Algoritmos Voraces

Problema del TSP

A. Herrera, A. Moya, I. Sevillano, J.L. Suarez

12 de mayo de 2015

Introducción

- En esta presentación se proporcionan varias soluciones para el problema del viajante de comercio.
- El código, los resultados de las ejecuciones, las gráficas y los pdf asociados se puede encontrar en GitHub.

Explicación del problema

Resumen del enunciado

- Implementar un programa que proporcione soluciones para el problema del viajante del comercio empleando las heurísticas del vecino más cercano y la mejor inserción, así como otra adicional propuesta por el propio equipo. El programa debe proporcionar el recorrido obtenido y la longitud de dicho recorrido.
- Realizar un estudio comparativo de las tres estrategias empleando un conjunto de datos de prueba.

Vecino Más Cercano

Características. Identificación Greedy.

- **Conjunto de candidatos.** Las ciudades del problema.
- **Candidatos usados.** Ciudades ya visitadas. La solución parcial.
- **La función solución.** Nos basta con comprobar si todas las ciudades están visitadas.
- **La función de selección.** Para cada ciudad C, se devuelve aquella ciudad S del conjunto de candidatos no usados E, tal que $dist(S, C) = \min\{dist(P, C) : P \in E\}$. Es decir, la ciudad más cercana a C.
- **La función objetivo.** La distancia total del recorrido, que es la que tratamos de optimizar.

Vecino Más Cercano

Pseudocódigo.

```
def TSPVecinoMasCercano
  solucion = []

  # Comenzamos añadiendo una ciudad inicial sobre
  # la que se empiezan a buscar los vecinos
  # (puede ser cualquiera)
  solucion.push(conjuntoCiudades[0])

  candidatos = conjuntoCiudades
  candidatos.erase(cojuntoCiudades[0])
```

Vecino Más Cercano

Pseudocódigo.

```
                        # (función solución)
  while solucion.numeroCiudades < totalCiudades

    # Vamos buscando el vecino más cercano a la
    # última ciudad insertada, con nuestra
    # función de selección previamente indicada.
    C = candidatos.vecinoMasCercano(solucion.get(
      solucion.numCiudades-1))
    solucion.push(C)
    candidatos.erase(C)
  end
  return solucion
end
```

Mejor Inserción

Características. Identificación greedy.

- **Conjunto de candidatos.** Las ciudades del problema.
- **Candidatos usados.** Ciudades ya visitadas. La solución parcial.
- **La función solución.** Nos basta con comprobar si todas las ciudades están visitadas.
- **La función objetivo.** La distancia total del recorrido, que es la que tratamos de optimizar.
- En general, estas características coinciden para cualquier heurística sobre el TSP

Mejor Inserción

Características. Identificación greedy.

- **La función de selección.** Dada una solución parcial $\{S_i\}_{,i=1,...,n}$, obtiene $S_{n+1} := C \in E = \{$ candidatos no usados $\}$, y una permutación que reinserta S_{n+1}:

$$\pi_j : \{1, ..., n+1\} \rightarrow \{1, ..., n+1\}$$

$$\pi_j(k) = k \; \forall k < j, \pi_j(n+1) = j, \text{ y } \pi_j(k) = k+1 \; \forall j < k < n+1)$$

De forma que $\{S_{\pi_j(i)}\}_{i=1,...,n+1}$ minimiza la distancia de la solución parcial.

Construimos de forma inductiva, insertando ciudades en la posición que minimiza el recorrido.

Mejor Inserción

A. Herrera, A. Moya, I. Sevillano, J.L. Suarez

Pseudocódigo

```
def TSPMejorInsercion
  # Comenzamos añadiendo una ciudad inicial sobre
  # la que se buscarán las mejores inserciones.
  solucion = []
  solucion.push(conjuntoCiudades[i])

  candidatos = conjuntoCiudades
  candidatos.erase(cojuntoCiudades[i])
```

Mejor Inserción

Pseudocódigo

```
                              # (función solución)
  while solucion.numeroCiudades < totalCiudades

    # Buscamos la ciudad y la posición en la que
    # insertarla según la heurística de la mejor
    # inserción. Esta es nuestra función de
    # selección.
    [C,posicion]=candidatos.mejorInsercion(solucion)
    solucion.insert(C,posicion)
    candidatos.erase(C)
  end
  return solucion
end
```

Optimización de la Colonia de Hormigas

A. Herrera, A. Moya, I. Sevillano, J.L. Suarez

Introducción

- Algoritmo bioinspirado: se basa en la naturaleza, imitando su comportamiento para encontrar la solución a un problema.
- Las hormigas salen en grupo a buscar alimentos. Acaban formando caminos nítidos entre origen y destino.
- Además, son capaces de transformar con el tiempo los caminos en caminos mínimos.
- Las hormigas conocen secretos del TSP desde el principio de los tiempos. ¿Cómo?

Optimización de la Colonia de Hormigas

A. Herrera, A. Moya, I. Sevillano, J.L. Suarez

Explicación

Conceptos clave

- Las hormigas con ciegas.
- Se guían por el olfato, a través de feromonas.
- Cuando se desplazan, las hormigas dejan un rastro de feromonas, que es seguido por otras hormigas y por ella misma para volver.
- Las feromonas se evaporan con el tiempo.
- Hay hormigas despistadas: se olvidan de seguir feromonas.

Optimización de la Colonia de Hormigas

A. Herrera, A. Moya, I. Sevillano, J.L. Suarez

Hormigas rápidas.

- La hormiga ha encontrado una ruta corta.
- Irá y volverá en poco tiempo: soltará más feromonas.
- Habrá más hormigas que sigan ese camino.

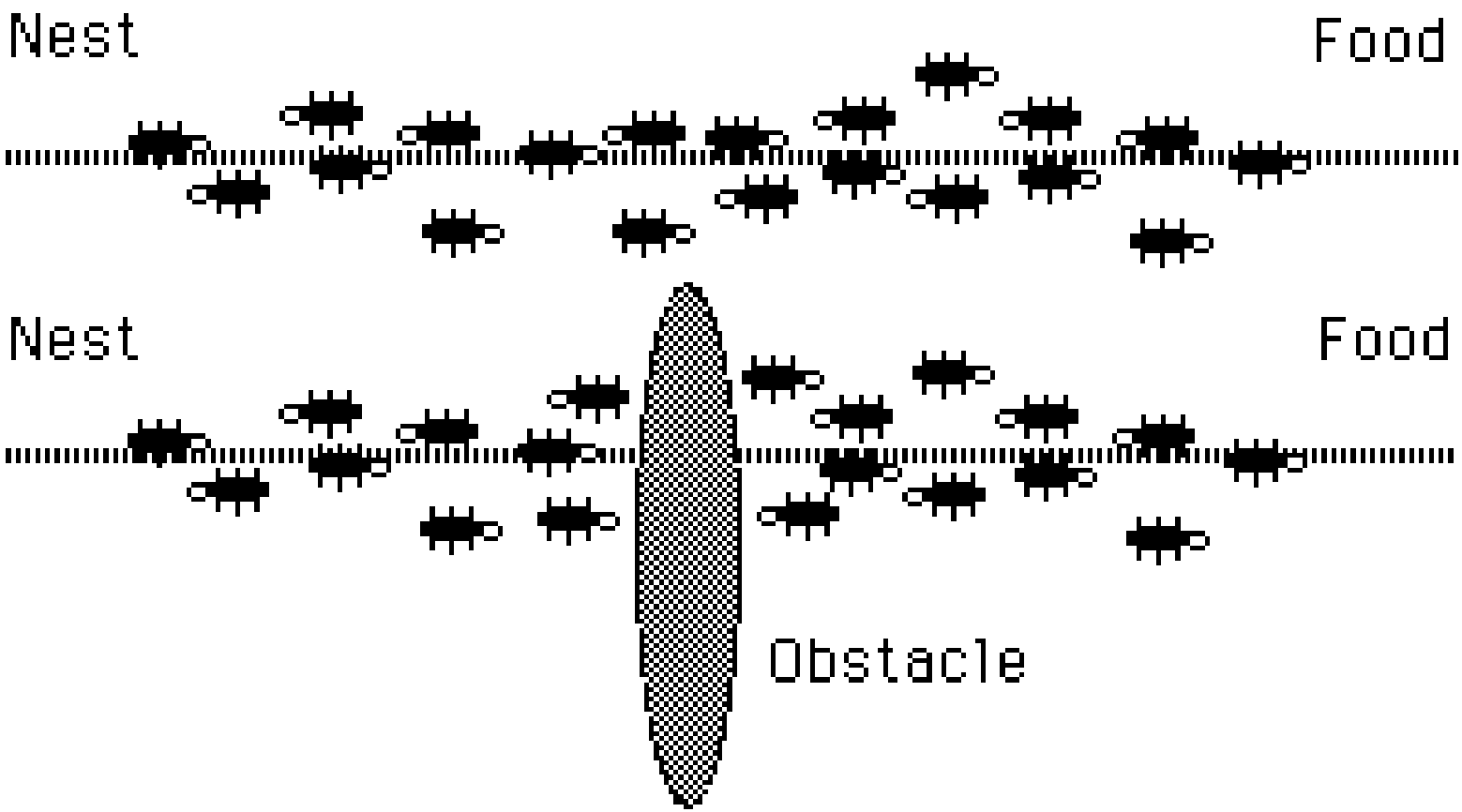
Hormigas lentas.

- La hormiga ha encontrado una ruta larga.
- Tardará más tiempo en ir y volver: se evaporarán más feromonas.
- Cada vez menos hormigas seguirán el camino. Acabará desapareciendo.

Optimización de la Colonia de Hormigas

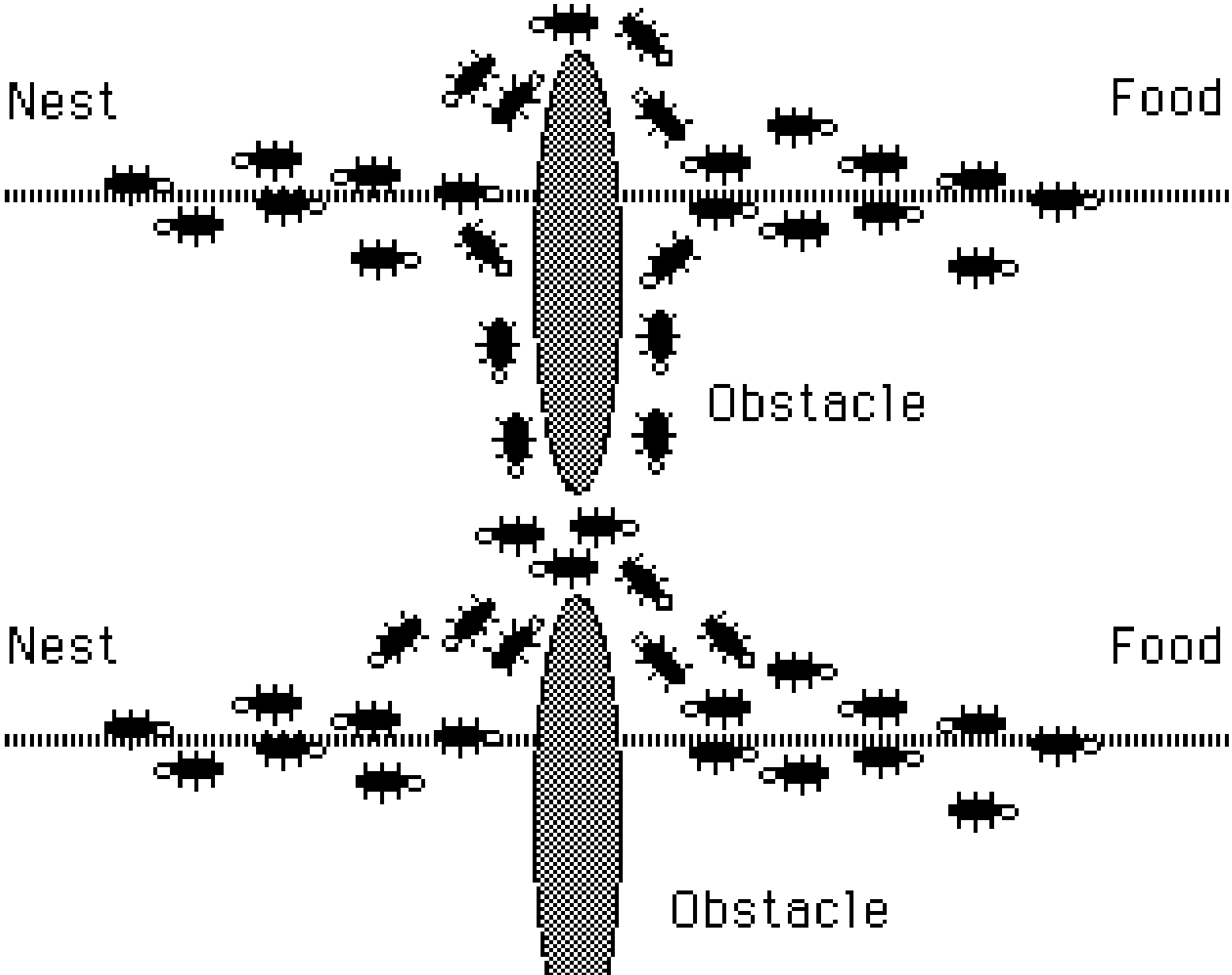
- Cada vez que una hormiga encuentra un camino, será más o menos seguido según su calidad.
- Las hormigas siempre acaban encontrando un buen camino.
- Las hormigas nos proporcionan una nueva heurística que trataremos de desarrollar.

Optimización de la Colonia de Hormigas



Optimización de la Colonia de Hormigas

A. Herrera, A. Moya, I. Sevillano, J.L. Suarez



Optimización de la Colonia de Hormigas

Características. Identificación greedy

- **Conjunto de candidatos.** Las ciudades del problema.
- **Candidatos usados.** Ciudades ya visitadas. La solución parcial.
- **La función solución.** Nos basta con comprobar si todas las ciudades están visitadas.
- **La función de selección.** Basada en leyes probabilísticas proporcionales a las feromonas. No es una función propiamente dicha. Tiene cierto grado de aleatoriedad. Evoluciona en el tiempo junto con la distribución de feromonas.
- **La función objetivo.** La distancia total del recorrido, que es la que tratamos de optimizar.

Optimización de la Colonia de Hormigas

Detalles de implementación.

- Mapa de feromonas. Grafo/matriz con la misma estructura que el mapa de ciudades. En lugar de distancias, almacena cantidad de feromonas.
- Hormigas. Agentes encargados de almacenar y recuperar soluciones, moverse según las probabilidades que determinan las feromonas y soltar feromonas mmientras avanzan.
- Algoritmo iterativo. Las hormigas necesitan muchos viajes para alcanzar buenas soluciones. Cuantas más iteraciones, mejor solución.

Optimización de la Colonia de Hormigas

Pseudocódigo

```
def TSPAntsColonyOptimization(numIteraciones)
  solucion = []

  while i < numIteraciones

    for hormiga in coloniaHormigas
      hormiga.iniciaRuta(rand)
    end
```

Optimización de la Colonia de Hormigas

Pseudocódigo

```
# Hacemos que las hormigas hagan un camino
# completo para obtener así una solución.
for i in 1..numeroCiudades
  for hormiga in coloniaHormigas
    # Función de selección probabilística.
    C = hormiga.determinarSiguienteCiudad()
    # La hormiga avanza en su recorrido y
    # añade feromonas.
    hormiga.avanza(C)
  end
end
```

Optimización de la Colonia de Hormigas

Pseudocódigo

```
    # Una vez las hormigas han obtenido la solución,
    # nos vamos quedando con la mejor.
    for hormiga in coloniaHormigas
      solucionHormiga = hormiga.getSolucion
      if(solucion == [] or
        solucionHormiga.coste() < solucion.coste())
          solucion = solucionHormiga
      end
    end
    # Repetimos el proceso el número de iteraciones
    # que se pide en el argumento. Las feromonas se
    # irán modificando y dando lugar cada vez a
    # soluciones mejores.
  end
```

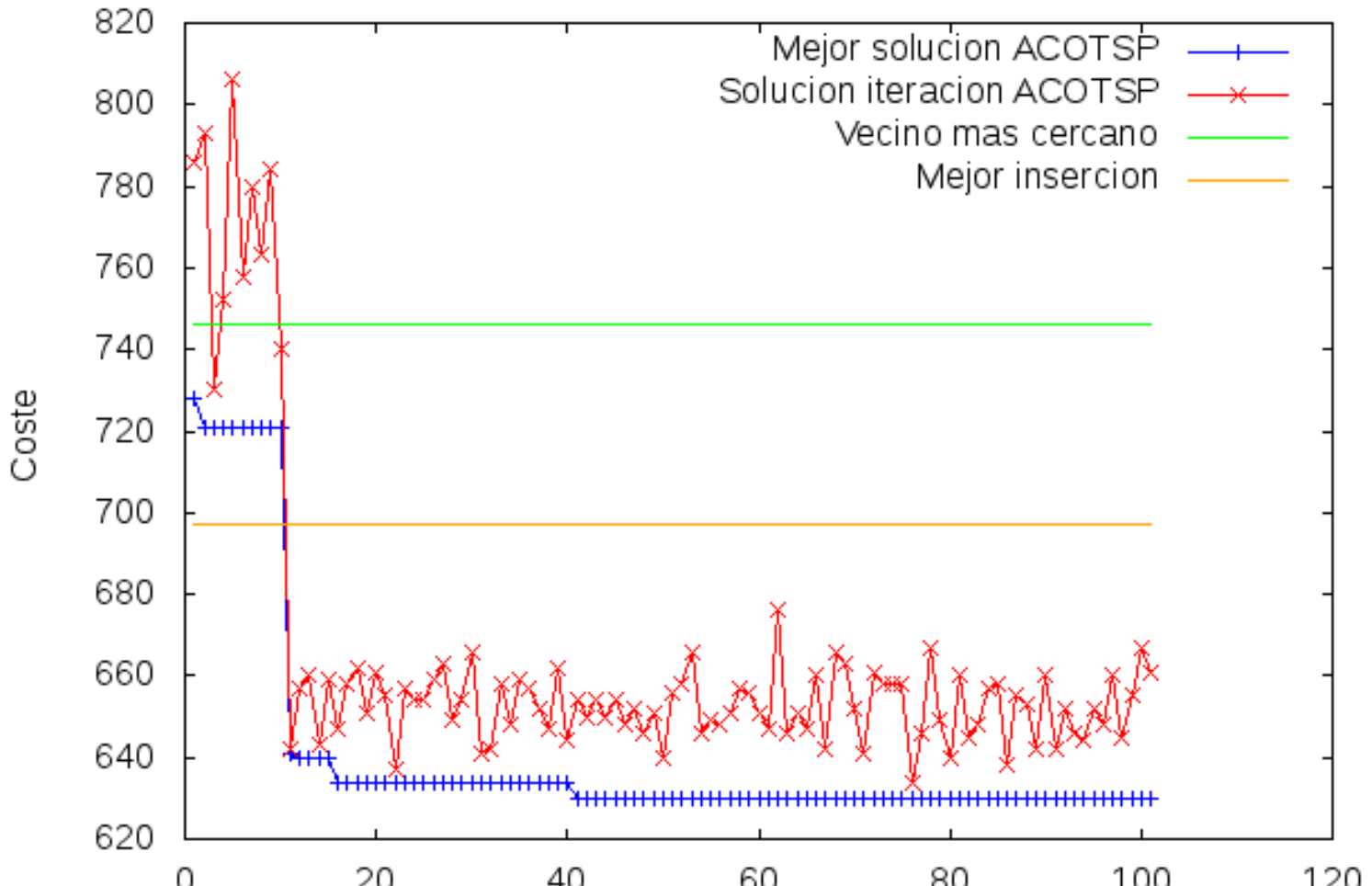
A. Herrera, A. Moya, I. Sevillano, J.L. Suarez

Eficiencia y mejoras

- Menos eficiente que Vecino Más Cercano y Mejor Inserción: muchas iteraciones vs construcción de una única solución.
- No se limita a una única solución. La solución mejora en el tiempo gracias al carácter aleatorio.
- Su eficacia mejora drásticamente al combinar el algoritmo con la búsqueda local: mejorar las soluciones en vecindades relativamente pequeñas.
- Al aplicar las feromonas sobre los recorridos mejorados las soluciones convergen a óptimos más rápidamente.

A. Herrera, A. Moya, I. Sevillano, J.L. Suarez

Mejora del algoritmo por iteraciones



Optimización de la Colonia de Hormigas

Detalles técnicos.

- Medida de las feromonas: deben premiar las rutas más cortas. Valor inversamente proporcional a las distancias.
- Inicialización de las feromonas: todas igual. Siempre cantidades mayores que cero. Es necesario un umbral inferior para evitar errores de división por cero.
- Actualización de las feromonas: $f_{i,j} = f_{i,j} + \frac{1}{dist(i,j)}$
- Evaporación de las feromonas: $f_{i,j} = f_{i,j}(1-\rho), \forall i \neq j$, ρ es el parámetro de evaporación. Nunca se sobrepasa el umbral. Se puede omitir en la práctica.

A. Herrera, A. Moya, I. Sevillano, J.L. Suarez

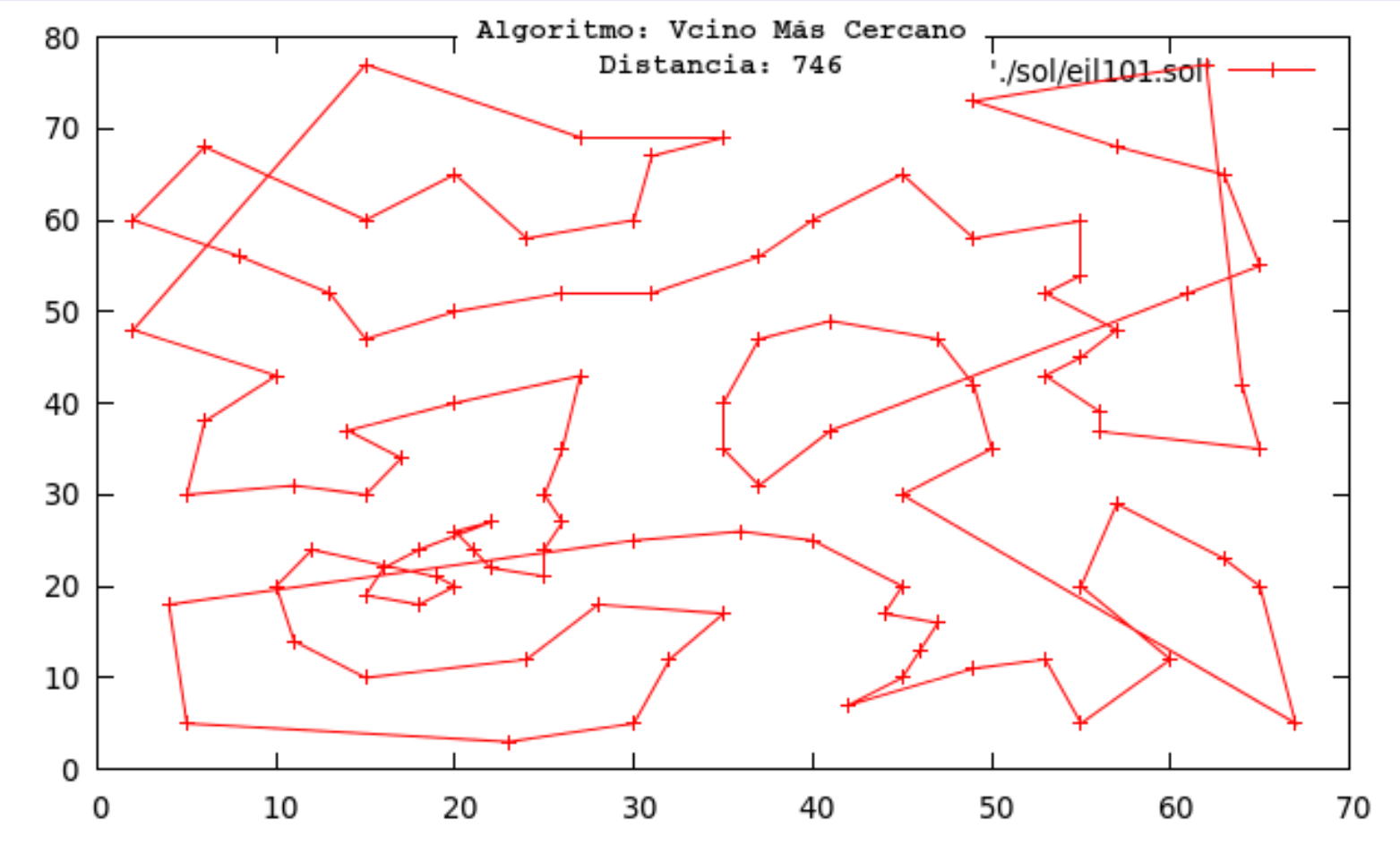
Detalles técnicos.

- Cálculo de la probabilidad: $\alpha :=$ influencia de feromonas, $\beta :=$ influencia de ciudades, $C :=$ conjunto de ciudades, $i, j \in C, i \neq j$

$$p_{i,j} = \frac{f_{i,j}^{\alpha} + (\frac{1}{d_{i,j}})^{\beta}}{\sum_{c \in C - \{i\}} f_{i,c}^{\alpha} + (\frac{1}{d_{i,c}})^{\beta}}$$

Comparación de las heurísticas

Vecino Más Cercano



Comparación de las heurísticas

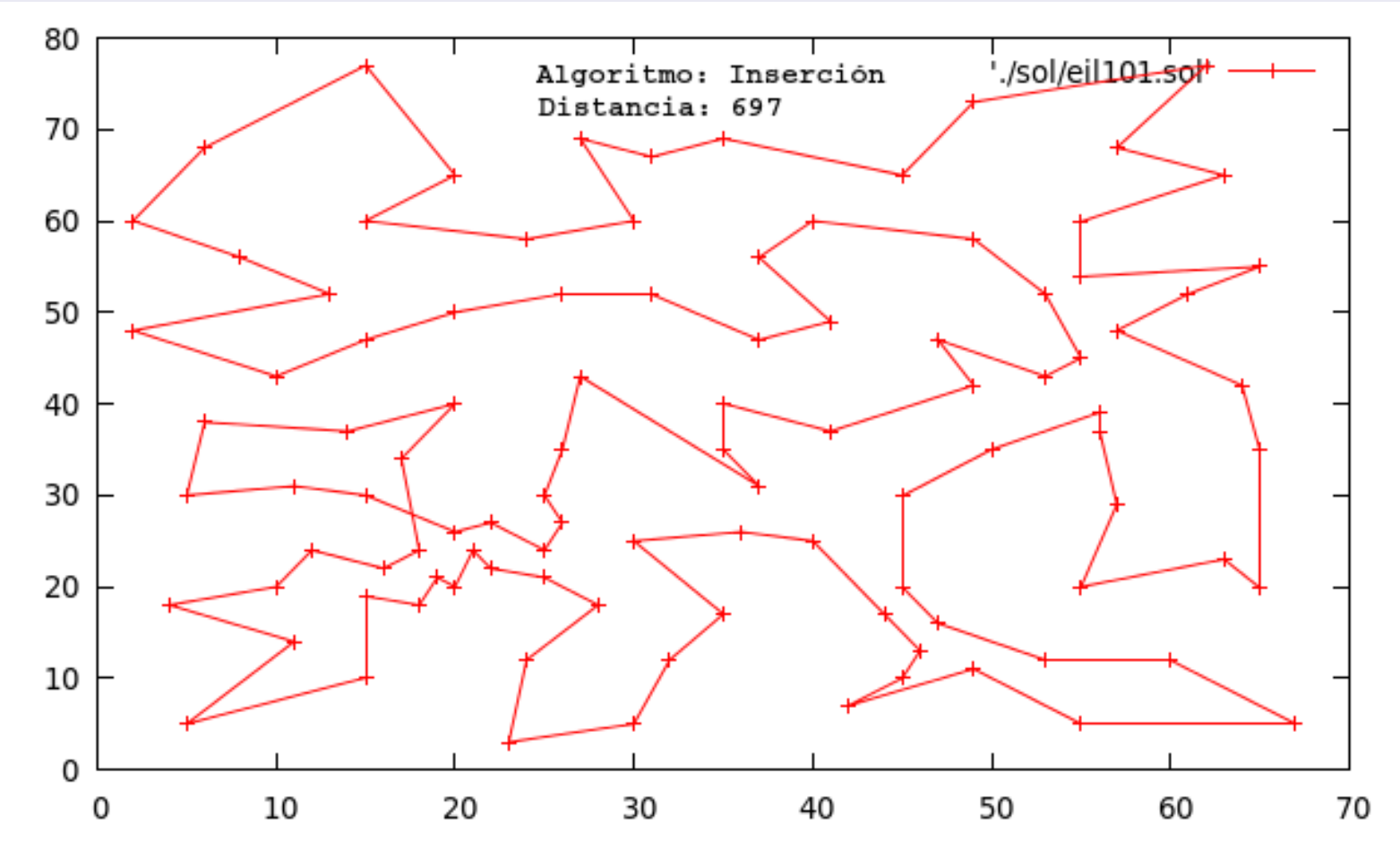
Vecino Más Cercano

- Solución aceptable. El camino es abordable.
- Problema: los últimos candidatos a vecino pueden originar arcos demasiado largos.

Comparación de las heurísticas

A. Herrera, A. Moya, I. Sevillano, J.L. Suarez

Mejor Inserción



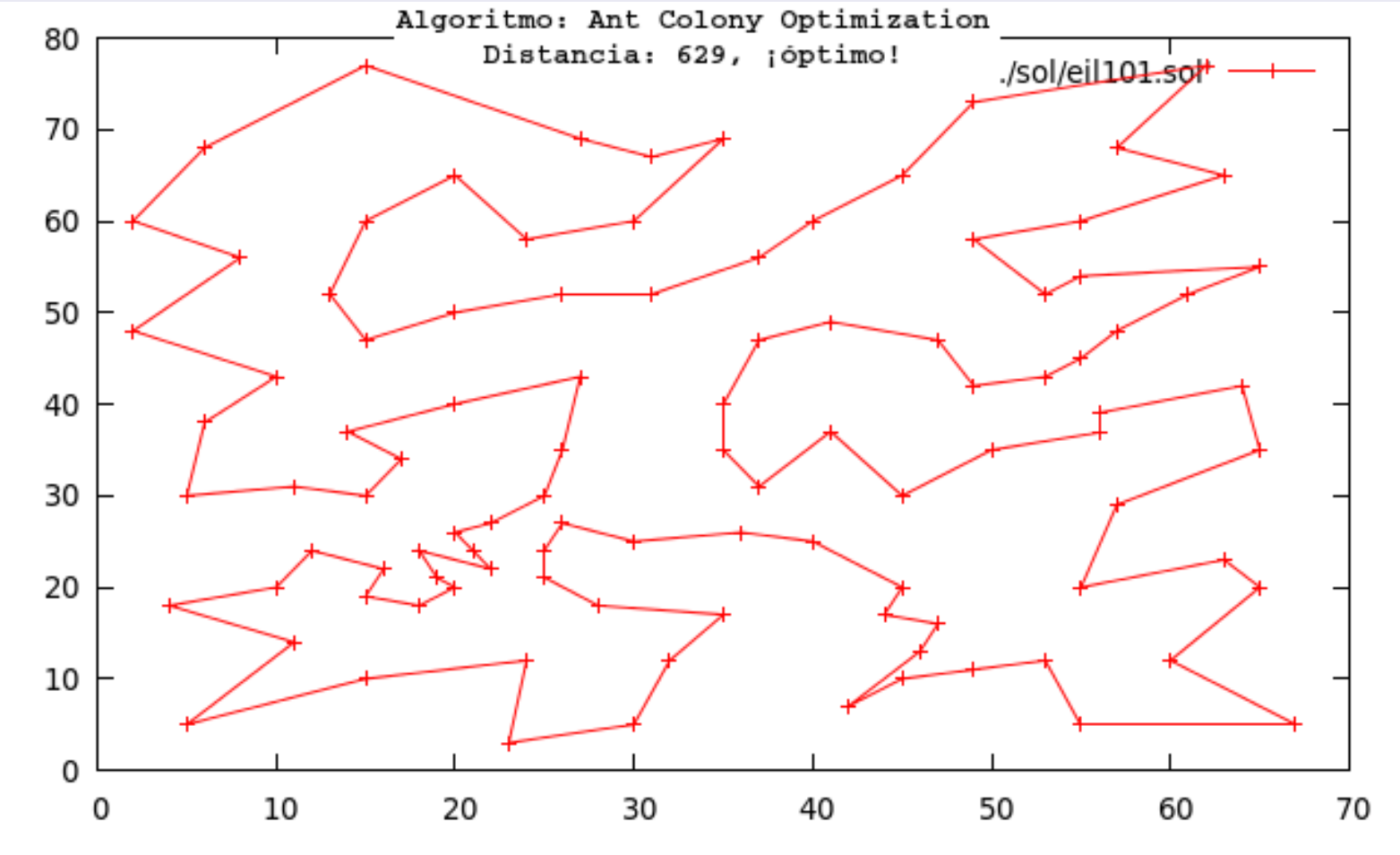
Comparación de las heurísticas

Mejor Inserción

- Buena solución. Ya no se aprecian arcos demasiado largos.
- Inconveniente: la secuencia de ciudades marca la tendencia de la solución.

Comparación de las heurísticas

Optimización de la Colonia de Hormigas



Comparación de las heurísticas

Optimización de la Colonia de Hormigas

- **En este caso, ¡las hormigas han encontrado el óptimo!**

Fin de la presentación

¡Gracias por su atención!