Metaheurísticas
Práctica 1

Hill Climbing y Simulated Annealing

Damián Martínez Ávila

José Ángel Expósito Férnandez

Gabriel Cordero Contreras

4 de marzo de 2022

Enlace al código

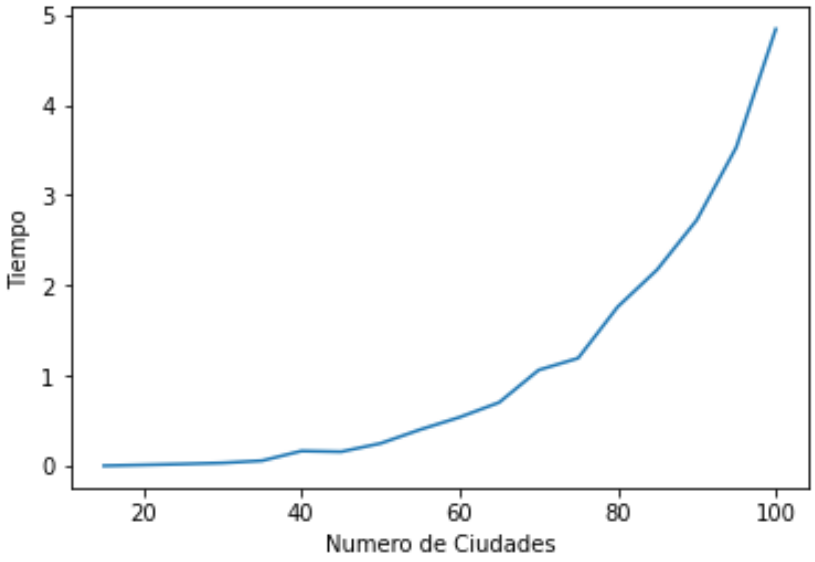
Índice

1. Hill Climbing

1.1. Escalabilidad del Algoritmo

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

A medida que crece el número de ciudades, el tiempo aumenta de forma exponencial, ya que es necesario explorar un mayor número de vecinos hasta llegar a un mínimo.



1.2. Optimización de la Solución

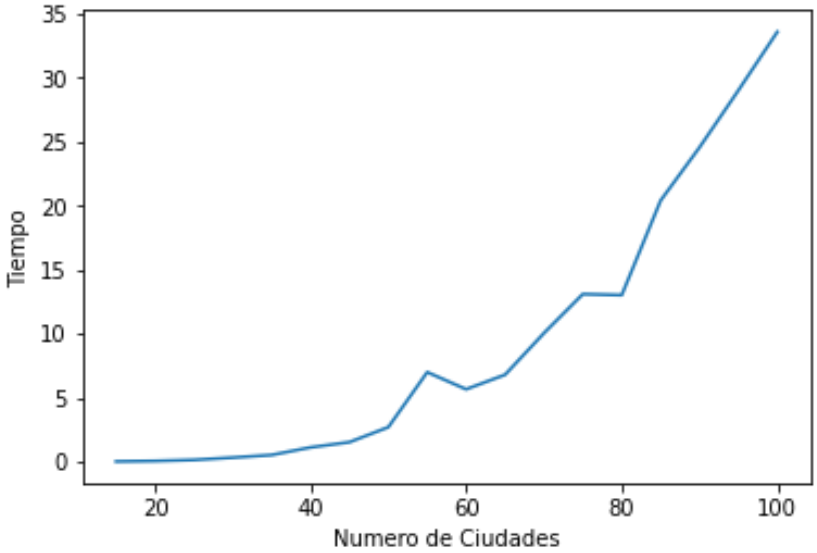
¿Obtiene siempre la mejor solución? ¿Por qué? ¿De qué depende?

No, ya que el mínimo local encontrado puede no ser el mínimo global. El resultado depende de la zona de inicialización.

1.3. Iterated Local Search

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

En la mayoría de casos si, ya que en este caso, cuando se queda atrapado en un mínimo local, al introducir una perturbación puede llegar a un mínimo distinto que suele ser mejor que el primer mínimo encontrado. Por otra parte, al tener que hacer más comparaciones el tiempo de ejecución aumenta considerablemente.

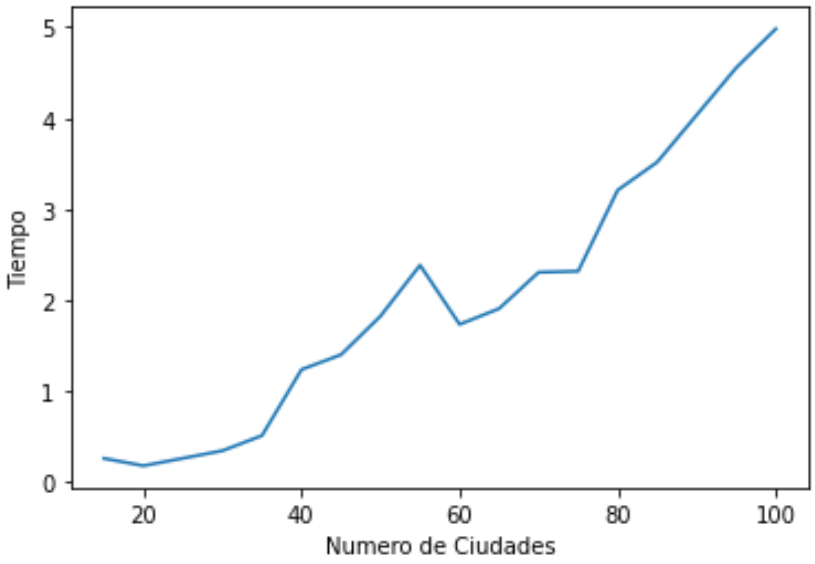


2. Simulated Annealing

2.1. Escalabilidad del Algoritmo

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

A medida que el número de ciudades crece, el algoritmo necesita un mayor tiempo para evaluar las nuevas posibles soluciones, aumentando el tiempo de cómputo.



2.2. Optimización de la Solución

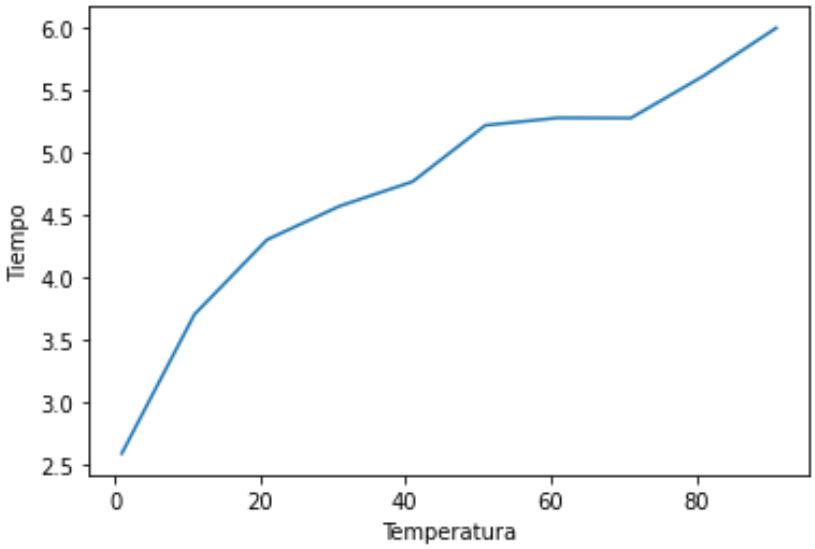
¿Obtiene siempre la mejor solución? ¿Por qué? ¿De qué depende?

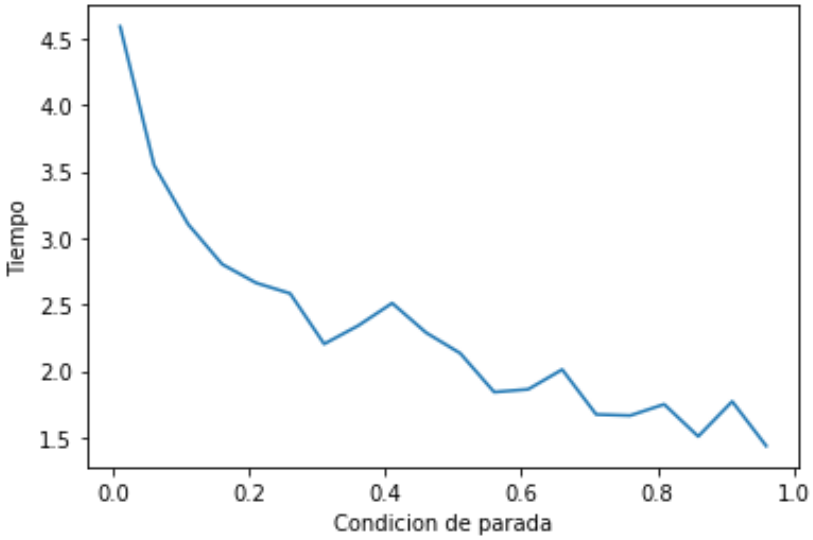
Siempre va a tender a alcanzar la solución óptima, pero podría ser que en las primeras iteraciones, cuando acepta peores soluciones, no explore el sub-espacio de soluciones óptimo y decida seguir buscando en otro sub-espacio de soluciones con un óptimo local en el que se quede atrapado

2.3. Criterio de Inicio y Parada

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

Para 100 ciudades, la temperatura varía de 1 a 100, podemos apreciar un incremento del tiempo de ejecución a medida que aumentamos la temperatura inicial. Mientras que para la condición de parada, se aprecia que mientras mayor temperatura se tiene para detener la búsqueda, el algoritmo tarda un menor tiempo.





2.4. Funciones de Enfriamiento

Modifica el código para utilizar diferentes funciones de enfriamiento:

- **Logarítmico.** $T_k = \frac{\alpha T_0}{ln(1+K)}$ **To es la temperatura inicial; es la velocidad de enfriamiento (valores menores de 1 aceleran el enfriamiento); k es la iteración en la que te encuentras.**

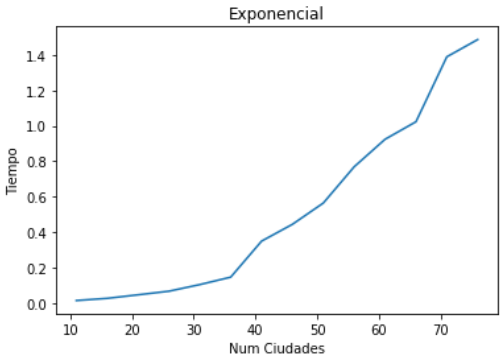
- **Geométrico.** $T_k = \alpha^K T_0$ **To es la temperatura inicial; es la velocidad de enfriamiento (valores menores de 1); k es la iteración en la que te encuentras.**

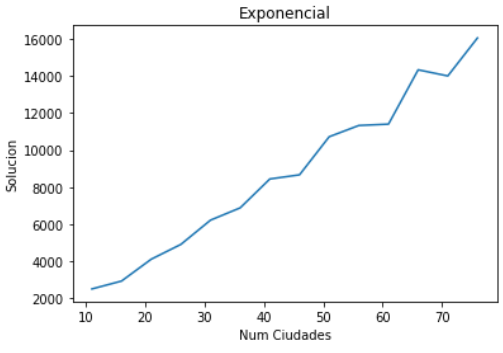
¿Cómo afectan estas funciones a los resultados finales? ¿Por qué? Ayúdate representando los valores de estas funciones. Busca nuevas funciones y compáralas a las anteriores.

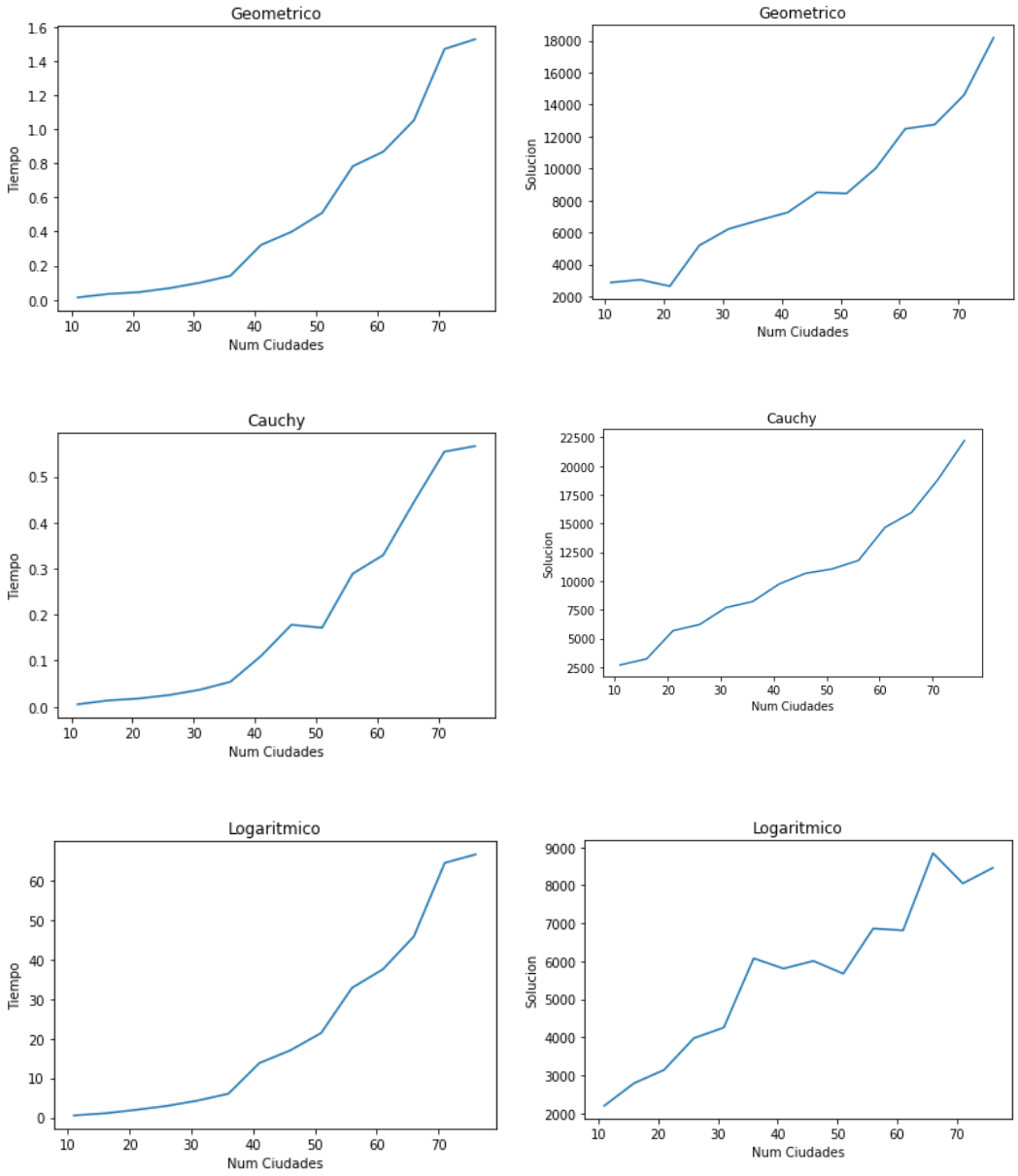
Las nuevas funciones de enfriamiento que vamos a utilizar son:

- Cauchy. $T_k = \frac{T_0}{1+k}$ To es la temperatura inicial; es la velocidad de enfriamiento (valores menores de 1 aceleran el enfriamiento); k es la iteración en la que te encuentras.
- Exponencial. $T_k = \alpha T_k$ To es la temperatura inicial; es la velocidad de enfriamiento (valores menores de 1); k es la iteración en la que te encuentras.

Se muestran las gráficas de los cuatro métodos de enfriamiento utilizados (Exponencial, Geométrico, Cauchy y Logarítmico), en una de ella se representa el tiempo respecto al número de ciudades y en la otra la solución obtenida frente al número de ciudades.







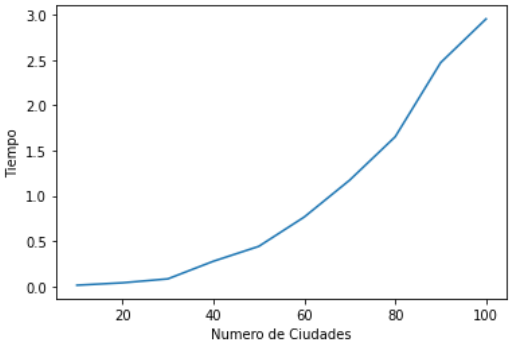
Podemos apreciar que usando una función logarítmica, el tiempo empleado es mucho mayor al resto, por lo que se exploran más soluciones y por tanto se obtiene una mejor solución en comparación con el resto de métodos. En el resto de métodos, al enfriarse de manera más rápida, el número de soluciones exploradas es menor, por tanto las soluciones obtenidas son mucho mayores al uso del logarítmico.

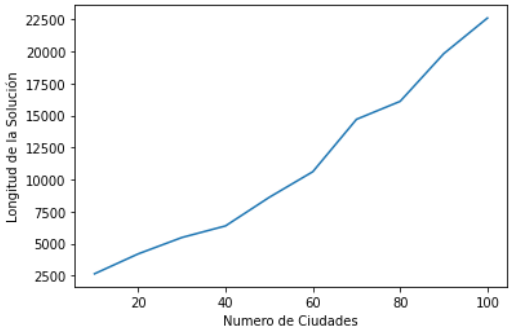
2.5. Mejora del Algoritmo

¿Cómo mejorarías el algoritmo? Por ejemplo, recalentando cada cierto tiempo. Modifica el código con esta y cualquier otra mejora que se te ocurra. Analiza los resultados.

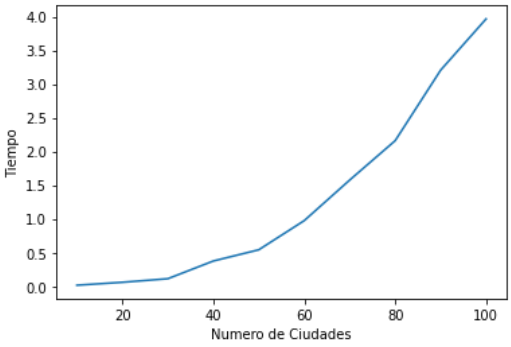
Al aplicar el recalentado, se aprecia que el algoritmo tarda un mayor tiempo, pero que gracias a este tiempo extra, las soluciones obtenidas son mejores a las obtenidas sin el recalentado.

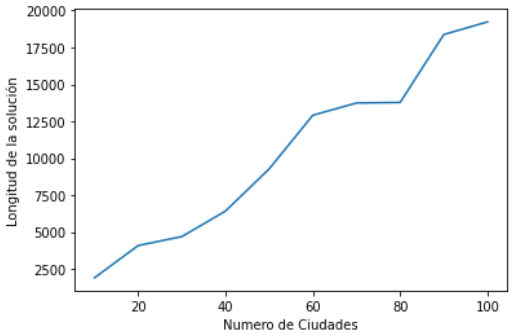
Ejecución del algoritmo sin modificar (sin aplicar recalentado).





Ejecución del algoritmo modificado (aplicando recalentado).





Referencias

- Codigo Fuente de la Práctica: https://github.com/p82maavd/Metaheuristicas/blob/main/Practica1Metaheuristica.ipynb
- Apuntes de Moodle: https://moodle.uco.es/m2122/course/view.php?id=1674