Universidad de Córdoba

Escuela Politécnica Superior de Córdoba

Grado en Ingeniería Informática

Metaheurísticas

Práctica 1

Hill Climbing y Simulated Annealing

Alberto Alcaraz Gutiérrez *i02algua*
Mario Berrios Carmona *i12bercm*
Miguel Ángel García Moreno *i02gamom*
Francisco Moreno Cano *i02mocaf*

Curso 2022/2023

Índice

Índice de figuras

Índice de cuadros

1. Introducción

En esta práctica se estudiarán dos técnicas heurísticas distintas: *Hill Climbing*, o algoritmo de escalada simple; y *Simulated Annealing* , o enfriamiento simulado. *Hill Climbing* trata de buscar una solución en un ámbito local en un tiempo razonable, mientras que el *Simulated Annealing* tratará de encontrar una solución aceptable en el ámbito global, aceptando o no soluciones peores según una probabilidad que vendrá dada por la "temperatura", la cual bajará a lo largo de la ejecución del algoritmo, llegando a mejores resultados.

Ambos algoritmos se pueden mejorar para que sean capaces de encontrar mejores soluciones, como es el caso de la adición de *Iterated Local Search* (búsqueda local iterada) para el *Hill Climbing* o cambios en la función de enfriamiento en *Simulated Annealing*.

Se tratará, por medio del análisis de los resultados de varias pruebas experimentales, la eficiencia y la complejidad de los dos métodos, como mejoran estos resultados después de retocar los algoritmos para añadir las mejoras, y como varía los resultados al usar un método u otro. Todos estos experimentos constan de 100 ejecuciones del algoritmo en cuestión, teniendo así una mejor idea del comportamiento medio del mismo. A su vez, todos los experimentos realizados han sido ejecutados en los ordenadores propios de la UCO para poder mantener esta variable constante a lo largo de la práctica.

Para probar los dos métodos se hará uso de dos programas codificados en el lenguaje *Python* (*HillClimbing.py* y *SimAnnealing.py)*, además de un conjunto de *datasets* que tratan de emular el problema del viajante del comercio (*TSP*). Todos estos archivos provienen de la *Moodle* de la asignatura. Sin embargo, se han realizado modificaciones para implementar las mejoras que proponen los ejercicios e indicar por línea de comandos el valor de los hiperparámetros y el *dataset* a estudiar. Estos ficheros serán adjuntados a este informe.

2. Hill Climbing

Analiza en detalle cómo se comporta el algoritmo a medida que aumentamos la complejidad del problema (espacio de búsqueda). Considera el número de ciudades para ello. Para ello, analiza el espacio de búsqueda.

Este experimento utiliza cada uno de los *datasets* facilitados en la práctica y analiza el tiempo de computo de cada uno de ellos. Para ello se ha modificado el fichero *HillClimbing.py* de tal forma que estipule el tiempo de computo necesario para ejecutar el algoritmo una sola vez durante las 100 ejecuciones que compone el experimento. Acto seguido se ha calculado el tiempo medio de las 100 iteraciones y se ha graficado. Como se observa en la figura 1, la curva de tiempo crece exponencialmente con respecto al número de ciudades.

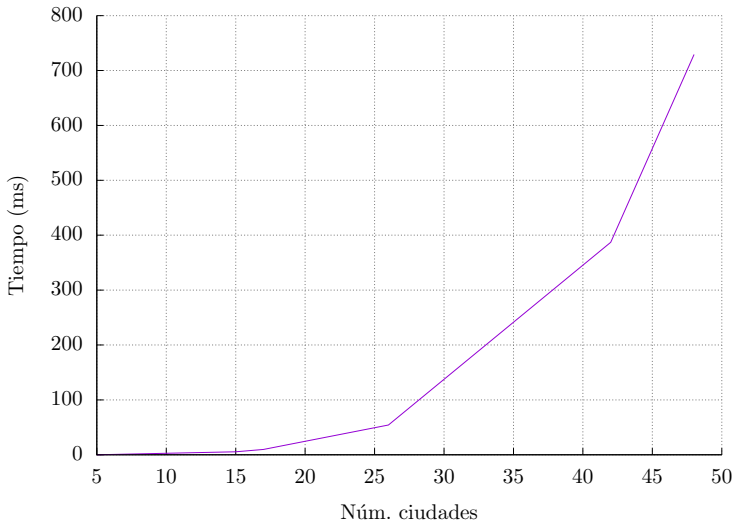


Figura 1: Tiempo medio con respecto al número de ciudades en *Hill Climbing*

Esto se puede relacionar con el espacio de búsqueda. En el problema *TSP*, suponiendo un grafo denso, el espacio de búsqueda puede calcularse mediante las permutaciones del conjunto de todos los nodos (n). El número de permutaciones corresponde al factorial $n!$. Esto muestra como el espacio de búsqueda aumenta a medida que aumenta el número de ciudades, haciendo que para llegar al mínimo local más cercano, el número de iteraciones necesarias sea mayor.

Realiza la experimentación que consideres oportuna para contestar a las siguientes cuestiones. Utiliza las pruebas y resultados experimentales que necesites: ¿El algoritmo obtiene siempre la mejor solución? ¿Por qué? ¿De qué depende?

Para estudiar si el algoritmo es capaz de encontrar siempre la solución óptima, se han considerado la mejor solución y la solución media obtenidas en las 100 ejecuciones realizadas. En la tabla 1 se encuentran los resultados experimentales junto con un porcentaje de variación (última columna). Este porcentaje indica cuánto más largo es el camino proporcionado por la solución media respecto al que indica la solución óptima a través de la ecuación 1.

$$Variación = \frac{media - óptimo}{óptimo} \cdot 100 \tag{1}$$

Analizando los datos del experimento, se llega a la conclusión de que el algoritmo no siempre encuentra la mejor solución. De hecho, se observa que cuantas más ciudades contenga el problema (cuanto mayor sea el espacio de búsqueda), más difícil es para el algoritmo encontrar la solución óptima. Además, se aprecia que conforme crece el número de ciudades la solución media se aleja cada vez más de la óptima (figura 2).

	Ciudades	Óptimo	Mejor solución (frecuencia)	Media	Media - Óptimo (%)
FIVE	5	19	19 (81)	19.38	0.38 (2.0)
P01	15	291	291 (18)	332.82	41.82 (14.371)
GR17	17	2085	2085 (2)	2227.79	142.79 (6.848)
FRI26	26	937	937 (1)	1139.45	202.45 (21.606)
DANTZIG42	42	699	838 (1)	1008.64	309.64 (44.298)
ATT48	48	33523	39457 (1)	48245.17	14722.17 (43.917)

Cuadro 1: Comparativa entre resultados experimentales y óptimos en *Hill Climbing*

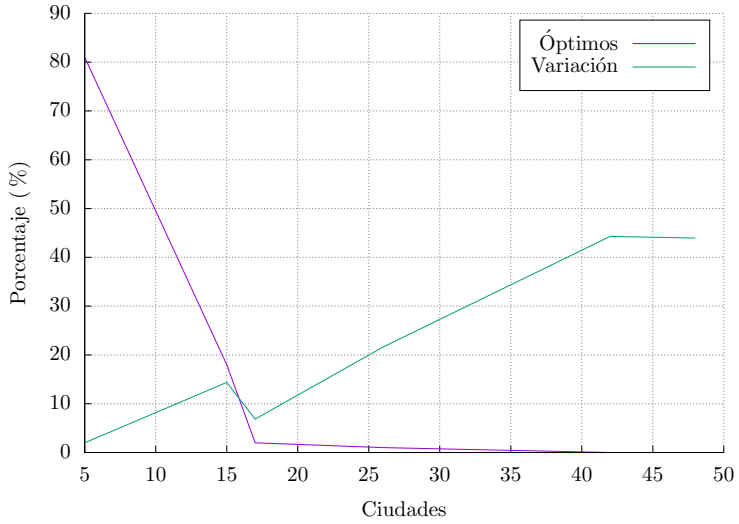


Figura 2: Porcentaje de aciertos y variación con respecto al óptimo en *Hill Climbing*

La razón de este suceso es el propio funcionamiento del algoritmo. *Hill Climbing* tiene una componente estocástica, la cual indica el estado inicial dentro del espacio de búsqueda. A partir de ahí traza la trayectoria hasta, en este caso, el mínimo local más cercano. Este mínimo local puede corresponder con el mínimo global, pero cuanto mayor sea el espacio de búsqueda, más difícil es que coincidan, reduciendo así la frecuencia de óptimos encontrados.

En cuanto a la variación entre media y óptimo, su crecimiento se debe a un motivo similar. Cuanto mayor es el espacio de búsqueda, mayor número de soluciones posibles, y por tanto, mayor facilidad de que valor del óptimo global sea más distante a cualquier óptimo local.

Realiza las modificaciones que consideres oportunas sobre el código proporcionado para obtener un *Iterated Local Search*. Una vez hecho esto, realiza la experimentación que consideres oportuna para contestar a las siguientes cuestiones. Utiliza las pruebas y resultados experimentales que necesites: ¿El algoritmo obtiene siempre la mejor solución? ¿Por qué? ¿De qué depende?

Una vez añadidas las modificaciones necesarias al fichero *Hill Climbing.py* para obtener el *Iterated Local Search*, se han ejecutado 100 iteraciones del algoritmo, cada una con 10 perturbaciones. Este número se ha elegido debido al tiempo de ejecución que se necesitaría si se aumenta. Estas perturbaciones se han realizado mediante una serie de permutaciones entre dos elementos de la solución. El número de permutaciones se ha calculado como el 20 % del número de ciudades con el que se está trabajando, siendo el mínimo número de permutaciones 2 para que no se escojan soluciones vecinas. Los resultados de este procedimiento aparecen en la tabla 2.

Se puede observar que, de nuevo, no siempre se obtiene la solución óptima. La principal razón de este hecho es que se sigue trabajando con un algoritmo con una componente estocástica, lo que provoca

	Ciudades	Óptimo	Mejor solución (frecuencia)	Media	Media - Óptimo (%)
FIVE	5	19	19 (96)	19.08	0.08 (0.421)
P01	15	291	291 (16)	330.64	39.64 (13.622)
GR17	17	2085	2085 (5)	2246.02	161.02 (7.723)
FRI26	26	937	960 (1)	1155.40	218.40 (23.308)
DANTZIG42	42	699	820 (1)	1027.5	328.5 (46.996)
ATT48	48	33523	36206 (1)	48730.91	15207.91 (45.365)

Cuadro 2: Comparativa entre resultados experimentales y óptimos en *Iterated Local Search*

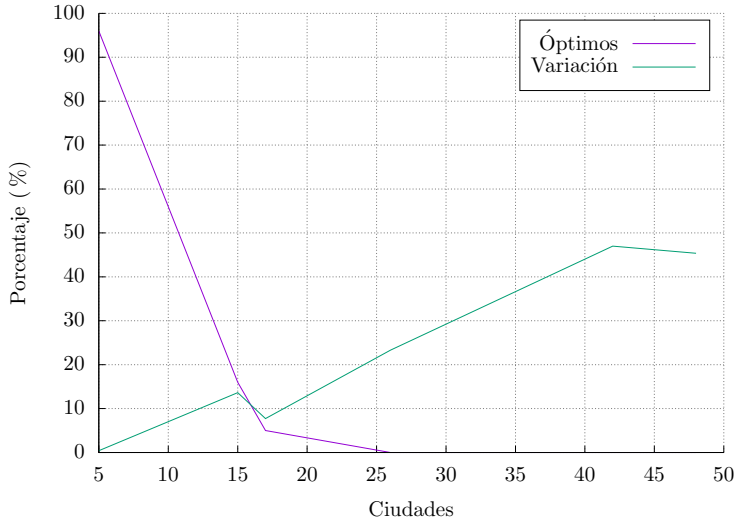


Figura 3: Porcentaje de aciertos y variación con respecto al óptimo en *Iterated Local Search*

que encontrar una mejor o peor solución dependa del inicio aleatorio que se haya seleccionado para el problema específico. También indicar como factor importante, que aunque añadir una perturbación permite "escapar" del mínimo local que puede no corresponderse con la solución óptima del problema, la elección de una perturbación adecuada es imprescindible si se busca la máxima efectividad en el algoritmo.

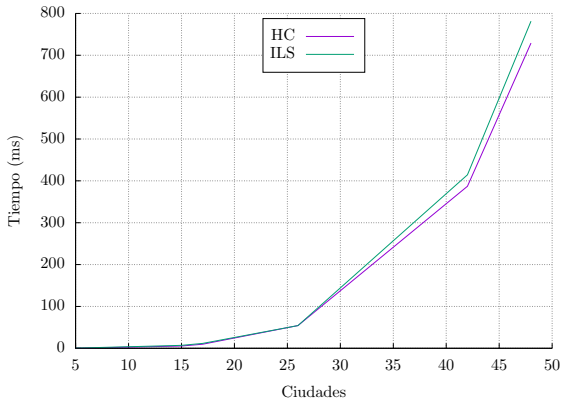
Observando la variación entre media y óptimo, igual que sucedió en el algoritmo base de *Hill Climbing* conforme aumenta el número de ciudades (cuanto mayor es el espacio de búsqueda), crece esta diferencia, lo que se traduce en que proporcionalmente existe más probabilidad de que el algoritmo finalice en un mínimo local alejado de la solución óptima correspondiente al problema. Como se puede advertir, los resultados finales en *Iterated Local Search* (cuadro 2) empeoran con respecto a *Hill Climbing* sin ninguna mejora (cuadro 1), este hecho se comentará en el apartado de comparativa próxima.

Realiza una comparativa entre los dos algoritmos para comprobar cual se comporta mejor. Analiza tiempos, caminos obtenidos, y cualquier otra cosa que se te ocurra para comparar.

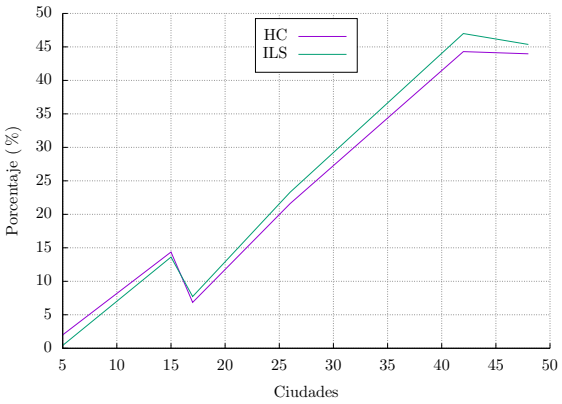
Tras el análisis de los resultados obtenidos al ejecutar el algoritmo original frente al algoritmo mejorado, podemos observar que las diferencias entre ambas son mínimas. La diferencia del tiempo de ejecución (figura 4a) de ambos algoritmos es despreciable. Esto se debe a que el número de perturbaciones que ejecuta *Iterated Local Search* es muy bajo. Si este número se eleva, el tiempo de ejecución de *ILS* aumentaría proporcionalmente, acentuando la diferencia entre ambos.

Seguidamente, en la figura 4b, se aprecia que la variación cambia de tal forma que es más favorable para *ILS* con valores de ciudades pequeños, empeorando a medida que el espacio de búsqueda aumenta. Una vez más, esto se debe al bajo número de perturbaciones realizadas en *ILS*. En el caso de que el número de perturbaciones fuese muy elevado, *Iterated Local Search* debería tener valores mejores que *Hill Climbing* sin esta mejora, ya que aumentaría la probabilidad de obtener una mejor solución.

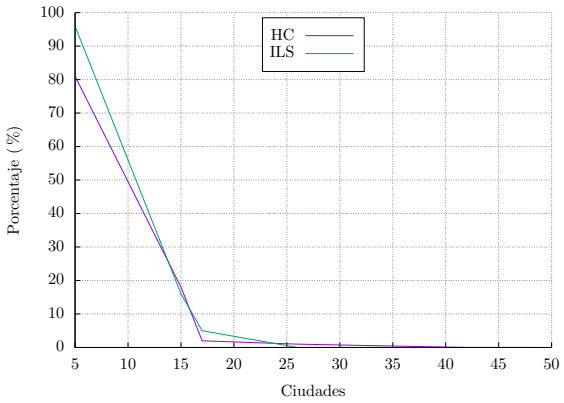
Por último, respecto a la diferencia en la frecuencia de óptimos (figura 4c), *ILS* devuelve mejores resultados para valores pequeños. Esto debe mantenerse independientemente del número de ciudades si se realizan las perturbaciones suficientes.



(a) Tiempo de ejecución



(b) Variación



(c) Frecuencia de óptimos

Figura 4: Comparativa entre *HC* y *ILS*

Con respecto a los datos obtenidos puede afirmarse que *Iterated Local Search* es mejor, ya que aumenta la probabilidad en una sola ejecución de encontrar mejores resultados. El único defecto de *ILS* frente al algoritmo base de *Hill Climbing* sería el tiempo de cómputo, por el hecho obvio de que debe realizar las perturbaciones que se le indiquen.

Comparando estos resultados con la teoría, *ILS* debería destacar siempre sobre el algoritmo base de *Hill Climbing*, como así corrobora la experimentación realizada. La diferencia que encontramos se debe a esa componente estocástica a la hora de elegir la solución inicial.

3. Simulated Annealing

Analiza en detalle cómo se comporta el algoritmo a medida que aumentamos la complejidad del problema (espacio de búsqueda). Considera el número de ciudades para ello. Para ello, analiza el espacio de búsqueda.

Al igual que en el análisis de algoritmo de *Hill Climbing*, se han realizado 100 iteraciones de *Simulated Annealing* con los datos de cada *dataset*. En cada ejecución se obtienen los tiempos medios consumidos en las 100 iteraciones, los cuales se encuentran representados en la figura 5.

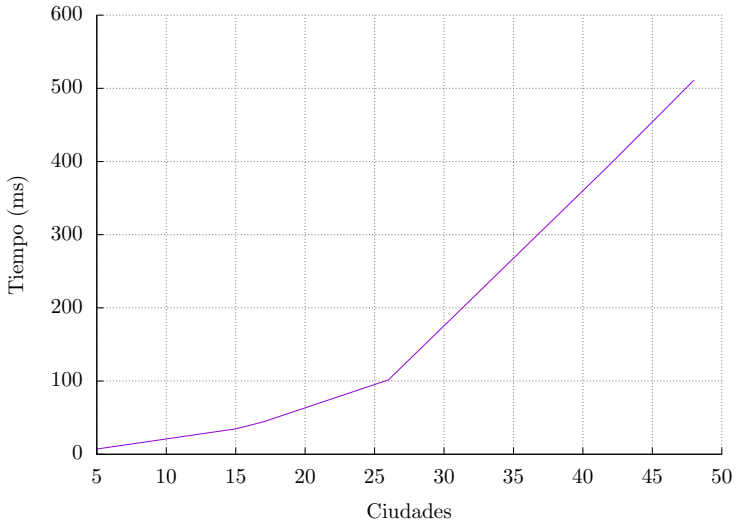


Figura 5: Tiempo medio con respecto al número de ciudades en *Sim. Annealing*

De igual forma que con el algoritmo *Hill Climbing*, *Simulated Annealing* aumenta el tiempo de computo de forma drástica a medida que el espacio de búsqueda incrementa. Este crecimiento es de orden exponencial.

Realiza la experimentación que consideres oportuna para contestar a las siguientes cuestiones. Utiliza las pruebas y resultados experimentales que necesites: ¿El algoritmo obtiene siempre la mejor solución? ¿Por qué? ¿De qué depende?

Al igual que en el apartado anterior, se han recogido los valores de las 100 ejecuciones. En la tabla 3 se muestran los distintos resultados.

	Ciudades	**Óptimo**	**Mejor solución (frecuencia)**	**Media**	**Media - Óptimo (%)**
FIVE	5	19	19 (100)	19.0	0.0 (0.0)
P01	15	291	291 (13)	337.67	46.67 (16.038)
GR17	17	2085	2085 (1)	2289.68	204.68 (9.817)
FRI26	26	937	1058 (1)	1284.67	347.67 (37.105)
DANTZIG42	42	699	1173 (1)	1402.35	703.35 (100.622)
ATT48	48	33523	57796 (1)	69715.86	36192.86 (107.964)

Cuadro 3: Comparativa entre resultados experimentales y óptimos en *Sim. Annealing*

Se observa que con números de ciudades bajos se encuentra con mucha más frecuencia el valor óptimo del problema, mientras que a medida que aumentamos el espacio de búsqueda, esta frecuencia va disminuyendo. Este comportamiento se debe a la componente estocástica que presenta *Simulated Annealing*, provocando que al aumentar el espacio de búsqueda sea menos probable llegar al mínimo global.

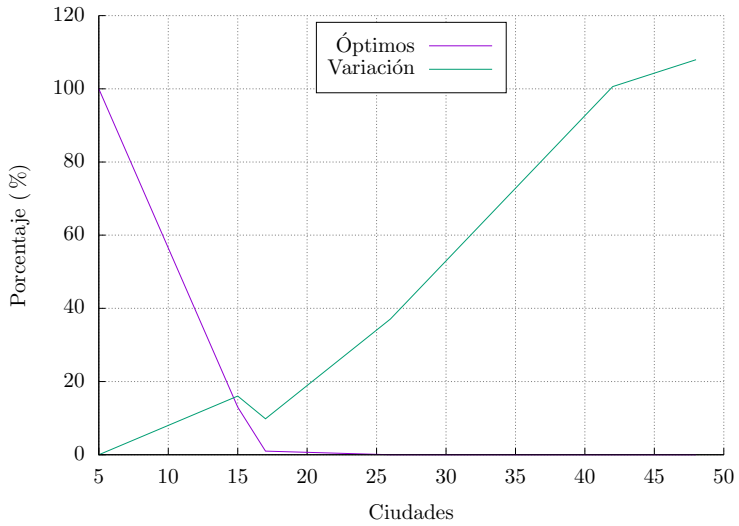


Figura 6: Porcentaje de aciertos y variación con respecto al óptimo en *Simulated Annealing*

Es importante entender que el cuadro 3 y la figura 6 se ha realizado teniendo una temperatura inicial, T_0, igual a 10 y un coeficiente de enfriamiento, α, de 0.99. En apartados posteriores se analizará como se comporta el algoritmo al modificar estos valores.

Realiza las modificaciones que consideres oportunas sobre el código proporcionado para considerar diferentes criterios de parada y temperatura inicial. Una vez hecho esto, realiza la experimentación que consideres oportuna para contestar a las siguientes cuestiones. Utiliza las pruebas y resultados experimentales que necesites. ¿Ha variado el comportamiento? ¿De qué manera? ¿Por qué?.

Para estudiar este apartado se han realizado dos experimentos con el mismo *dataset*, siendo este FRI26. En el primero de ellos (cuadro 4) se ha variado la temperatura inicial (T_0) sin variar el coeficiente de enfriamiento (α) fijándolo en 0.99. En el segundo experimento (cuadro 5) se ha modificado α manteniendo T_0 fijo en 10. Al mantener en ambos experimentos el mismo *dataset* podemos comprobar el efecto en los resultados finales.

Temperatura	**Óptimo**	**Mejor solución (frecuencia)**	**Media**	**Media - Óptimo (%)**
0.5	937	1186 (1)	1457.75	520.75 (55.576)
1		1235 (1)	1405.85	468.85 (50.037)
2		1127 (1)	1377.64	440.64 (47.027)
5		1101 (1)	1309.97	372.97 (39.805)
10		1067 (1)	1280.07	343.07 (36.614)
20		1062 (1)	1270.7	333.7 (35.614)
50		1067 (1)	1244.51	307.51 (32.819)

Cuadro 4: Comparativa variando temperatura en el *dataset* FRI26

En el primer experimento puede observarse como, a medida que la temperatura inicial va elevándose, la variación de la media con respecto al óptimo va disminuyendo. Es decir, cuanto menor es la temperatura menos probabilidad hay de escoger una solución vecina con peor solución al problema. Esto provoca que a una baja temperatura, sea más difícil salir de un mínimo local, quedándose en él en lugar de seguir explorando el espacio de búsqueda.

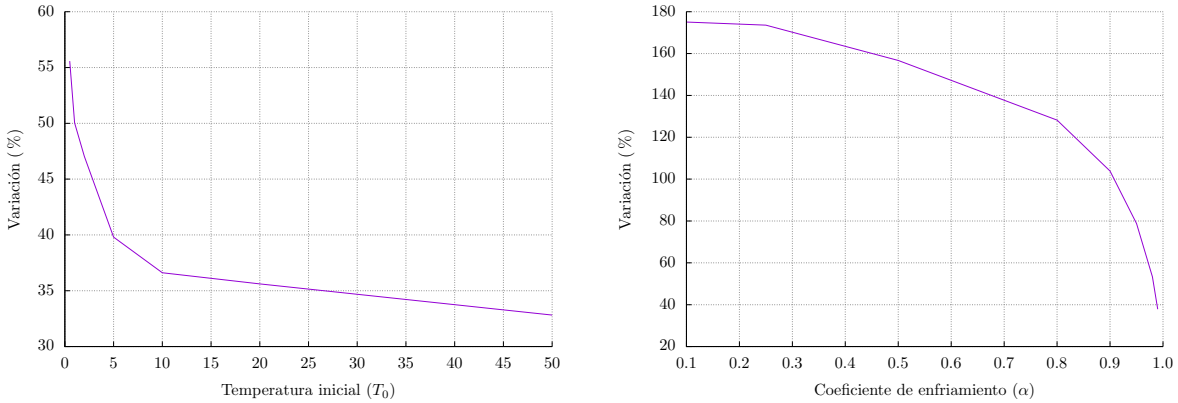
En el segundo experimento se observa un comportamiento similar. A medida que el coeficiente de enfriamiento va aumentando, la variación de la media con respecto al óptimo va disminuyendo. La razón

Coeficiente de enfriamiento	**Óptimo**	**Mejor solución (frecuencia)**	**Media**	**Media - Óptimo (%)**
0.1	937	2137 (1)	2577.41	1640.41 (175.070)
0.25		2142 (1)	2563.63	1626.63 (173.6)
0.5		1989 (1)	2405.22	1468.22 (156.694)
0.8		1792 (1)	2138.4	1201.4 (128.218)
0.9		1576 (1)	1910.3	973.3 (103.874)
0.95		1370 (1)	1674.98	737.98 (78.76)
0.98		1199 (1)	1436.49	499.49 (53.307)
0.99		1105 (1)	1291.66	354.66 (37.851)

Cuadro 5: Comparativa variando coeficiente de enfriamiento en el *dataset* FRI26

de este comportamiento, se debe a que cuanto mayor sea α, más lentamente disminuye la temperatura, lo que se traduce en una mayor facilidad de salir de un mínimo local, y por tanto de seguir explorando el espacio de búsqueda.

En vista de estos resultados se puede observar que tanto el aumento de la temperatura inicial, T_0, como el aumento del coeficiente de enfriamiento, α, provoca un decremento de la variación, sin embargo, se desconoce si el grado en que la modifican es similar. En la figura 7 se muestra como influye el cambio de T_0 y de α al porcentaje de variación entre media y óptimo.



(a) Cambio de temperatura inicial (T_0)

(b) Cambio de coeficiente de enfriamiento (α)

Figura 7: Evolución al cambio de hiperparámetros

Con respecto a T_0 (figura 7a), una variación en este parámetro cuando su valor es bajo, o lo que es igual, cuando la probabilidad de seleccionar un vecino peor es baja, muestra que tiene una importancia elevada con respecto al comportamiento del algoritmo. Por otro lado, a partir de cierto valor ($T_0 \geq 10$ en este *dataset*) se observa poca diferencia en los resultados obtenidos por el algoritmo.

Estudiando los resultados al variar el valor de α, observamos que si este toma un valor pequeño, no tiene mucha influencia. Sin embargo, si el valor que toma es mayor ($\alpha \geq 0{,}8$ en este *dataset*), el resultado sí cambia de forma considerable. La explicación a dicho fenómeno es que a partir de cierto valor de α la variación de temperatura es demasiado rápida, dando poco tiempo para escoger un vecino peor y así poder explorar mejor el espacio de búsqueda.

Modifica el código para utilizar diferentes funciones de enfriamiento:

- **Logarítmico**
- **Geométrico**

Realiza la experimentación que consideres oportuna para contestar las siguientes cuestiones usando los experimentos: ¿Cómo afectan estas funciones a los resultados finales? ¿Por qué? Si lo deseas, puedes buscar nuevas funciones y compararlas a las anteriores.

En la figura 8 se muestra el comportamiento teórico de las distintas funciones implementadas en el *script SimAnnealing.py* a lo largo de quinientas iteraciones. Además de las dos propuestas en el enunciado, se ha añadido una tercera con un comportamiento lineal [1].

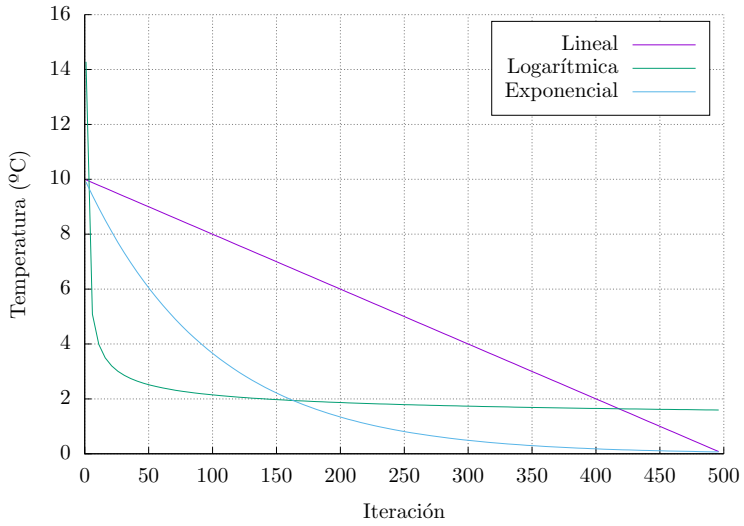


Figura 8: Curva de las distintas funciones de enfriamiento

La función lineal (ecuación 2) disminuye de forma constante la temperatura a medida que aumenta el numero de iteraciones, lo que supone que la probabilidad de elegir una solución vecina peor que la actual decremente de la misma forma. Eso se traduce en un mayor número de saltos a lo largo de la ejecución, lo que es bueno en el caso de que haya muchos óptimos locales. Para comprobar su comportamiento teórico se han tomado los valores $n = 500$ y $T_n = 0$ para una bajada suave de la temperatura.

$$T_k = T_n + (T_0 - T_n)\left(\frac{n-k}{n}\right) \tag{2}$$

Por el contrario, la función logarítmica (ecuación 3) tiene una pendiente negativa muy pronunciada al principio hasta llegar a cierto valor, donde comienza una asíntota horizontal. Este descenso tan pronunciado hace que la probabilidad de seleccionar una solución vecina peor que la actual disminuya proporcionalmente. Por ende, esta función sería aconsejada para los casos en los que los óptimos locales sean pocos o se quiera centrar en una búsqueda local de forma rápida.

$$T_k = \frac{\alpha \cdot T_0}{ln(1+k)} \tag{3}$$

Por último tenemos la función geométrica (ecuación 4). Esta función es un compromiso entre la función logarítmica y la función lineal. La función base que se propone en la práctica tiene este comportamiento, aunque está implementada de forma recursiva (ecuación 5)

$$T_k = \alpha^k \cdot T_0 \tag{4}$$

$$T_k = \alpha \cdot T_{k-1} \tag{5}$$

Experimentalmente (cuadro 6, 7 y 8) se puede observar como los valores no son muy distintos a los conseguidos mediante la función base (ecuación 5). Esto se observa a lo largo de todos los *datasets*. La única excepción es con la función de enfriamiento logarítmica, la cual funciona considerablemente mejor en espacios de búsqueda grandes (cuadro 7). Ha de tenerse en cuenta que esta función se ha utilizado con unos hiperparámetros diferentes ($T_0 = 1$, $\alpha = 0{,}35$) debido a su propia naturaleza, ya que la existencia de la asíntota horizontal comentada anteriormente, provoca un tiempo de ejecución excesivo utilizando los mismos que en el resto de funciones.

	Ciudades	**Óptimo**	**Mejor solución (frecuencia)**	**Media**	**Media - Óptimo (%)**
FIVE	5	19	19 (98)	19.04	0.04 (0.211)
P01	15	291	291 (20)	325.47	34.47 (11.845)
GR17	17	2085	2088 (1)	2285.5	200.5 (9.616)
FRI26	26	937	1044 (1)	1298.47	361.47 (38.577)
DANTZIG42	42	699	1158 (1)	1403.82	704.82 (100.833)
ATT48	48	33523	56978 (1)	70435.27	36912.27 (110.110)

Cuadro 6: Comparativa utilizando la función de enfriamiento lineal en *Sim. Annealing*

	Ciudades	**Óptimo**	**Mejor solución (frecuencia)**	**Media**	**Media - Óptimo (%)**
FIVE	5	19	19 (100)	19.0	0.0 (0.0)
P01	15	291	291 (18)	332.46	41.46 (14.247)
GR17	17	2085	2085 (1)	2255.9	170.9 (8.197)
FRI26	26	937	975 (1)	1193.32	256.32 (27.355)
DANTZIG42	42	699	930 (1)	1217.67	518.67 (74.202)
ATT48	48	33523	47824 (1)	61790.53	28267.53 (84.323)

Cuadro 7: Comparativa utilizando la función de enfriamiento logarítmica en *Sim. Annealing*

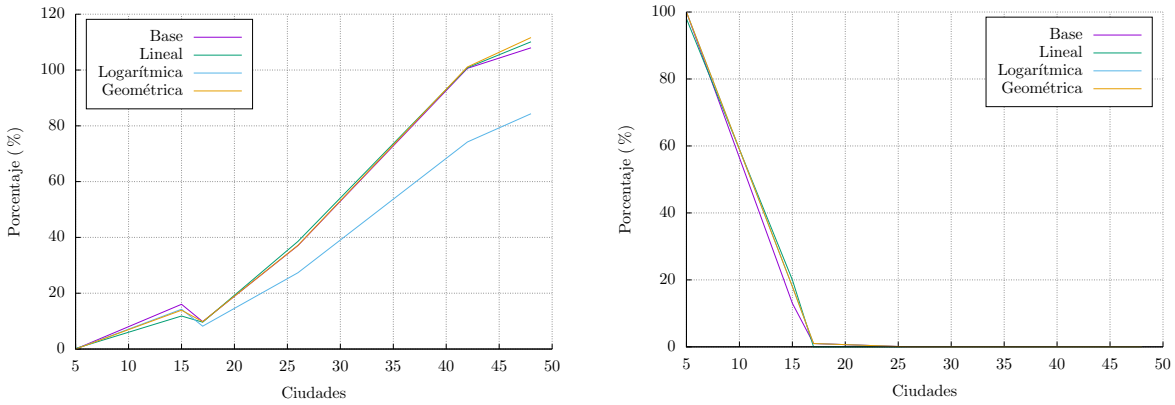
	Ciudades	**Óptimo**	**Mejor solución (frecuencia)**	**Media**	**Media - Óptimo (%)**
FIVE	5	19	19 (100)	19.0	0.0 (0.0)
P01	15	291	291 (18)	331.48	40.48 (13.911)
GR17	17	2085	2085 (1)	2289.84	204.84 (9.824)
FRI26	26	937	1090 (1)	1285.37	348.37 (37.179)
DANTZIG42	42	699	1170 (1)	1405.76	706.76 (101.110)
ATT48	48	33523	56701 (1)	70940.93	37417.93 (111.619)

Cuadro 8: Comparativa utilizando la función de enfriamiento geométrica en *Sim. Annealing*

¿Cómo mejorarías el algoritmo? Modifica el código con la idea que se te ocurra y analiza los resultados.

Dos técnicas que pueden mejorar el algoritmo son el *backtracking* y el recalentamiento. La mejora de *backtracking* implementada consiste en almacenar en memoria aquellas soluciones alcanzadas justo antes de obtener un incremento positivo, es decir, de obtener una solución peor que la anterior. De esta forma permitimos al algoritmo acceder a la mejor solución ya almacenada. Esto puede ser útil debido a la propia naturaleza de *Simulated Annealing*, el cual puede finalizar con una solución peor que las retenidas en memoria.

Por otra parte, la mejora de recalentamiento consiste en aumentar la temperatura en algún punto del algoritmo. En nuestro caso este aumento se realiza cuando se selecciona una peor solución, es decir, en el mismo momento en el que se aplica *backtracking*. Otra forma de aplicar esta mejora sería mediante un



(a) Variación (b) Óptimos

Figura 9: Comparativa entre las funciones de enfriamiento

aumento de la temperatura constante en cada iteración, independientemente del vecino que se elija. Con esta técnica se consigue realizar una búsqueda más amplia, ya que al subir la temperatura del algoritmo este aceptará más fácilmente desplazarse a vecinos que incrementen la longitud de la solución, pudiendo explorar una mayor parte del espacio de búsqueda y evitar quedarse atrapado en mínimos locales.

Por último, una posible mejora que no se ha implementado sería cambiar la función que mide la probabilidad de seleccionar una peor solución, que en este caso ha sido:

$$e^{-\Delta C/temp}$$

siendo ΔC el incremento en la longitud de la solución producido al seleccionar un nuevo vecino. Otra mejora posible sería cambiar la condición de parada (actualmente fija a 0.05). Esta condición de parada podría ser cualquier otro valor, o incluso uno dinámico.

Una vez implementadas estas mejoras, se han realizado una serie de pruebas con las diversas funciones de enfriamiento mencionadas en el apartado anterior (cuadro 9 para la función de enfriamiento base, cuadro 10 para la lineal, cuadro 11 para la logarítmica y cuadro 12 para la geométrica).

	Ciudades	**Óptimo**	**Mejor solución (frecuencia)**	**Media**	**Media - Óptimo (%)**
FIVE	5	19	19 (100)	19.0	0.0 (0.0)
P01	15	291	291 (14)	333.43	42.4 (14.581)
GR17	17	2085	2085 (1)	2282.77	197.77 (9.485)
FRI26	26	937	1042 (1)	1270.27	333.27 (35.568)
DANTZIG42	42	699	1180 (1)	1398.6	699.6 (100.086)
ATT48	48	33523	59636 (1)	70995.61	37472.61 (111.782)

Cuadro 9: Comparativa utilizando la función de enfriamiento proporcionada en *Sim. Annealing* con las mejoras de recalentamiento y *backtracking*

Tras analizar estos resultados se puede observar que son bastante similares a los obtenidos antes de realizar las modificaciones, por lo que en la práctica no se aprecian mejoras tras añadir *backtracking* y recalentamiento. Esto difiere con la teoría, ya que tanto *backtracking* como calentamiento proporcionan mejoras al algoritmo. Podría deberse a que, por la forma de la función objetivo de los *datasets* utilizados, el recalentamiento aplicado no sea suficiente para que el algoritmo llegue a explorar posibles soluciones que no se explorarían sin recalentamiento. También podría deberse a la elección incorrecta de hiperparámetros debido a que no se ha estudiado exhaustivamente cuales son los mejores, o a la aleatoriedad de los *datasets*.

	Ciudades	Óptimo	Mejor solución (frecuencia)	Media	Media - Óptimo (%)
FIVE	5	19	19 (98)	19.04	0.04 (0.211)
P01	15	291	291 (17)	328.28	37.28 (12.811)
GR17	17	2085	2085 (1)	2308.56	223.56 (10.722)
FRI26	26	937	1100 (1)	1296.2	359.2 (38.335)
DANTZIG42	42	699	1179(1)	1417.01	718.01 (102.720)
ATT48	48	33523	56182 (1)	71791.11	38268.11 (114.155)

Cuadro 10: Comparativa utilizando la función de enfriamiento lineal en *Sim. Annealing* con las mejoras de recalentamiento y *backtracking*

	Ciudades	Óptimo	Mejor solución (frecuencia)	Media	Media - Óptimo (%)
FIVE	5	19	19 (100)	19.0	0.0 (0.0)
P01	15	291	291 (21)	331.93	40.93 (14.065)
GR17	17	2085	2085 (1)	2240.01	155.01 (7.345)
FRI26	26	937	975 (1)	1189.51	252.51 (26.949)
DANTZIG42	42	699	980 (1)	1225.51	526.51 (75.323)
ATT48	48	33523	50692(1)	60866.15	27343.15 (81.565)

Cuadro 11: Comparativa utilizando la función de enfriamiento logarítmica en *Sim. Annealing* con las mejoras de recalentamiento y *backtracking*

	Ciudades	Óptimo	Mejor solución (frecuencia)	Media	Media - Óptimo (%)
FIVE	5	19	19 (100)	19.0	0.0 (0.0)
P01	15	291	291 (12)	333.54	42.54 (14.619)
GR17	17	2085	2095 (1)	2282.77	197.77 (9.485)
FRI26	26	937	1043 (1)	1285.16	348.16 (37.156)
DANTZIG42	42	699	1139 (1)	1391.81	692.81 (99.114)
ATT48	48	33523	57721 (1)	70030.78	36507.78 (108.904)

Cuadro 12: Comparativa utilizando la función de enfriamiento geométrica en *Sim. Annealing* con las mejoras de recalentamiento y *backtracking*

Para concluir, es interesante analizar el tiempo de cómputo de este algoritmo con las mejoras implementadas y compararlo sin dichas mejoras. Se observa (cuadro 13) que en cuanto a tiempo la diferencia entre ellos es minúscula. La razón de esto es sencilla. Si observamos las ecuaciones de las distintas funciones de enfriamiento (ecuaciones 2, 3 y 4) observamos que las variables T_0, α y T_k son fijas, haciendo que el valor de k (número de iteraciones) sea fijo también. La única función que no cumple esto es la función base (ecuación 5), la cual es recursiva y no depende de ningún valor fijo más que α. Aún así, los valores obtenidos para esta función son similares debido a que el ratio de calentamiento tiene que ser muy pequeño para que pueda llegar a la condición de parada. Para este experimento (cuadro 13), el aumento de la temperatura usando la función base ha sido un 1 % de la temperatura de la iteración anterior (utilizando un porcentaje mayor a este la temperatura aumenta tanto que nunca llega al valor umbral cuando $T_0 = 10$)

Implementando el recalentamiento no se nota mejoría con respecto a no implementarlo. Aún incrementando la temperatura, a la siguiente iteración el valor de la misma vuelve a un valor determinado por la iteración que se lleve a cabo. Se puede observar esto analizando las ecuaciones de enfriamiento propuestas, ya que ninguna de ellas (excepto en la base como se ha comentado en el párrafo anterior) utiliza la temperatura actual como variable.

	Base (ms)	Base mjr (ms)	Lineal (ms)	Lineal mjr (ms)	Log (ms)	Log mjr (ms)	Geom (ms)	Geom mjr (ms)
FIVE	7.631	7.465	7.071	7.278	16.424	15.868	7.895	7.821
P01	35.564	36.861	34.624	35.758	77.329	76.437	36.883	36.227
GR17	46.721	48.019	45.622	45.216	98.565	96.86	46.27	48.043
FRI26	110.297	106.866	101.012	98.128	223.143	227.894	109.030	108.377
DANTZIG42	418.074	418.536	396.121	389.535	863.298	867.292	417.891	409.113
ATT48	522.223	529.822	504.430	485.218	1111.313	1096.908	523.937	515.676

Cuadro 13: Comparativa de tiempos de las funciones de enfriamiento con y sin mejoras en *Sim. Annealing*

Utiliza todo el conocimiento que has adquirido hasta este punto, así como los resultados experimentales que tienes, para comparar *Hill Climbing* (y cualquier modificación) con *Simulated Annealing* (y cualquier modificación). ¿A qué conclusiones llegas?

A modo de conclusión, en el análisis realizado en los apartados anteriores se observa que el algoritmo *Hill Climbing* realiza una búsqueda local, analizando una parte pequeña del espacio de búsqueda y haciendo aleatorio el encontrar soluciones óptimas. Mediante la modificación de *Iterated Local Search* se consigue mejorar este problema ligeramente, aunque no de forma significativa.

Comparando lo anterior con *Simulated Annealing*, este realiza una búsqueda global que explora una mayor parte del espacio de búsqueda debido a la influencia de la temperatura comentada en apartados anteriores. A priori ayudaría a encontrar mejores soluciones que con *Hill Climbing*. Sin embargo, el uso de este algoritmo sin ninguna mejora ha producido unos resultados experimentales peores en los *datasets* de mayor tamaño. Tras la implementación de *backtracking* y recalentamiento no se han conseguido cambios significativos con respecto al algoritmo base, aunque se esperase aumentar la probabilidad de llegar a soluciones óptimas.

Para realizar una comparación más exhaustiva de estos dos algoritmos se han seleccionado las modificaciones que mejores resultados han dado. En el caso de *Hill Climbing* se ha optado por *Iterated Local Search* (cuadro 2, figura 1). Para *Simulated Annealing* se ha escogido la función logarítmica con la mejora implementada de *backtracking* y función de calentamiento (cuadro 11). Además de estos dos algoritmos, se ha decidido incluir *Simulated Annealing* con la función de enfriamiento lineal y los siguientes valores para los hiperparámetros: $T_0 = 100$, $\alpha = 0{,}999$ y $n = 1000$ (ecuación 2)(cuadro 14).

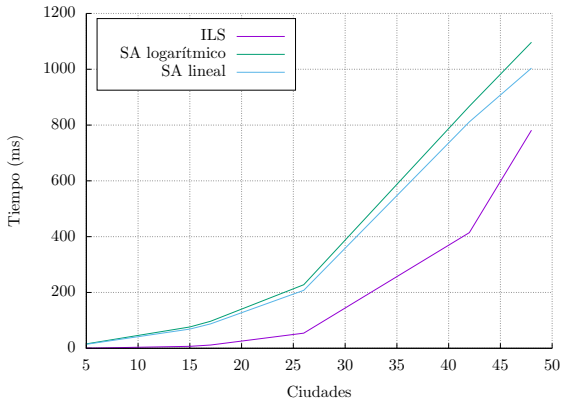
	Ciudades	Óptimo	Mejor solución (frecuencia)	Media	Media - Óptimo (%)
FIVE	5	19	19 (100)	19	0 (0.0)
P01	15	291	299 (1)	372.59	81.59 (28.037)
GR17	17	2085	2085 (4)	2193.51	108.51 (5.204)
FRI26	26	937	1122 (1)	1339.76	402.76 (42.983)
DANTZIG42	42	699	1297 (1)	1519.24	820.24 (117.34)
ATT48	48	33523	51016 (1)	51355.33	27832.33 (83.024)

Cuadro 14: Resultados con función de enfriamiento lineal, $T_0 = 100$, $\alpha = 0{,}999$ y 1000 iteraciones, y mejoras de *backtracking* y recalentamiento

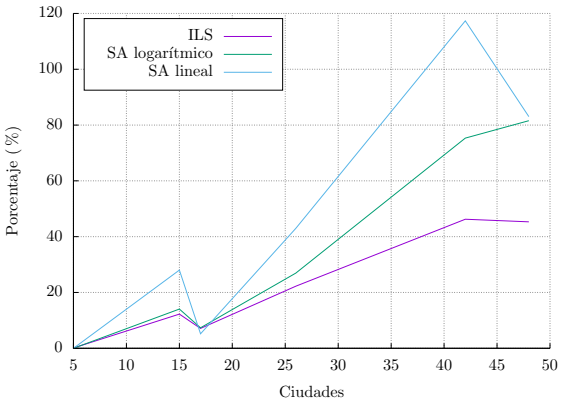
En cuanto a la complejidad del algoritmo se observa que los tres tienen un tiempo de cómputo exponencial (figura 10a), aunque *Simulated Annealing* tarda más debido al número de iteraciones que ejecuta. Esto se ve respaldado por la teoría expuesta a lo largo de este documento.

En cuanto a la variación obtenida (figura 10b) en los tres experimentos, el mejor resultado nos lo brinda *ILS*. Dicho resultado se explica a través de una mala elección de hiperparámetros, ya que *Simulated Annealing*, al ser una búsqueda global debería encontrar una mejor solución conforme se aumenta el

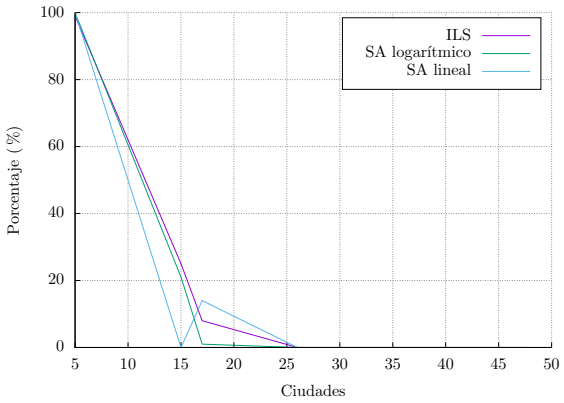
espacio de búsqueda. Cabe recalcar el comportamiento de *Simulated Annealing* con función lineal, el cual experimenta una mejoría notable cuando el óptimo global a encontrar es muy elevado. Sería interesante utilizar dicho algoritmo con un conjunto nuevo de *datasets* para verificar su comportamiento.



(a) Tiempo



(b) Variación



(c) Óptimo

Figura 10: Comparativa entre *ILS* y *SA* con función logarítmica

Por último, la frecuencia de óptimos (figura 10c) encontradas a lo largo de las 100 ejecuciones que compone el experimento, muestra que los tres algoritmos se comportan de manera similar. Al igual que con la variación, *Simulated Annealing* debería mejorar el resultado obtenido con *ILS*. Una vez más este comportamiento puede deberse a una mala elección de los hiperparámetros. De nuevo mencionar la actuación de *Simulated Annealing* con la función de enfriamiento lineal. Puede observarse cómo en los *datasets* con valores óptimos elevados mejora su resultado.

Como conclusión final, aunque *Simulated Annealing* debería ofrecer mejores resultados teóricamente, esto no es satisfecho por los experimentos realizados a lo largo de la práctica. Puede ser debido a los *datasets*, los cuales juegan un papel importante a la hora de elegir el algoritmo adecuado, y a la elección de hiperparámetros, que requieren de un estudio propio para el correcto funcionamiento del algoritmo debido a su importancia.

Referencias

[1] crankshaft, "A comparison of cooling schedules for simulated annealing (artificial intelligence)." http://what-when-how.com/artificial-intelligence/a-comparison-of-cooling-schedules-for-simulated-annealing-artificial-intelligence/.