito ⟵ ∞;
begin
    for ciudadInicial ∈ conjuntoCiudades do
        ciudadActual ⟵ ciudadInicial;
        circuito ⟵ ciudadInicial;
        costeCircuito ⟵ 0;
        for i = 2 to n do
            mejorVecino ⟵ EncontrarVecinoMasCercano(ciudadActual);
            circuito ⟵ circuito ∪ mejorVecino;
            costeCircuito ⟵ costeCircuito + costeMejorVecino;
            ciudadActual ⟵ mejorVecino;
        end
        cerrar circuito con ciudadInicial;
        actualizar costeCircuito;
        if costeCircuito < costeMejorCircuito then
            guardar circuito y actualizar costeMejorCircuito;
        end
    end
    return circuito, costeMejorCircuito;
end
```

4.2. Resultados.

Conjunto de datos	Óptimo	Resultado
att48	33524	37897
berlin52	7542	8181
eil51	426	482
eil76	538	608
eil101	629	746
gr96	512	621
gr120	1666	1836
lin105	14379	16935
pr76	108159	130921
st70	675	796

Figura 1: Resultados obtenidos por la heurística del vecino más cercano.

5. Estrategia de la inserción más económica (O(n^3)).

La idea subyacente a esta heurística es que en cada iteración se prueba cada ciudad candidata en todas las posibles posiciones donde se podría insertar. Por ello, la Función Selección consiste en escoger, de entre las candidatas en cada momento, la ciudad cuya inserción provoque el menor aumento del coste del circuito.

5.1. Pseudocódigo.

El pseudocódigo del algoritmo es el siguiente:

```
function Greedy(conjuntoCiudades);
circuito ⟵ SolucionParcial();
begin
    for i = 3 to n do
        for ciudad ∈ candidatos do
            posInsercion ⟵ EncontrarInsercionMasEconomica(ciudad);
            if Aumento(ciudad,posInsercion) <
            Aumento(mejorCiudad,posMejorInsercion) then
                posMejorInsercion ⟵ posInsercion;
                mejorCiudad ⟵ ciudad;
            end
        end
        Insertar(mejorCiudad, posMejorInsercion, circuito);
    end
    coste ⟵ CalcularCoste(circuito);
    return circuito, coste;
end
```

5.2. Resultados.

Conjunto de datos	Óptimo	Resultado
att48	33524	34682
berlin52	7542	8286
eil51	426	454
eil76	538	602
eil101	629	697
gr96	512	578
gr120	1666	1745
lin105	14379	16126
pr76	108159	115546
st70	675	741

Figura 2: Resultados obtenidos por la heurística de la inserción más económica.

6. Estrategia de inserción radial (O(n^2)).

En esta heurística de cosecha propia, la idea es fijar una ciudad situada aproximadamente en el centro de todas las demás y construir un circuito parcial. Posteriormente, se tomará como mejor candidata aquella ciudad que esté más lejos de la ciudad central, insertándola en la posición que provoque el menor incremento de la longitud del circuito.

La Función Selección asociada consiste, por tanto, en escoger en cada paso la ciudad más lejana al centro en la posición menos costosa.

6.1. Pseudocódigo.

El pseudocódigo asociado a este algoritmo se muestra a continuación:

```
function Greedy(conjuntoCiudades);
circuito ⟵ SolucionParcial();
begin
    ciudadCentral ⟵ CiudadMasCercanaAlCentro();
    for i = 3 to n do
        ciudadAInsertar ⟵ CiudadMasLejana(ciudadCentral);
        posInsercion ⟵ EncontrarInsercionMasEconomica(ciudadAInsertar);
        Insertar(ciudadAInsertar,posInsercion,circuito);
    end
    coste ⟵ CalcularCoste(circuito);
    return circuito,coste;
end
```

6.2. Resultados.

Conjunto de datos	Óptimo	Resultado
att48	33524	65884
berlin52	7542	12907
eil51	426	740
eil76	538	1083
eil101	629	1246
gr96	512	960
gr120	1666	3868
lin105	14379	32858
pr76	108159	250946
st70	675	1536

Figura 3: Resultados obtenidos por la heurística de inserción radial.

7. Comparación de resultados.

En este apartado se muestra con más claridad lo que ya se podía comprobar en secciones anteriores: la heurística que mejores resultados ha dado es la de la inserción más económica, seguida de la del vecino más cercano y de la de inserción radial.

Es destacable, aunque de esperar, la diferencia entre las dos primeras y la inserción radial, ya que esta última es de nuestra cosecha y compite frente a dos estrategias ya consolidadas.

Si nos centramos en las estrategias de vecino más cercano e inserción más económica, podemos observar que, en general, su rendimiento es muy bueno si consideramos su naturaleza *Greedy*; en particular, y como ya comentábamos, la inserción más económica consigue a veces unos resultados especialmente competitivos.

7.1. Tabla comparativa de la calidad de las soluciones.

Conjunto de datos	**Vecino más cercano**	**Inserción más económica**	**Inserción radial**
att48	13,044	3,454	96,528
berlin52	8,472	9,865	71,135
eil51	13,145	6,573	73,71
eil76	13,011	11,896	101,301
eil101	18,601	10,811	98,092
gr96	21,289	12,891	87,5
gr120	10,204	4,742	132,173
lin105	17,776	12,15	128,514
pr76	21,045	6,83	132,016
st70	17,926	9,78	127,56

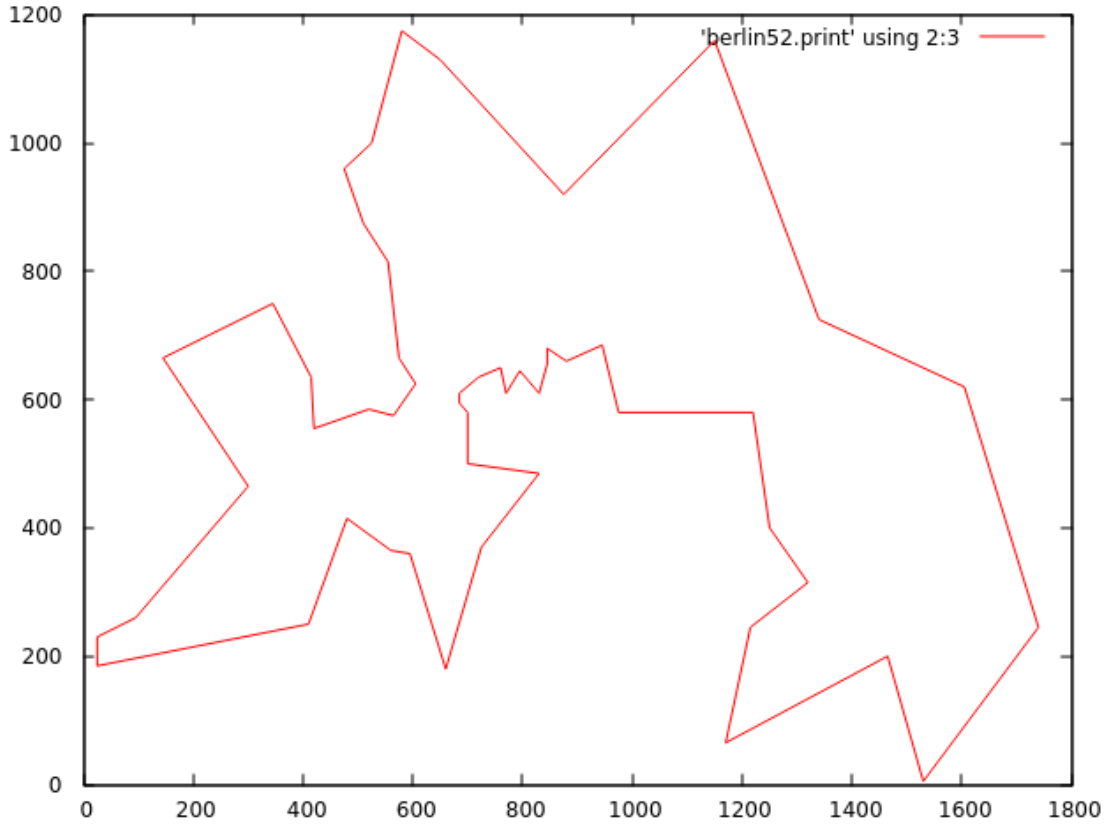
Figura 4: Desviaciones porcentuales respecto al óptimo en los caminos encontrados por los tres algoritmos.

7.2. Comparativa gráfica de soluciones.

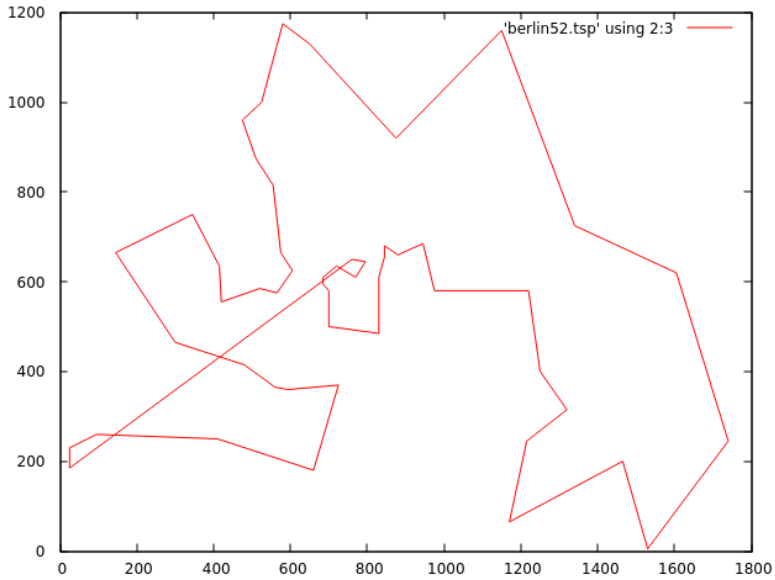
A continuación, mostraremos los recorridos óptimos de tres instancias del problema junto con las soluciones encontradas por las tres heurísticas implementadas.

7.2.1. Instancia *berlin52*.

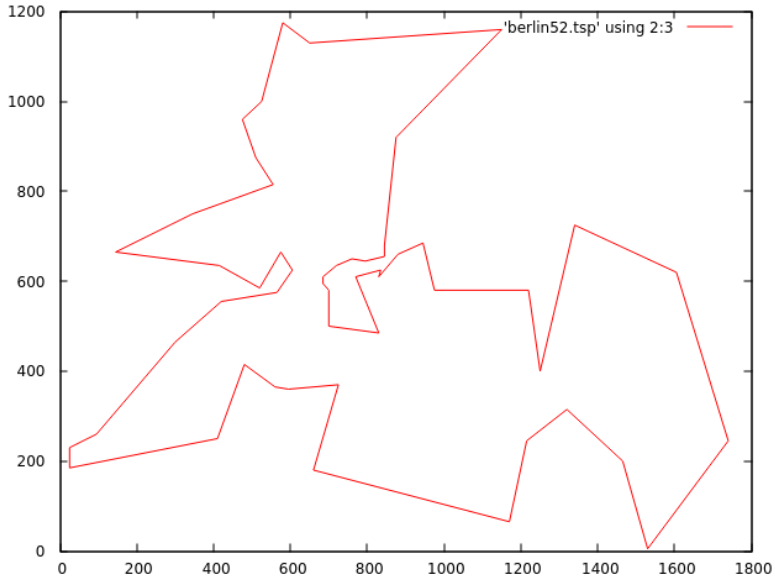
Solución óptima:



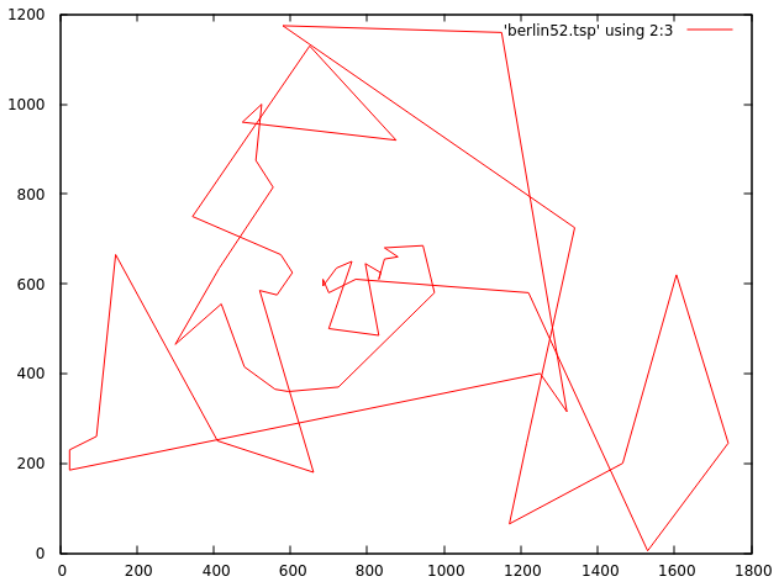
Heurística del vecino más cercano:



Heurística de la inserción más económica:

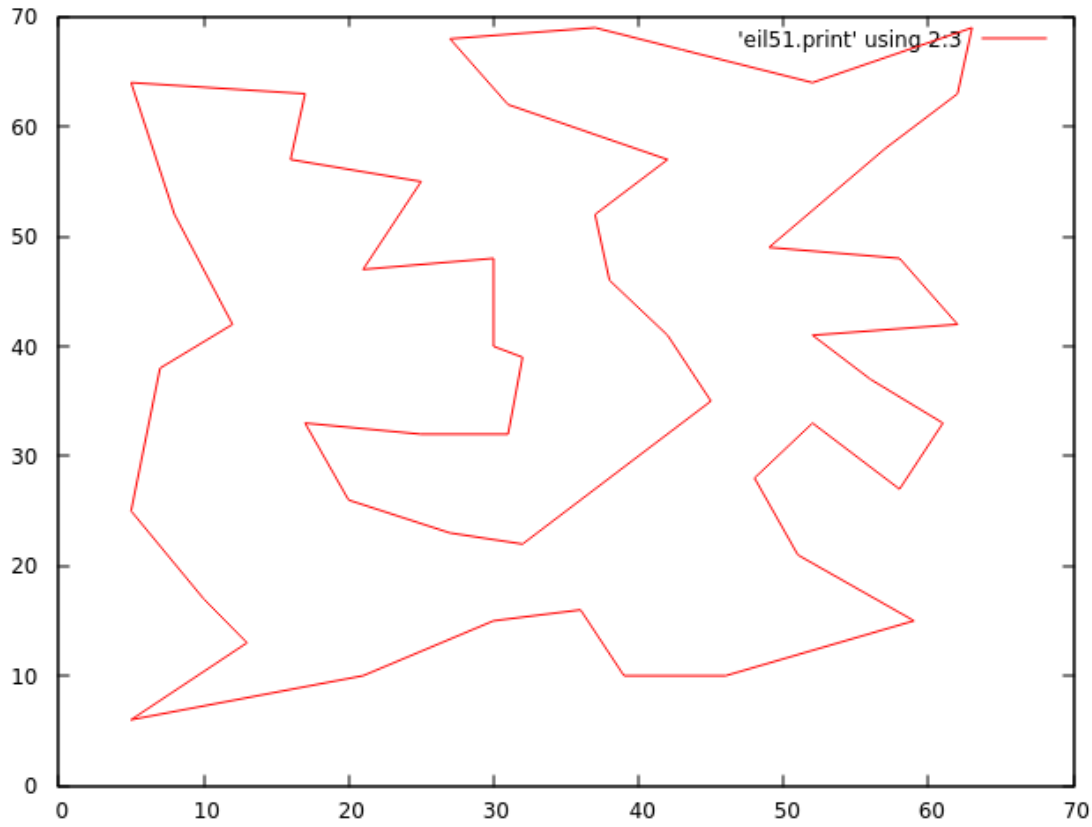


Heurística de la inserción radial:

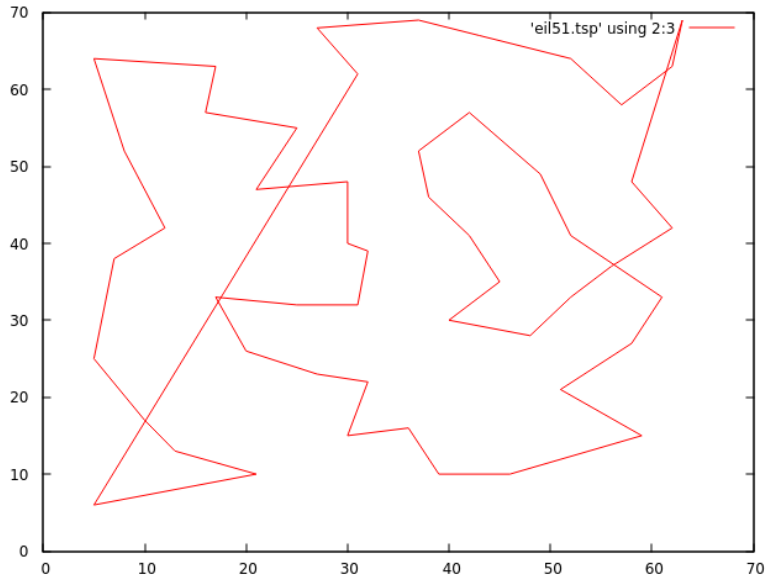


7.2.2. Instancia *eil51*.

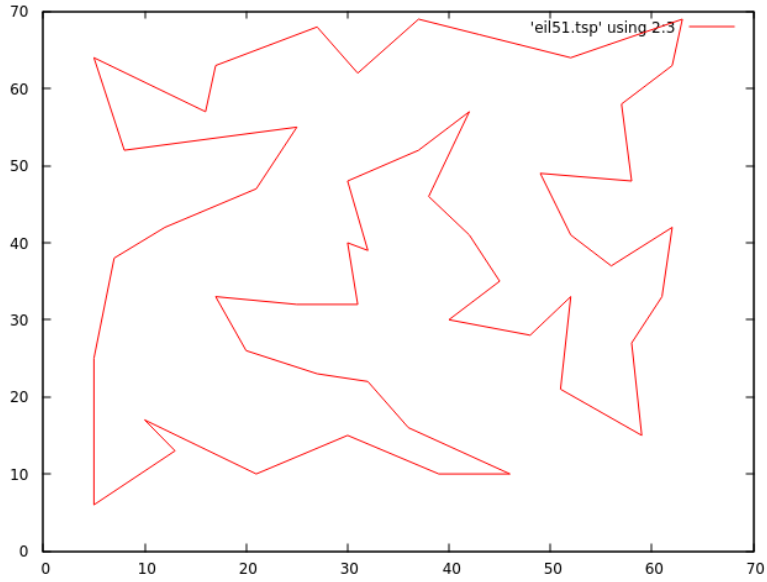
Solución óptima:



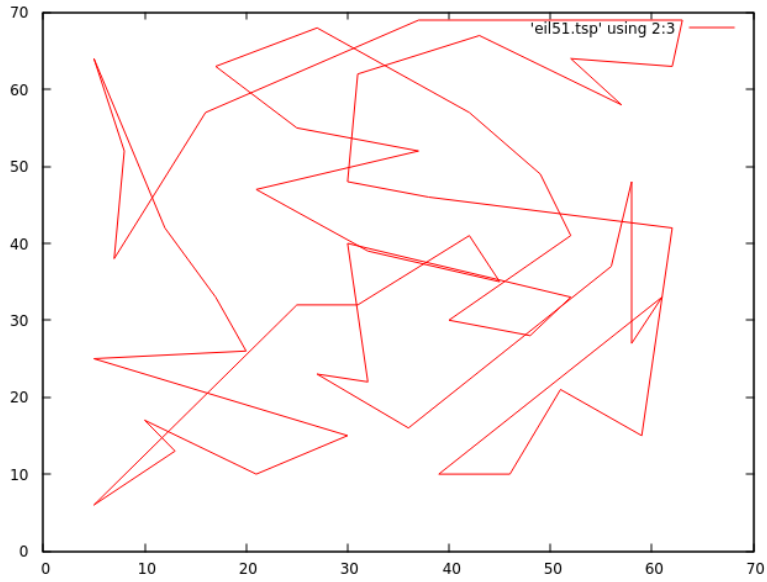
Heurística del vecino más cercano:



Heurística de la inserción más económica:

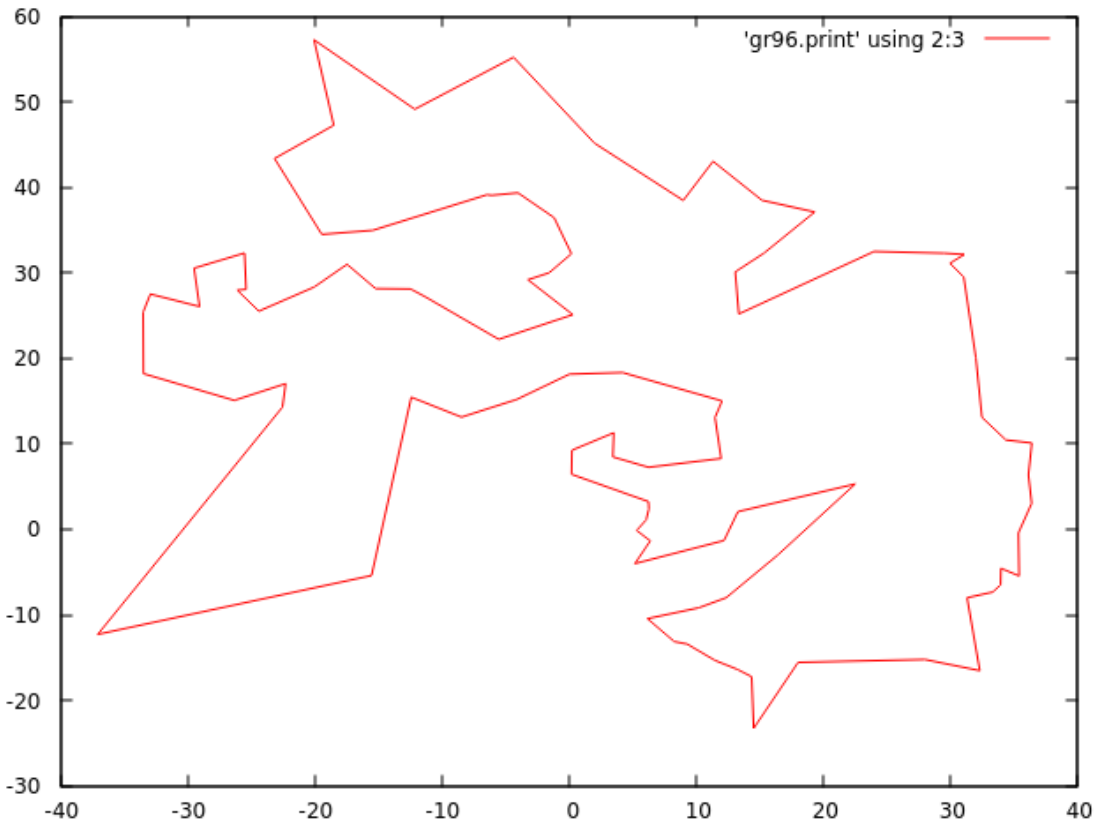


Heurística de la inserción radial:

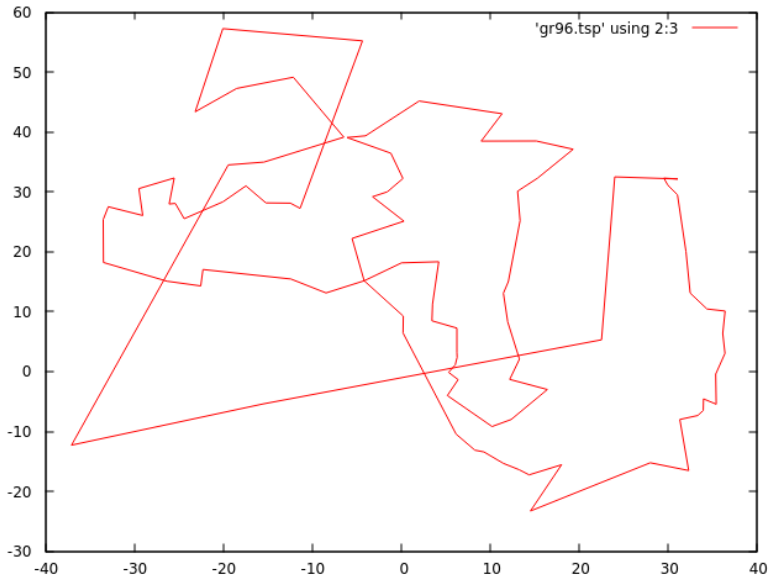


7.2.3. Instancia *gr96*.

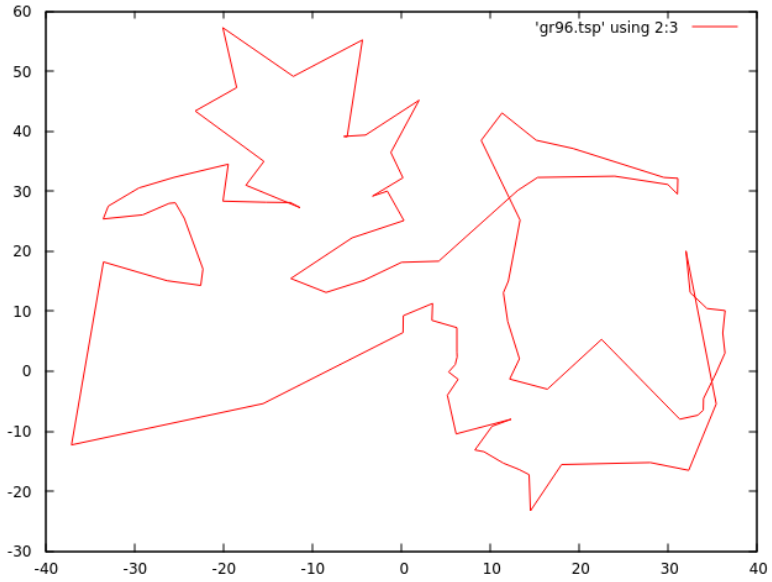
Solución óptima:



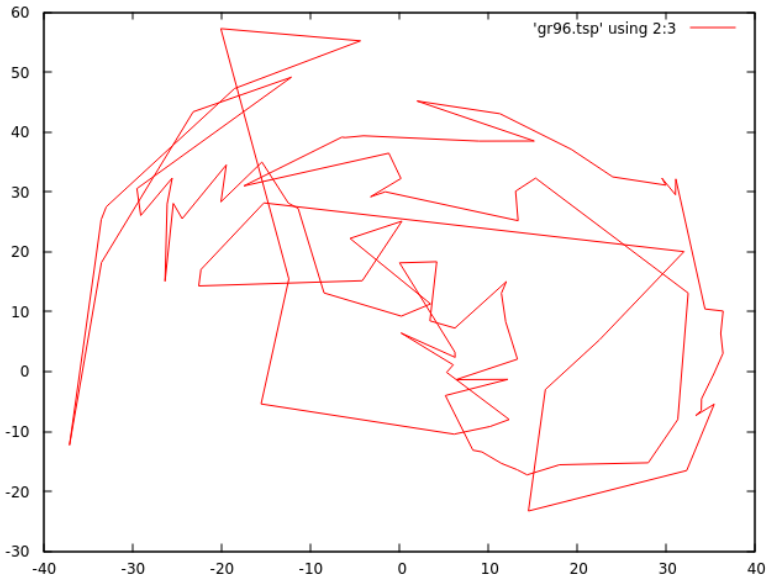
Heurística del vecino más cercano:



Heurística de la inserción más económica:



Heurística de la inserción radial:



8. Conclusión.

En primer lugar, lo más importante es que la naturaleza de este tipo de algoritmos permite obtener en muchos casos una solución aceptable para el problema del viajante de comercio en muy poco tiempo, pero no la óptima: debido a que es voraz, a veces ignorará en favor de la optimalidad local rutas que un ser humano podría identificar y que a la larga serían más cortas. Se ha sugerido que, si en las etapas finales del algoritmo las elecciones son considerablemente más costosas que al inicio, es bastante probable que haya circuitos mucho mejores.

En segundo lugar, debemos tener en cuenta que el método de selección de candidatos es crucial para que el algoritmo *Greedy* ofrezca todo su potencial. Como ha quedado patente a lo largo de este documento, la elección de una buena heurística es lo que marca la diferencia, y más si tenemos en cuenta que las soluciones obtenidas mediante estos algoritmos son frecuentemente empleadas como punto de partida para estrategias más sofisticadas.