# IA - Práctica 1

## Búsqueda Local

Grupo 14


Autores:

Pol Aracil Borrego - aracil.pol@estudiantat.upc.edu
Amadeu Moya Sardà - amadeu.moya@estudiantat.upc.edu
Juan Jiménez Pardo - juan.jimenez.pardo@estudiantat.upc.edu

Índice

# Identificación del problema

## Descripción del problema

Este problema plantea un escenario en el que una región geográfica de 100 x 100 km2 está equipada con S sensores que capturan y almacenan datos a diferentes velocidades de 1, 2, o 5 Mb/s. Las capacidades de captura de los sensores siguen una distribución uniforme, haciendo que haya el mismo número de sensores de 1, 2 y 5 Mb/s. Estos sensores tienen la capacidad de transmitir la información capturada durante un segundo de forma instantánea (para este problema suponemos que el tiempo de transmisión es 0).
Los sensores pueden enviar la información directamente a un centro de datos o a otro sensor que pueda recibir la conexión y actuar como intermediario. Cada sensor puede recibir un máximo de 3 conexiones por parte de otros sensores.
Para optimizar la transmisión y reducir la distancia de las conexiones, los sensores pueden almacenar hasta dos veces el volumen de datos que capturan. Esto implica que un sensor pueda transmitir como máximo tres veces el volumen de datos que captura.
Un sensor envía toda la información que tiene almacenada a un único destino, es decir, no puede dividir la información que tiene almacenada y enviarla a dos puntos diferentes.
Adicionalmente, hay C centros de datos distribuidos en la región, cada uno con una capacidad de recepción de 150 Mb/s con una limitación de 25 conexiones simultáneas. Los centros de datos únicamente pueden recibir información por parte de los sensores.
El objetivo del problema es determinar una red de conexión eficiente entre los sensores y centros de datos que minimice el coste total de transmisión y maximice la cantidad de información recogida por los centros de datos. Otro requisito importante es que todos los sensores tienen que estar conectados directa o indirectamente a un centro de datos.
El coste de una conexión se calcula en función de la distancia euclídea entre emisor y receptor y el volumen de datos transmitidos, siguiendo la fórmula:

$$coste(x,\ y)\ =\ d(x,y)^2\ *\ v(x)$$

Donde:

- $d(x,y)$ es la distancia euclídea entre el sensor emisor (x) y el centro de datos o sensor receptor (y).
- $v(x)$ es el volumen de datos transmitidos por el sensor emisor (x).

Queremos resolver el problema para una instancia en concreto de la cual conocemos la capacidad de los sensores y su ubicación. También conocemos el número de centros de datos y su ubicación.

# Restricciones y objetivos

El problema consta con las siguientes restricciones:

- **Conectividad completa:** Todos los sensores deben estar conectados directa o indirectamente a un centro de datos.
- **Capacidad de almacenamiento de un sensor:** Un sensor puede almacenar hasta dos veces el volumen que captura.
- **Capacidad de transmisión de un sensor:** Un sensor puede transmitir hasta tres veces el volumen que captura. Es decir, puede enviar la información que captura más la que almacena.
- **Capacidad de almacenamiento de un centro de datos:** Un centro de datos puede almacenar un máximo de 150 Mb/s.
- **Límite de conexiones realizadas por los sensores:** Un sensor solo se puede conectar a un único sensor o centro de datos y envía toda la información que pueda transmitir.
- **Límite de conexiones recibidas para los sensores:** Un sensor puede recibir la conexión de hasta 3 sensores.
- **Límite de conexiones recibidas para los centros de datos:** Un centro de datos puede recibir conexiones de hasta 25 sensores.
- **Pérdida de información por parte de un sensor:** Si un sensor recibe más datos de los que se pueden almacenar, estos se pierden.
- **Pérdida de información por parte de un centro de datos:** Si un centro de datos recibe más datos de los que puede almacenar, estos se pierden.

Por otro lado, tiene los siguientes objetivos:

- **Minimizar el coste total de la transmisión:** Para ello hay que reducir el coste de las conexiones entre sensores y entre sensores y centros de datos.
- **Maximizar la información transmitida:** Para ello hay que maximizar la cantidad de datos que reciben (y que puedan almacenar) los centros de datos.
- **Optimizar la topología de la red:** Como resultado de los dos puntos anteriores, es necesario diseñar una infraestructura de conexiones que optimice el uso de las capacidades de los sensores. Esta infraestructura debe evitar, en medida de lo posible, la sobrecarga de cualquier sensor para garantizar que se maximice la cantidad de información que llega a los centros de datos. De la misma manera, se debe minimizar el coste de transmisión, y para ello, tenemos que aprovechar la capacidad de realizar conexiones intermedias entre sensores.
- **Conocer las conexiones:** Para cada sensor queremos saber a que sensor o centro de datos se conecta.

# Estado del problema

Para poder trabajar de forma computacional con este problema, lo mejor es representarlo como un grafo dirigido ponderado, donde tenemos:

- **Nodos:**
  - Un nodo para cada sensor.
  - Un nodo para cada centro de datos.
- **Aristas:**
  - Puede existir una arista que conecta un sensor con otro sensor.
  - Puede existir una arista que conecta un sensor con un centro de datos.
  - Los nodes de los centros de datos solo tienen aristas incidentes; no emiten aristas de salida.
  - Los nodos de los sensores pueden tener aristas incidentes. Cada sensor tiene una arista de salida que tiene como destino otro sensor, diferente a él, o un centro de datos.
- **Peso de las aristas:**
  - Determinado por la fórmula del coste: $coste(x,\ y)\ =\ d(x,y)^2\ *\ v(x)$
- **Estado del sistema:**
  - Posibles configuraciones de conexiones de los sensores hacia otros sensores o centros de datos.
- **Acciones permitidas:**
  - Modificar las conexiones para redistribuir el flujo de información.

# Problema de búsqueda local

El problema de optimización de la red de sensores se ajusta a los problemas de búsqueda local por las siguientes razones:

- **Espacio de estados grande y complejo:** Hay una cantidad exponencial de posibles configuraciones entre sensores y centros de datos, donde la gran mayoría de ellas no son una solución que cumpla con las restricciones del problema.
- **Función objetivo clara:** Se puede evaluar cualquier solución en función del coste total de transmisión y de la información transmitida.
- **Soluciones vecinas definidas:** Las soluciones vecinas pueden generarse realizando cambios locales, como, por ejemplo, intercambiar la conexión entre dos sensores.
- **No existe un algoritmo exacto:** Debido a las restricciones de combinatoria de este problema, se puede considerar un problema NP-difícil que no se puede resolver con algoritmos polinómicos en un tiempo razonable para instancias grandes.
- **Optimización por heurística:** Algoritmos como el Hill Climbing o el Simulated Annealing permiten explorar soluciones cercanas y aproximarse al óptimo global sin necesidad de explorar todo el espacio de soluciones. Con lo cual, podemos partir de una solución calculada de forma computacionalmente simple, y a partir de ahí buscar por el espacio de soluciones.

# Estado del problema y representación

Un estado del problema se representa mediante dos arreglos de objetos: uno de tipo SensorInfo y otro de tipo CentroInfo. Estos tipos extienden respectivamente a Sensor y Centro, incorporando información adicional relevante para la gestión de conexiones.

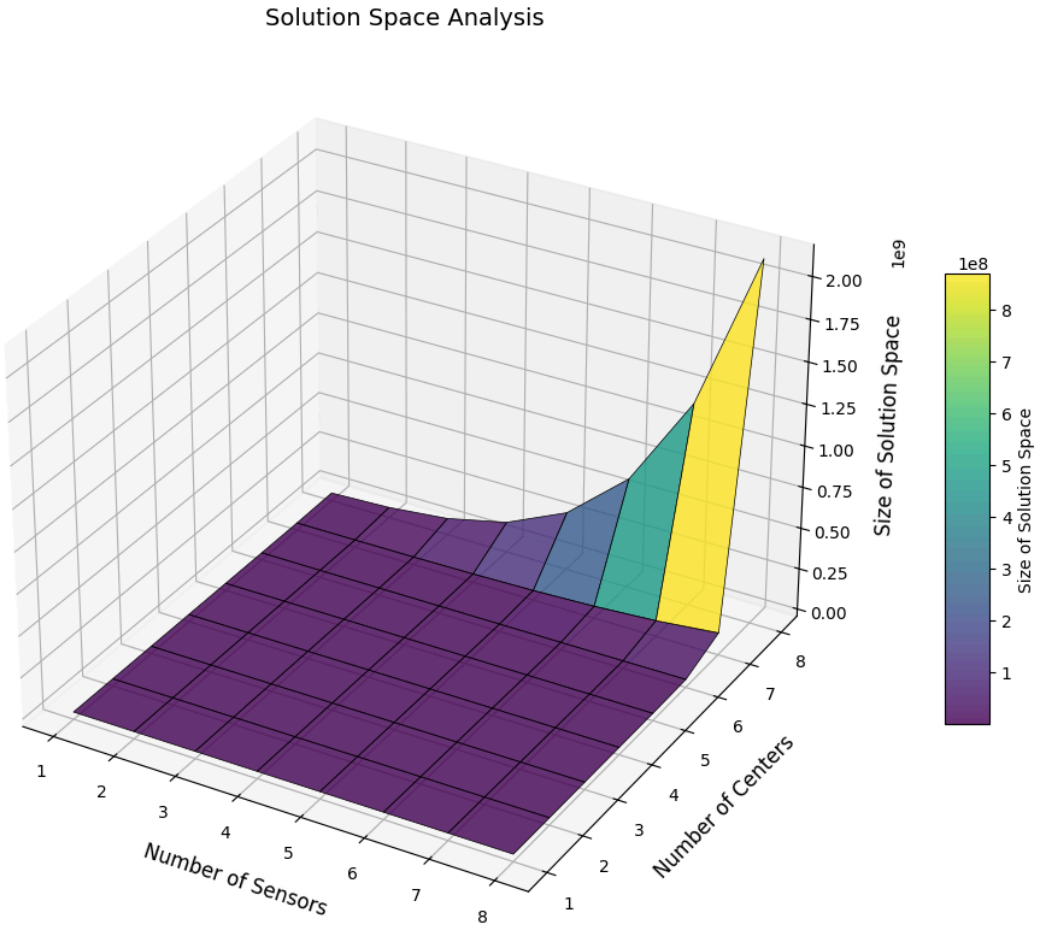
- SensorInfo amplía la funcionalidad de un Sensor, añadiendo detalles sobre su capacidad restante, el número de conexiones entrantes aún disponibles, y registrando a qué Centro o Sensor está conectado.

- CentroInfo, por su parte, extiende a Centro, incluyendo información sobre su capacidad restante y el número de conexiones entrantes aún disponibles.

Gracias a esta estructura de datos, es posible extraer una solución al problema, ya que se mantiene un registro de la conexión de cada sensor. Además, esta representación permite generar estados sucesores válidos porque tenemos la información sobre la disposición de las conexiones, junto con la capacidad restante y las conexiones entrantes disponibles de los sensores y los centros.

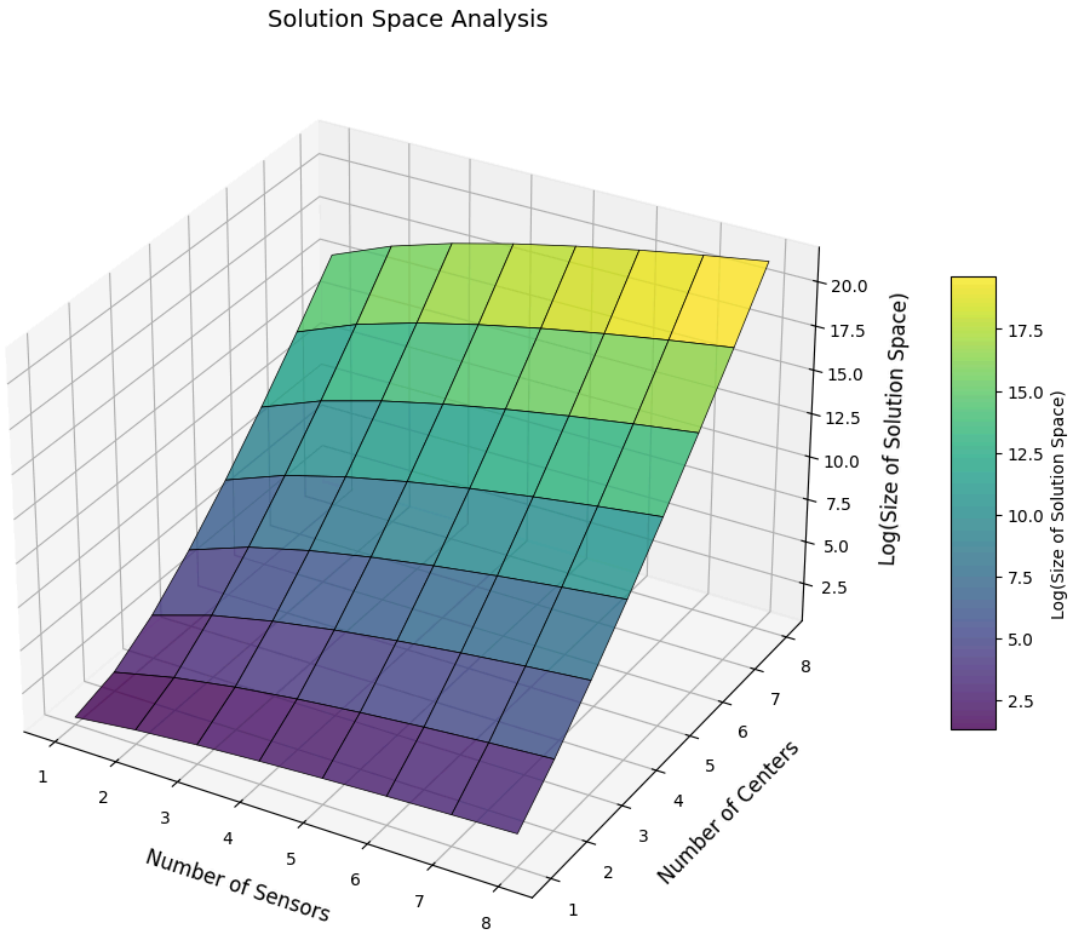
## Análisis del tamaño del Espacio de Búsqueda

Con los operadores que hemos definido en la sección de Representación y análisis de los operadores, nuestro espacio de búsqueda se reduce al espacio de soluciones.
Analizar el tamaño del espacio de soluciones de forma analítica es muy complicado. La intuición nos dice que crece exponencialmente en función de sensores y centros, pero queremos verificarlo. Para ello, hemos diseñado un experimento que visita la totalidad del espacio de soluciones para todos los casos con sensores entre 1 y 8, y centros entre 1 y 8.




Estos son los resultados que hemos obtenido. Como se puede observar, el tamaño del espacio de soluciones se dispara rápidamente cuando el número de sensores y centros empieza a crecer hasta tal punto que para 9 o 10 sensores ya es computacionalmente inviable.

El problema que tenemos es que sabemos que crece muy rápido, pero no podemos asegurar cual es el tipo de crecimiento de la función.




Para confirmar de manera empírica nuestra hipótesis de que el tamaño del espacio de soluciones crece exponencialmente, aplicamos un escalado logarítmico, con el que ahora podemos ver claramente una linealidad. De este modo podemos decir que estamos bastante seguros que el tamaño del espacio de soluciones crece exponencialmente.

# Representación y análisis de los operadores

Para encontrar una buena solución al problema planteado en el enunciado usando Hill Climbing o Simulated Annealing, hemos seleccionado dos operadores que nos permiten desplazarnos por el espacio de soluciones sin salirnos.

## Operador de Cambiar Conexión

El primer operador se llama “cambiar conexión”, y consiste en cambiar el sensor/centro al cual un sensor dado está conectado, por otro sensor/centro respetando las condiciones de aplicabilidad. Por ejemplo, si tenemos el sensor S1 que está conectado a X, aplicando el operador sobre S1 y otro sensor/centro Y que cumple las condiciones de aplicabilidad, tenemos que S1 pasa a estar conectado a Y.

## Condiciones de Aplicabilidad

Para mantenernos en el espacio de soluciones y respetar las condiciones de aplicabilidad debemos hacer unas cuantas comprobaciones antes de aplicar el operador:

1. El sensor/centro al que se va a conectar el sensor debe ser diferente de él mismo. Si no hacemos esta comprobación estamos permitiendo que se generen ciclos en el grafo, y no se cumpliría la condición de que todo sensor debe estar conectado directa o indirectamente a un centro de datos.
2. El sensor sobre el cual aplicamos el operador no debe ser accesible desde el nuevo sensor/centro al cual estará conectado. (En el caso de que sea un centro, esta propiedad se cumple trivialmente porque los centros no se conectan a nada). Si no hacemos esta comprobación estamos permitiendo que se generen ciclos en el grafo, y no se cumpliría la condición de que todo sensor debe estar conectado directa o indirectamente a un centro de datos.
3. El sensor/centro al que se va a conectar el sensor debe tener una cantidad de conexiones entrantes restantes mayor que 0.

Para mejorar la exploración también hacemos la siguiente comprobación:

4. El sensor/centro al que se va a conectar el sensor debe ser diferente al sensor/centro al que está conectado actualmente. (Sabemos que el sensor estará conectado a algún sensor/centro porque siempre nos movemos por el espacio de soluciones). Si no hacemos esta comprobación estamos permitiendo el operador de identidad, que no ayuda en la exploración porque nos lleva de una solución a ella misma.

## Efectos

Aplicar el operador de “cambiar conexión” comprobando las condiciones anteriores nos mantiene en el espacio de soluciones.

Vamos a demostrarlo por inducción. Como caso base, al haber aplicado el operador 0 veces tenemos una solución porque el estado inicial es una solución válida. Ahora, suponiendo que partimos de una solución válida, vamos a ver que llegamos a una solución válida después de aplicar el operador. Por un lado tenemos que el sensor/centro al que está conectado el sensor sobre el que aplicamos el operador seguirá teniendo un número válido de conexiones restantes (aumentar un número de conexiones restantes genera un número válido de conexiones restantes). Por otro lado, el sensor/centro al que pasará a estar conectado el centro también seguirá teniendo un número válido de conexiones restantes (una de las condiciones de aplicabilidad que comprobamos es que su número de conexiones restantes sea mayor que 0. Decrementar un número de conexiones restantes mayor que 0 da un número de conexiones restantes válido). Por último, comprobamos que el sensor/centro al que nos conectamos no tenga accesibilidad al sensor (si es un centro es trivial), cosa que nos asegura que el sensor estará indirectamente conectado a un centro, porque el sensor al que nos conectamos tiene que estar directa o indirectamente conectado a un centro. (de otro modo el estado antes de aplicar el operador no sería solución válida).

### Factor de Ramificación

Ahora vamos a analizar el factor de ramificación del operador tomando en cuenta el caso en el que se maximiza, que se da cuando el operador se puede aplicar siempre mientras no intentemos conectar el sensor consigo mismo o con el sensor/centro al que ya está conectado. Este caso en el que se maximiza se da, por ejemplo, en un estado dónde haya tantos sensores como centros, y cada sensor está conectado a un centro diferente.

Dados $x \equiv nSensores$ e $y \equiv nCentros$, tenemos que $f(x, y) = x * (x + y - 2)$ es la función que calcula el factor de ramificación máximo en función del número de sensores y centros. La interpretación de esta función es que por cada sensor, podemos cambiar su conexión a cualquiera de los sensores y centros exceptuando a sí mismo y al sensor/centro al que ya está conectado.

En notación big O, el factor de ramificación sería $O(x^2 + x * y)$.

## Operador de Intercambiar Conexión

El segundo operador se llama “intercambiar conexión”, y consiste en intercambiar las conexiones de dos sensores a la vez. Por ejemplo, si tenemos dos sensores S1 y S2 que están conectados con X e Y respectivamente, aplicando el operador sobre S1 y S2 obtenemos que S1 pasa a estar conectado con Y, y S2 con X.

### Condiciones de Aplicabilidad

Para mantenernos en el espacio de soluciones y respetar las condiciones de aplicabilidad debemos hacer una comprobación muy importante antes de aplicar el operador:

1. Los dos sensores sobre los cuales aplicamos el operador no deben ser accesibles entre sí. Si no hacemos esta comprobación estamos permitiendo que se generen ciclos en el grafo, y no se cumpliría la condición de que todo sensor debe estar conectado directa o indirectamente a un centro de datos.

Para mejorar la exploración también hacemos la siguiente comprobación:

2. Los sensores/centros a los que están conectados ambos sensores deben ser diferentes. Si no hacemos esta comprobación estamos permitiendo el operador de identidad, que no ayuda en la exploración porque nos lleva de una solución a ella misma.

### Efectos

Este operador es muy parecido al operador al de “cambiar conexión” pero tiene una propiedad muy interesante: no modifica el número de conexiones restantes de ningún sensor/centro. Esta propiedad puede ser útil para explorar zonas del espacio de soluciones dónde los sensores/centros no tengan demasiadas conexiones restantes disponibles. En este escenario, el factor de ramificación del operador de “cambiar conexión” sería muy pequeño, y el operador de “intercambiar conexión” nos ayudaría a generar más nodos para visitar.

Este operador también cumple la propiedad de que nos mantiene en el espacio de soluciones. La demostración sería parecida a la del operador anterior. Las condiciones relacionadas con las conexiones máximas de los sensores y los centros se cumplen sin tener que comprobar nada porque al intercambiar las conexiones, no estamos modificando la cantidad de conexiones restantes de los sensores/centros a los que nos conectamos. Por otro lado, dado que estamos comprobando la accesibilidad de los sensores entre si, sabemos que al intercambiar las conexiones, ambos sensores seguirán estando directa o indirectamente conectados a un centro.

### Factor de Ramificación

Para este operador, un ejemplo de estado que maximiza el factor de ramificación podría ser el que hemos puesto de ejemplo para el anterior operador: mismos sensores que centros y cada sensor conectado a un centro diferente.

Dados $x \equiv nSensores$ e $y \equiv nCentros$, tenemos que $f(x, y) = \sum_{i=0}^{x-1}(x - 1 - i) = \sum_{i=0}^{x-1}(x - 1) - \sum_{i=0}^{x-1}(i) = x * (x - 1) - \frac{(x-1)*x}{2} = \frac{(x-1)*x}{2}$ es la función que calcula el factor de ramificación máximo en función del número de sensores y centros. Como se puede ver, en este caso la cantidad de centros no influye. La interpretación de esta función (el primer paso) es que por cada sensor, podemos intercambiar su conexión con la de cualquier otro sensor exceptuando consigo mismo. También tenemos que ir descartando repetidos, porque intercambiar S1 con S2 es lo mismo que intercambiar S2 con S1. Si nos imaginamos una tabla bidimensional de tamaño x*x donde cada casilla representa una posible pareja de sensores, estaríamos contando solo la diagonal superior para evitar repetidos.

En notación big O, el factor de ramificación sería $O(x^2)$.

### Elección de los Operadores

Hemos acabado escogiendo ejecutar los algoritmos con la combinación de los dos operadores porque, tal como se explica en el Experimento 1, se obtienen resultados ligeramente mejores, y nos permite explorar una mayor porción del espacio de soluciones que si solo utilizáramos uno de los dos operadores.

# Análisis de la función heurística

## Factores que intervienen en la heurística

A la hora de diseñar la heurística, hemos considerado dos factores principales, el coste y la información enviada con éxito a los centros de datos.

### Coste

Para calcular el coste de una conexión, hemos tenido en cuenta dos variables; La distancia entre el emisor y el receptor y la cantidad de información enviada por el emisor. El coste sigue la siguiente fórmula:

$$coste\ (x, y)\ =\ d(x, y)^2\ *\ v(x)$$

La distancia máxima $d(x, y)$ entre dos elementos es de 141.42 km (suponiendo que el emisor está en la coordenada (0,0) y el receptor en la coordenadas (100, 100)). Al elevar este factor al cuadrado obtenemos un rango de valores para de $d(x, y)^2$ de 0 a 20000.
Por otro lado, el rango de información enviada $v(x)$ oscila entre 1 y 15. El valor 1 corresponde a un sensor de 1Mb/s que solo transmite la información capturada en tiempo real. Por otro lado, el valor 15 corresponde a un sensor de 5 Mb/s que almacena 10Mb/s y envía toda esa información al destino.
Finalmente, dado que el coste es el producto de estas dos variables, el valor del coste de una única conexión puede oscilar entre 0 y 300000.

### Información enviada con éxito

Para calcular la información enviada, solo hace falta ver cuánta información están recibiendo los centros de datos con éxito.
Los centros de datos pueden almacenar hasta 150Mb/s cada uno, es decir, si reciben más de 150Mb/s de información por segundo, esta se pierde.
Sabemos la capacidad de captura de los sensores sigue una distribución uniforme, es decir, para un problema con 100 sensores, tenemos que 34 tienen una capacidad de captura de 1Mb/s, 33 con una capacidad de 2 Mb/s y 33 con una capacidad de 5Mb/s, dando un total de 265 Mb/s.
Teniendo estos datos, para un problema con 100 sensores, tenemos un rango de valores que oscila entre 0 Mb/s y 265 Mb/s de información enviada con éxito. El caso de 0 Mb/s corresponde al caso donde se pierde toda la información. Por otro lado, el caso de 265 Mb/s corresponde al caso donde toda la información es enviada con éxito.

## Funciones heurísticas probadas

Para analizar el comportamiento de los parámetros mencionados anteriormente, diseñamos una serie de funciones heurísticas para estudiar su comportamiento.

### Heurística 1: Coste

Esta función heurística únicamente tenía en cuenta el coste de la solución. Como consecuencia, el coste era muy bajo pero a su vez, la cantidad de información enviada con éxito también lo era, haciendo que la condición de la solución de maximizar los datos enviados no estuviera satisfecha, haciendo que la solución final no fuera una buena solución.

Para verificar este comportamiento, realizamos diversas simulaciones con las mismas características que en el experimento 1 obtuvimos los siguientes resultados:

| Tiempo medio | Coste Medio | Throughput medio |
|---|---|---|
| 6.703 | 28.302 | 42 |

Como podemos observar, aunque el coste de la solución es bastante bajo, teniendo en cuenta que el rango de valores para una única conexión puede oscilar entre 0 y 300000, los datos enviados con éxito se alejan mucho del máximo que se puede obtener en un escenario con 100 sensores y 4 centros de datos, que es 265.

## Heurística 2: Información enviada con éxito

Esta función heurística, al contrario que la anterior, solo tenía en cuenta la cantidad de datos enviados con éxito. Como resultado solo se obtenían soluciones que maximizan la cantidad de datos enviados con éxitos sin tener en cuenta el coste, haciendo que la mayoría de veces el coste de la solución fuera muy alto, haciendo que la condición de la solución de minimizar el coste no estuviera satisfecha, haciendo que la solución final obtenida no fuera una buena solución.
Para verificar este comportamiento, realizamos diversas simulaciones con las mismas características que en el experimento 1 obtuvimos los siguientes resultados:

| Tiempo medio | Coste Medio | Throughput medio |
|---|---|---|
| 1.701 | 2.378.366 | 265 |

Como podemos observar, aunque los datos enviados con éxito son máximos para un escenario con 100 sensores y 4 centros de datos, el coste es demasiado elevado haciendo que las soluciones sean realmente malas.

## Heurística 3: Coste - Información enviada con éxito

Esta función heurística buscaba relacionar los dos parámetros que hacen que una solución sea buena. El problema de esta función, es que, al no tener los factores ponderados, el coste, al tener un rango de valores mucho más grande que el de la información enviada con éxito, hiciera que tuviera el mismo comportamiento que con la heurística 1, descartando de esta manera esta implementación.
Para verificar este comportamiento, realizamos diversas simulaciones con las mismas características que en el experimento 1 obtuvimos los siguientes resultados:

| Tiempo medio | Coste Medio | Throughput medio |
|---|---|---|
| 6.718 | 28.259 | 43 |

Como podemos observar, los datos son realmente muy similares a los de la heurística 1.

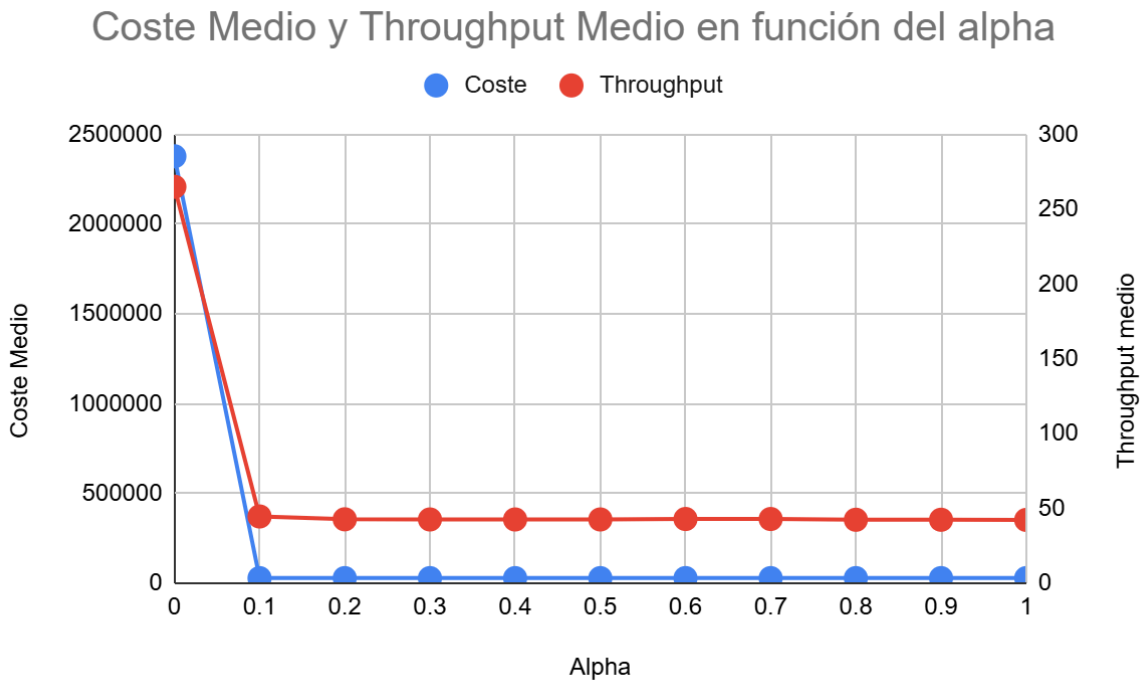
### Heurística 4: Coste - Información enviada con éxito (Ponderados)

Esta función heurística buscaba relacionar el coste con la información enviada con éxito, ponderando ambos factores con la fórmula:

$$alpha * coste\ total - (1 - alpha) * información\ enviada\ con\ éxito$$
$$alpha \in [0, 1]$$

Dado a la gran diferencia de rangos entre los valores, el valor de alpha tenía que tomar valores extremadamente pequeños (del orden de 0.00025 para que empezará a tener un comportamiento deseado), haciendo que fuera muy difícil ajustar el valor de alpha para [poder experimentar y ajustar la fórmula al comportamiento deseado, descartando así esta opción.

Para verificar este comportamiento, realizamos diversas simulaciones con las mismas características que en el experimento 1 obtuvimos los siguientes resultados:



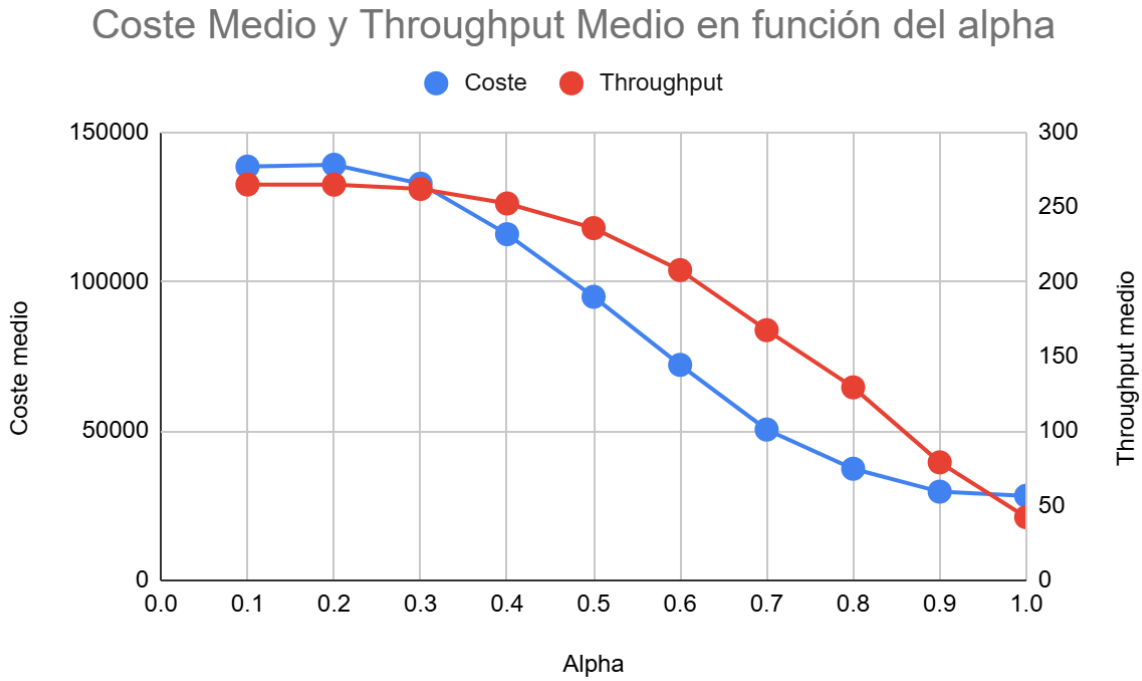

Como podemos observar el valor deseado del *alpha* se encuentra en el rango entre 0 y 0.1, que se ajusta al valor de 0.00025 que habíamos obtenido de forma experimental, confirmando así que esta no era la manera correcta de balancear los valores.

## Función heurística final

Después de analizar los casos anteriores llegamos a la conclusión de que teníamos que multiplicar la información enviada con éxito por un factor que hiciera que el rango de valores entre el coste y la información enviada con éxito fuera similar para ello, diseñamos la siguiente heurística:

$$alpha * coste\ total - (1 - alpha) * 1000 * información\ enviada\ con\ éxito$$
$$alpha \in [0, 1]$$

Haciendo que el rango de valores para la información enviada con éxito, para un escenario concreto de 100 sensores, pasará de [0, 265] a [0, 265000]. De esta manera conseguimos que el peor caso del coste de una única conexión y el mejor caso de la información enviada con éxito fueran bastante similares (300000 para el coste y 265000 para la información enviada).
Finalmente, con el fin de reducir el valor final de la heurística, aplicamos un alpha de 0.4 haciendo que el rango del coste de una única solución estuviera entre [0, 120000] y el rango de valores para la información enviada con éxito fuera de [0, 159000].
Esta heurística nos permite obtener soluciones finales donde el coste y la información enviada con éxito están equilibradas, encontrando un punto intermedio entre maximizar la información enviada con éxito y minimizar el coste total de la conexión, obteniendo así soluciones finales mucho más cercanas al requisito del problema.
Para confirmar que estábamos en lo cierto, hicimos varias simulaciones, con las mismas condiciones que en el experimento 1, de las cuales pudimos extraer los siguientes datos:




Los datos del punto 0 no se añaden porque distorsionan la gráfica, igualmente, en este escenario con 100 sensores y 4 centros de datos, los resultados obtenidos en el punto 0 siguen el mismo comportamiento que con la heurística 2.
Como podemos observar en la gráfica, a partir del valor de *alpha* 0.4 la caída del valor medio de información enviada con éxito y del coste se acentúa, haciendo que, para nosotros, ese sea el mejor punto para encontrar un equilibrio entre coste e información enviada con éxito.

# Elección y generación del estado inicial

Para poder usar los algoritmos de Hill Climbing y Simulated Annealing es necesario aportar una solución inicial, la cual servirá de punto de partida.

En nuestro caso, decidimos crear 2 tipos de soluciones iniciales, una mala y otra buena, ya que de esta forma se puede apreciar como con una solución que se aproxime más a lo que se busca se puede encontrar mejores soluciones e incluso en menos tiempo.

En el experimento 2 analizamos ambas soluciones iniciales para determinar cuál es la mejor a usar en el resto de casos.

## Solución mala

Esta solución se basa en seleccionar el primer centro de datos disponible y crear una línea de sensores de tal forma que las distancias entre los sensores sea máxima.

Esto implica que el coste final será muy alto y el throughput será como mucho igual a la capacidad máxima del sensor conectado al centro de datos.

De esta forma, conseguimos una solución inicial muy mala y, al aplicar los algoritmos de búsqueda local, podremos ver que tanto puede evolucionar desde este punto.

El coste temporal de encontrar esta solución tal y como está implementada es de $O(n^2)$, siendo n el número de sensores.

## Solución buena

Esta solución se basa en algoritmos de redes de flujo. En concreto, iteramos por los sensores como puntos de inicio de los flujos y buscamos con Dijkstra el camino más corto disponible hacía un centro.

Durante esta búsqueda, nos aseguramos de que el throughput total que hay desde el sensor inicial y los que hay por el camino sea transferido sin pérdidas, asegurándonos de que el throughput total sea máximo y con un coste relativamente pequeño.

El coste temporal de esta solución es $O(t \cdot (n^2 + n \cdot c) \cdot \log(n + c))$, siendo t el throughput total, n el número de sensores y c el número de centros de datos. Esto es en el peor de los casos, pero en la realidad normalmente no es un algoritmo pesado por cómo está diseñado el problema.

# Escenarios propuestos y preguntas del enunciado

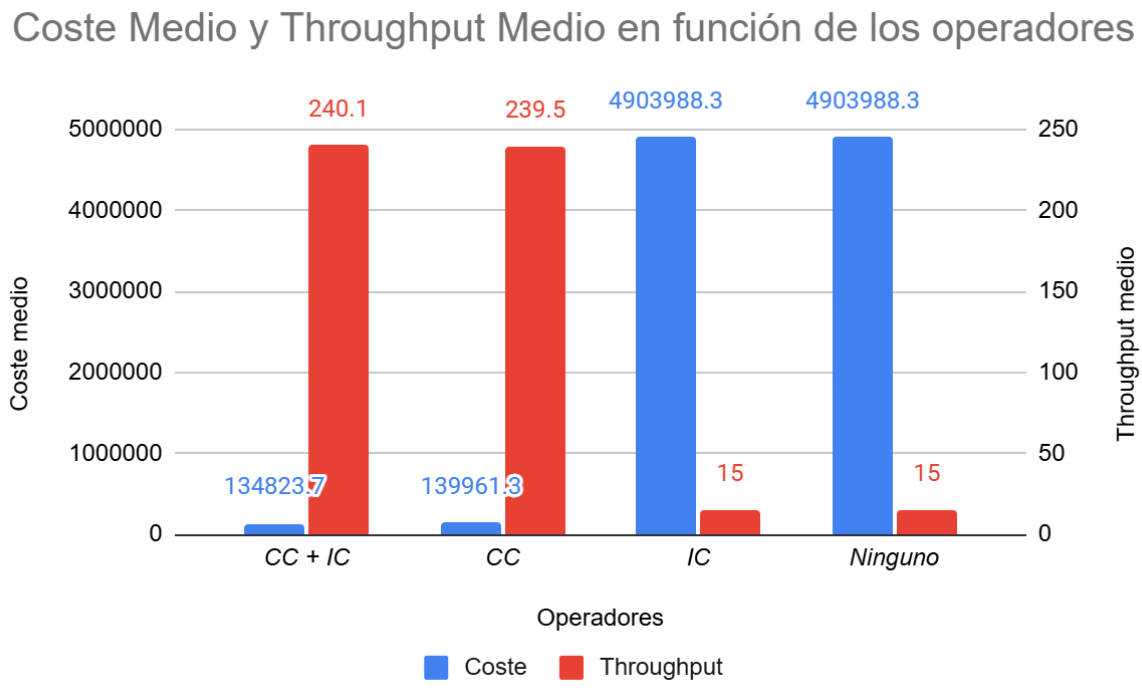
## Experimento 1: Influencia de los operadores

El experimento 1 consiste en determinar qué conjunto de operadores da mejores resultados para la heurística que hemos determinado y para una de las dos soluciones que hemos definido.

Nosotros hemos decidido trabajar con la heurística explicada en el apartado "Función heurística final" y hemos utilizado la "Solución mala". Las operaciones utilizadas son:

- CC - Cambiar Conexión: La función de este operador es cambiar el destino de la conexión de un sensor (Nunca podrá ser el mismo).
- IC - Intercambiar Conexión: La función de este operador es intercambiar el destino entre dos sensores. Es decir, el sensor 1 pasará a tener el destino 2 y viceversa (asegurando en todo momento que el nuevo destino no sea uno mismo).

Los datos obtenidos después de la ejecución del experimento, en un escenario con 100 sensores y 4 centros de datos, son los siguientes:




Con esta gráfica podemos observar que los mejores resultados se dan para el caso CC + IC y para CC. Dado que los resultados son muy similares, finalmente nos hemos decantado por la opción CC + IC porque, potencialmente, permite explorar más estados.

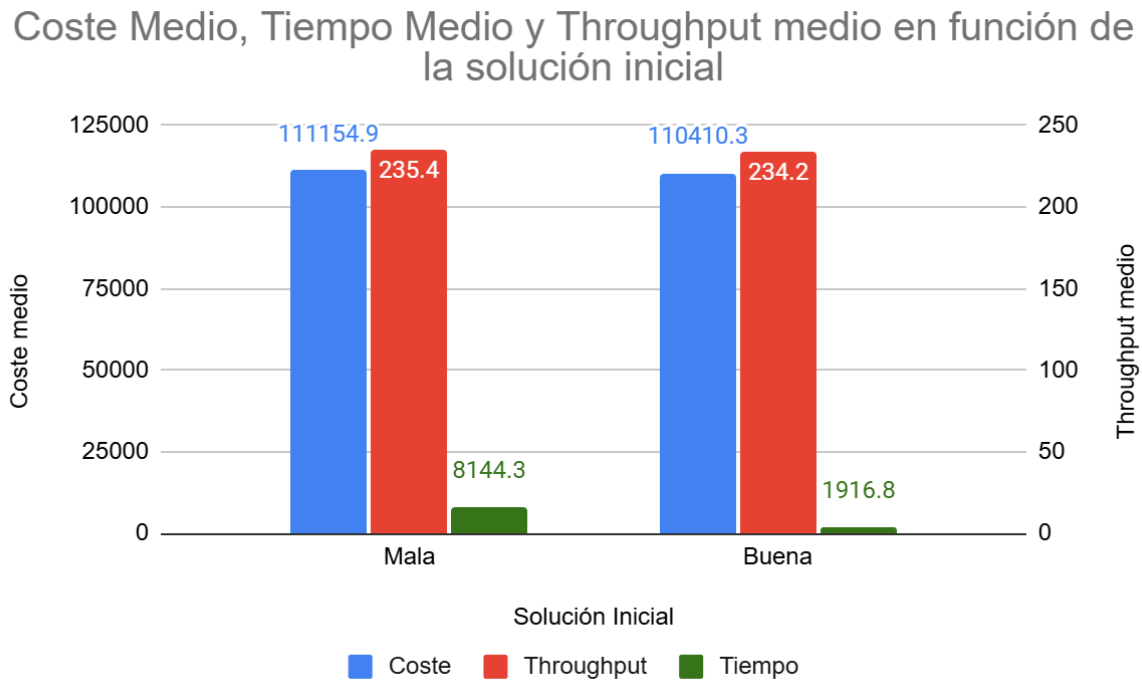
Respecto al tiempo de ejecución obtuvimos los siguientes resultados:

| Operadores | Tiempo medio |
|---|---|
| CC + IC | 7923.4 |
| CC | 4530.9 |
| IC | 0.8 |
| Ninguno | 0 |

Podemos ver, tomando como referencia el caso de *IC* y *Ninguno* sabiendo que con la solución utilizada no exploran ningún estado, que *CC* y *CC+IC* exploran ambos más estados y como hemos mencionado anteriormente, nos quedamos con la opción de *CC+IC* porque potencialmente explora más.
Con experimentó fijamos la función heurística y los operadores para el resto de experimentos.

## Experimento 2: Influencia de la solución inicial

El experimento 2 consiste en determinar qué solución inicial da mejores resultados a partir de las conclusiones obtenidas en el experimento 1.
Con estas condiciones y para un escenario con 100 sensores y 4 centros de datos, los datos obtenidos son los siguientes:



Con esta gráfica podemos ver, que con las condiciones dadas, las dos soluciones iniciales tienen un coste y un throughput prácticamente idéntico. En este caso, únicamente difieren en el tiempo de ejecución, teniendo la solución buena un tiempo medio de ejecución de 6 segundos menos que el caso de la solución inicial mala. Esto nos ha hecho decantarnos por la solución inicial buena, ya que al tener un tiempo medio de ejecución menor nos permitirá explorar espacios de búsqueda más grandes en un tiempo inferior.
Con este experimento fijamos la solución inicial utilizada para el resto de experimentos.

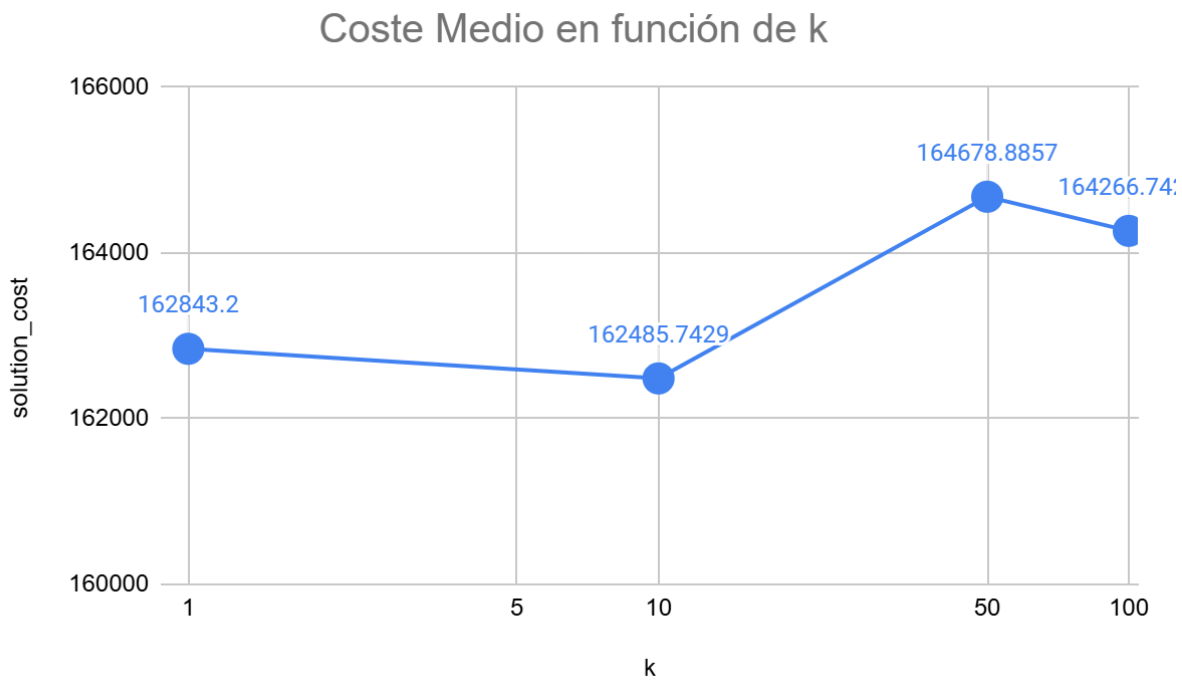
## Experimento 3: Parámetros del Simulated Annealing

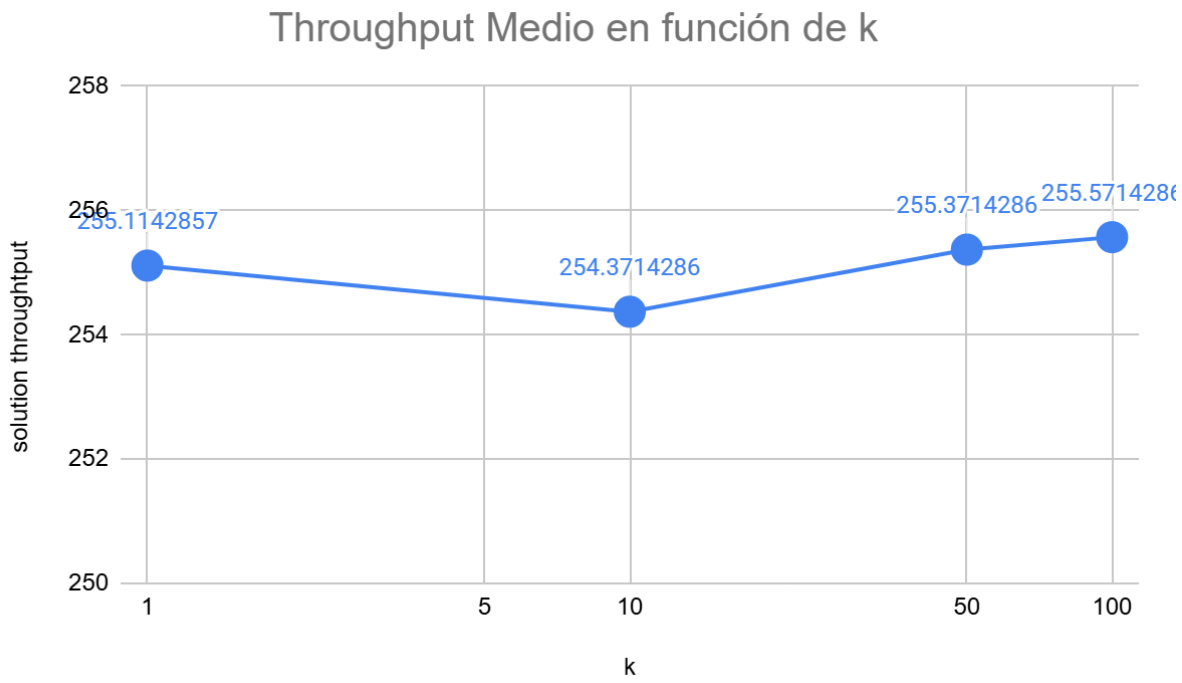
El experimento 3 consiste en determinar qué parámetros del algoritmo de Simulated Annealing dan mejores resultados con el mismo escenario de los experimentos anteriores, misma heurística, mismos operadores y misma solución inicial.

Primeramente, escogimos que haríamos 500 iteraciones, ya que el costo para crear sucesores hace que ejecutar una iteración de Simulated Annealing sea bastante costoso, por lo que decidimos encontrar un punto medio para que no tardase tanto en ejecutarse pero que pudiera hacer varias iteraciones.

Además, también decidimos que se hiciese una iteración por cada cambio de temperatura para centrarnos en los otros parámetros.

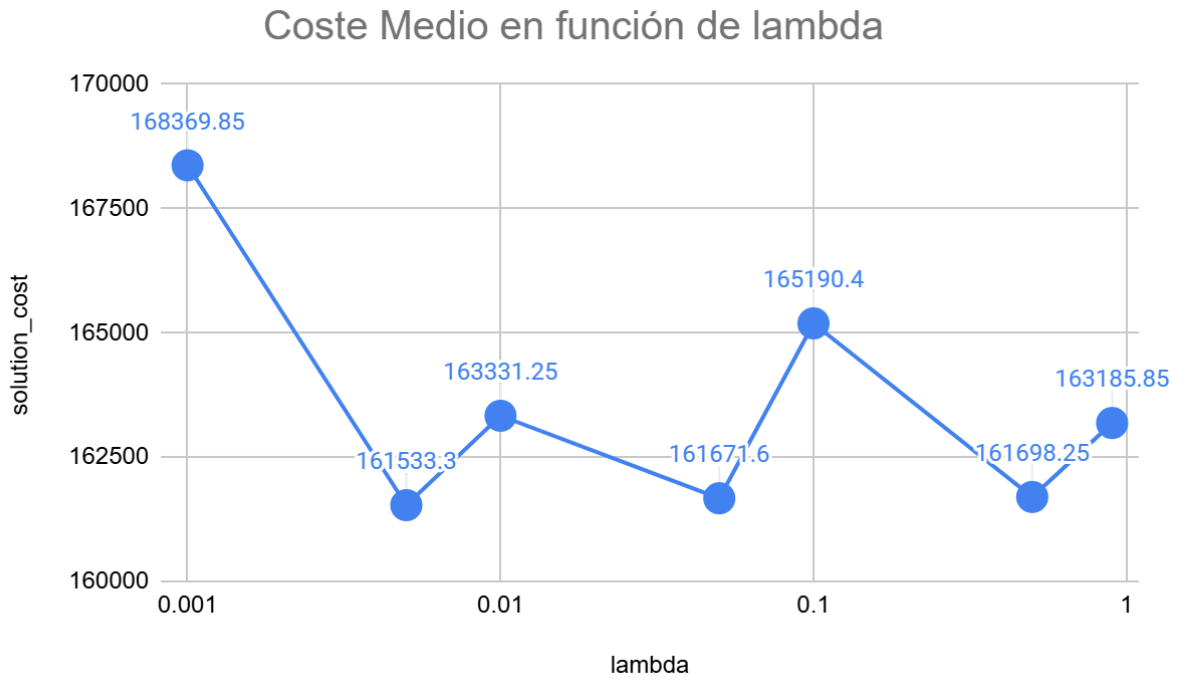
Una vez especificadas estas condiciones, ejecutamos el experimento y estos son los resultados:

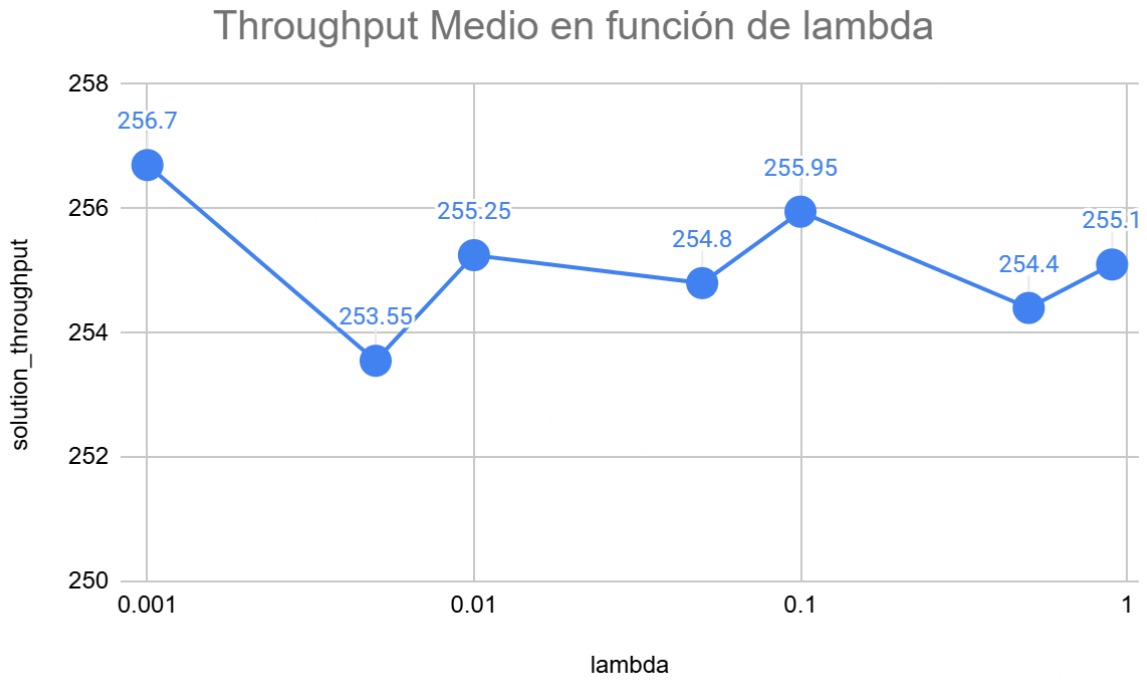





En el caso del parámetro k, a pesar de que no es muy grande, podemos ver que el coste es más bajo cuando k = 1 y k = 10. Por otro lado, en el caso del throughput es más alto en los casos distintos a k = 10.

A pesar de que los resultados no son tan distintos, se puede ver que cuando k = 1 esta parece dar mejores resultados, por lo que fijamos este valor para k.






En el caso del parámetro lambda, vemos que cuando esta está alrededor de 0.001, el throughput es bueno, pero su coste es bastante más alto.

En los demás casos, podemos ver que los parámetros que destacan más en términos de coste son 0.005, 0.05 y 0.5. Por otro lado, en el caso del throughput destacan los mayores a 0.01.

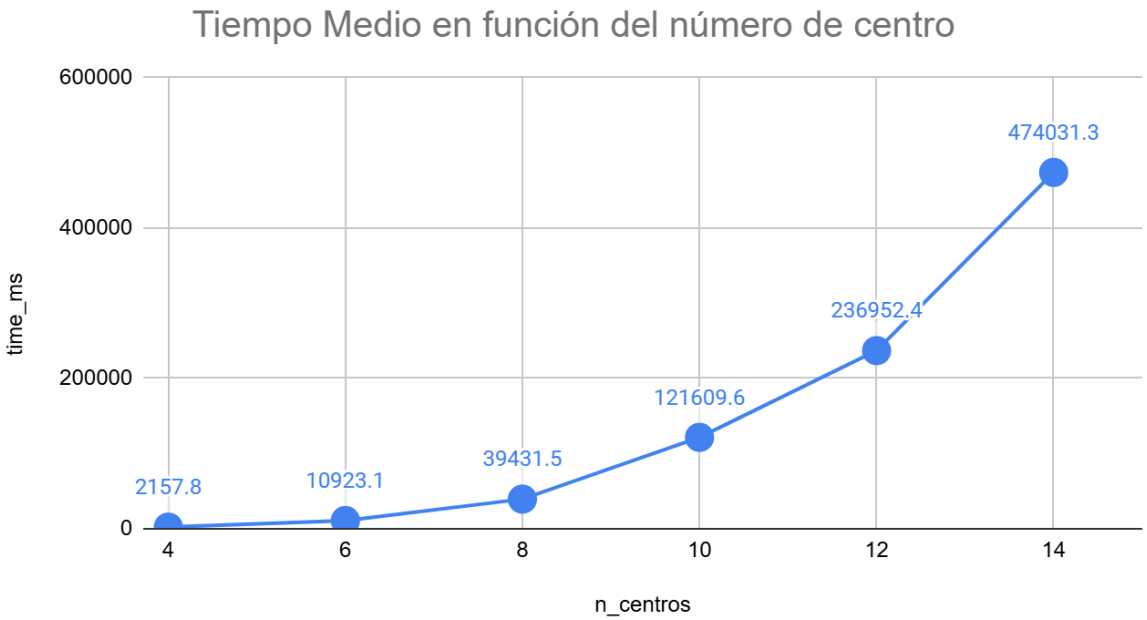
En este métrica los resultados tampoco varían mucho, por lo que en este caso decidimos fijar lambda = 0.05, ya que es el parámetro con mejores y más equilibrados resultados.

Debido al alto costo de crear sucesores, el algoritmo de Simulated Annealing no parece ser una gran opción, ya que para poder dar mejores resultados como esperaríamos sería necesario conseguir hacer muchas más iteraciones, pero esto implicaría un alto coste de tiempo.

## Experimento 4: Tiempo en función de la proporción 4:100

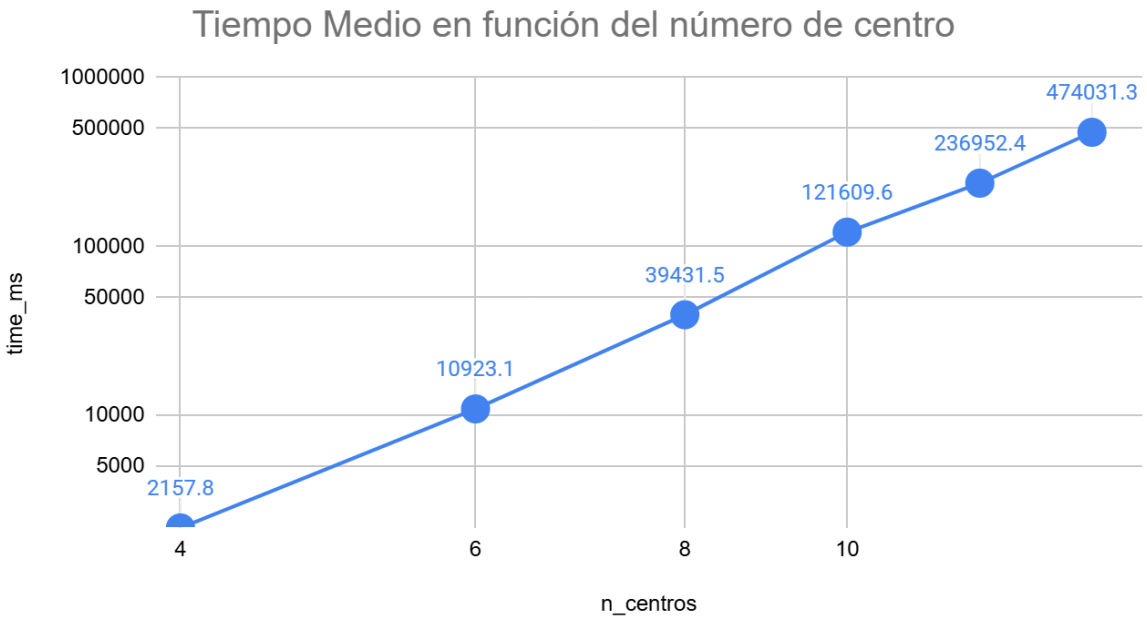
El experimento 4 consiste en determinar cómo evoluciona el tiempo con una proporción de 4:100 para los parámetros de sensores y centros de datos. Para ello, se usa la misma heurística, el algoritmo Hill Climbing y se aumenta de 2 en 2 el número de centros siguiendo la proporción para determinar el número de sensores.

Una vez establecidas las condiciones, los resultados del experimento son los siguientes:




Como se puede apreciar, el coste no es lineal y va aumentando cada vez más a medida que se aumentan los centros de datos y los sensores proporcionalmente.

Sin embargo, hay que determinar si el coste es polinómico o exponencial, por lo que ajustaremos los 2 ejes logarítmicamente.

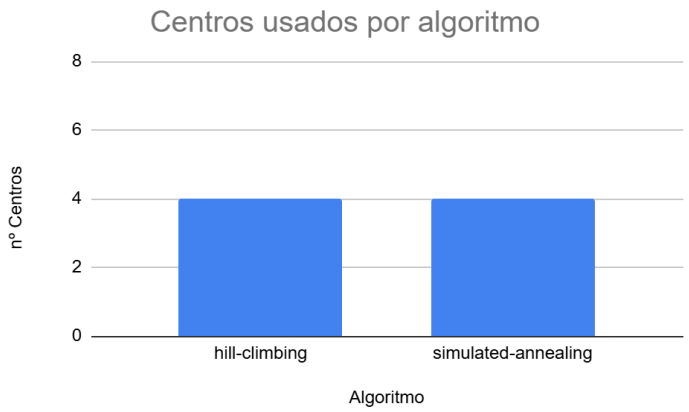



En este caso, podemos ver como la evolución de la función es lineal, por lo tanto podemos concluir que el coste temporal es polinómico.

# Experimento 5: Porcentaje de centros utilizados en los casos anteriores

El experimento 5 consiste en determinar si hay resultados en los experimentos anteriores que no utilizan todos los centros de datos debido a que las proporciones de recibir datos de todos los centros son mucho mayores a las de captura de todos los sensores.

Usando las condiciones de los experimentos anteriores y usando los 2 algoritmos disponibles, ejecutamos el experimento obteniendo los siguientes resultados:



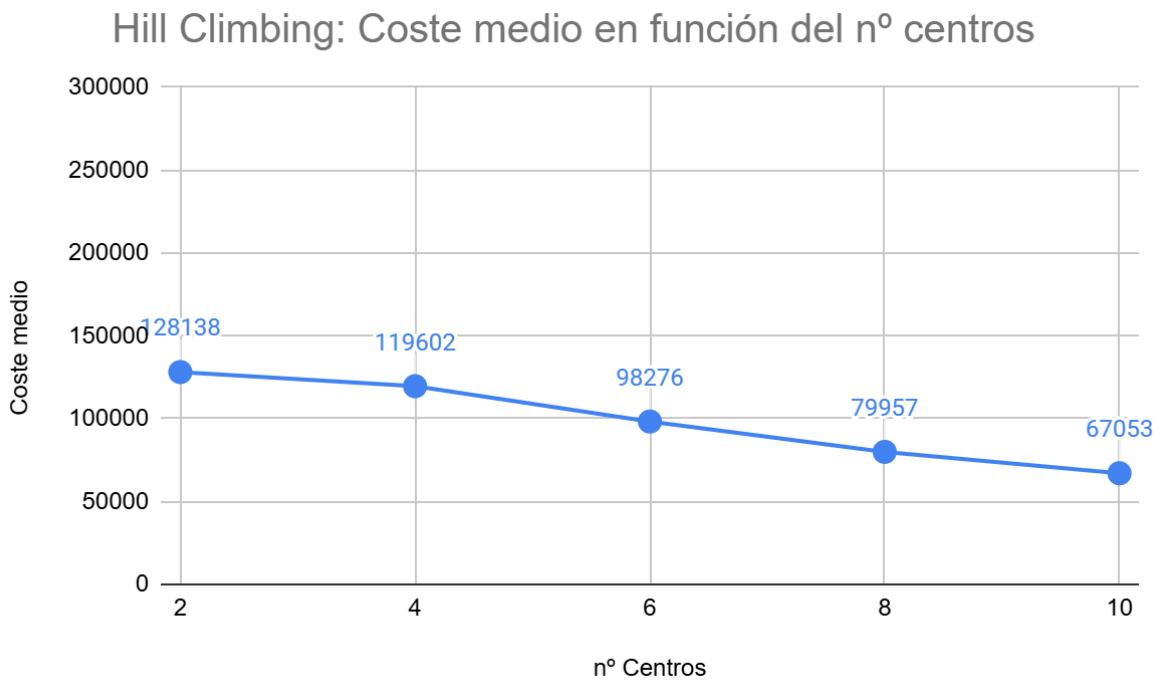

Como se puede ver, en ambos casos se utilizan todos los centros de datos disponibles como se esperaba, ya que de esta forma se aprovecha mejor las distancias entre caminos.

# Experimento 6: Hill Climbing vs Simulated Annealing

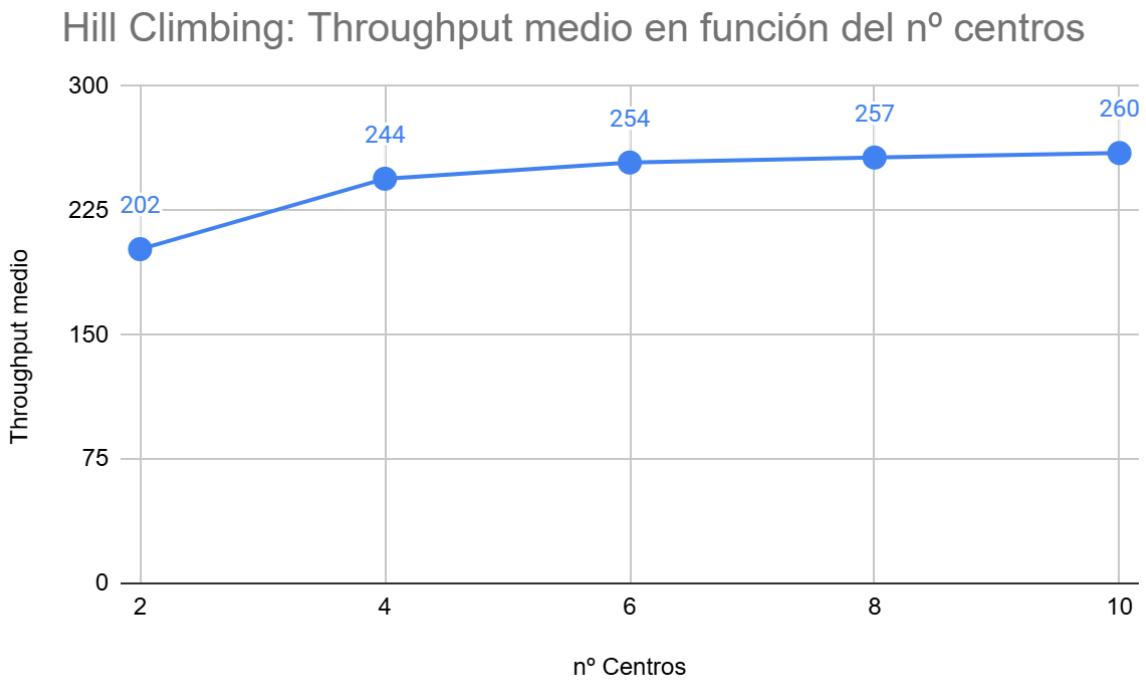
Para el experimento 6, se nos pide ver cómo afecta el número de centros de datos en el coste de la solución. También se nos pide ver cuántos de estos centros son utilizados en cada caso.
El número de sensores para este experimento se fija a 100, y los centros de datos se incrementan de dos en dos hasta llegar a 10.
También se nos pide analizar estos datos para cada uno de los siguientes algoritmos; Hill Climbing y Simulated Annealing.

## Hill Climbing

Dadas las condiciones mencionadas anteriormente y utilizando el algoritmo de Hill Climbing, los datos relacionados con el coste de la solución final, son los siguientes:

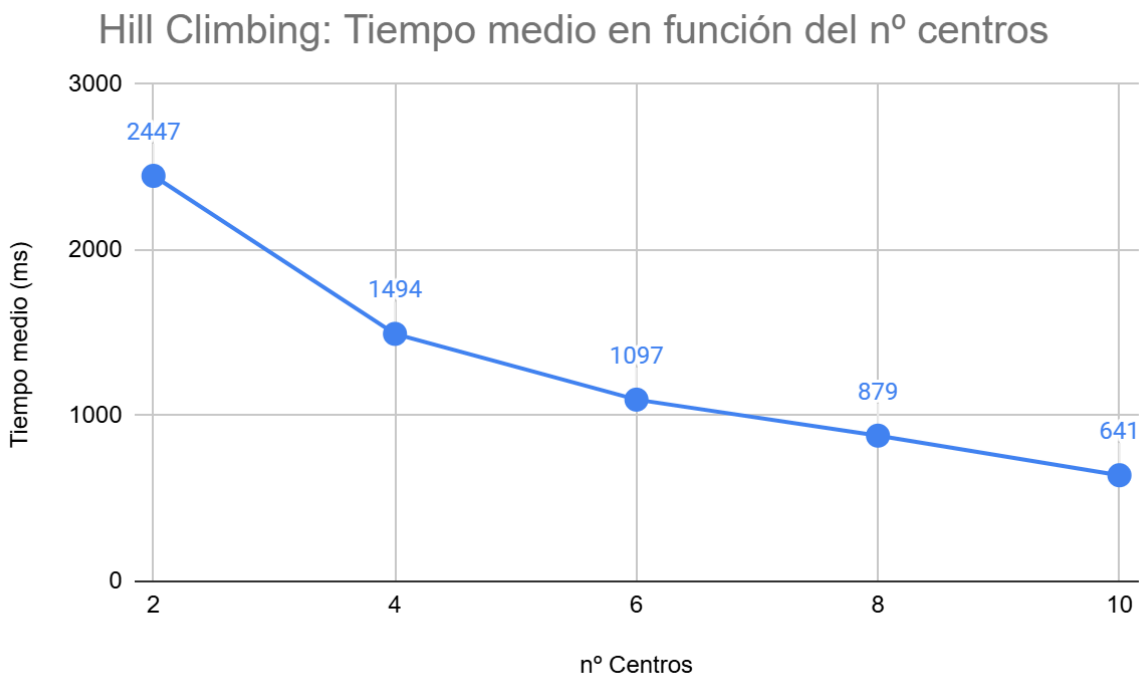


Podemos observar que, a medida que se añaden más centros de datos, el coste de la solución va disminuyendo progresivamente.
Por otro lado, los datos obtenidos relacionados con la información enviada con éxito, son los siguientes:

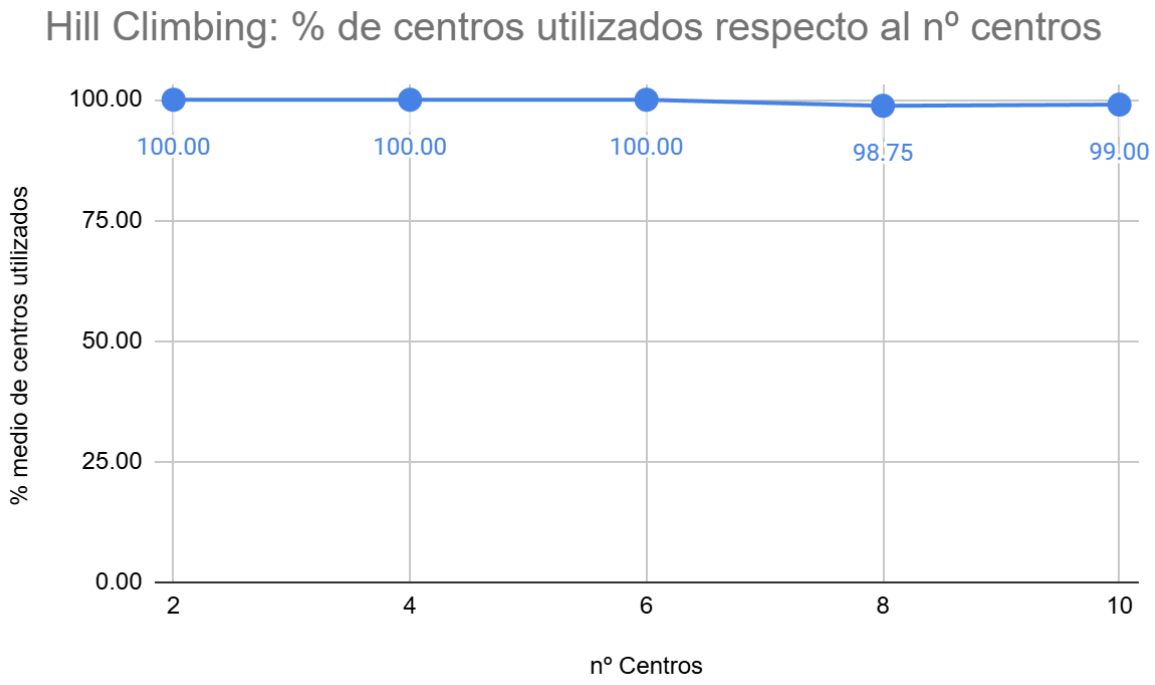



Podemos observar que, a medida que se aumenta el número de centros, los datos enviados con éxito se acercan cada vez más al máximo de las condiciones de esta solución, ya que, para un caso con 100 sensores la información enviada con éxito máximo tiene un límite superior de 265 Mb/s.

Por su parte, los datos obtenidos referentes al tiempo medio de ejecución del algoritmo, son los siguientes:



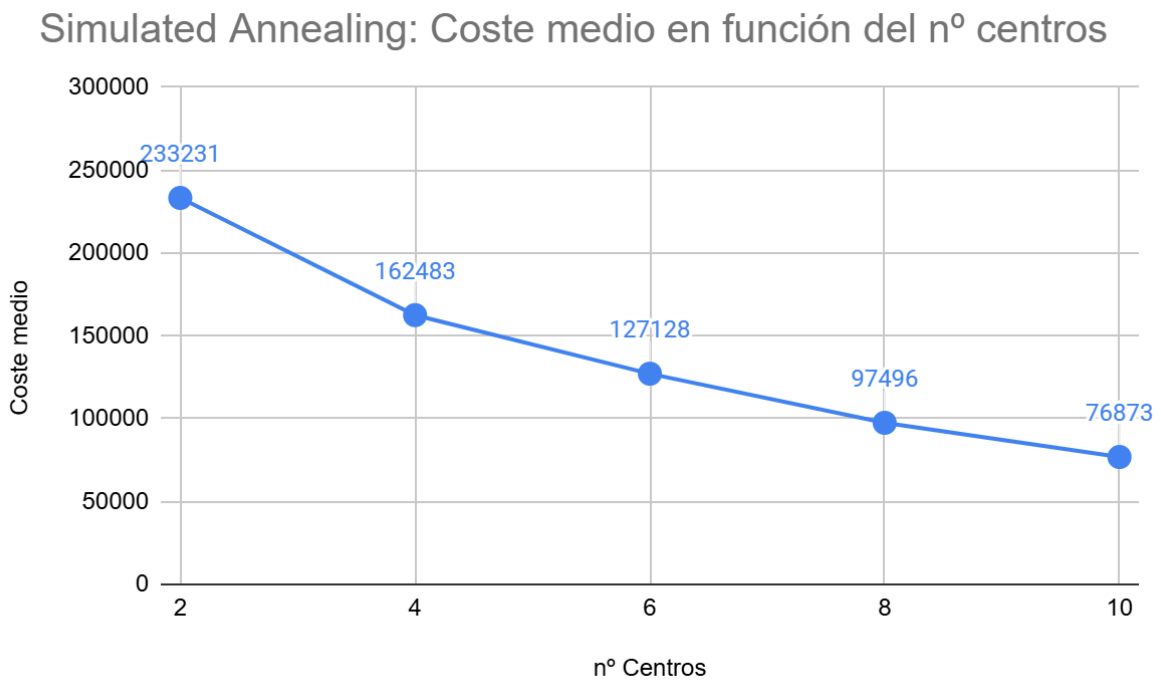

Podemos observar que, igual que en el caso del coste, a medida que se añaden centros de datos disminuye el tiempo de ejecución.
Por último, los datos obtenidos sobre el porcentaje de uso de los centros de datos, son los siguientes:



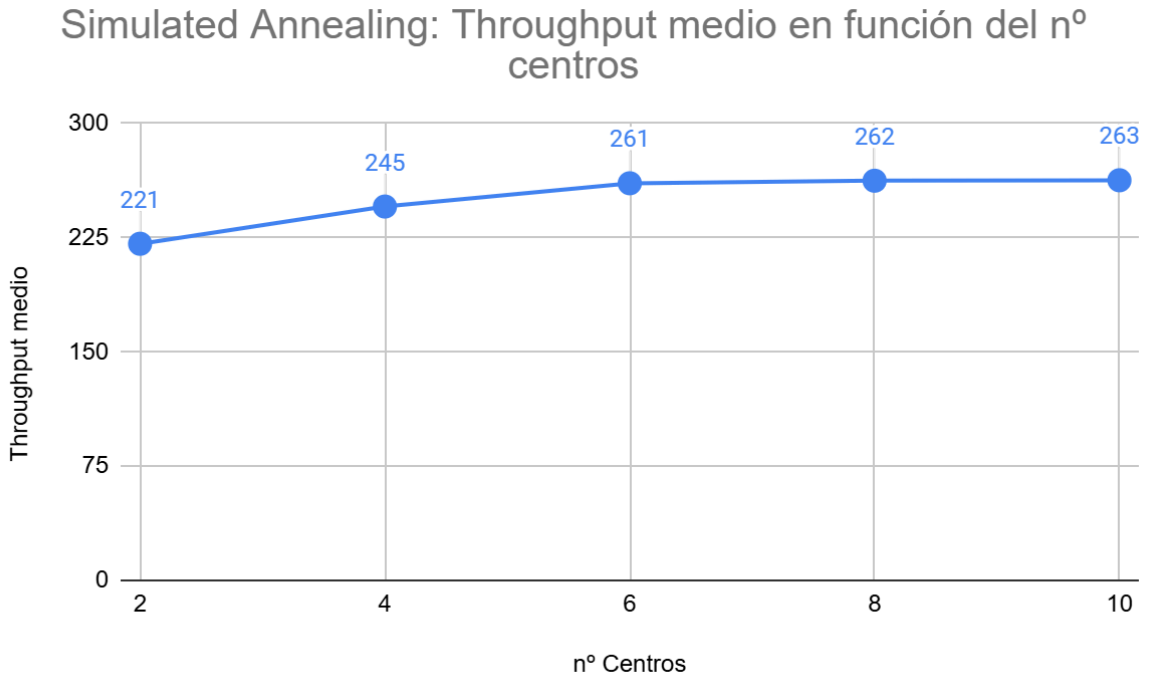

Podemos observar que prácticamente en todos los escenarios, se prioriza utilizar el máximo de centros de datos posibles.

## Simulated Annealing

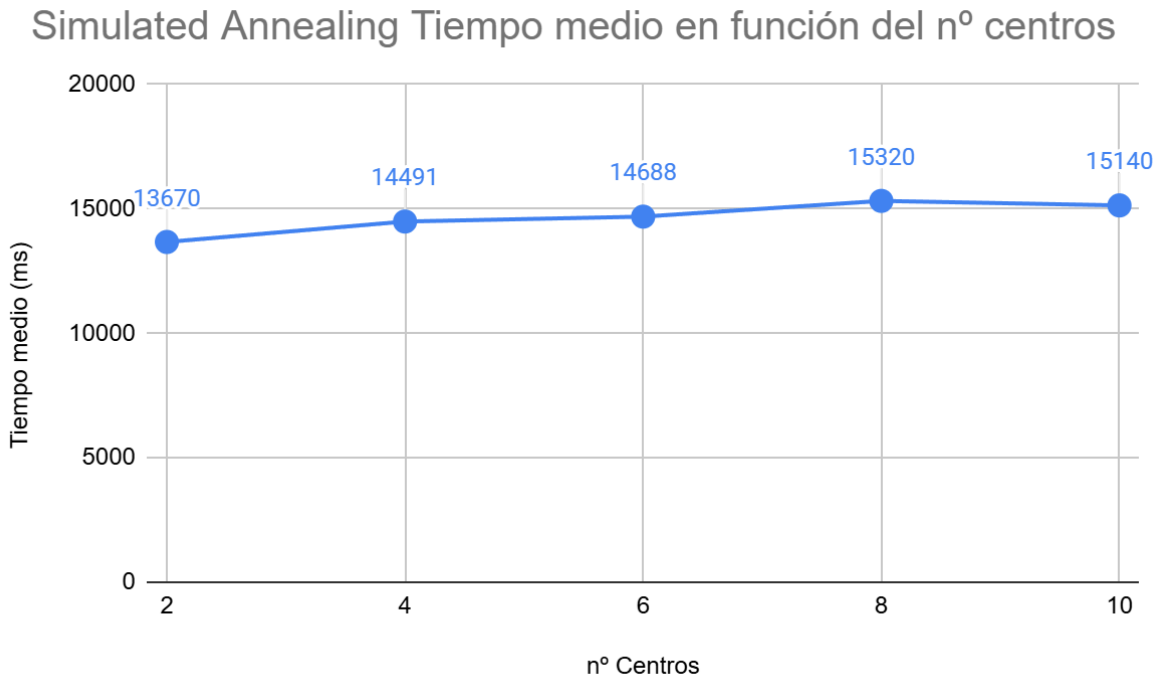
Dadas las condiciones mencionadas anteriormente y utilizando el algoritmo de Simulated Annealing con los parámetros obtenidos en el experimento 3, los datos relacionados con el coste de la solución final, son los siguientes:




Podemos observar que, a medida que se añaden más centros de datos, el coste de la solución va disminuyendo progresivamente.
Por otro lado, los datos obtenidos relacionados con la información enviada con éxito, son los siguientes:

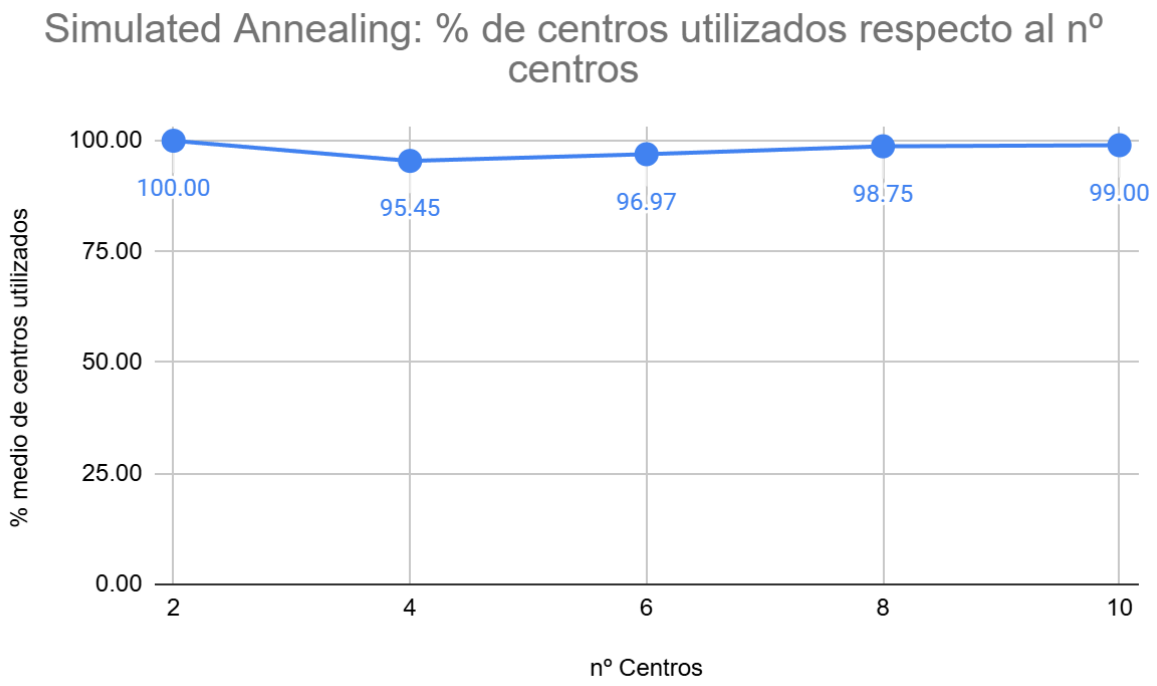


Podemos observar que, a medida que se aumenta el número de centros, los datos enviados con éxito se acercan cada vez más al máximo de las condiciones de esta solución, ya que, para un caso con 100 sensores la información enviada con éxito máximo tiene un límite superior de 265 Mb/s.
Por su parte, los datos obtenidos referentes al tiempo medio de ejecución del algoritmo, son los siguientes:




Dado que el simulated annealing, ejecuta siempre las mismas iteraciones independientemente del tamaño del problema, cabría esperar el mismo tiempo para cada escenario, pero observamos que no es así . Esto es debido a que a medida que se añaden centros de datos, el espacio de búsqueda incrementa ligeramente, haciendo que, para cada iteración haya más estados posibles que visitar implicando un mayor tiempo de cómputo.

Por último, los datos obtenidos sobre el porcentaje de uso de los centros de datos, son los siguientes:




Podemos observar que prácticamente en todos los escenarios, se prioriza utilizar el máximo de centros de datos posibles.

## Conclusiones

De este experimento podemos concluir que el porcentaje de uso de los centros de datos siempre es cercano al 100%. Por otro lado, podemos ver una diferencia entre el coste y los datos enviados con éxito en cada caso.

Para el coste, podemos ver que la disminución de este es mucho más pronunciada en el Simulated Annealing que en el Hill Climbing. Esto es debido a que el Hill Climbing encuentra soluciones menos costosas desde un inicio.

Para los datos enviados con éxito, podemos ver que el Simulated Annealing obtiene resultados ligeramente superiores que en el Hill Climbing, esto es debido a la naturaleza del algoritmo, que le permite salirse de mínimos locales y le permite explorar más allá, mientras que el Hill Climbing no tiene esta capacidad, y se queda estancado en el primer mínimo local que encuentra.

Como la heurística intenta maximizar los datos enviados con éxito y reducir el coste total, es normal que el Simulated Annealing tenga un mayor coste medio ya que, tiene más datos enviados con éxito de media.

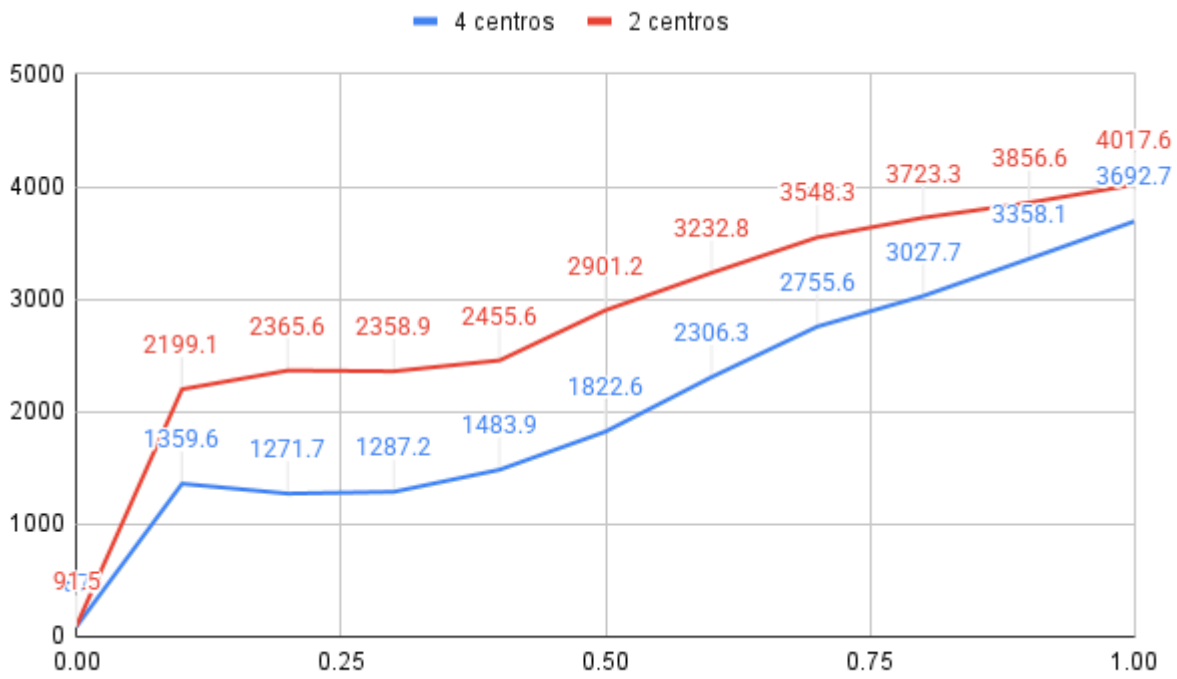
Lo mismo para el Hill Climbing, con la única diferencia que se queda estancado en el primer mínimo local que encuentra.

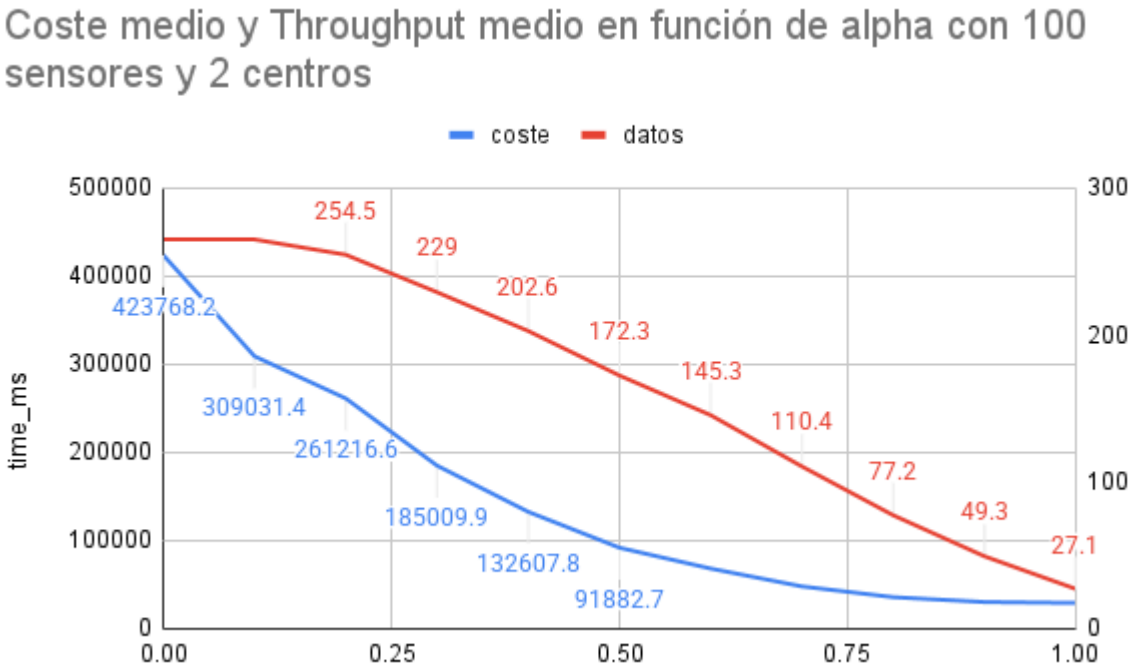
Los tiempos medios no pueden ser comparados entre ambos algoritmos ya que tienen un comportamiento totalmente diferente el uno del otro.

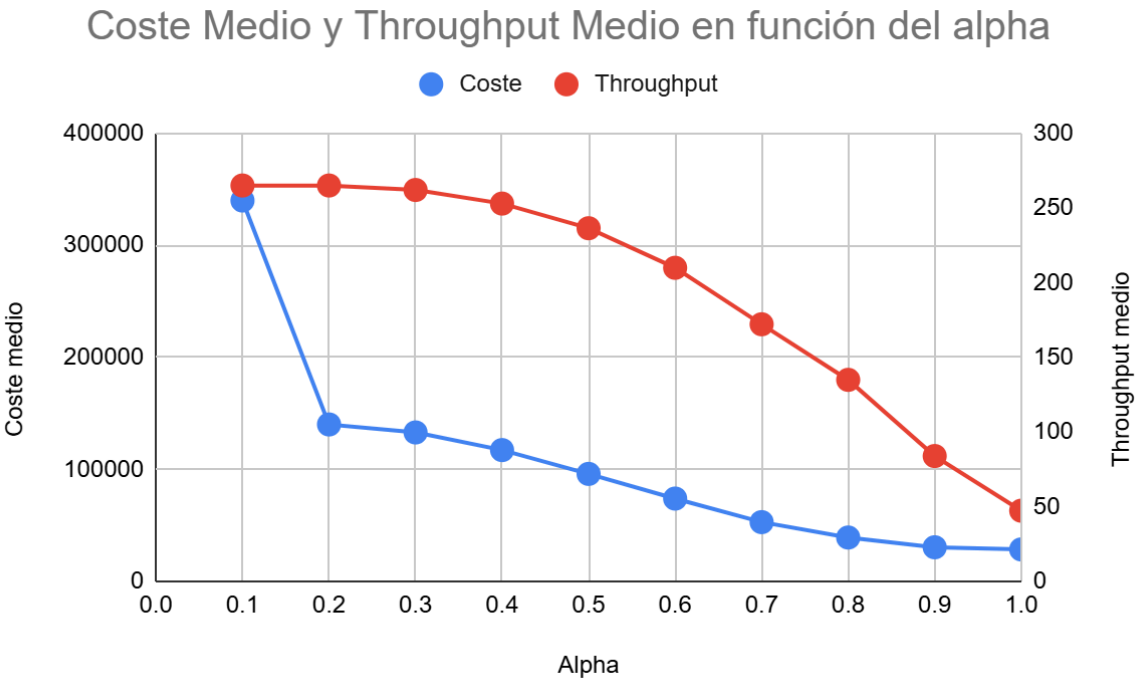
# Experimento 7: Pruebas de ponderaciones

Para ejecutar este experimento vamos a trabajar con la heurística seleccionada en la sección Función heurística final, que es la que hemos estado utilizando hasta ahora. La particularidad ahora es que ejecutamos con 100 sensores y 2 centros en vez de 100 sensores y 4 centros. Vamos a ver cómo afecta este cambio a los resultados obtenidos.

En cuanto a los tiempos de ejecución, con 2 centros en vez de 4 tenemos unos tiempos de ejecución mayores. Con 4 centros se puede converger a un mínimo local más rápido, reduciendo así el tiempo de ejecución.






Para alpha = 0, que da más peso al volúmen de datos enviados, tenemos los siguientes resultados:

| | **Coste** | **Throughput** |
|---|---|---|
| 100 sensores, 4 centros | 162038.3 | 265 |
| 100 sensores, 2 centros | 423768.2 | 265 |

En el caso de 100 sensores con 2 centros, conseguimos un Coste 2,62 veces mayor que con 100 sensores y 4 centros.

En ambos casos se puede observar que el coste aumenta cuando mejoramos el volúmen de datos enviados, y disminuye cuando empeoramos el volúmen de datos enviados. Cuando intentamos mejorar uno, el otro empeora. Sobre 0.4 y 0.5 encontramos un punto que consideramos un compromiso razonable entre los datos que perdemos y la mejora que tenemos sobre el coste.

Si comparamos los resultados obtenidos podemos observar que con 100 sensores y 4 centros se obtienen mejores resultados. En concreto, el coste cae de forma más abrupta, y el volumen de datos transmitidos tiene una caída más suave al principio.