Documentación de la práctica de búsqueda local

1er Cuatrimestre - curso 2016/2017

UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

Mario Fernández Villalba
Juan Miguel de Haro Ruiz
Carlos Roldán Montaner

ÍNDICE

1. Identificación del problema

Una compañía ficticia llamada Ázamon, que se dedica a organizar cada día los paquetes que debe enviar a una ciudad, nos ha pedido resolver un problema con los siguientes elementos:

- Paquetes, cada uno con un peso y prioridad determinados. El peso de los paquetes es siempre múltiplo de 0.5 kg y no superior a 10 kg. La prioridad de un paquete indica en cuántos días como máximo se entregará. Hay tres tipos de prioridad:

- **Día siguiente:** El paquete será entregado en un día con un coste de 5 euros.
- **2-3 días:** El paquete será entregado en un período de 2 a 3 días con un coste de 3 euros.
- **4-5 días:** El paquete será entregado en un período de 4 a 5 días con un coste de 1.5 euros.

- Ofertas, que son las distintas maneras que tenemos de llevar los paquetes a su destino. Cada oferta tiene un peso máximo determinado (entre 5 y 50 kg en intervalos de 5 kg), el precio por kilogramo transportado y el número de días en los que los paquetes llegarán a su destino (entre 1 y 5 días).

- Coste, que definimos como el dinero que le cuesta a la empresa enviar los paquetes a una ciudad. Depende de a qué ofertas de transporte asignemos cada paquete, ya que el precio por kilogramo es diferente en cada oferta.

Si un paquete llega antes de lo esperado según su prioridad, consideramos que el cliente está más contento con la empresa y por lo tanto es más probable que vuelva a comprar en Ázamon. Mediremos la **felicidad** de un cliente como los días de antelación con los que llega el paquete que ha solicitado a su destino.

Las restricciones del problema son las siguientes:

- Un paquete no puede llegar a su destino más tarde de lo que indica su prioridad.
- Todos los paquetes deben ser entregados.
- La suma del peso de todos los paquetes de una oferta no puede superar el peso máximo de la oferta.

Los objetivos del problema son minimizar el coste y maximizar la felicidad total de los clientes.
Una solución del problema consiste en asignar todos los paquetes disponibles a ofertas respetando las restricciones. Esto implica que hay una solución por cada asignación de paquetes válida que exista. Por lo tanto, identificamos el problema como de búsqueda local ya que tendremos que buscar una solución aceptable según los objetivos del problema de entre todo el espacio de soluciones.

2. Estado del problema y representación

Una solución del problema es un conjunto de asignaciones de n paquetes a m ofertas, el coste de estas y la felicidad que proporcionan a los clientes cumpliendo todas las restricciones del problema. Si no las tenemos en cuenta podemos estimar una cota superior para el espacio de búsqueda. Considerando que cada paquete puede ser asignado a m ofertas, el número de combinaciones posibles con n paquetes será $\boldsymbol{m^n}$.

Valoramos tres maneras distintas de representar un estado del problema:

1) Representar las asignaciones en un vector v de mapas de paquetes. Cada v_i contiene un conjunto de paquetes asignados a la oferta y la suma de todos sus pesos. Esta opción tiene un coste espacial de $n+m$, aunque también hay que considerar que habrá un coste adicional por los punteros de los mapas del vector. Esta representación implica que los cambios de las asignaciones tendrán coste logarítmico debido al uso de mapas.
2) Representar las asignaciones en un único vector v de enteros. Cada v_i contiene el índice de la oferta donde está asignada el paquete i. Esta opción tiene un coste espacial de n. Esta representación implica que los cambios de las asignaciones son constantes pero habrá coste lineal en n para calcular el peso actual de una oferta.
3) Es la misma representación que en 2) pero con un vector w adicional de números reales. Cada w_i contiene la suma de los pesos de los paquetes asignados a la oferta i. Esta representación implica un coste constante para cambios en las asignaciones y para calcular el peso actual de una oferta.

Estas tres implementaciones del estado nos permiten representar todas las soluciones del espacio de soluciones del problema, pero decidimos escoger la 3) ya que en cuanto a eficiencia temporal es la mejor y solo nos añade un coste espacial lineal respecto a m.

3. Representación y análisis de los operadores

Para movernos por el espacio de soluciones hemos considerado dos operadores: mover un paquete e intercambiar paquetes.

El operador de mover un paquete cambia su asignación de una oferta a otra.
Sus condiciones de aplicabilidad son los siguientes:

- La suma del peso del paquete más el peso de todos los paquetes asignados a la nueva oferta no puede sobrepasar el peso máximo de la nueva oferta.
- La prioridad del paquete debe ser concorde con los días de entrega de la nueva oferta.

Sus efectos son los siguientes:

- Se elimina la felicidad que proporcionaba el paquete en su anterior oferta y se añade la felicidad que proporciona en la nueva.
- Se elimina el coste del paquete en su anterior oferta y se añade su coste en la nueva.
- Al peso actual de la oferta a la que se asigna el paquete se le añade el peso de éste y a la oferta a la que estaba asignado anteriormente, se le sustrae.

Sea *n* el número de paquetes y *m* el número de ofertas podemos estimar una cota superior del factor de ramificación:

$$O(n * (m - 1))$$

El operador de intercambiar un paquete asigna cada paquete a la oferta del otro.
Sus condiciones de aplicabilidad son los siguientes (sean los dos paquetes a intercambiar P1 y P2, y sus respectivas ofertas O1 y O2):

- La suma del peso del P1 más el peso de todos los paquetes asignados a O2 menos el peso de P2 no puede sobrepasar el peso máximo de O2. De manera análoga, la suma del peso de P2 más el peso de todos los paquetes asignados a O1 menos el peso de P1 no puede sobrepasar el peso máximo de O1.
- La prioridad de P1 debe ser concorde con los días de entrega de O2 y la prioridad de P2 debe ser concorde con los días de entrega de O1.

Sus efectos son los siguientes:

- Se elimina la felicidad que proporcionaba el P1 en O1 y se añade la felicidad que proporciona en O2. Simétricamente, se elimina la felicidad que proporcionaba P2 en O2 y se añade la felicidad que proporciona en O1.
- Se elimina el coste de P1 en O1 y se añade su coste O2. Simétricamente, se elimina el coste de P2 en O2 y se añade su coste en O1.
- Se sustrae en O1 el peso de P1 y se añade el de P2. Se sustrae en O2 el peso de P2 y se añade el de P1.

Sea *n* el número de paquetes y *m* el número de ofertas podemos estimar una cota superior del factor de ramificación:

$$\sum_{i=1}^{n-1}(n-i) \;=\; n*(n-1) - \sum_{i=1}^{n-1} i \;= n*(n-1) - \frac{n*(n-1)}{2} = \frac{n^2-n}{2} \;=\; O(n^2)$$

Estos operadores no nos sacan del espacio de soluciones válidas, ya que si todos los paquetes están asignados a una oferta y aplicamos cualquier operador, quedan igualmente asignados y cumpliendo las restricciones. Además, el operador de intercambiar no genera todas las soluciones del espacio ya que nunca podremos alterar el número de paquetes asignado a cada oferta, por lo cual lo aplicamos juntamente con el operador de mover que si las genera. Decidimos usar sólo el operador de mover, la explicación detallada del porqué está disponible en el **apartado 6.1**.

4. Análisis de la función heurística

En el problema se nos pide buscar soluciones siguiendo dos criterios distintos: una donde el coste de las asignaciones y de almacenamiento sea mínimo y otra donde el coste sea mínimo y la felicidad sea máxima. Consecuentemente, hemos diseñado dos funciones heurísticas para los dos criterios citados.

La primera función heurística devuelve el coste de la solución actual, por lo tanto éste será el único factor que interviene. Al sólo haber un factor interviniendo en la función no es necesario ponderar. Durante la búsqueda, el algoritmo siempre irá a soluciones donde el coste sea menor debido a que queremos minimizar el coste de las asignaciones y de almacenamiento.

La segunda función heurística devuelve el coste de la solución actual menos la felicidad de ésta multiplicada por un factor. Ya que la felicidad es menor que el precio es necesario este factor para ponderar. El factor nos indica cuánto dinero vale una unidad de felicidad, por lo tanto aumentar este número provocará que el algoritmo vaya por soluciones donde la felicidad sea mayor aunque su coste también aumente.

Hemos escogido estas dos funciones porque son las que nos han parecido más razonables de entre todas las posibles.

5. Elección y generación del estado inicial

Para generar las soluciones iniciales hemos diseñado dos algoritmos:

Generación aleatoria

Ordenamos los paquetes por prioridad ascendente (primero estarán los paquetes que tienen que llegar antes) e iremos asignando cada uno a una oferta aleatoria, cumpliendo siempre las restricciones del problema. La ordenación sirve para que las ofertas con menos días de entrega se llenen con paquetes que se deban entregar antes. Haciendo esto imposibilitamos que, por ejemplo, un paquete de prioridad 0 no pueda colocarse en ninguna oferta debido a que las únicas ofertas donde pueda ser situado (las de 1 día de entrega) estén llenas con paquetes de mayor prioridad.

Con este algoritmo los paquetes quedarán repartidos uniformemente por todas las ofertas. El coste temporal es *O(n)* donde *n* es el número de paquetes, ya que hacemos solo una asignación por paquete. No podemos saber cuál será la bondad de la solución porque el resultado es aleatorio.

Generación ordenada

Ordenamos los paquetes de la misma forma que en el primer algoritmo y las ofertas ascendentemente por días de entrega. Intentamos llenar cada oferta con todos los paquetes que aún no están asignados y que al hacerlo siguen cumpliendo las restricciones, es decir, asignamos paquetes a cada oferta hasta que se alcanza su capacidad de peso máxima y entonces pasamos a la siguiente.

Con este algoritmo los paquetes quedarán repartidos en las primeras ofertas, y las que tardan más en realizar la entrega quedarán vacías o parcialmente vacías. El coste temporal es *O(n)* donde *n* es el número de paquetes, ya que cada paquete solo se asignará una vez. La felicidad de la solución inicial será alta, ya que asignaremos los paquetes a las ofertas que tarden menos, pero a la vez éstas son las que tienen un precio por kg mayor, por lo tanto el precio será alto también.

De estos dos algoritmos hemos escogido el primero, la explicación detallada del porqué está disponible en el **apartado 6.2**.

6. Experimentos

6.1 Influencia de los operadores

Observación	Pueden existir combinaciones de operadores mejores que otras, es decir, que nos lleven a una solución mejor o más rápido que otras.
Planteamiento	Escogemos diferentes grupos de operadores y observamos sus soluciones.
Hipótesis	Todos los grupos de operadores son iguales (H_o) o hay algunos mejores que otros.
Método	● Elegimos 25 semillas de manera aleatoria, una para cada réplica. ● Ejecutamos 1 experimento para cada semilla con el operador mover. ● Ejecutamos 1 experimento para cada semilla con los operadores mover y intercambiar. ● Experimentos con problemas de 100 paquetes y proporción 1.2 ● Usamos el algoritmo Hill Climbing. ● Medimos el coste, el número de pasos y el tiempo de búsqueda.

Una vez ejecutados los experimentos hemos recopilado los datos en gráficas para su posterior análisis. Las medias de éstos son las siguientes:

	Mover	**Mover y Intercambiar**
Coste promedio	962.8898	961.2158
#Pasos promedio	45.36	48.8
Tiempo(ms) promedio	19.28	116.28

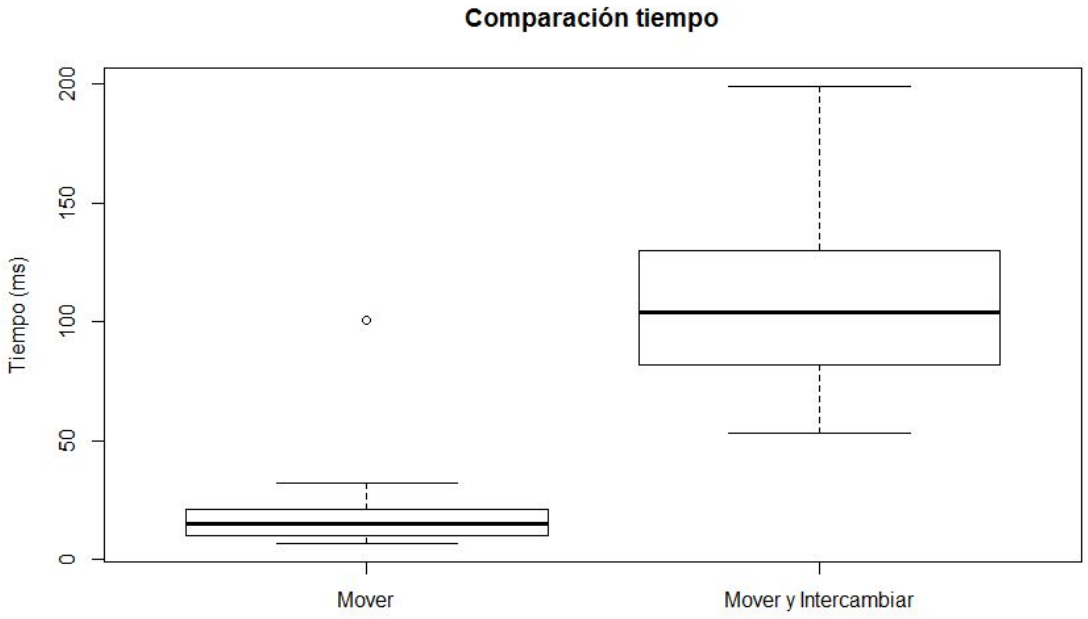

Figura 1 - Comparación del tiempo de ejecución de los dos conjuntos de operadores.

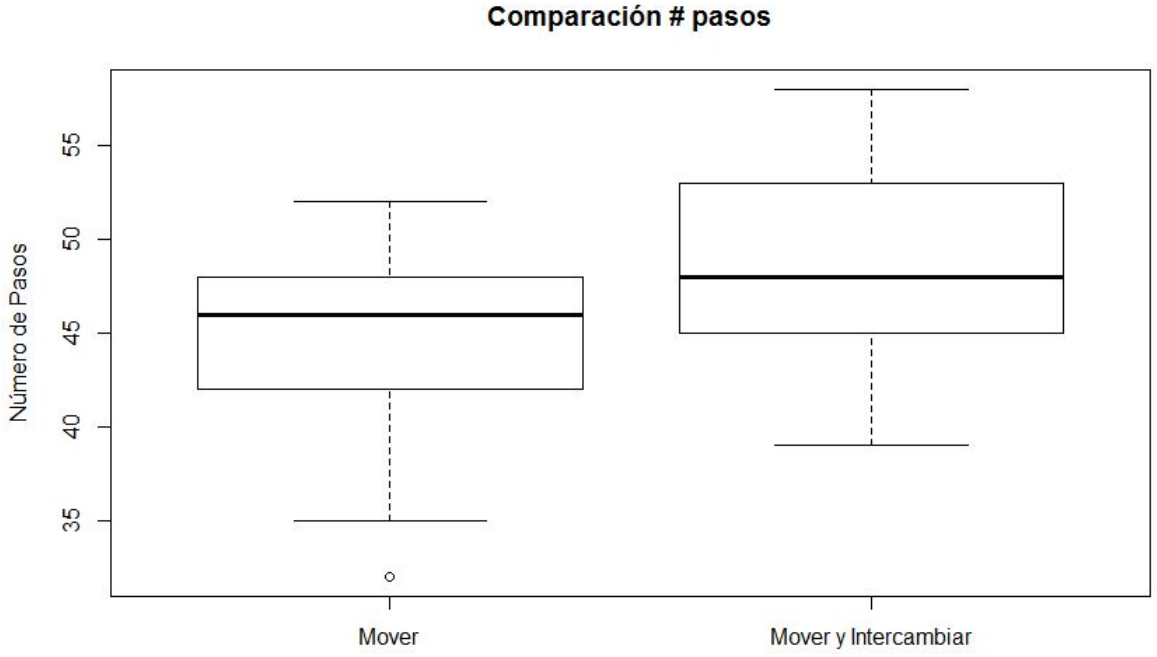

Figura 2 - Comparación del número de pasos de los dos conjuntos de operadores.

Como se puede observar en la gráfica de la ***Figura 1*** y en las medias, el operador de mover tarda siempre menos en tiempo. Esto es debido al factor de ramificación, ya que el de mover es $O(n*(m-1))$ y el de mover e intercambiar es $O(n*(m-1))+O(n^2)$.
En cambio, en cuanto al número de pasos no está tan claro cuál es mejor. Observando los datos recogidos podemos ver que el operador de mover siempre hace menos pasos que el de mover e intercambiar, en media unos 3 pasos menos, lo cual no es muy significativo.

En cuanto al precio, hemos omitido la gráfica ya que solo con analizar los datos podemos observar que la diferencia entre los resultados es mínima. El conjunto de operadores de mover e intercambiar en la mayoría de casos es mejor, pero con una diferencia media de 1€, por lo tanto tampoco es significativo. Estos resultados podrían deberse a que la combinación de mover e intercambiar permite al algoritmo moverse más ampliamente por el espacio de soluciones. Por lo tanto, es más probable que llegue a un resultado mejor pero en más pasos, ya que tiene más opciones que considerar a la hora de generar los sucesores.

Como conclusión, podemos deducir a partir de los datos que tanto el operador de mover como el de mover e intercambiar obtienen el mismo resultado en pasos y coste, pero el de mover es significativamente más rápido, por lo que fijamos este operador para el resto de experimentos.

6.2. Influencia de la solución inicial

Observación	Pueden existir métodos de inicialización mejores que otros.
Planteamiento	Escogemos dos algoritmos de generación del estado inicial.
Hipótesis	Todos los algoritmos de generación de estado inicial son iguales (H_o) o hay algunos mejores que otros.
Método	● Elegimos 25 semillas de manera aleatoria, una para cada réplica. ● Ejecutamos 1 experimento para cada semilla con la generación ordenada. ● Ejecutamos 5 experimentos para cada semilla con la generación aleatoria, y hacemos la media de los 5 resultados. ● Experimentos con problemas de 100 paquetes y proporción 1.2 ● Usamos el algoritmo Hill Climbing. ● Medimos el coste, el número de pasos y el tiempo de búsqueda.

	Ordenado	Aleatorio
Coste promedio	997.7596	998.21236
#Pasos promedio	61.32	45.48
Tiempo (ms) promedio	19.72	17.56

Como se puede observar, los pasos que realiza el Hill Climbing con el generador ordenado son mayores que los del aleatorio. Esto se debe a que la solución que genera el primer algoritmo está más lejos de un mínimo (local o absoluto). El motivo es que los paquetes están asignados a las primeras ofertas, que son las que tienen más coste por kg, por lo que la búsqueda debe realizar más pasos.

En cambio, el generador aleatorio empieza con una configuración que, con mayor probabilidad, estará más cerca de un mínimo que el generador ordenado. El tiempo de ejecución es proporcional al número de pasos

El coste varía muy poco de un generador a otro. En promedio, el ordenado tiene menos de 1€ de diferencia y analizando los datos en algunas ocasiones uno supera al otro y viceversa. Esto puede ser porque al ser aleatorio, el segundo algoritmo generará una solución que lleve a un mínimo que podrá ser mejor o peor que el mínimo al que puede llegar el generador ordenado.

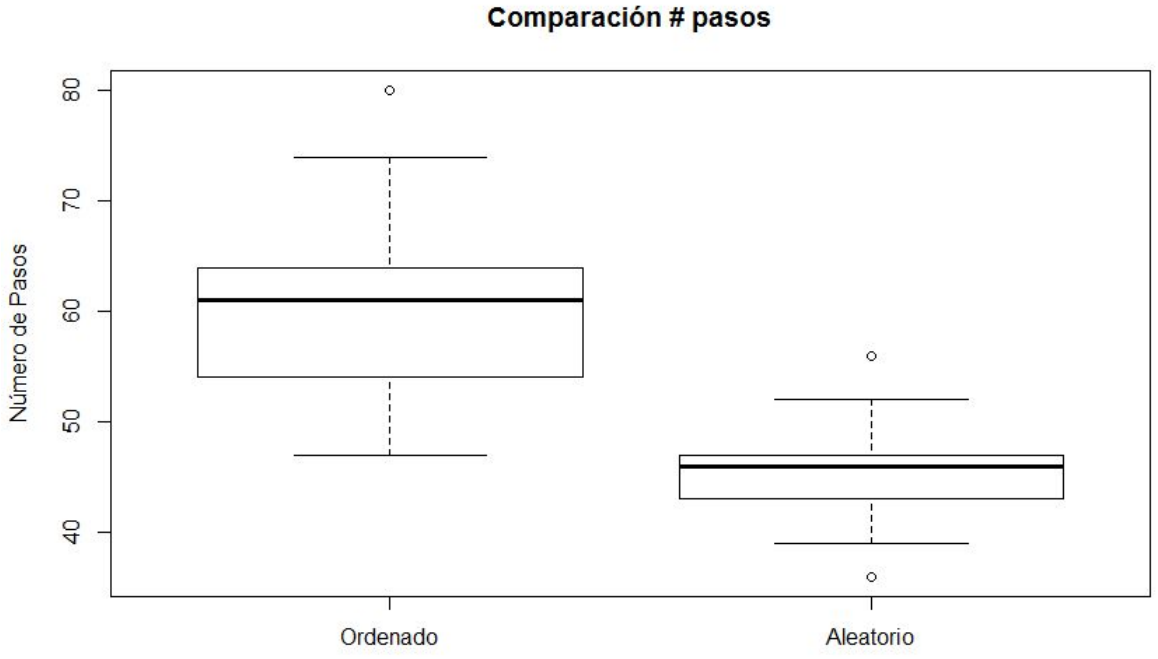

Figura 3- Comparación del número de pasos del generador ordenado con el aleatorio.

Como conclusión, podemos confirmar que el generador aleatorio será mejor ya que generará un estado inicial que llevará a un resultado muy similar pero con menos pasos y, consecuentemente, en menor tiempo.

6.3. Influencia de los parámetros del Simulated Annealing

Observación	Dependiendo de los valores *k* y *lambda* el algoritmo Simulated Annealing puede dar mejor o peores resultados que el algoritmo Hill Climbing.
Planteamiento	Escogemos diferentes valores de *k* y *lambda* y observamos el comportamiento de la búsqueda.
Hipótesis	Los parámetros *k* y *lamba* no influyen en la bondad de las soluciones de Simulated Annealing (H_o) o si influyen.
Método	● Elegimos 25 semillas de manera aleatoria, una para cada réplica. ● Estudiamos el comportamiento del Simulated Annealing para $k \in \{5, 50, 500, 5000, 50000\}$ y $lambda \in \{0.1, 0.01, 0.001, 0.0001\}$. ● Ejecutamos 5 experimentos para cada semilla con el algoritmo Simulated Annealing probando todas las combinaciones posibles de *k* y *lambda*, y hacemos la media de los 5 resultados. ● Experimentos con problemas de 100 paquetes y proporción 1.2 ● Usamos el algoritmo Simulated Annealing con 10000 iteraciones y cambios de temperatura cada 100 iteraciones. ● Calculamos la media del coste de cada combinación de *lambda* y *k* en las 25 réplicas. ● Una vez escogidos los mejores valores de *k* y *lambda*, comparamos el tiempo de ejecución y la bondad de la solución de Simulated Annealing y Hill Climbing en un mismo escenario. Para ello elegimos 25 semillas de manera aleatoria y hacemos 25 réplicas.

K/LAMBDA	0.1	0.01	0.001	0.0001
5	935.73984	935.32616	933.83624	954.48012
50	935.77392	935.22108	934.19032	993.944
500	935.77768	935.19492	934.33112	998.94564
5000	935.72716	935.65424	938.10696	998.09892
50000	935.45272	935.30844	975.57356	997.92868

	Coste	Tiempo (ms)
Hill Climbing	951.446	18
Simulated Annealing	952.931	9

Una vez calculadas las medias, podemos observar que la mejor combinación de *lambda* y *k* en las 25 réplicas del experimento ha sido *k = 5* y *lambda = 0.001*, así que a partir de ahora fijamos estos valores para posteriores experimentos. La figura 4 nos muestra la evolución del coste del problema para una ejecución con *k = 5* y *lambda = 0.001*. Observamos que con 8000 iteraciones es suficiente ya que a partir de aquí la probabilidad de aceptar estados peores es casi nula. En la siguiente tabla tenemos las medias de los costes y tiempos de los dos algoritmos. Podemos observar que aunque la bondad de la solución dada por Hill Climbing y Simulated Annealing es casi la misma, Simulated Annealing se ejecuta más rápido que Hill Climbing. Es por esto que determinamos que Simulated Annealing es mejor algoritmo de búsqueda que Hill Climbing.

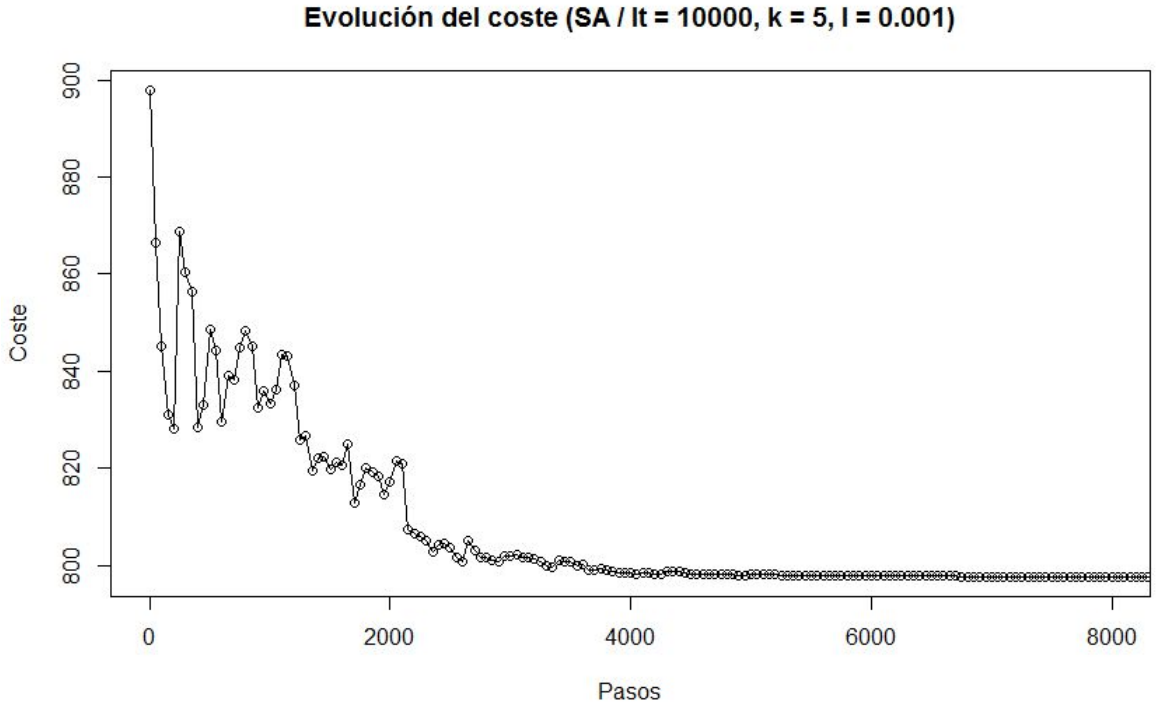

Figura 4- Evolución del coste en la ejecución del Simulated Annealing.

6.4. Influencia de los parámetros de generación de escenarios

Observación	Los parámetros de generación de escenarios (número de paquetes y proporción del peso transportable) afectan al tiempo de ejecución para hallar la solución.
Planteamiento	Ejecutamos el algoritmo de Hill Climbing con diferente proporción del peso transportable.
Hipótesis	El parámetro de proporción provoca un aumento cuadrático o mayor del tiempo.
Método	● Elegimos 10 semillas aleatoriamente, y por cada una generamos un conjunto de 100 paquetes diferentes. ● Ejecutamos el algoritmo de Hill Climbing para cada conjunto de paquetes 15 veces. ● Para cada una de las 15 ejecuciones aumentamos la proporción del peso transportable en 0.2 y generamos un nuevo conjunto de ofertas, empezando en 1.2 (de 1.2 hasta 4). ● Medimos el tiempo de ejecución para cada experimento.

Proporción	Media Tiempo
1.2	19.8
1.4	40.9
1.6	64.7
1.8	110.8
2	155.1
2.2	250.4
2.4	307.4
2.6	410.5
2.8	487
3	574.3
3.2	723
3.4	894.9
3.6	988.5
3.8	1181.3
4	1348.7

En los resultados del experimento podemos observar la misma tendencia: el crecimiento en tiempo de ejecución no es lineal, sino que este va aumentando a medida que lo hace la proporción. Esto se debe al factor de ramificación de los operadores, ya que aumentar la proporción implica aumentar el número de ofertas, por lo que según el coste temporal *O(n*(m-1))* del operador de mover y del tamaño del espacio de búsqueda $O(m^n)$, el tiempo total será más elevado.

Aunque no sea del todo evidente el grado de la función que define el crecimiento, podemos afirmar con bastante seguridad que será polinómico, ya que si comparamos con funciones polinómicas de grado 3 y 4 (**Figura 5**) el tiempo se mantiene en medio. Además, el factor de ramificación también es polinómico en cuanto al número de ofertas y paquetes. Suponemos también que el espacio de soluciones (de tamaño exponencial) no influye directamente en el algoritmo de búsqueda, sino que influye en el número de soluciones a explorar y en la posible existencia de más mínimos locales.

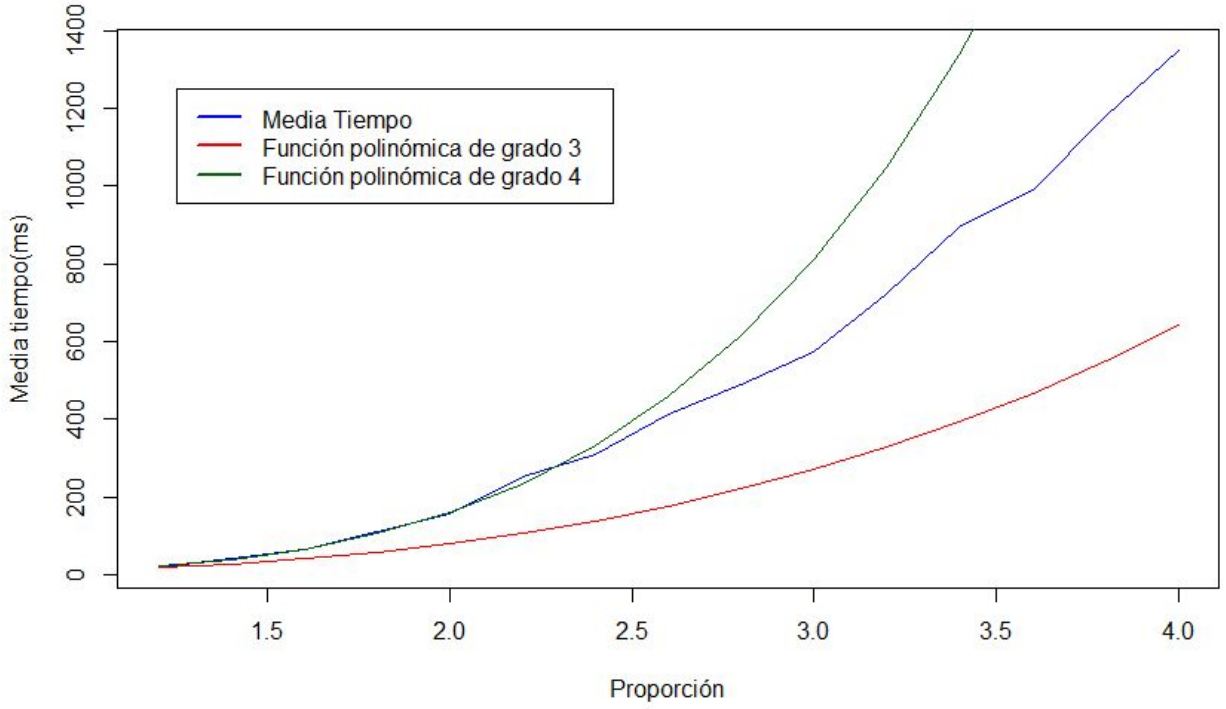

Figura 5 - Función del tiempo respecto a la proporción.

Observación	Los parámetros de generación de escenarios (número de paquetes y proporción del peso transportable) afectan al tiempo de ejecución para hallar la solución.
Planteamiento	Ejecutamos el algoritmo de Hill Climbing con diferente cantidad de paquetes.
Hipótesis	El parámetro de número de paquetes provoca un aumento cuadrático o mayor del tiempo.
Método	● Ejecutamos el siguiente experimento 10 veces. ● Creamos conjuntos de paquetes de 100, 150, 200, 250 y 300 y ejecutamos Hill Climbing para cada conjunto. ● Medimos el tiempo de ejecución.

Paquetes	Media tiempo
100	19.9
150	98.8
200	303
250	913.9
300	2266.9

En este caso se puede también observar la misma tendencia, el crecimiento. Si lo comparamos con una función polinómica de grado 5 y de grado 4 como podemos ver en la **Figura 6** observamos que la función de tiempo se mantiene entre las dos. Esto se debe a los mismos motivos explicados en el anterior experimento, el factor de ramificación de los operadores y el tamaño del espacio de búsqueda, que provocan un aumento de posibilidades a la hora de generar sucesores y un posible incremento de mínimos.

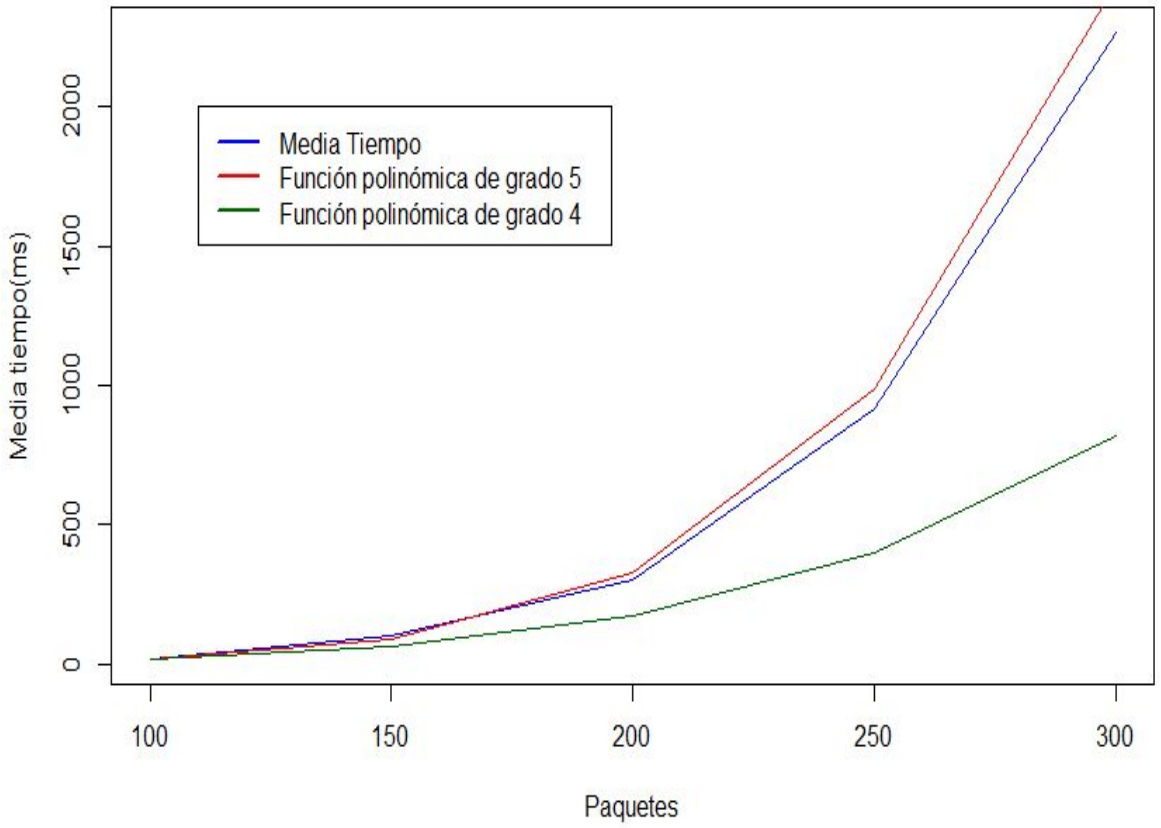

Figura 6 - Función del tiempo respecto al número de paquetes.

6.5. Influencia del parámetro de proporción en el coste

Observación	Los parámetros de generación de escenarios (número de paquetes y proporción del peso transportable) afectan al coste de la solución.
Planteamiento	Ejecutamos el algoritmo de Hill Climbing con diferente proporción del peso transportable.
Hipótesis	El parámetro de proporción provoca una disminución del coste de la solución final.
Método	● Elegimos 10 semillas aleatoriamente, y por cada una generamos un conjunto de 100 paquetes diferentes. ● Ejecutamos el algoritmo de Hill Climbing para cada conjunto de paquetes 15 veces. ● Para cada una de las 15 ejecuciones aumentamos la proporción del peso transportable en 0.2 y generamos un nuevo conjunto de ofertas, empezando en 1.2 (de 1.2 hasta 4). ● Medimos el coste para cada experimento.

Proporción	Media Coste
1.2	953.1875
1.4	931.189
1.6	918.8065
1.8	912.8685
2	895.577
2.2	895.029
2.4	886.3625
2.6	876.272
2.8	868.6875
3	862.73
3.2	863.105
3.4	861.9005
3.6	855.302
3.8	846.812
4	839.8015

Con una rápida observación del gráfico en la **Figura 7** y de la tabla de medias se puede deducir que el precio disminuye proporcionalmente a la proporción del peso transportable. Intuimos que es debido a que si aumentamos el número de transportes, habrá más ofertas donde el precio será bajo, por lo que el algoritmo tendrá más opciones para asignar los paquetes a estas ofertas, aunque por ese mismo motivo el coste temporal será mayor. Esto quiere decir que la calidad de la solución dependerá del tiempo que gastemos ejecutando y aunque sea un tiempo polinómico, el grado de éste es mayor que el de la función del coste. Por lo tanto aumentar la proporción valdrá la pena hasta un cierto punto en el que el aumento del tiempo será muy grande comparado con la disminución del coste.

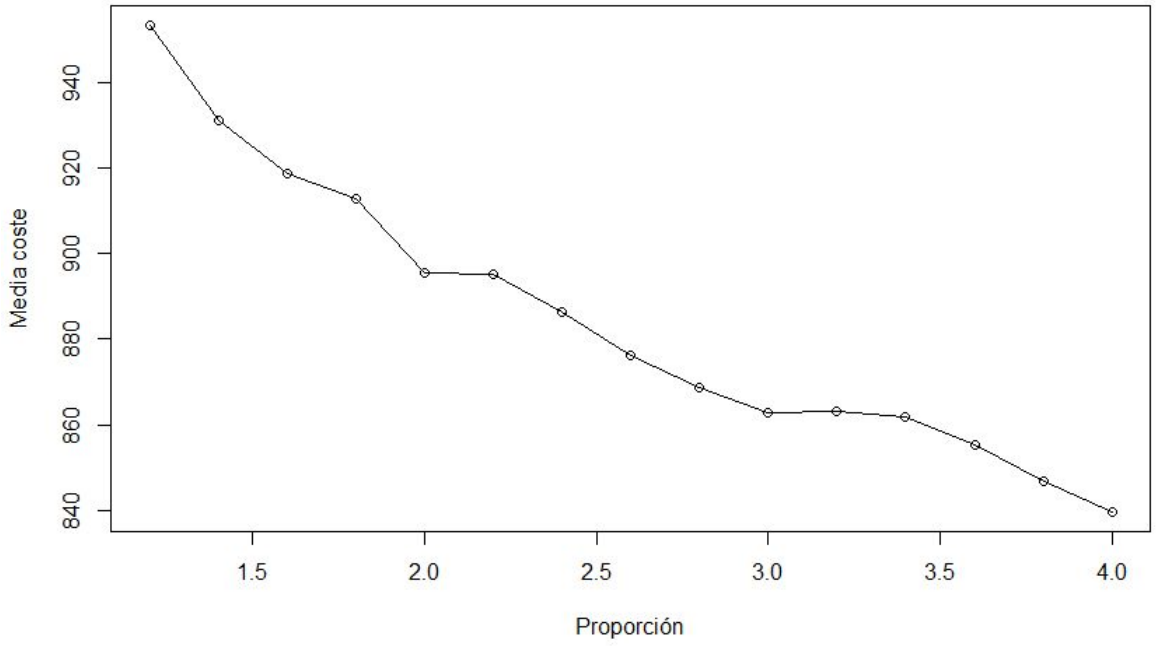

Figura 7-Función del coste de transporte y almacenamiento respecto a la proporción.

6.6. Influencia de la ponderación de la felicidad en Hill Climbing

Observación	La ponderación de la felicidad en la función Heurística del Hill Climbing puede afectar a los costes de transporte y almacenamiento, y también al tiempo de búsqueda.
Planteamiento	Modificamos la ponderación de la felicidad en la función heurística y observamos cómo repercute en los resultados.
Hipótesis	El coste y tiempo de búsqueda se mantienen iguales al modificar la ponderación de felicidad (H_o) o hay variaciones.
Método	● Elegimos 5 semillas de manera aleatoria, una para cada réplica. ● Empezamos con una factor de ponderación de 1 para la felicidad. ● Ejecutamos 15 experimentos para cada semilla,incrementando la ponderación cada vez. ● Experimentos con problemas de 100 paquetes y proporción 1.2 ● Usamos el algoritmo Hill Climbing. ● Medimos el número de pasos, el tiempo de búsqueda y el coste de transporte y almacenamiento.

Ponderación	Media Coste	Media Tiempo
1	927.972	24
2	959.697	16.6
3	993.308	13.6
4	1036.025	18.4
5	1062.979	10.4
6	1075.824	10.8
7	1089.936	16
8	1105.084	9
9	1114.897	7
10	1114.956	7.2
11	1113.942	7
12	1109.847	7
13	1116.18	8.8
14	1116.096	6.2
15	1116.262	6.4

Como podemos observar en la **Figura 8**, a medida que incrementamos la ponderación de la felicidad, el coste de transporte y almacenamiento incrementa, hasta llegar a un punto en que se mantiene constante. Esto es debido a que según aumenta la ponderación, la felicidad cobra más importancia que el coste en la función heurística, hasta llegar al punto en que ésta es tan importante que el coste no influye en el cálculo del heurístico.

Además, también podemos ver que a medida que aumentamos la ponderación, disminuye el tiempo, hasta que pasa a ser constante. Esto se debe a que al principio, cuando la ponderación es baja, el Hill Climbing tiene en cuenta estados sucesores con mayor felicidad o con menor precio; pero a medida que la felicidad cobra importancia, comienza a descartar los estados con menor precio, escogiendo sólo aquellos que proporcionan mayor felicidad.

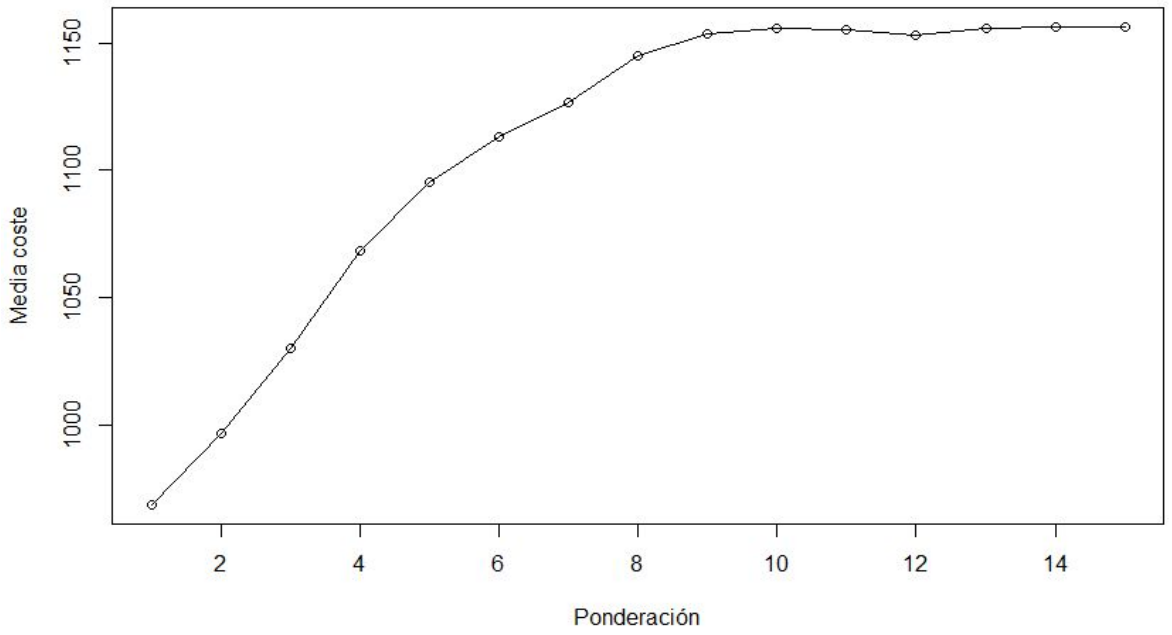

Figura 8 - Función de la media del coste del Hill Climbing en función de la ponderación de la felicidad.

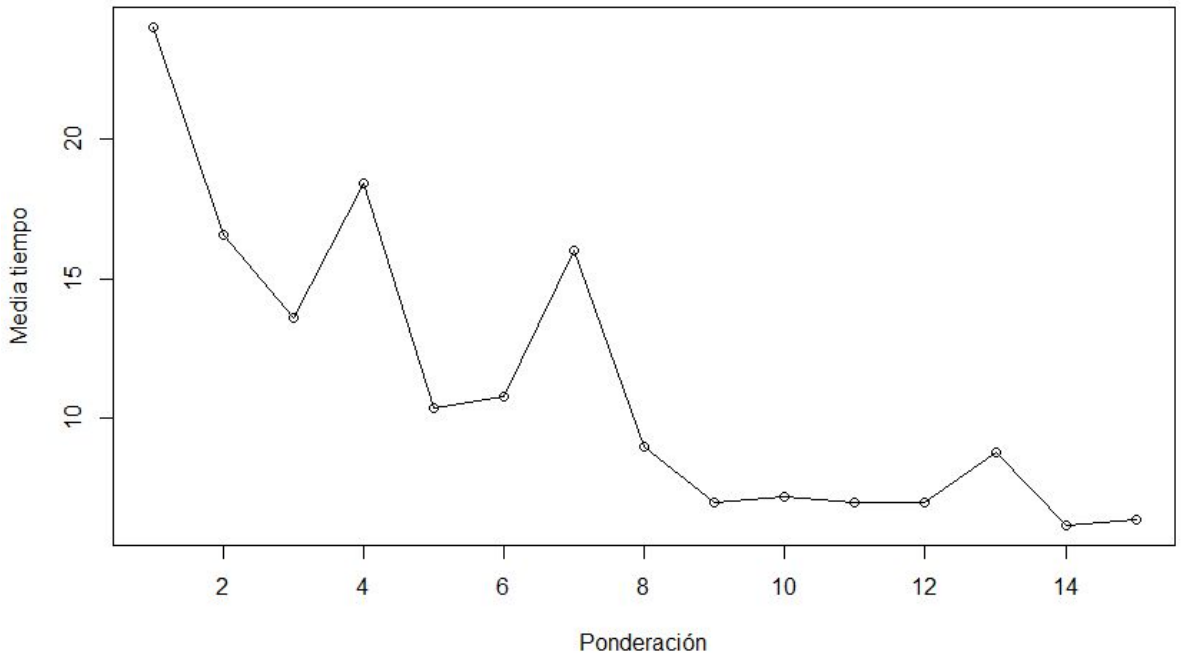

Figura 9 - Función de la media de tiempo de ejecución del Hill Climbing en función de la ponderación de la felicidad.

6.7. Influencia de la ponderación de la felicidad en Simulated Annealing

Observación	La ponderación de la felicidad en la función Heurística del Simulated Annealing puede afectar a los costes de transporte y almacenamiento, y también al tiempo de búsqueda.
Planteamiento	Modificamos la ponderación de la felicidad en la función heurística y observamos cómo repercute en los resultados.
Hipótesis	El coste y tiempo de búsqueda se mantienen iguales al modificar la ponderación de felicidad (H_o) o hay variaciones.
Método	• Elegimos 5 semillas de manera aleatoria, una para cada réplica. • Empezamos con una factor de ponderación de 1 para la felicidad. • Ejecutamos 15 experimentos para cada semilla,incrementando la ponderación cada vez. • Experimentos con problemas de 100 paquetes y proporción 1.2. • Usamos el algoritmo Simulated Annealing con parámetros *k = 5* y *lambda = 0.001*. • Medimos el coste de transporte y almacenamiento.

Como podemos observar en la **Figura 10**, a medida que incrementa la ponderación de la felicidad, el precio también aumenta hasta el punto en que se mantiene prácticamente constante. No es del todo constante (a diferencia de en el Hill Climbing) debido a la aleatoriedad del Simulated Annealing, que hace que a partir del mismo estado inicial genere soluciones distintas.

Esto es debido al mismo motivo que en el experimento 6.6.

No obstante, a diferencia de en el Hill Climbing, el tiempo en el Simulated Annealing es constante independientemente de la ponderación de la felicidad.

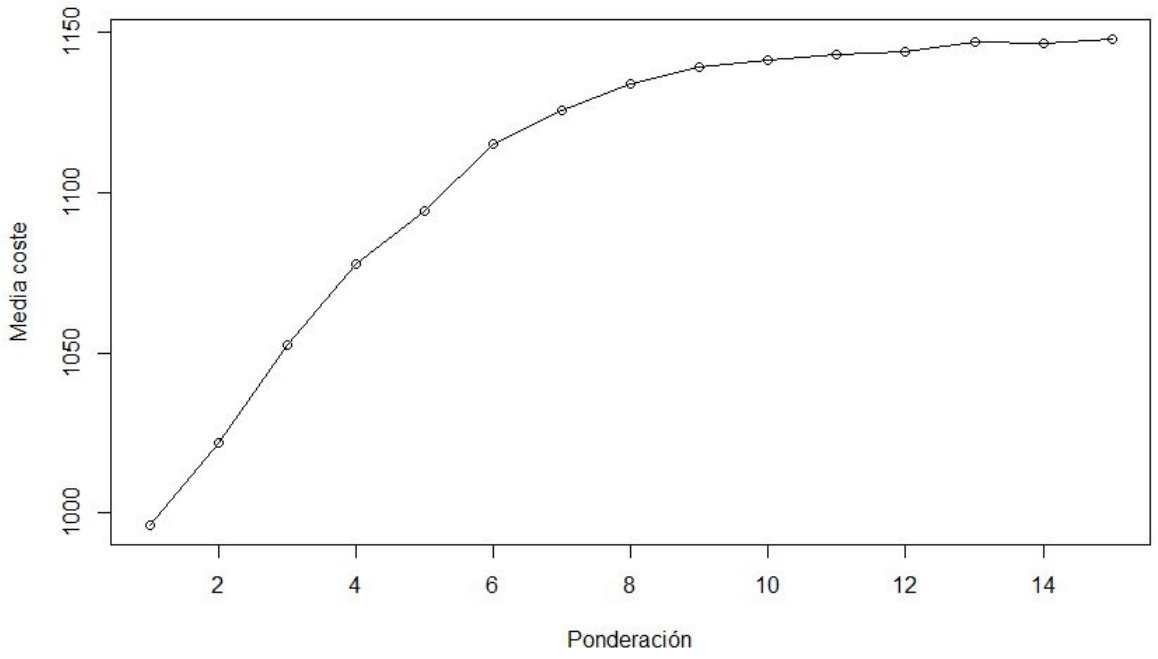

Figura 10 - Función de la media del coste del Simulated Annealing en función de la ponderación de la felicidad.

6.8. Influencia del coste de almacenamiento

Si aumentamos el coste de almacenamiento, implica que el coste de los transportes que tardan más días en entregar un paquete (que son los que tienen un coste por kg mas barato) aumentará, lo que significa que el algoritmo considerará en menor medida asignar paquetes a estos transportes, por lo tanto intentará asignar más a las ofertas que tardan menos, lo que también implica más felicidad.

Así, aumentar el precio de almacenamiento implica aumentar el coste de la solución final pero con mayor felicidad. Lo contrario pasará si lo bajamos, el algoritmo buscará más opciones donde los paquetes están asignados a ofertas que tarden más días porque serán más baratos, lo que se traducirá en una disminución de la felicidad.