Intel.ligència Artificial

(Laboratorio)

Práctica de Búsqueda Local

Documentación

Curso 2020/2021 2Q

- Jesús Benítez Díaz
- Mauro García Lorenzo
- Javier Rivera Hernández

UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

Índice:

1.Descripción del problema **3**
1.1 Un problema de búsqueda local 3
1.2 Elementos del problema 3
1.3 Restricciones de la solución 4

2. Estado y representación **5**
2.1 Espacio de búsqueda 5
2.2 Estado inicial 5
2.3 Estado final 5
2.4 Operadores 6
2.5 Factor de ramificación 6
2.6 Estructuras de datos 7
2.7 Atributos de la clase 8
2.8 Algoritmos generadores 8
2.8.1 Generador 1 8
2.8.2 Generador 2 8
2.9 Funciones heurísticas 9

3. Experimentación **10**
3.1 Estudio sobre el conjunto de operadores a elegir 10
3.2 Estudio sobre la estrategia para generar la solución inicial 12
3.3 Estudio de los parámetros utilizados en el Simulated Annealing 14
3.4 Variación de la proporción sensores-centros 16
3.6 Variación del coste de las conexiones en función del número de centros 18
3.7 Estudio sobre la variación de la ponderación en la heurística 22

4. Comparación de los dos algoritmos **24**
4.1 Coste temporal de la búsqueda 24
4.2 Bondad de las soluciones 25

Apéndice **26**

1.Descripción del problema

En esta práctica se nos plantea un problema relacionado con las redes de distribución, en el cual tendremos una serie de sensores que envían paquetes de información con el objetivo de que lleguen a los centros de datos.

Nuestro objetivo en esta práctica será tratar de crear una configuración (conectar un nodo a otro) que minimice la pérdida de información, así como el coste de la instalación. Todo esto siguiendo una serie de reglas que veremos a continuación.

1.1 Un problema de búsqueda local

Una vez analizado el problema, hemos decidido tratarlo mediante una búsqueda local por las siguientes razones:

En primer lugar porque lo que se nos pide es buscar una optimización para una solución del problema, es decir, no necesitamos ni recoger ni analizar datos del proceso para llegar a la solución. Las funciones heurísticas de nuestro problema no buscarán aproximar un coste para llegar a un estado, sino que calcularán el valor heurístico de este a partir de las propiedades que mencionaremos más adelante.

Además, como más adelante trataremos, el espacio de búsqueda aumenta de forma cuadrática en función de los sensores del problema (suponiendo que hay más sensores que centros), pero la complejidad temporal crece exponencialmente, lo cual supondría unas dimensiones demasiado grandes para aplicar un algoritmo de búsqueda explicado distinto.

1.2 Elementos del problema

Durante el análisis realizado en la primera semana hemos tenido en cuenta varios elementos. En primer lugar todos los datos que nos brinda el problema, como el número de sensores y centros (las dimensiones del espacio de soluciones aumentarán de forma exponencial en función del número de centros, como comentaremos más adelante); o la posición de estos en un área geográfica de 100 x 100 km, de la cual dependerá el coste de transmisión total del problema. También tendremos en cuenta la capacidad de almacenamiento y de transmisión de los mismos, siendo un factor determinante a la hora de escoger la configuración más óptima.

Otro de los elementos que tendremos en cuenta será la representación del estado sobre la cual profundizaremos más adelante, y las conexiones sensor-sensor sensor-centro que configuremos.

Por último mencionar las variables que valoraremos en la heurística, como es la cantidad de información transmitida y el coste de la transmisión.

1.3 Restricciones de la solución

Para que una solución sea considerada como válida debe cumplir una serie de requisitos.

- La red ha de conectar todos los sensores a los centros directa o indirectamente.
- No se pueden establecer más de 25 conexiones a un centro de datos o más de 3 a un sensor
- Cada sensor puede recibir de otros sensores y almacenar hasta dos veces el volumen de datos que captura
- Un sensor puede transmitir en una conexión como máximo tres veces el volumen de datos que captura
- No pueden existir ciclos en la red de distribución

2. Estado y representación

2.1 Espacio de búsqueda

En un principio hemos interpretado (erróneamente) que un sensor podía conectarse a un valor ilimitado de sensores y/o centros, por lo que el espacio de búsqueda sería sustancialmente superior al del problema real.

Asumiendo ya que un sensor tiene una sola conexión saliente, el espacio de búsqueda será las distintas combinaciones de n conexiones, siendo *n* el número de sensores.

Teniendo en cuenta lo mencionado en el apartado anterior el espacio de soluciones será de *n*(m + (n - 1))* (m centros de recogida de información). Para cada sensor las conexiones posibles (1 sola a la vez) serán todos los sensores menos él mismo y todos los centros.

Por lo tanto, a pesar de que el espacio de búsqueda sea cuadrático en función del número de centros más el número de sensores, se explorarán espacios repetidos con asiduidad, por lo que el coste temporal de aplicar algoritmos de búsqueda local como Simulated Annealing o Hill Climbing tenderá a tener una complejidad exponencial en función del número de centros y sensores.

2.2 Estado inicial

Un estado inicial será una solución que cumpla con las restricciones indicadas por el enunciado. Desde este estado se comenzará a explorar el espacio de soluciones. Es importante, por ello, que este sea lo más óptimo posible, pudiendo ahorrar tiempo de ejecución. Para asegurarnos de ello hemos implementado varios algoritmos generadores (unos más óptimos que otros), que más adelante comentaremos.

2.3 Estado final

Un estado final será una solución que cumpla con las restricciones indicadas por el enunciado que, a su vez, maximiza el valor de la solución según la heurística que hemos diseñado en el espacio explorado. Esta solución será, por lo tanto, un máximo local o global.

2.4 Operadores

Para explorar el espacio de soluciones hemos planteado utilizar 2 conjuntos de operadores distintos:

El primero de ellos consta simplemente de un operador, como es el de *swap*. En un principio recorreremos sensor por sensor enlazándolo con un sensor o centro aleatorio distinto de él mismo y del nodo al que estaba conectado originalmente. En caso de que la solución sea válida tras intercambiar la nueva conexión por la antigua, el cambio será efectuado. En caso contrario se deshará la modificación.

El segundo conjunto implementado es la combinación de 2 operadores. Uno de ellos es el *swap* descrito con anterioridad, mientras que el otro podríamos denominarlo "swap 2 a 2".

Consiste en intercambiar (si cumple con las restricciones del problema) las conexiones de 2 sensores, es decir, el elemento al que estaba conectado en primera instancia un hipotético sensor *a*, será ahora la conexión del sensor *b* y viceversa.

Ninguno de los dos operadores tiene condiciones de aplicabilidad, ya que ambos operadores pueden aplicarse en cualquiera de las soluciones del espacio de soluciones que se exploran. Aun así, ambos operadores tienen ciertos valores restringidos. En el caso del primer operador, no está permitido cambiar la conexión de destino por sí mismo, es decir, que se conecte a sí mismo. En el caso del segundo operador no podemos hacer swap de un sensor con sí mismo o con otro sensor al cual esté conectado (tanto por una conexión de entrada o salida).

Hemos escogido el primer conjunto de operadores porque es versátil y nos permite explorar todo el espacio de soluciones.

En el caso del segundo conjunto de operadores, porque el doble swap nos permite hacer cambios expandiendo menos nodos y puede suponer una mejora en la eficiencia del programa. Además en combinación con el cambio de conexión de salida nos permite explorar todo el espacio de soluciones.

2.5 Factor de ramificación

Según nuestra interpretación del problema y los operadores que utilizaremos, desde un sensor con el primer conjunto de operadores podremos establecer $n - 2 + m$ conexiones distintas (solo una simultánea), es decir, todos los centros del problema y todos los sensores menos él mismo y al que estaba conectado originalmente (si ya estaba enlazado a un centro la ecuación no variará).

Tenemos entonces que operando swaps entre conexiones para un sensor podremos acceder a *n - 2 + m* estados nuevos, por lo que nuestro factor de ramificación será de *n * (n - 2 + m)*.

En el caso del segundo conjunto de operadores formados por el cambio de conexión de destino y el swap de conexión de destino entre dos sensores, tenemos que el operador de cambio de conexión de destino nos permite acceder a n * (n - 2 + m) como hemos explicado antes.

Por otro lado, el operador de swap de conexión de destino entre dos sensores permite cambiar cualquier nodo por otro que no sea sí mismo, o a los que esta conectados, ya que al hacer el swap de la conexión de destino acabaríamos con un sensor conectándose a sí mismo, lo cual va en contra de la representación de nuestro problema. Por lo tanto el número de estados a explorar es n * (n-2-g), siendo g el número de sensores que se conectan a otro sensor.

El factor de ramificación total del conjunto de operadores 2 es la suma de los factores de ramificación de cada operador, ya que no existen estados en la intersección de los conjuntos de estados generados de cada operador. Por lo tanto el factor de ramificación total es O(n * (n - 2 + m) + n * (n-2-g)).

2.6 Estructuras de datos

Para almacenar los datos de los estados del problema hemos decidido utilizar 3 vectores:

- Vector de conexiones: el tamaño de este será igual al número de sensores del problema. Cada sensor y centro tendrá asociado un valor numérico, por lo que en cada posición se almacenará el valor del elemento al que está asociado dicho sensor.

  El valor numérico asociado es un índice i tal que: 0 <= i < Num_Centros + Num_Sensores, siendo un sensor si 0 <= i < Num_sensores y centro si se encuentra entre Num_sensores <= i < Num_Centros + Num_Sensores

- Vectores estáticos: habrá 2 vectores estáticos que almacenarán los datos del problema (uno para sensores y otro para centros) con datos de posición y capacidad. Esto reduce el coste espacial abstrayendo a la clase información que no es única de cada estado.

La complejidad espacial de esta representación es lineal en función del número de sensores del problema, por lo tanto es una representación eficiente para los estados. Además, como todos los sensores tienen exclusivamente una conexión de salida, nos ahorramos expresar las conexiones como una matriz/lista de adyacencia; a su vez, esta representación nos permite seguir usando la ordenación topológica para validar el estado y obtener su información y demás datos.

2.7 Atributos de la clase

Además de los vectores mencionados con anterioridad emplearemos variables estáticas, como el número de sensores y centros, la semilla para generar los valores, el tipo de generador de soluciones iniciales, el conjunto de operadores o el algoritmo a utilizar. Por último incluiremos las variables que nos permiten evaluar la solución (cantidad de información transmitida, coste total y centros utilizados).

2.8 Algoritmos generadores

2.8.1 Generador 1

Mientras haya centros con menos de 25 conexiones entrantes iremos conectando los sensores a estos (distribuyendolos equitativamente). En caso de que haya sensores sin conectar, estos se conectarán ordenadamente a los que ya tenían una conexión establecida (primero a los que están a distancia 1 de un centro, luego a los de distancia 2...). El coste de este algoritmo será de n, siendo n el número de sensores problema.

La razón por la cual hemos implementado este algoritmo es debido a que hemos observado que, si conectamos un sensor directamente a un centro, en ningún caso se producirá una pérdida de información, además es una forma eficiente de crear una solución.

2.8.2 Generador 2

Es el algoritmo más sencillo posible. Numeraremos los sensores y los conectaremos uno al consecutivo (por el orden del valor numérico establecido). Tras finalizar las conexiones entre los sensores, asociaremos el último al primer centro (también por el valor numérico asociado). Hemos implementado este algoritmo para aprovecharnos de la sencillez y eficiencia del mismo (eficiencia del generador, no del problema completo).

2.9 Funciones heurísticas

A la hora de diseñar nuestra heurística debemos tener en cuenta los dos siguientes factores:

- El coste de la instalación, el cual nos piden que sea mínimo.
- La información que se transmite a los centros, la cual debe ser maximizada.

Siguiendo esto, construimos la siguiente fórmula heurística:

$$costeTotalInstalación/10000.0 \; - \; infoTransmitida$$

El algoritmo de la clase AIMA de Hill Climbing trata de minimizar la función heurística, Simulated Annealing siempre trata de minimizarlo.

Por eso mismo la información transmitida está negada, ya que así conseguimos que su aportación a la función sea mínima cuando la información que reciben los centros es máxima. Análogamente, el coste de instalación es positivo porque queremos minimizarlo, por lo tanto su aportación a la función heurística es mínima cuando este es mínimo.

La ponderación que aparece en la función, es decir, 0.0001 * coste de instalación, se debe a que el coste de instalación, es 4 órdenes de magnitud más grande que el coste de instalación, por lo que así conseguimos igualar la implicación que tienen ambas en la función heurística.

Por lo tanto, el efecto que tiene esta función heurística es maximizar la información minimizando el coste. Se observa que tiene más prioridad maximizar la información que minimizar el coste, ya que debido a la ponderación del coste, el cambio de una conexión que genere una mejora en la información va a tener un impacto mayor que el impacto que genera un cambio en el coste.

3. Experimentación

3.1 Estudio sobre el conjunto de operadores a elegir

Observación	El conjunto de operadores que usemos a la hora de generar sucesores influye en la solución final.
Planteamiento	Usamos dos conjuntos de operadores: -El primero será un swap entre 2 sensores -El segundo será el anterior combinado con un swap 2 a 2 (comentados con anterioridad).
Hipótesis	Ambos operadores dan el mismo resultado (H0) o uno de los dos da un mejor resultado.
Método	- Generamos 10 semillas aleatorias - Ejecutaremos 2 experimentos por cada semilla, uno por conjunto de operadores. - Experimentamos con 100 sensores y 4 centros. - Compararemos resultados para determinar qué conjunto funciona mejor

Después de haber realizado los experimentos, y al ver que los datos son relativamente pocos hemos plasmado los resultados en la siguiente tabla (Figura 1):

	Coste Heurístico		Nodos Expandidos		Tiempo(ms)	
semilla(C/S)	Conjunto 1	Conjunto 2	Conjunto 1	Conjunto 2	Conjunto 1	Conjunto 2
80-80	253	252,8	105	116	3822	7883,4
450-210	248,3	249,5	114	118	4149	8058
289-542	250,3	251,3	96	116	3659,2	7942,6
855-452	244,5	245,5	112	121	4146,2	9309,2
56-746	248,5	249,8	97	139	3957	9820,4
1022-458	245,6	247,7	106	135	4172,6	9448,4
120-88	254,4	254,8	112	127	4217,8	8927,8
1525-45	253,4	254,1	104	112	4036,2	8234,6
54-888	248,7	248,8	111	115	4148	7968,2
1294-458	253,5	254,4	108	121	3964,8	8409

Figura 1 : Datos de coste heurístico, nodos expandidos y tiempo de ejecución para ambos conjuntos

Tras analizar los resultados del experimento (Figura 1) observamos que la hipótesis no se cumple. Si nos fijamos en el coste heurístico y en los nodos expandidos, podemos ver que son mejores los resultados para el segundo conjunto de operadores. Este es, de media, 0,9 puntos superior al primero en el coste heurístico, y 13,5 en los nodos expandidos (ver Figura 2).
Sin embargo, comprobamos que el tiempo de ejecución es mucho mayor para el conjunto 2, tal y como podemos comprobar en la siguiente tabla, una diferencia de más del doble.

	Coste	Nodos	Tiempo
Con1	249,97(3,4309)	106,5(6,2227)	4027,28(178,7277)
Con2	250,87(3,1347)	122(8,9567)	8600,16(717,1084)

Figura 2 : Datos de media y desviación media de la gráfica de la Figura 1

Hemos creado un box plot (Figura 3) para observar mejor la diferencia en el tiempo de ejecución.

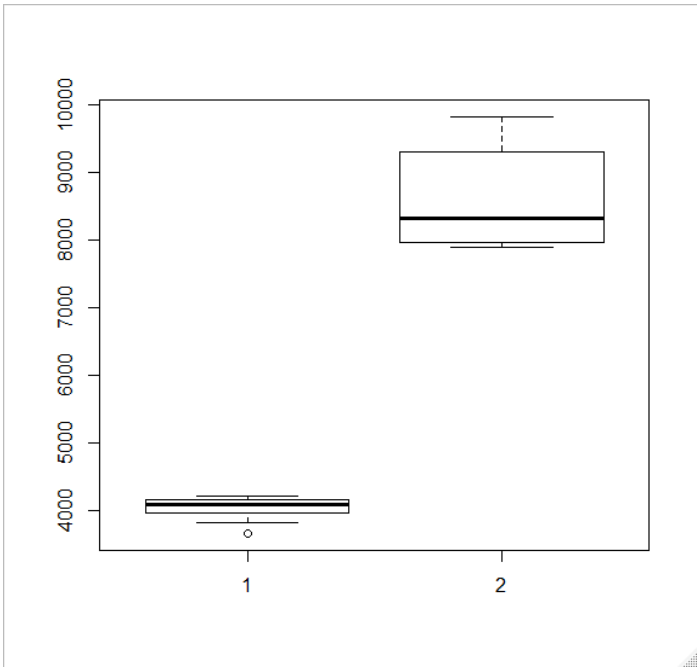


Figura 3 : BoxPlot del tiempo de ejecución para ambos conjuntos

Tomando entonces la diferencia en coste heurístico y el número de nodos explorados por el tiempo de ejecución, podemos concluir que la observación del experimento se cumple, y que la hipótesis nula planteada no es cierta (el primer conjunto de operadores obtiene un mejor resultado).

3.2 Estudio sobre la estrategia para generar la solución inicial

Observación	La estrategia que usemos a la hora de construir la solución inicial influye en el tiempo para encontrar una solución válida
Planteamiento	Usaremos dos estrategias de generación (comentadas en el punto 2.3)
Hipótesis	Ambas estrategias generan una solución con la misma calidad (H0) o una de las dos es mejor que la otra
Método	- Generamos 10 semillas aleatorias - Ejecutaremos 2 experimentos por cada semilla, uno por cada estrategia de generación - Experimentamos con 100 sensores y 4 centros. - Compararemos resultados para determinar qué generador crea una solución inicial mejor

Para la realización de este experimento hemos utilizado el conjunto de operadores que mejor resultado dio en el anterior apartado, es decir, el primero. Igual que en el experimento anterior, al ver que los datos son relativamente pocos, hemos plasmado los resultados en la siguiente tabla (Figura 4):

	Coste Heurístico		Nodos Expandidos		Tiempo(ms)	
semilla(C/S)	Generador 1	Generador 2	Generador 1	Generador 2	Generador 1	Generador 2
80-80	252,5	252,6	105	135	3340	4262
450-210	248,3	248,6	114	135	3801	4225
289-542	250,3	250,9	96	131	3080	4185
855-452	244,5	245,1	112	129	3729	4059
56-746	250,4	250,1	103	132	3316	4461
1022-458	245,6	246,8	106	135	3309	4197
120-88	254,4	254,2	112	134	3679	4146
1525-45	253,4	253,6	104	141	3415	4397
54-888	248,7	248,3	111	130	3632	4263
1294-458	253,5	253,6	108	130	3529	4045

Figura 4 : Datos de coste heurístico, nodos expandidos y tiempo de ejecución para ambos generadores

Teniendo en cuenta los datos mostrados podemos concluir que tanto la observación como la hipótesis alternativa se cumplen. A pesar de que el coste heurístico no sea un factor determinante a la hora de decantarnos por un generador u otro, el tiempo de ejecución sí. La diferencia en este campo es, en algunos casos, superior a 1 segundo (de media 741 puntos en la siguiente tabla), lo que se convierte en una diferencia demasiado grande desde el punto de vista computacional.

Otro aspecto revelador es el número de nodos expandidos, siendo esto también mayor para todas las pruebas con el generador 2 (Figuras 4 y 5). Deducimos entonces, que el espacio explorado es mayor para este y, por lo tanto, que el estado inicial generado se encuentra más lejos de la solución final.

	Coste	Nodos	Tiempo
Gen1	250,16(3,3929)	107,1(5,4457)	3483(228,5267)
Gen2	250,38(3,1411)	133,2(3,5839)	4224(131,9360)

Figura 5 : Datos de media y desviación media de la tabla de la Figura 4

Apoyándonos en la Figura 6, que representa el Box Plot de los tiempos de ejecución del experimento, podemos observar que, a pesar de que el espectro de valores posibles es mucho menor para el segundo generador (entorno a 400 ms de intervalo frente a los casi 800 del primero), la mediana es mucho menor para este último (del orden de unos 750 ms de diferencia).

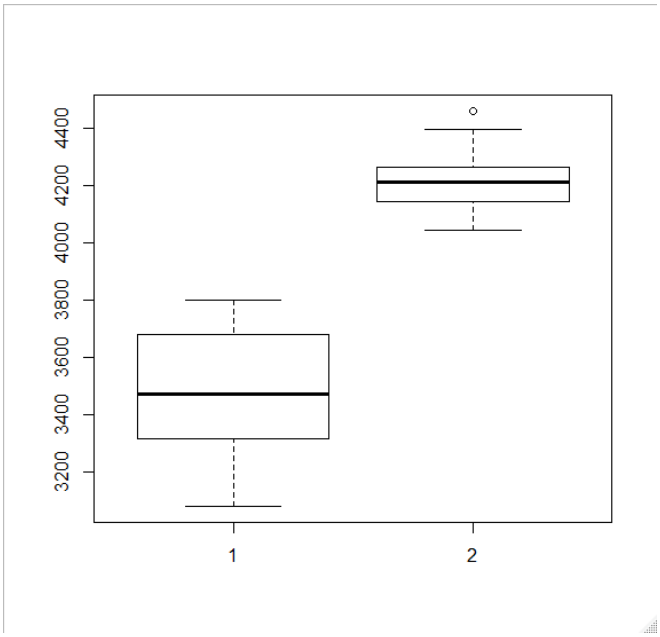


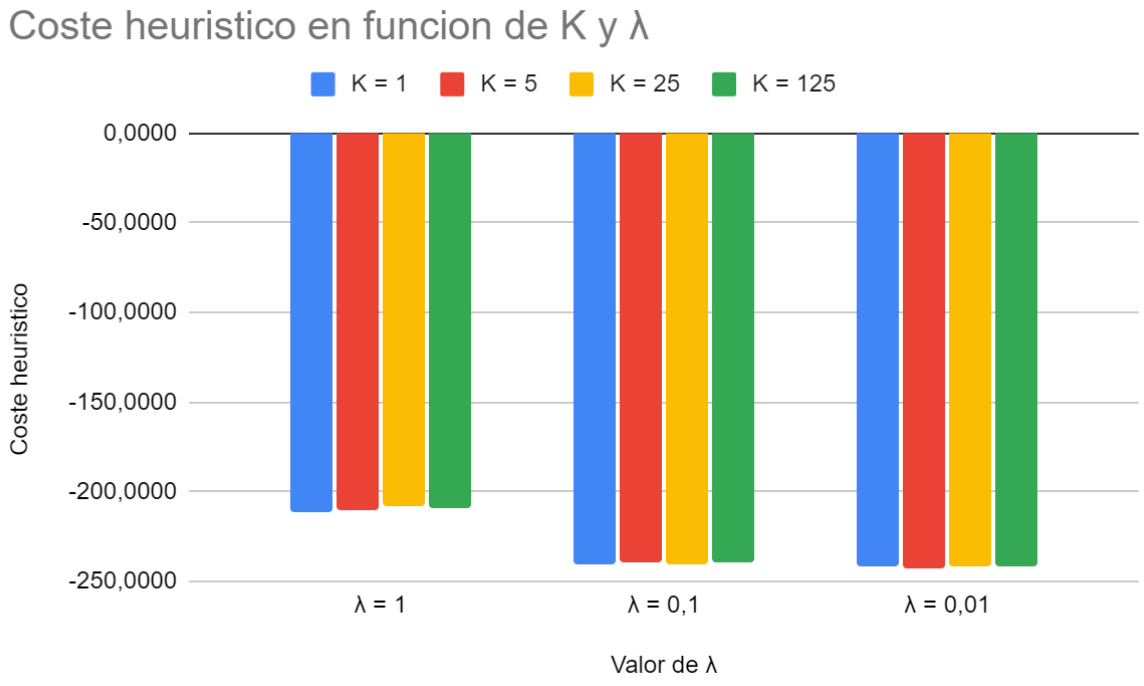
Figura 6 : BoxPlot del tiempo de ejecución para ambos generadores

Tras la realización de la experimentación podemos concluir que la observación del experimento se cumple, y que la hipótesis nula planteada no es cierta (la primera estrategia para generar la solución obtiene un mejor resultado).

3.3 Estudio de los parámetros utilizados en el Simulated Annealing

Variación de las variables de k y lambda

Observación	Hay unos valores λ y K que nos permiten obtener un mejor coste heurístico y/o tiempo
Planteamiento	Elegimos combinaciones de K y λ distintas y observamos el coste heurístico del estado final y el tiempo que tardamos en ejecutarlo con dichos parámetros. El número de iteraciones será 10000 para homogeneizar el impacto de los parámetros K y λ, y se incrementará la temperatura cada 10 iteraciones.
Hipótesis	No existe diferencia de coste heurístico y/o tiempo al cambiar los parámetros (H0) o si que la hay.
Método	- Generamos 6 semillas aleatorias - Ejecutaremos 12 experimentos por semilla, uno para cada combinación de K = 1, 5, 25, 125 y λ = 1, 0.1, 0.01. - Ejecutaremos 2 experimentos por cada semilla y cada par de λ y K - Experimentamos con 100 sensores y 4 centros. - Compararemos resultados obtenidos



Coste heurístico en función de K y λ				
	K = 1	K = 5	K = 25	K = 125
λ = 1	-211,0502	-210,1332	-208,5252	-209,5120
λ = 0,1	-240,0788	-239,8196	-240,5632	-239,9456
λ = 0,01	-241,9624	-242,7284	-242,0173	-241,3709

Figura 7 : datos de coste heurístico en función de K

Observamos en la Figura 7 que para λ = 0.1 o 0.01 el coste heurístico obtenido es mejor, pero el tiempo también es mucho más grande que para el caso con λ = 1 como se observa en la Figura 7. Los valores de K no generan cambios apreciables, excepto en el tiempo de ejecución (Figura 8) para K = 5, donde con λ = 1 o 0.1 se obtiene un tiempo mejor, mientras que con λ = 0.01 con K = 5 el tiempo de ejecución aumenta drásticamente. λ = 1 funciona más rápido cuando K es 125. λ = 0.01 funciona más rápido cuando K = 25.

Por lo tanto, podemos decir que si buscamos rapidez a la hora de ejecutar usaremos λ = 1 y K = 125. En cambio, si queremos obtener el mejor coste heurístico a cambio de un mayor tiempo de ejecución usaremos λ = 0.1, ya que se obtienen resultados en la heurística similares a λ = 0.01 pero con un tiempo de ejecución menor. Se obtiene un menor coste de ejecución con λ = 0.1 para K = 5. Afirmamos, por lo tanto, que la observación es correcta al refutar la hipótesis nula.

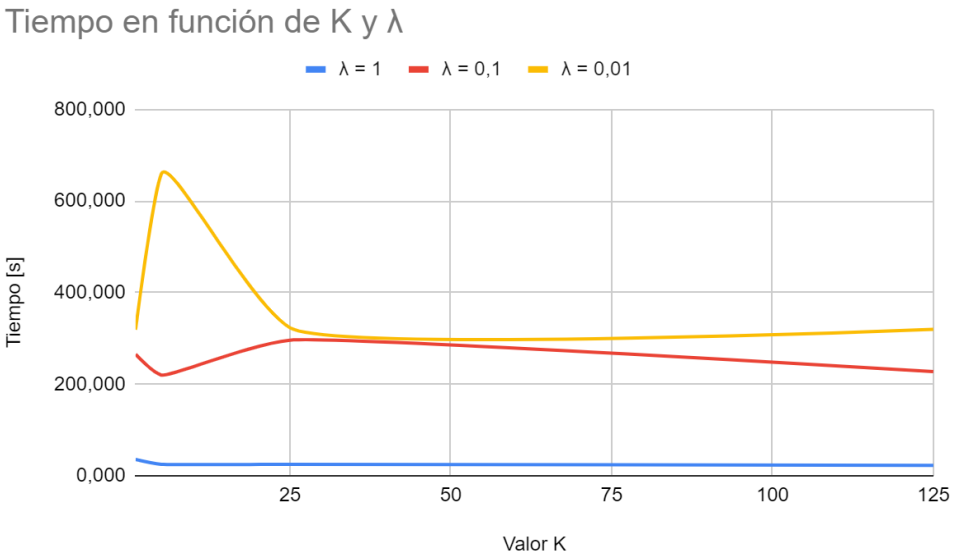


Figura 8 : gráfica que representa el coste heurístico en función de K y λ

3.4 Variación de la proporción sensores-centros

Observación	La proporción de centros-sensores influye en el tiempo de ejecución
Planteamiento	Usamos diferentes proporciones centros-sensores
Hipótesis	La proporción a la hora de medir el tiempo de ejecución no influye en el tiempo de ejecución (H0) o habrá proporciones que si
Método	- Generamos 8 semillas aleatorias - Ejecutaremos 1 experimentos por cada semilla - Empezamos experimentando con 100 sensores y 4 centros. - Iremos incrementando de 2 en 2 hasta que se vea la tendencia - Compararemos resultados para determinar qué conjunto funciona mejor

Una vez realizado el experimento, hemos decidido representar la evolución del tiempo de ejecución respecto a la proporción en la siguiente gráfica.

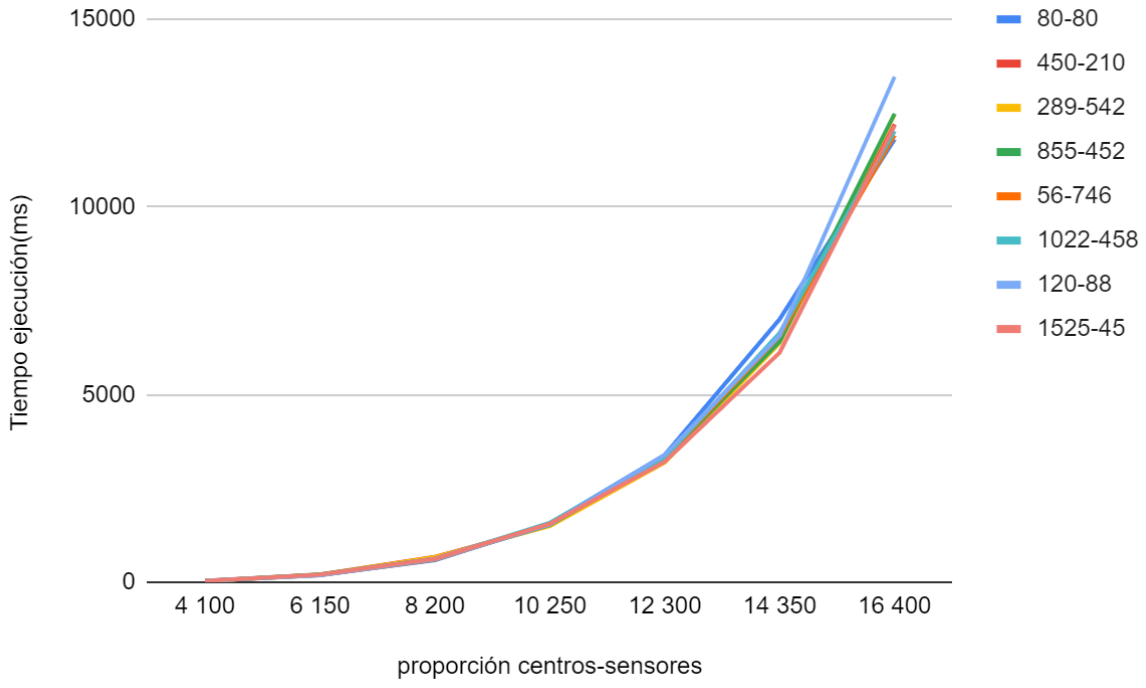


Figura 9 : gráfica que representa la variación del tiempo de ejecución respecto a la proporción centros-sensores

Podemos observar una clara tendencia creciente en el tiempo de ejecución (figura 9). Hasta que la proporción llega a 10 centros y 250 sensores el tiempo para encontrar una solución es unas 3 o 4 veces mayor que el anterior, y una vez que sobrepasa esa proporción se comienza duplicar. En este experimento no hemos podido realizar la simulación para las 10 semillas

debido al elevado tiempo que conlleva una única ejecución cuando las proporciones sobrepasan esta proporción.

La gráfica toma esta forma debido, principalmente, al crecimiento del espacio de soluciones a medida que crece el número de sensores y centros. Tal y como hemos mencionado con anterioridad, el factor de ramificación es de *n * (n - 2 + m)* (siendo *n* el número de sensores y *m* el de centros), por lo que a medida que crecen estos, el tiempo de ejecución tiende a ser exponencial.

Tras analizar los resultado concluimos que nuestra observación se cumple, y la hipótesis nula no es cierta. La proporción sensores-centos influye en el tiempo de ejecución.

3.5 Estudio sobre el uso de los centros de datos

Observación	Siguiendo la proporción centro/sensor 4:100 todos los centros se utilizan.
Planteamiento	Ejecutamos el programa con diferentes valores de centros y sensores que sigan el ratio 4:100.
Hipótesis	No se utilizan todos los centros de datos y existe una proporción centros de datos/sensores para aquellos que no se utilizan (H0) o todos los centros se aprovechan.
Método	- Ejecutamos para 6 semillas aleatorias cada par de valores. - Ejecutamos 5 veces para cada semilla. - Elegiremos los siguientes pares: 4/100, 6/150, 8/200. - Ejecutamos el experimento con Hill Climbing y Simulated Annealing

Se observa que sea cual sea el par de número de centros/número de sensores, todos los centros son aprovechados. Eso se debe, deducimos, a dos factores:

En primer lugar, un centro puede recibir 25 conexiones, por lo tanto a expensas de la optimización en coste e información, a cada centro le pertocan 25 sensores exactamente si estos se conectarán directamente al centro (100/25 = 4).

En segundo lugar, debido al generador de solución inicial 1, que lo que hace es conectar cada sensor a un centro mientras no se superen las 25 conexiones de sensores a centros, ya que así se consigue una solución fácil de construir en la cual no se pierde información de ningún sensor. Esto condiciona que el espacio explorado por el algoritmo de búsqueda tiendan a ser soluciones en las cuales todos los sensores se encuentren conectados directamente a centros a excepción de aquellos que se han conectado a otros sensores para optimizar el coste.

Con Simulated Annealing también obtenemos que se utilizan todos los centros.

Por lo tanto, es bastante verosímil el resultado obtenido, ya que todos los sensores tienden a repartirse 1:25 con cada centro. Tenemos entonces que la hipótesis nula es falsa, y en consecuencia, la observación es cierta.

3.6 Variación del coste de las conexiones en función del número de centros

Hill Climbing

Observación	El número de centros influye en el coste de instalación
Planteamiento	Fijamos los sensores en 100 y aumentaremos el número de sensores
Hipótesis	Aumentar el número de centros no influye en el coste de instalación (H0) o influye.
Método	- Generamos 10 semillas aleatorias - Ejecutaremos 3 experimentos por cada semilla, uno por cada factor que vamos a medir (coste de instalación, centros utilizados y tiempo de ejecución) - Empezamos experimentando con 100 sensores y 4 centros - Iremos incrementando de 2 en 2 los centros hasta llegar a 10 - Compararemos resultados para determinar qué conjunto funciona mejor

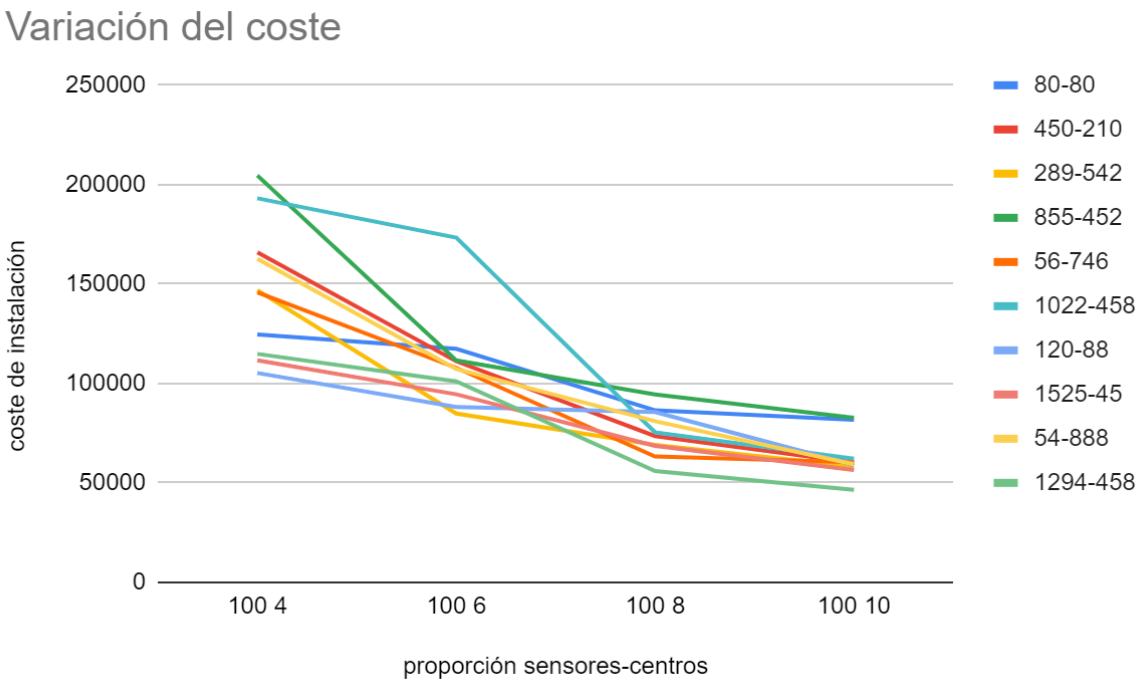


Figura 10 : variación del coste de las conexiones en función de la proporción sensores-centros

Como podemos observar con claridad en la gráfica anterior (Figura 10), existe una clara tendencia descendente del coste de la transmisión a medida que crece el número de centros de datos. Esto es debido a lo mencionado en el punto anterior; tanto el algoritmo generador como la heurística planteada propician el uso de todos los centros. Por eso si se añaden más centros tenemos mayor de posibilidad de encontrar centros más cercanos a los sensores y por tanto que se mejore el coste de instalación.

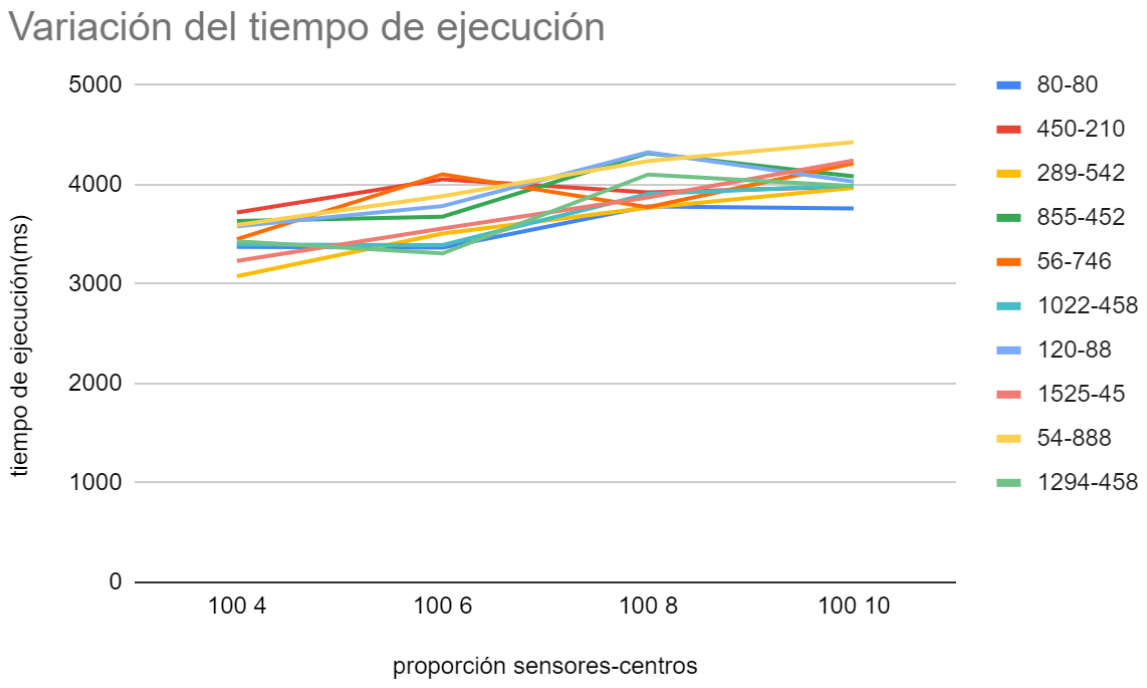


Figura 11 : variación del tiempo de ejecución de las conexiones en función de la proporción sensores-centros

En cuanto al tiempo de ejecución se aprecia una mínima tendencia creciente.

Tras recoger datos que reflejaban el número de centros que se empleaban en función del número de centros del problema, hemos observado que en todos los casos todos los centros recibían al menos una conexión. Esto se debe a la posible excesiva ponderación en nuestra heurística de la información transmitida, lo que provoca que siempre primará la conexión sensor-centro, aprovechándose así de una información transmitida máxima.

Podemos concluir que la observación se cumple y, por contra, la hipótesis nula no.

Simulated Annealing

Observación	El número de centos puede influir en el coste de instalación
Planteamiento	Fijamos los sensores en 100 y aumentaremos el número de sensores.
Hipótesis	Aumentar el número de centros no influye en el coste de instalación(H0) o influye.
Método	- Generamos 2 semillas aleatorias - Ejecutaremos 5 experimentos por cada semilla, uno por cada factor que vamos a medir (coste de instalación, centros utilizados y tiempo de ejecución) - Empezamos experimentando con 100 sensores y 4 centros - Iremos incrementando de 2 en 2 los centros hasta llegar a 10 - Usaremos K = 1, λ = 0.1, 10000 iteraciones totales, y 10 iteraciones por incremento de temperatura. - Compararemos resultados para determinar qué conjunto funciona mejor

Con Simulated Annealing se obtienen resultados muy similares a Hill Climbing tal y como se observa en las Figuras 12 y 13.

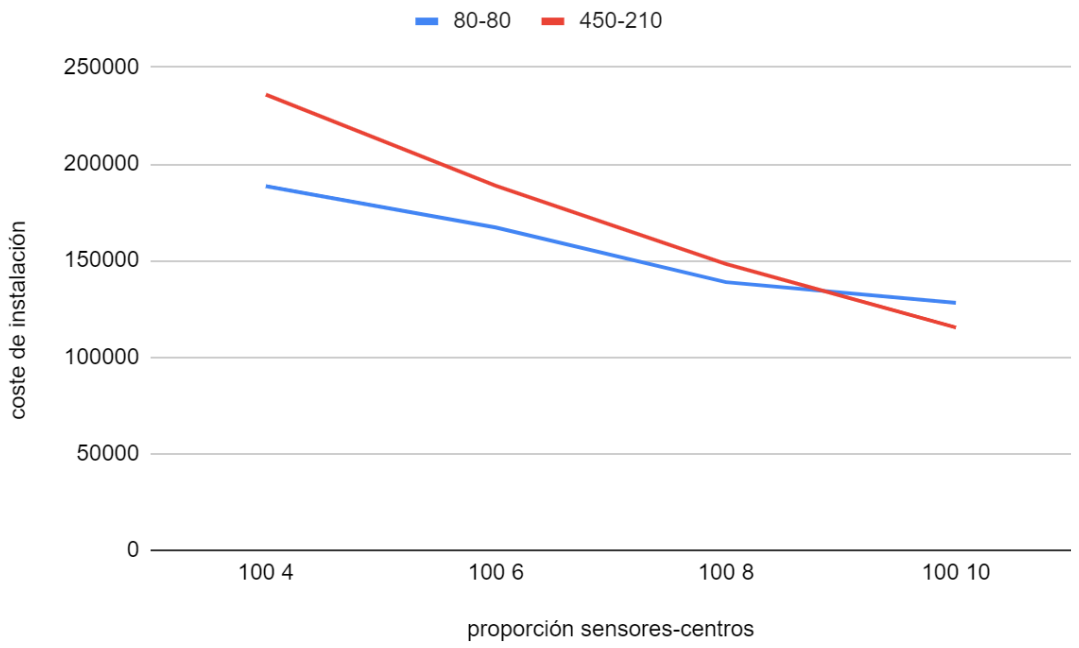


Figura 12 : Variación del coste de instalación respecto a la proporción sensores-centros

El coste observa una tendencia descendente como con Hill Climbing. Creemos que se debe a la misma causa, ya que al haber más centros aumenta la cantidad de conexiones posibles a estos y se puede encontrar con mayor posibilidad uno con menor coste de instalación por la distancia a la que se encuentra.

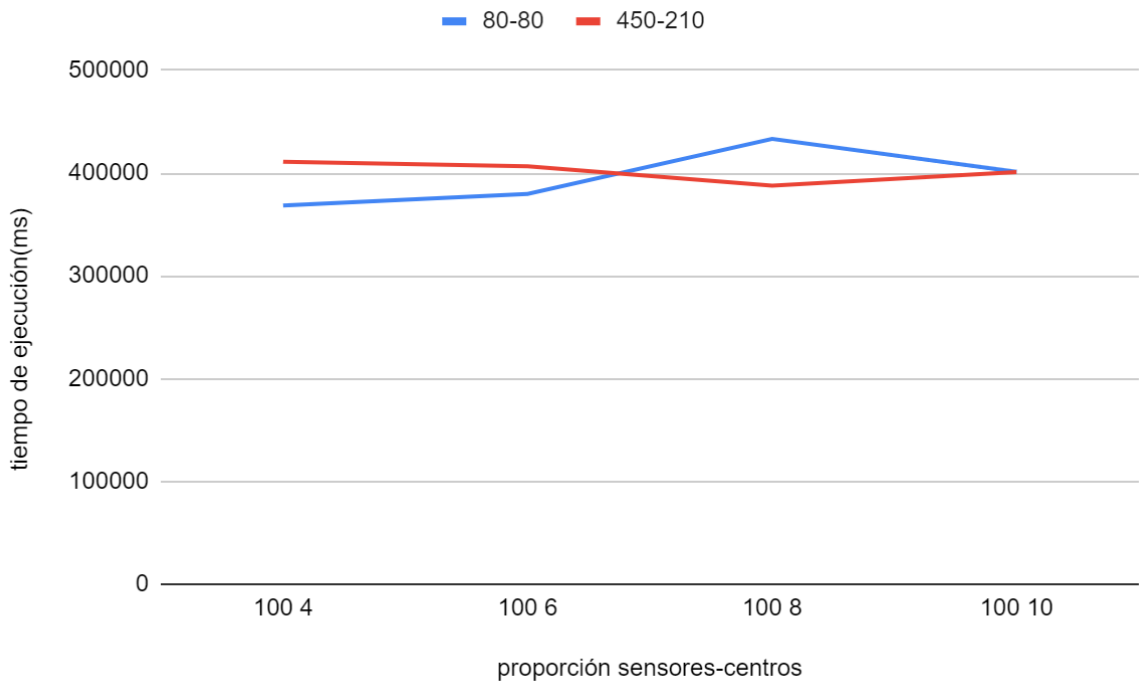


Figura 13 : Variación del tiempo de ejecución respecto a la proporción sensores-centros

En el tiempo de ejecución se observa una tendencia creciente en la semilla 80-80 y una tendencia decreciente en la semilla 450-210. No hemos ejecutado más semillas debido al alto tiempo de computación que requiere ejecutar varias veces Simulated Annealing para una semilla, pero suponemos que de hacerlo se observaría una tendencia creciente en general como en Hill Climbing.

Por lo tanto refutamos la hipótesis nula y la observación de que el número de centros afecta al tiempo de ejecución y el coste de instalación es cierta.

3.7 Estudio sobre la variación de la ponderación en la heurística

Observación	La ponderación de la Información influye en el coste de red y en el tiempo de ejecución
Planteamiento	Usaremos 2 centros y 100 sensores
Hipótesis	Cambiar la ponderación de la información transmitida no influirá en el coste de la red/tiempo de ejecución (H0) o sí que influirá y puede que sea mayor o menor.
Método	- Generamos 10 semillas aleatorias - Ejecutaremos 2 experimentos por cada semilla, uno para el tiempo de ejecución y otro para el coste de conexión de la red - Siendo la función heurística CosteTotal/X - Info*Y - Empezaremos con X = 100, Y = 10

Tras la ejecución de los experimentos hemos plasmado los resultados en las gráficas de las figuras 14 y 15. En cuanto a la primera cuestión que plantea este experimento, vemos una clara tendencia decreciente en el tiempo de ejecución de casi un segundo y medio a medida que aumentamos la ponderación de la información.

La única explicación que le vemos a esta tendencia decreciente es que es más fácil optimizar la información que el coste, por eso al darle cada vez más importancia a la información, deja de tener en cuenta el coste y llega más rápido a una solución óptima en cuanto a información.

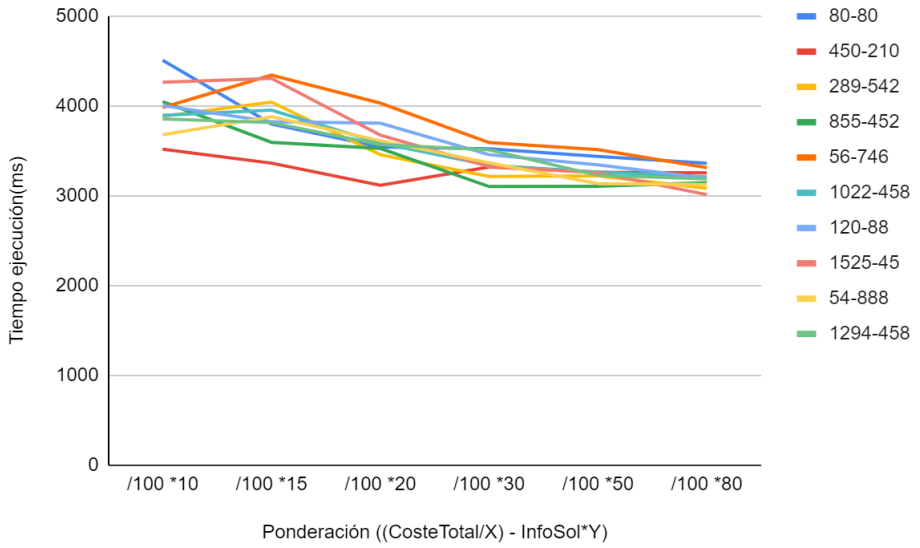


Figura 14 : Variación del tiempo de ejecución respecto a la ponderación de InfoSol

Todo esto que acabamos de comentar tiene relación directa con el incremento del coste. En la figura Y vemos el crecimiento del coste a medida de que incrementamos la ponderación, pero hasta un cierto punto, en el cual se estabiliza y no crece más. Este punto coincide siempre con la llegada del algoritmo a una solución óptima (máxima información posible).

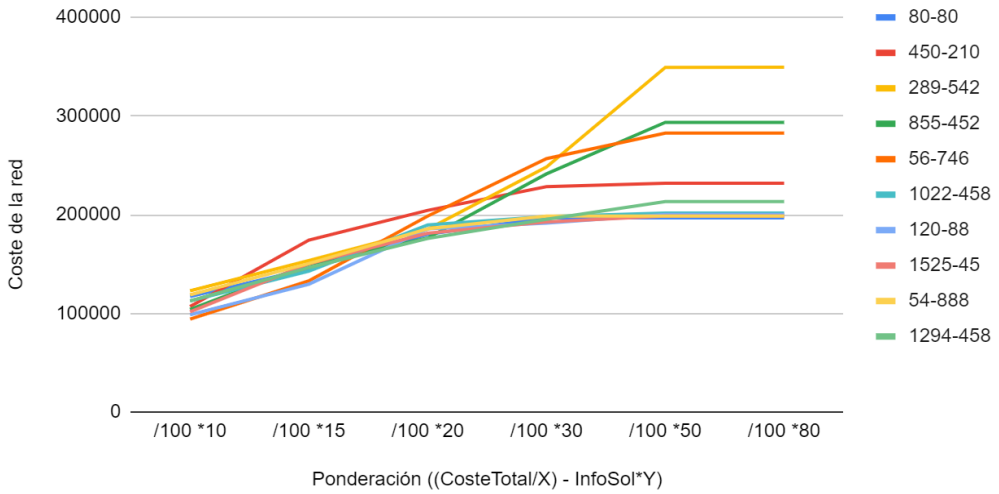


Figura 15 : Variación del tiempo de ejecución respecto a la ponderación de InfoSol

Por último, podemos concluir que la hipótesis nula no es cierta y que se cumple nuestra observación. El tiempo de ejecución ha disminuido y el coste de red ha aumentado considerablemente.

4. Comparación de los dos algoritmos

Después de haber realizado los experimentos anteriores podemos proceder a comparar los algoritmos Hill Climbing y Simulated Annealing para concluir cuál de los dos es más eficiente para el problema tratado en este trabajo. Para comparar ambos algoritmos nos vamos a centrar en dos puntos de estos, el coste temporal de la búsqueda y la bondad de las soluciones. Como hemos observado en el experimento 3, después de haber explorado los valores de los parámetros utilizados en el Simulated Annealing, hemos concluido que los valores para K y λ deben ser 5 y 0,1 respectivamente, para maximizar la eficiencia del algoritmo.

4.1 Coste temporal de la búsqueda

Como hemos visto en el apartado 3.6 (experimento 6), existe una gran diferencia en cuanto al tiempo de ejecución de un algoritmo al otro, obteniendo Hill Climbing resultados temporales de dos órdenes de magnitud menores.

En pro de evidenciar esta desigualdad hemos realizado un pequeño experimento ejecutando ambos algoritmos para unas mismas semillas que plasmaremos en la siguiente tabla:

	tiempo(s)	
semilla	Hill Climbing	SA
1532	3,157	221
5695	3,244	222
4334	2,809	221
6552	2,908	220
1987	3,2	216
1181	3,155	219
Media	3,078833333	219,8333333

Figura 16: Comparación tiempos de ejecución

Una vez más observamos la diferencia entre ambos algoritmos de búsqueda, obteniendo una desigualdad de 216 segundos de media. Podemos concluir entonces que, en cuanto al tiempo de ejecución el primer algoritmo obtiene, sin lugar a dudas, unos mejores resultados en términos de tiempo.

4.2 Bondad de las soluciones

Para comparar este aspecto de la solución hemos recogido los resultados de coste heurístico producto del experimento anterior.

Una vez más comprobamos cómo el algoritmo de Hill Climbing obtiene mejores resultados (una diferencia de 8 puntos en la media), como podemos ver en la siguiente tabla (Figura X).

	Heurística	
semilla	Hill Climbing	SA
1532	248,1	240,5067
5695	251,8	244,7352
4334	232,6	223,6013
6552	246,1	240,9794
1987	256,8	248,1555
1181	249,1	240,9397
Media	247,4166667	239,8196

Figura 17: Comparación de las heurísticas

Como conclusión extraemos que, como el algoritmo Hill Climbing obtiene mejores resultados tanto en tiempo de ejecución como en coste heurístico, y por lo tanto este es más óptimo para el problema que se nos ha planteado.

Apéndice

1. Proyecto de Innovación

1.1 Descripción del tema:

El piloto automático es un sistema avanzado de asistencia al conductor que mejora la seguridad y la comodidad al volante. Cuando se usa correctamente, Autopilot reduce su carga de trabajo general como conductor. 8 cámaras externas, un radar, 12 sensores ultrasónicos y una potente computadora a bordo brindan una capa adicional de seguridad para guiarlo en su viaje.

1.2 Reparto del trabajo:

Durante la realización de la práctica hemos llevado a cabo una búsqueda en paralelo de información individualmente. Previamente comentamos que puntos indispensables habría que tener en cuenta, para que después cada uno busque por su cuenta aquello que creía interesante comentar en el trabajo. Los tres temas a investigar por nuestra cuenta eran: redes neuronales, visión por computador, seguridad y accidentes. Una vez terminada esta búsqueda, pondremos ideas en común y descartaremos aquellas que no sean de gran interés.

1.3 Referencias:

- [1] Redes Neuronales https://towardsdatascience.com/teslas-deep-learning-at-scale-7eed85b235d3
- [2] Data Collection https://electrek.co/2017/05/06/tesla-data-sharing-policy-collecting-video-self-driving/
- [3] Visión por Computador https://heartbeat.fritz.ai/computer-vision-at-tesla-cd5e88074376
- [4] Redes Neuronales https://www.youtube.com/watch?v=S_TRI9vVDAM

- [5] Seguridad y accidentes https://www.forbes.com/sites/bradtempleton/2020/10/28/new-tesla-autopilot-statistics-show-its-almost-as-safe-driving-with-it-as-without/#:~:text=%E2%80%9CIn%20the%203rd%20quarter%2C%20we,every%202.42%20million%20miles%20driven.
- [6] Seguridad y accidentes https://www.forbes.com/sites/bradtempleton/2020/07/28/teslas-arent-safer-on-autopilot-so-researchers-calling-for-driver-monitoring-may-be-right/?sh=5825b9e21d73
- [7] Youtube: Autopilot on the road https://www.youtube.com/watch?v=fKXztwtXaGo&t=2s
- [8] Autopilot programming language https://analyticsindiamag.com/elon-musk-tesla-python-c-programming-language-tensorflow-pytorch/