Memoria de la práctica 1

Jaime Lorenzo Sánchez

Francisco Cruceira Lloret

Alejandro Del Moral Rodríguez

Francisco Julián Reyes Urbano

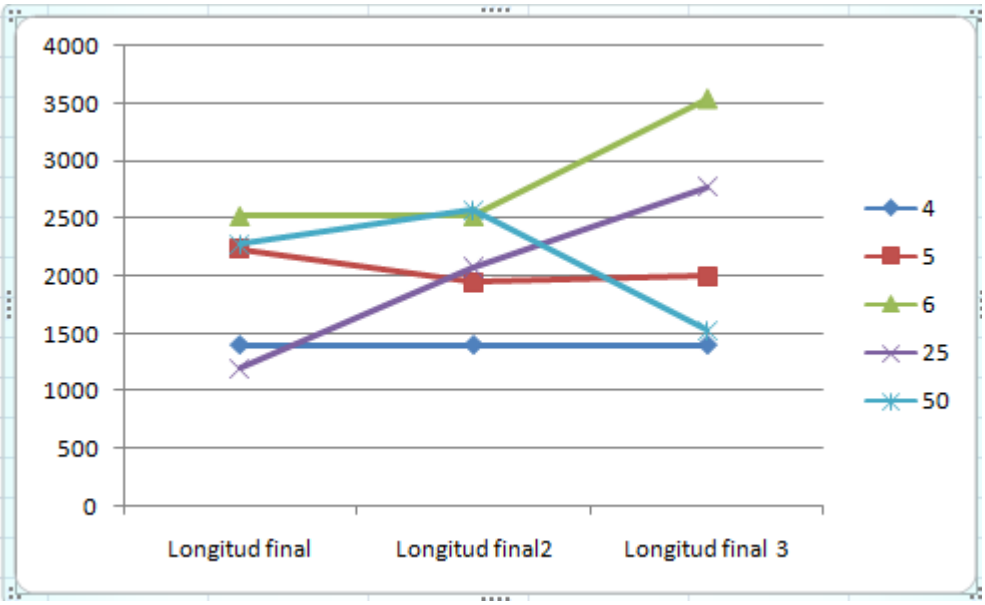
11 de febrero de 2022

Índice general

Capítulo 1

Hill Climbing

1.1. ¿Cómo se comporta este algoritmo a medida que aumentamos el problema (número de ciudades en el TSP)?



Si observamos el gráfico anterior, comprobamos que a medida que aumentamos el número de ciudades la longitud final del algoritmo es cada vez más variable.

También podemos comprobar que, conforme aumentamos el número de ciudades, también aumenta la longitud final de la solución. Esto provoca que sea más complicado alcanzar el

posible máximo global, debido a que el algoritmo localiza un mayor número de máximos locales.

Por tanto, este algoritmo es recomendable para problemas con un número de ciudades bajo.

1.2. ¿Obtiene siempre la mejor solución? ¿Por qué? ¿De qué depende?

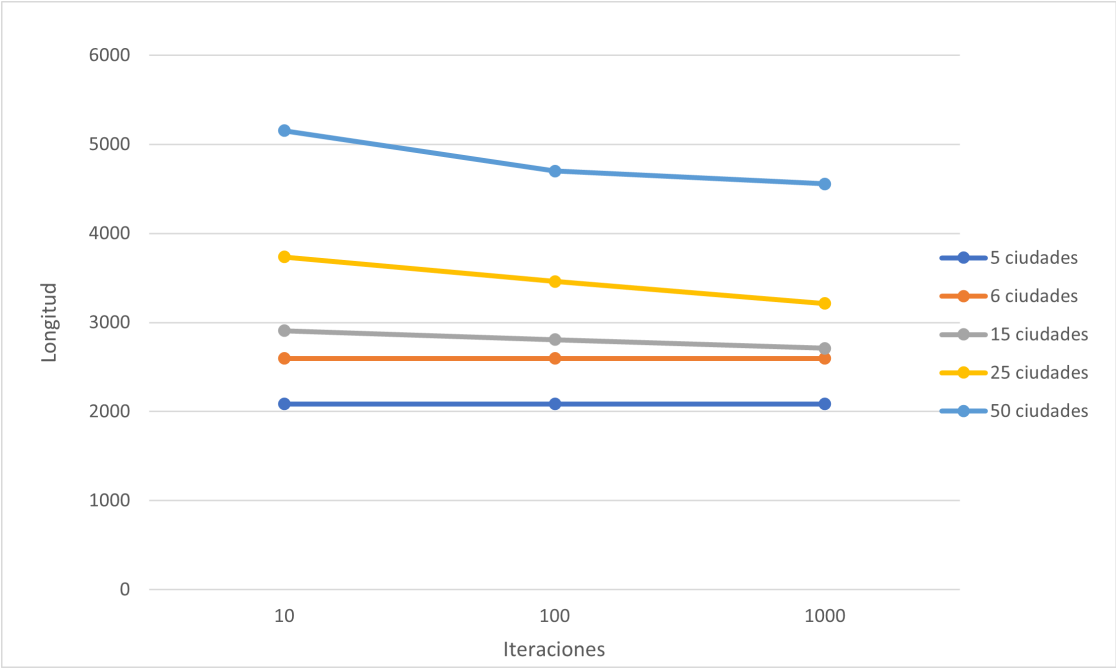
No siempre obtiene la mejor solución, porque el proceso termina cuando en un estado dado ninguno de los estados resultantes es mejor. Sin embargo, esto no garantiza que dicho estado sea el mejor estado del problema.

Este algoritmo depende de lo siguiente:

1. Existencia de máximos o mínimos locales: Pico más alto que sus vecinos, pero más bajo que el pico máximo global
2. Existencia de zonas del espacio de estados con estados vecinos iguales al estado actual.
3. Existencia de zonas del espacio con varios máximos o mínimos locales.

1.3. Modifica el código para comenzar la búsqueda de nuevo desde otra solución inicial (Iterated local search). ¿Has conseguido mejorar? ¿Por qué?

Modificamos el código para ejecutar n veces la función de HillClimbing e ir comparando la soluciones finales obtenidas en cada una de las n iteraciones, para obtener de esta forma la solución más óptima. En nuestro caso hemos utilizado un criterio de parada de 10, 100 y 1000 iteraciones para observar como influye a la vez que aumentamos el tamaño del problema desde 5 hasta 50 ciudades.Los valores de longitud obtenidos son los valores medios de 5 ejecuciones del programa para cada valor n de iteraciones.



Tras observar la gráfica podemos determinar que para tamaños muy pequeños del problema como 5 o 6 ciudades es indiferente el número de iteraciones, todo ello es debido a que el espacio de estados o soluciones es tan pequeño que es fácil explorarlo por completo y alcanzar por tanto su valor óptimo.

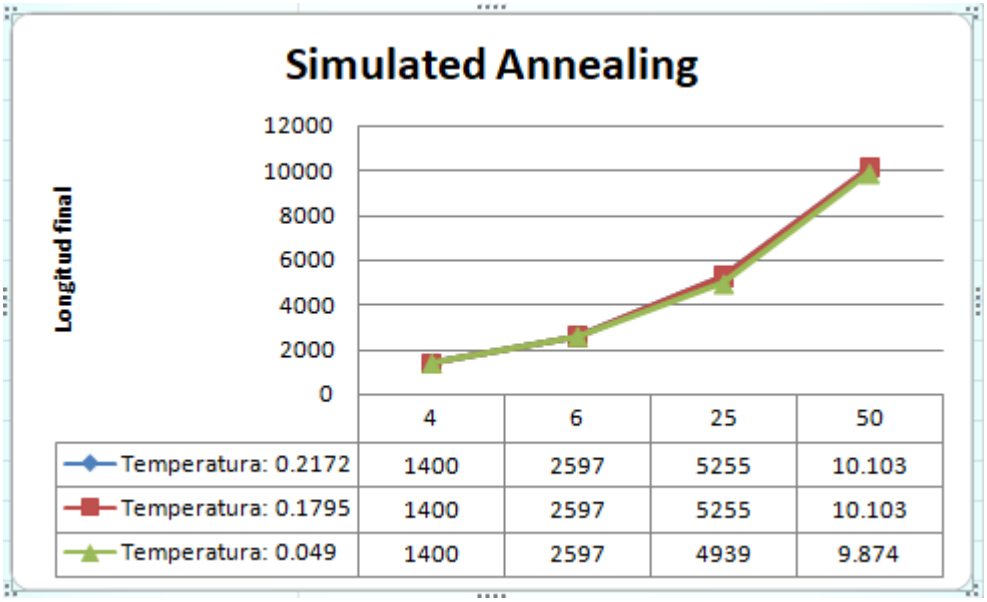
Para un tamaño considerable del problema como 15 ciudades se observa que disminuye la probabilidad de alcanzar el óptimo ya que al aumentar notoriamente el espacio de soluciones es más probable que el algoritmo se estanque en mínimos locales.Para este tipo de tamaños del problema ya se empieza a notar el efecto del número de iteraciones, ya que al generar un mayor número soluciones iniciales se incrementa la probabilidad de generar una solución inicial que permitan alcanzar mínimos locales cercanos al mínimo global o a el mismo.

Para tamaños del problema grandes como 25 y 50 ciudades se acentúa aun mas el efecto del número de iteraciones, es decir, conforme se incrementa el número de soluciones iniciales generadas se incrementa de forma notoria la probabilidad de alcanzar mínimos locales cercanos al global o a el mismo.

Capítulo 2

Simulated Annealing

2.1. ¿Cómo se comporta este algoritmo a medida que aumentamos el problema (número de ciudades en el TSP)?



Si observamos la gráfica anterior, vemos que este algoritmo mejora (se produce una optimización de la longitud final) cuando la temperatura es menor.

Debido a que la mejora es más visible cuando el número de ciudades es mayor, este algoritmo es recomendable para problemas con un número de ciudades elevado.

2.2. ¿Obtiene siempre la mejor solución? ¿Por qué? ¿De qué depende?

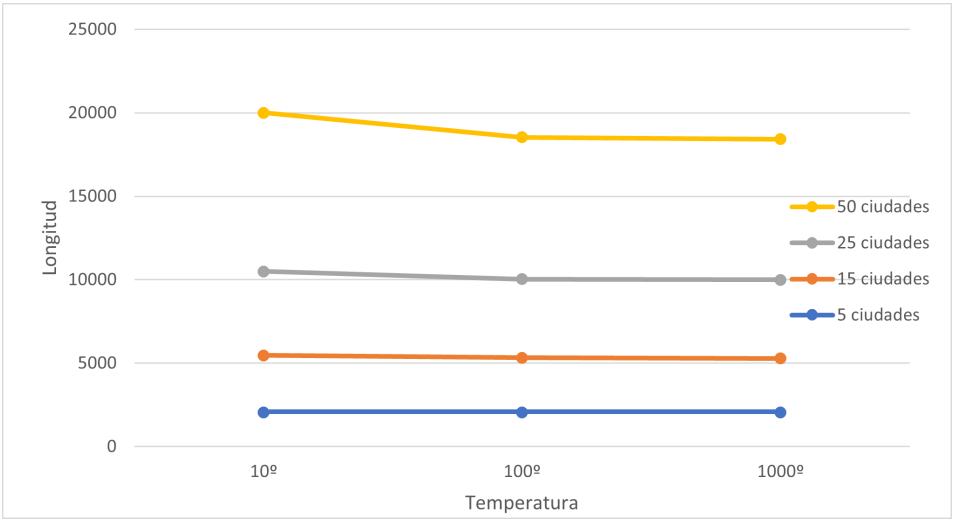
No siempre obtiene la mejor solución, porque la longitud final obtenida no siempre es constante (depende de la iteración).

Este algoritmo utiliza 2 criterios: una probabilidad de la configuración elegida (depende de la función de calidad del algoritmo) y de la temperatura de la iteración (depende de la temperatura inicial y del factor de enfriamiento utilizado).

2.3. Analiza cómo varía el comportamiento del algoritmo a medida que cambiamos el criterio de parada y la temperatura inicial

Con respecto al criterio de parada, si observamos la figura del ejercicio 2,1 podemos determinar que conforme disminuye el valor de temperatura mínimo para seguir iterando mejora la solución, ya que se obtiene menores valores longitud , esto es debido a que la temperatura es inversamente proporcional a la probabilidad de aceptar mejores soluciones, es decir, que a menor temperatura la probabilidad de aceptar peores soluciones disminuye y aumenta la de aceptar mejores soluciones por lo que el algoritmo se vuelve más restrictivo.

El efecto del criterio de parada comienza a ser efectivo para tamaños del problema considerables, ya que en tamaños pequeños del problema es relativamente fácil recorrer todo el espacio de soluciones y alcanzar el óptimo global.



Si observamos la figura anterior podemos comprobar que para tamaños del problema considerables se empieza a notar el efecto de partir de una temperatura inicial grande, ya que conforme aumenta la temperatura aumenta la probabilidad de aceptar peores soluciones lo que nos ayuda inicialmente a evitar caer en óptimos locales e intentar de esa forma alcanzar una zona cercana al óptimo global.

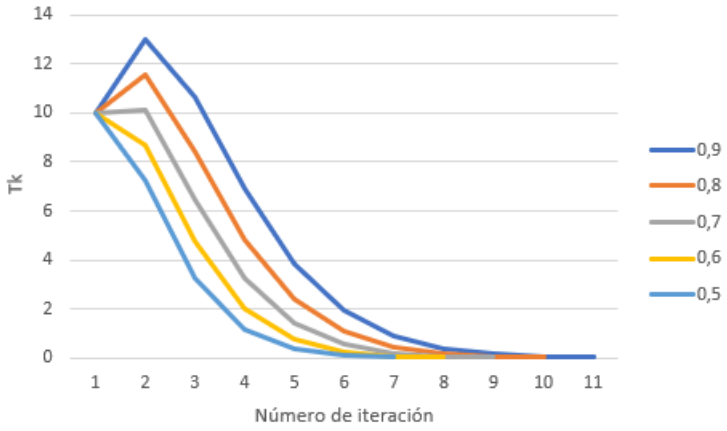
En nuestro caso podemos observar que si pasamos de una temperatura inicial de 10 a 100 grados se empieza a notar mejoras, pero en cambio la diferencia entre 100 y 1000 grados apenas es apreciable.Esto puede ser debido principalmente a dos factores:

- Con una temperatura de 100^{o} es suficiente para alcanzar valores cercanos óptimo global o a el mismo.
- Las zonas cercanas al óptimo global son unas soluciones tan malas que para poder saltarlas y se necesita una temperatura mucho mayor que 1000^{o}.

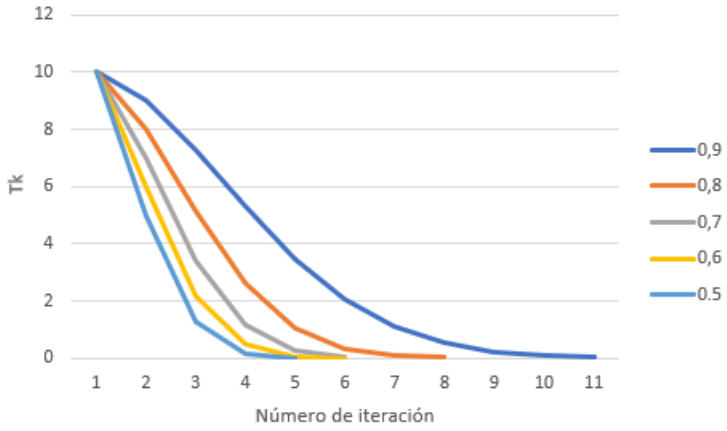
2.4. Modifica el código para utilizar funciones logarítmicas y geométricas de enfriamiento. ¿Cómo afectan estas funciones a los resultados finales? ¿Por qué? Ayúdate representando los valores de estas funciones. Busca nuevas funciones y compáralas a las anteriores

De la misma forma que al ejecutar el programa sin modificar ninguna función, la solución final varia, pero la longitud de la ruta final permanece igual. Por otra parte al aplicar las funciones se puede observar que el programa tarda menos en obtener los resultados ya que la temperatura desciende mucho más rápido si aplicamos dichas funciones. Según la función:

- Aplicando la función logarítmica, en la segunda iteración la temperatura sobrepasa el valor de la temperatura máxima establecida haciendo que empeore. Como se puede ver en la gráfica esto sucede en aquellos en los que alfa es mayor a 0,6.



- Aplicando la función geométrica, la temperatura baja mucho más rápido a medida que se va reduciendo alfa, como se puede ver en la siguiente gráfica.



Entre las funciones de enfriamiento existen otras como la función "fast simulated annealingz la función "GSA". A diferencia de las pedidas en el enunciado, estas funciones actúan de diferente manera. El "fast simulated annealing.[es] una función más popular que consiste en calcular la temperatura dividiendo la temperatura inicial entre el número de iteraciones más uno. En cuanto a la aceptación de peores soluciones, utiliza la temperatura, así así como la diferencia entre la evaluación de la función objetivo de la peor solución y la solución actual. Se calcula un valor entre 0 y 1 utilizando esta información, lo que indica la probabilidad de aceptar la peor solución. Luego, esta distribución se muestrea utilizando un número aleatorio, que, si es menor que el valor, significa que se acepta la peor solución. Por otra parte, el "GSA"se calcula de la siguiente manera: Tqa=Tqv/t siendo Tqa la temperatura de aceptación, Tqv la temperatura de visitación y t el número de iteraciones.Esta función consta de dos partes, la primera es un generador de números aleatorios y la segunda referente a la estructura del algoritmo. La eficiencia de este algoritmo depende de la generación de los números aleatorios.