Práctica de búsqueda local

Albert Pita, Carlos Gascón, Alejandro Alarcón

2º Cuatrimestre - Curso 2019/2020

UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

Índice general

Introducción

En esta práctica se plantean dos objetivos principales a realizar.

El primero es aprender diferentes técnicas basadas en la búsqueda local usando la librería de AIMA en Java para resolver problemas de esta índole. En particular las técnicas son dos, el algoritmo *Hill Climbing* y el *Simulated Annealing*.

El segundo es realizar un estudio comparativo entre los dos algoritmos mencionados con anterioridad mediante diversos experimentos para poder observar cual es mejor y en que situación.

Descripción del problema

La gran mayoría de aplicaciones que hay en internet requieren de una gran alta tolerancia a fallos y también una alta disponibilidad. En esta práctica se nos plantea el caso de un sistema de ficheros distribuido, el cual se presenta como una unidad pero realmente es un conjunto de servidores que contienen un conjunto de ficheros cada uno. La forma en que este funciona es bastante razonable frente a las condiciones impuestas por el propio internet, por cada fichero hay una o varias réplicas en los diferentes servidores del sistema, de manera que si un servidor cae haya una alta posibilidad que otra copia esté aún disponible.

La parte más importante de un sistema de ficheros distribuidos es el servidor que atiende las peticiones de los usuarios. De este va a depender que servidor atiende a una cierta petición de un usuario y si este recibe el archivo requerido en más o menos tiempo. Así que es esencial que las asignaciones que haga sean las óptimas para que el tiempo que el usuario este esperando sea el mínimo.

En esta práctica se asume que se está resolviendo el problema para uno de los servidores que distribuye las peticiones a un sistema de ficheros distribuido. Este recibirá un conjunto de peticiones de usuarios que necesitan acceder a ciertos ficheros y debe devolver una asignación de que servidor, de los que componen el sistema de ficheros distribuido, ha de satisfacer que petición.

Tanto los servidores como los usuarios están distribuidos geográficamente, de forma que el tiempo de transmisión de un fichero desde un servidor al usuario puede variar en función del servidor escogido.

Se dan herramientas para trabajar con el conjunto de servidores y el conjunto de peticiones recibidos. Gracias a estas el servidor distribuidor de peticiones tiene conocimiento de que servidores tienen un cierto fichero y también del tiempo de transmisión entre un servidor y un usuario.

La calidad en que se medirá las soluciones obviamente estará ligada al tiempo total de transmisión ya que como se ha dicho antes, se quiere que el usuario espere lo mínimo. Aun así se podrían dar casos en que algún servidor específico se sobrecargara y tuviera asignadas demasiadas peticiones. Es por eso que los puntos para evaluar una solución serán dos: el primero será que el tiempo del servidor con más tiempo de transmisión acumulado sea mínimo, y el segundo será que el tiempo total de transmisión también sea mínimo pero priorizando la repartición de asignaciones a la sobrecarga de un servidor.

El planteamiento del problema como una búsqueda en el espacio de soluciones es correcto. Eso es debido a que como se ha dicho antes, hay unos ciertos criterios a minimizar para que el usuario final quede satisfecho. Además dadas las condiciones del problema el servidor distribuidor tiene más facilidad a generar una solución inicial ya válida y mediante operadores moverse dentro del espacio de soluciones.

Representación del estado

Se empieza la resolución del problema planteando qué datos se deben incluir en el estado. Para ello es importante tener en cuenta el tamaño potencial del problema. Esto hace que se deba buscar una implementación con la cantidad mínima de información, optimizando así los recursos espaciales. Sin embargo, si se tiene un déficit de información en el estado, se estará aumentando el coste temporal debido a la computación, ya que se tendría que calcular reiteradamente en cada estado toda la información necesaria.

También se tiene que tener en cuenta que la implementación debe ser capaz de representar todos los posibles estados pertenecientes al espacio de soluciones del problema.

Las implementaciones del estado que fueron valoradas y estudiadas fueron:

- Una matriz de tamaño fijo de *#Servidores* por *#Peticiones*
- Una matriz de tamaño dinámico de anchura *#Servidores* y altura dependendiendo de las asignaciones que cada servidor tuviese
- Un vector que representara las asignaciones hechas. Cada una de sus posiciones representaría una petición y su contenido sería el servidor asignado a satisfacerla
- Dos vectores, uno que representara las asignaciones hechas como anteriormente se ha descrito y otro que representara el tiempo de transmisión acumulado por cada servidor dependiendo de las asignaciones que tuviese

La primera y segunda opción se desestimaron porque ocupaban demasiado espacio y había formas menos costosas y más fáciles de representar lo que estas hacían. Es posible que la segunda se pudiera haber usado pero tanto los heurísticos como los operadores hubieran sido o más costosos o ligeramente más difíciles de implementar.

La tercera opción se desestimó porque, aunque representa el problema y la solución que se quiere, es tan sintética que no se puede trabajar bien con ella.

La cuarta opción fue la mejor que se encontró. Esta basada en la tercera pero añadiendo un poco más de información para que sea más sencillo trabajar con ella. Se consideró que era la más equilibrada en cuanto a optimización temporal y espacial dados los operadores planteados a posteriori.

Aun así y dados los requerimientos a optimizar, se decidió añadir a esta última opción dos atributos más. El primero es un entero el cual representa el tiempo de transmisión total de todos los servidores y el segundo es también otro entero el cual representa el tiempo de transmisión del servidor con el tiempo de transmisión más grande. Eso se hizo para no tener que recalcular a cada nueva expansión esos valores y con un simple cálculo en los operadores tenerlos actualizados.

Así pues el estado tendrá como atributos:

- Un entero *server max time*, que indica el tiempo de transmisión del servidor que más tarda
- Un entero *total transmision time*, que indica el tiempo total de transmisión de todos los servidores

- Un vector de enteros *time per server*, que contiene el tiempo de transmisión de cada servidor
- Un vector de enteros *petition server*, que por cada petición almacena que servidor la va a satisfacer

A parte de todo eso, se decidió añadir como atributos estáticos las *request* y los *servers* del escenario planteado. La explicación a este hecho es que para ciertos cálculos se necesitan funciones de dichas clases y por eso se requieren.

Generando la solución inicial

El siguiente paso para la resolución del problema es generar una solución inicial, ya que, tratándose de un problema de búsqueda local, se debe entrar primero en el espacio de soluciones para después poder iniciar la búsqueda.

Cuando se genera una solución inicial es importante tener en cuenta que solo se estará generando una vez, así que el coste es relativamente despreciable.

Sin embargo, la calidad de esta solución es un importante factor a tener en cuenta: Una solución "demasiado buena", hace que la IA no tenga margen de maniobra y se estanque fácilmente en máximos o mínimos locales. Por otro lado, si la solución inicial es "demasiado mala", la búsqueda que tendrá que hacer la IA será innecesariamente grande.

Es por esto que se debe encontrar una solución que sea relativamente buena, pero que ceda a la IA un cierto margen de maniobra.

Por ello, se han valorado e implementado diversos generadores de soluciones iniciales que serán explicados a continuación y testeados en los experimentos.

Generador de baja calidad

Este es el primer generador que se pensó y valoró, ya que su funcionamiento es el más básico. También llamado en el código *Low quality generator*, tal y como su nombre indica, este generador produce soluciones de una calidad baja.

El criterio con el que crea una solución inicial es el siguiente: para cada petición que reciba le asigna el primer servidor que la pueda satisfacer sin tener en cuenta ningún otro factor más, como pueda ser el tiempo de transmisión entre el servidor y el usuario.

Generador del menor tiempo necesario

También llamado en el código *Less time needed generator*, este es el segundo generador que se valoró. Se consideró que era posible que los algoritmos que se quieren analizar trabajaran mejor con una solución inicial un poco mejor y más enfocada al segundo criterio así que se decidió implementar y observar este generador.

El criterio con el que crea una solución inicial es el siguiente: para cada petición que recibe itera sobre todos los posibles servidores que contienen el fichero requerido y le asigna a esta el servidor que menos tiempo de transmisión tenga entre el usuario que ha emitido la petición y este.

Generador del menor tiempo ocupado

También llamado en el código *Less time occupied generator*, este es el tercer generador que se valoró. Se pensó y enfocó para ver si las soluciones, trabajando con los diferentes algoritmos, eran mejores respecto al primer criterio.

El criterio con el que crea una solución inicial es el siguiente: para cada petición que recibe itera sobre todos los posible servidores que contienen el fichero requerido y le asigna a esta el servidor que tenga un menor tiempo de transmisión acumulado.

Análisis de los operadores

En este punto se plantearon los operadores que la IA podría usar para generar los nuevos sucesores en su búsqueda.

Se hizo una lista de todos los posibles operadores útiles que la IA podría usar y se hizo una valoración de cuáles realmente eran los mejores candidatos para analizar en profundidad en los experimentos.

La lista inicial de operadores de la que se partió fue:

- Crear servidor
- Eliminar servidor
- Añadir petición
- Eliminar petición
- Mover petición
- Intercambiar petición

Los dos primeros operadores se desestimaron enseguida, ya que el sistema de ficheros distribuidos no se puede modificar y tampoco se dan las herramientas necesarias para que se puedan implementar.

Los dos siguientes, añadir y eliminar petición, también fueron desestimados, ya que se valoró que no se podrían hacer experimentos individuales sobre ellos, pues si se parte de una solución inicial con las asignaciones ya hechas los dos operadores por si solos no podrían hacer nada, ya que resultarían o, en el caso de añadir, en nada o, en el caso de eliminar, en no soluciones. Se valoró también la posibilidad de combinarlas pero, a efectos prácticos serian como el operador mover o intercambiar, pero si no se controlase pudiendo dar lugar a no soluciones.

Por último se valoraron los operadores mover e intercambiar peticiones. Estos parecieron adecuados teniendo en cuenta las características de la búsqueda local en el espacio de soluciones, ya que en ningún momento se pueden salir de este y lo exploran todo. Además se podían estudiar tanto de forma individual como combinada.

Mover petición

En términos generales este operador mueve una petición de un servidor a otro. Está definido como:

$$\text{public void move\_petition(int } p\text{, int } s\text{)}$$

Internamente recibe una petición p y la reasigna a un servidor s. Pero para mantener la coherencia de todos los otros atributos del estado actualiza los tiempos en el *time per server* del servidor en el que estaba la petición y al que es reasignado. También actualiza el *total transmission time* y el *max server time*.

Intercambiar petición

En términos generales este operador intercambia dos peticiones de servidor. Está definido como:

public void swap_petitions(int $p0$, int $p1$)

Internamente recibe dos peticiones, $p0$ y $p1$, e intercambia los servidores a los que estas estaban asignados. Al igual que el anterior operador actualiza todos los atributos que se ven afectados, que en este caso son el tiempo acumulado de los dos servidores en *time per server*, el *total transmission time* y el *max server time*.

Generando los estados sucesores

Como en la anterior sección se ha explicado, al final se decidió que solo eran necesarios dos operadores. Utilizando estos la IA ha de ser capaz de generar los estados sucesores que explorará.

Conceptualmente solo se han creado tres generadores de estados sucesores que exploran todas las posibles combinaciones entre los operadores y són los que se explicaran posteriormente en esta sección, pero debido a que el funcionamiento del algoritmo *Hill Climbing* y el del *Simulated Annealing* es diferente se ha requerido crear tres generadores específicos para cada uno.

La diferencia entre ellos es que en el *Hill Climbing* se han de generar todos los posibles estados sucesores y retornarlos todos en una lista mientras que en el *Simulated Annealing* tan solo se tiene que generar uno de los posibles sucesores, escogido de forma aleatoria.

En las siguientes subsecciones se van a especificar los operadores usados y su factor de ramificación. Este último es en el caso del algoritmo *Hill Climbing* ya que el factor de los generadores del Simulated Annealing es siempre O(1).

Generador usando sólo el operador mover

En este caso el generador solo utiliza el operador de mover peticiones de sitio para generar los posibles sucesores.

Internamente este generador itera sobre todas las peticiones que el problema tenga y las va moviendo a otros servidores, creando nuevos estados cada vez que lo hace. Cada nuevo estado es añadido a una lista que contendrá todos los posibles estados sucesores.

Su factor de ramificación es $O(P \cdot S)$ donde P es el número de peticiones que tenga el problema y S el número de servidores disponibles en el sistema de ficheros distribuidos.

Generador usando sólo el operador intercambiar

En este caso el generador solo utiliza el operador de intercambio de peticiones para generar los posibles sucesores.

Internamente este generador itera sobre todas las peticiones que el problema tenga y las va intercambiando por todas las otras, creando nuevos estados por cada intercambio hecho. Cada nuevo estado es añadido a una lista que contendrá todos los posibles estados sucesores.

Su factor de ramificación es $O(P^2)$ donde P es el número de peticiones que tenga el problema.

Generador usando ambos operadores

En este caso el generador utiliza los dos operadores para generar los posibles sucesores.

Internamente es una combinación de los dos generadores anteriores iterando sobre todas las peticiones y añadiendo los nuevos estados generados por los dos operadores.

Su factor de ramificación es $O(P \cdot S + P^2)$ donde P es el número de peticiones que tenga el problema y S el número de servidores disponibles en el sistema de ficheros distribuidos.

Funciones de evaluación

En el enunciado de esta práctica se definen de forma muy concisa los criterios de la solución. El primero de ellos es minimizar el tiempo de transmisión de los ficheros para el servidor que necesita más tiempo para transmitir sus peticiones y el segundo es minimizar el tiempo total de transmisión de los ficheros teniendo en cuenta que los tiempos de transmisión de los servidores han de ser lo más similares posibles entre ellos.

Dados estos criterios se ha intentado encontrar diversas funciones de evaluación que trabajen con estos. Aun así no es lo único que se tuvo en cuenta, se vio también que con los operadores definidos anteriormente y como siempre se estarían usando variables de tiempo, no se podía trabajar con un sistema lineal, ya que se estarían generando una gran cantidad de mesetas cosa que afectaría negativamente a nuestros resultados, ya que este hecho es uno de los puntos débiles de los algoritmos a analizar.

Con estas restricciones en mente se decidió que se investigarían las funciones de evaluación descritas a continuación.

Tiempo del servidor con más tiempo de transmisión acumulado

Para esta función de evaluación se aprovechará la existencia del atributo *server max time* de la representación del estado, de modo que lo único que tendrá que hacer será devolver este valor.

Esta función de evaluación es útil para el cumplimiento de la primera restricción.

Tiempo de transmisión total de los servidores

Como en la sección de la representación del estado se ha mencionado, se decidió que habría un entero *total transmission time* que representaría el tiempo total de transmisión de todos los servidores.

Esta función de evaluación lo aprovecha y retorna ese valor del estado. El objetivo final es minimizar este valor y así poder valorar la solución por el segundo criterio aunque esta función no tenga en cuenta que los tiempos de transmisión de los servidores sean lo más parecidos entre si.

Media de tiempos de transmisión

Esta función de evaluación genera la media de los tiempos acumulados de todos los servidores. Esto lo hace dividiendo el entero *total transmission time* por el tamaño del atributo *time per server*.

Con eso se intenta que la restricción del segundo criterio se cumpla y que así los valores sean lo más similares entre sí.

Varianza

La varianza es un valor estadístico que mide la dispersión de un conjunto de datos respecto a su media. De este modo, si se consigue reducir la varianza, se estará reduciendo también

la dispersión de las muestras, con lo que se estará consiguiendo que los servidores tengan un tiempo de ocupación homogéneo.

Sin embargo, si solo se usara la varianza, lo único que se estaría generando son tiempos homogéneos, pero no se estarían minimizando. Por lo tanto, se debería combinar con algún otro parámetro que ayude a minimizar, al mismo tiempo, los tiempos de transmisión.

Para solucionar esto se ha multiplicado la varianza por la media. Con esta aproximación se podrá lograr minimizar ambos factores a la vez.

Entropía

La entropía hace referencia a la cantidad de desorden en un sistema. De este modo, se buscará conseguir que los tiempos acumulados de los servidores sean lo más homogéneos posibles, reduciendo así el desorden.

Desafortunadamente, es difícil encontrar una expresión adecuada para la entropía dado el problema planteado. Además, se debe tener en cuenta que, del mismo modo que con la varianza, se debe estar minimizando a la vez que homogeneizando los tiempos de transmisión totales.

Para solucionar esto y al igual que con la varianza se ha multiplicado la entropía por la media. Con esta aproximación se podrá lograr minimizar ambos factores a la vez.

Experimentos

Experimento 1

Uno de los elementos esenciales de un problema de búsqueda local son los operadores. Dependiendo de si estos son elegidos correctamente o no determinará el hecho de que se explore correctamente todo el espacio de soluciones, al igual que también es posible que unos sean más rápidos que otros o den mejores resultados. A priori no se puede determinar cuál será el más adecuado para el problema, ya que estos no han sido probados.

Para demostrar la hipótesis de que hay operadores mejores que otros se va a realizar un experimento probando los diferentes operadores que en anteriores puntos se han explicado evaluando sus resultados.

Se van a considerar dos operadores distintos: el de mover una petición de un servidor a otro y el de cambiar dos peticiones de sitio. Utilizando estos se van a probar tres métodos diferentes: el primero será utilizando solo el primer operador, el segundo será utilizando solo el segundo operador y el tercero será utilizando una combinación de ambos. Estos coinciden con los generadores de estados sucesores descritos.

Para poder hacer el experimento correctamente debemos garantizar que todos los experimentos con cada hipótesis se realizan en las mismas condiciones. Es por eso que se realizaran diez réplicas del experimento donde se probará cada método explicado anteriormente. Para cada experimento individual se utilizará la misma semilla de números aleatorios, de manera que cada réplica sea idéntica en las pruebas para cada método. Los otros parámetros iniciales serán iguales en todos los experimentos. Valdrán, el número de usuarios que piden ficheros 200, el número máximo de peticiones por usuario 5, el número de servidores 50 y el número de replicaciones 5. El método para generar la solución inicial siempre será el *low quality*, la heurística utilizada será la *server max time*, para así optimizar el primer criterio, y el algoritmo utilizado el *Hill Climbing*.

Como en todos los experimentos se parte de una situación igual podemos comparar que operador optimiza mejor el primer criterio. También nos puede interesar el tiempo que tardaron en solucionarse los problemas y el número de pasos que se hicieron hasta llegar a la solución.

Podemos resumir las características de este experimento en la siguiente tabla:

Observación	Pueden haber operadores que obtengan mejores resultados
Planteamiento	Se escogen diferentes operadores y se observan sus soluciones
Hipótesis	Todos los operadores dan los mismos resultados (H0) o hay unos operadores mejores que otros
Método	▪ Se elegirán 10 semillas aleatorias, una por cada réplica ▪ Se ejecutará 1 experimento por cada semilla aleatoria solo para el operador mover una petición de un servidor a otro ▪ Se ejecutará 1 experimento por cada semilla aleatoria solo para el operador intercambiar dos peticiones ▪ Se ejecutará 1 experimento por cada semilla aleatoria para la combinación de los dos operadores anteriores ▪ Se experimentará con los parámetros usando los siguientes valores fijos: • Número de usuarios: 200 • Número máximo de peticiones: 5 • Número de servidores: 50 • Número de replicaciones: 5 ▪ Se usará el generador *low quality* ▪ Se usará la heurística *server max time* ▪ Se usará el algoritmo *Hill Climbing* ▪ Se medirán diferentes parámetros y resultados para compararlos

En la siguiente tabla se pueden ver los resultados de los experimentos como se han definido. Se añade a esta la semilla usada para que los experimentos se puedan volver a hacer en caso de necesitar comprobación.

		Tiempo máximo			Pasos			Tiempo		
Réplica	Semilla	Move	Swap	Both	Move	Swap	Both	Move	Swap	Both
1	9397	51624	61532	66490	591	173	156	9.509	30.718	34.543
2	3463	70832	36422	29348	385	297	409	6.926	64.756	116.185
3	3001	80412	34731	45929	227	369	229	3.614	54.737	40.011
4	7793	97248	57487	27115	257	300	640	5.086	103.301	298.21
5	9806	92770	35529	63418	190	394	191	3.46	89.989	66.597
6	4349	80151	56738	22046	328	224	653	5.648	52.201	199.795
7	3131	104679	53886	38544	259	346	431	4.595	95.663	137.772
8	8055	70049	51647	51647	362	188	188	5.725	34.01	45.501
9	4690	101186	50967	55483	218	307	281	4.038	87.829	90.537
10	5501	82772	40348	44454	273	284	263	4.955	76.463	89.126

Cuadro 1: Resultados del experimento 1

En las gráficas que vienen a continuación en el eje horizontal se marcan los métodos usados como 0, 1 y 2. En este caso són el operador de mover, el operador de intercambiar y la combinación de ambos, respectivamente.

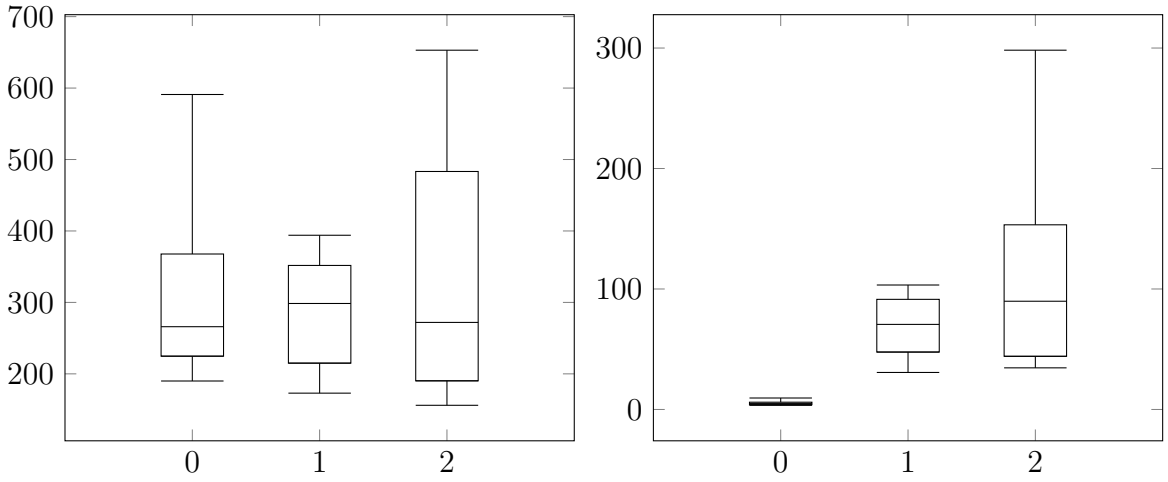
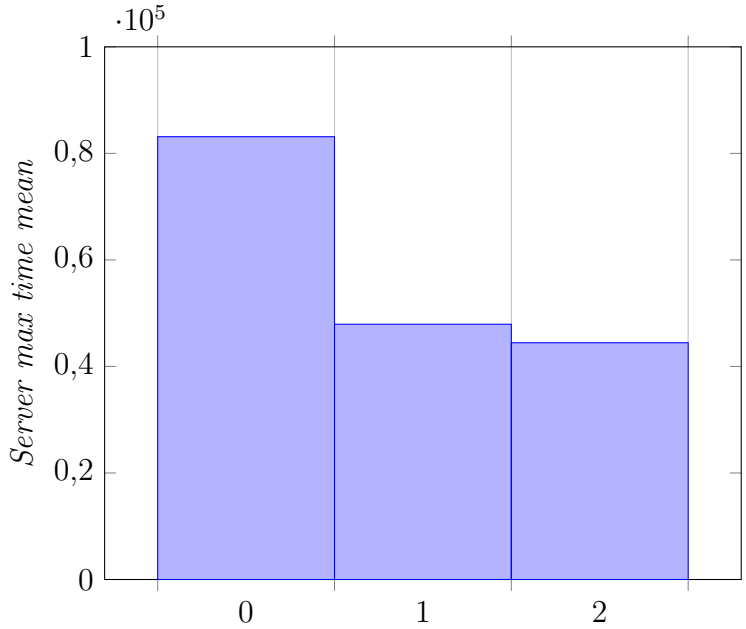


Figura 1: Distribución del número pasos y del tiempo de ejecución

En la figura 1 se puede apreciar mejor la distribución del número de pasos y el tiempo de ejecución.

Como se puede ver, el operador de mover una petición es el más rápido de todos respecto al tiempo pero el que peor calidad de solución da. Este último hecho se puede comprobar viendo la figura 2. Así que como resultado este operador queda descartado.

Los otros dos operadores ha analizar serian el de intercambiar peticiones y la combinación de los dos anteriores. Fijándose en el número de pasos se puede ver que el operador de intercambiar llega antes a un óptimo local mientras que la combinación de los dos tarda más y es más disperso. Respecto al tiempo que tardan, el operador intercambiar tarda menos lo cual es lógico porque no tiene que crear ni valorar tantos estados sucesores como el combinado. Comparando la calidad de la solución dada por cada uno se puede ver que son bastante parecidas, es por eso que se ha decidido medir la media de los tiempos máximos obtenidos para decidir cuál se va a escoger. Esta se puede ver en la figura 2.



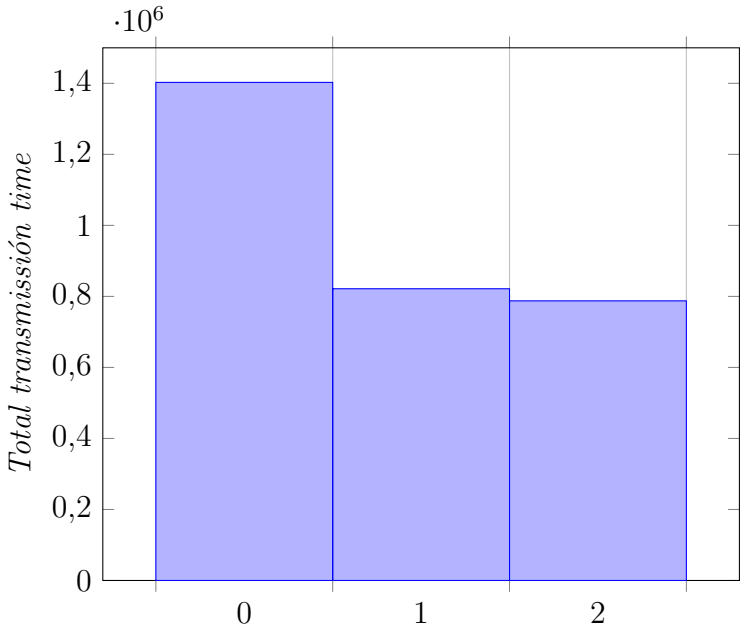
Esta última gráfica está escalada, como se puede observar arriba a la izquierda, a -10^5.

Gracias a ella se puede observar que la media aritmética del operador de intercambiar es ligeramente más alta. Aun así se ha decidido optar por valorar también el segundo criterio para tener todos los factores en cuenta a la hora de tomar la decisión.

		Tiempo total de transmisión		
Réplica	Semilla	Move	Swap	Both
1	9397	1166031	955944	1010236
2	3463	1357502	711439	625920
3	3001	1240390	603525	736655
4	7793	1532225	944728	618283
5	9806	1453673	667019	996990
6	4349	1441284	976024	560799
7	3131	1492794	820302	678435
8	8055	1355134	893996	893996
9	4690	1499137	855631	904479
10	5501	1488200	786472	848755

Cuadro 2: Resultados del tiempo total de transmisión del experimento 1

Como se puede observar los resultados entre ellos siguen siendo bastante parecidos, por ello se ha vuelto a hacer la media para observar cuál ha dado mejores resultados.



Esta última gráfica está escalada, como se puede observar arriba a la izquierda, a -10^6.

Viendo que tanto en el primer criterio como con el segundo la combinación de los dos operadores siguen dando mejores resultados, se ha decidido utilizar definitivamente estos.

Habiéndose analizado los resultado se puede afirmar que la hipótesis inicial (H0) planteada era errónea y por lo tanto hay operadores mejores que otros.

Experimento 2

En este tipo de problemas otro factor que es muy importante es el generador de la solución inicial. Si este genera una solución demasiado buena los algoritmos no podrán hacer nada, ya que no sería necesario y eso sería malo porque realmente no sabríamos si es un máximo o mínimo local o no. El caso contrario en que la solución es demasiado mala tampoco nos interesa, ya que muy probablemente los algoritmos tarden demasiado en encontrar un óptimo local. De los que han sido definidos y explicados en secciones anteriores a priori no podemos determinar cuál será el mejor, ya que se desconoce la forma que tendrá el espacio de soluciones.

Para probar la hipótesis de que hay generadores de solución inicial mejores que otros vamos a realizar un experimento probando los diferentes operadores que en anteriores puntos se han explicado evaluando sus resultados.

Para poder hacer el experimento correctamente debemos garantizar que todos los experimentos con cada hipótesis se realizan en las mismas condiciones. Es por eso que se realizaran diez réplicas del experimento donde se probaran cada generador de solución inicial planteado. Para cada experimento individual se utilizará la misma semilla de números aleatorios, de manera que cada réplica sea idéntica en las pruebas para cada generador. En el caso de los operadores dadas las conclusiones obtenidas del primer experimento se usaran siempre la combinación de mover e intercambiar peticiones. Los otros parámetros iniciales serán iguales en todos los experimentos. Valdrán, el número de usuarios que piden ficheros 200, el número máximo de peticiones por usuario 5, el número de servidores 50 y el número de replicaciones 5. La heurística utilizada será la *server max time* para así optimizar el primer criterio y el algoritmo usado será el *Hill Climbing*.

Como en todos los experimentos se parte de una situación igual podemos comparar que generador da mejores resultados respecto al primer criterio. También nos puede interesar el tiempo que tardaron en solucionarse los problemas y el número de pasos que se hicieron hasta llegar a la solución.

Podemos resumir las características de este experimento en la siguiente tabla:

Observación	Pueden haber generadores de soluciones iniciales que obtengan mejores resultados
Planteamiento	Se escogen los diferentes generadores de soluciones iniciales y se observan sus soluciones
Hipótesis	Todos los generadores de soluciones iniciales dan los mismos resultados (H0) o hay generadores mejores que otros
Método	▪ Se elegirán 10 semillas aleatorias, una por cada réplica ▪ Se ejecutará 1 experimento por cada semilla aleatoria solo para el generador *low quality* ▪ Se ejecutará 1 experimento por cada semilla aleatoria solo para el generador *less time needed* ▪ Se ejecutará 1 experimento por cada semilla aletoria solo para el generador *less time occupied* ▪ Se experimentará con los parámetros usando los siguientes valores fijos: • Número de usuarios: 200 • Número máximo de peticiones: 5 • Número de servidores: 50 • Número de replicaciones: 5 ▪ Se usarán los operadores de mover e intercambiar combinados ▪ Se usará la heurística *server max time* ▪ Se usará el algoritmo *Hill Climbing* ▪ Se medirán diferentes parámetros y resultados para compararlos

En la siguiente tabla se pueden ver los resultados de los experimentos como se han definido.

		Tiempo máximo			Pasos			Tiempo		
Réplica	Semilla	LQ	LT	LO	LQ	LT	LO	LQ	LT	LQ
1	7968	37461	10468	11696	329	61	411	79.62	16.895	110
2	5759	30387	8080	15550	540	172	315	160.8	59	111.9
3	7871	31423	8276	13138	401	143	348	88.3	32.61	109.4
4	7469	50474	9798	17703	309	78	251	123.6	21.666	76.3
5	5635	45833	10454	15952	384	93	342	168	34.393	148.1
6	3871	30968	10819	28407	392	35	37	127.5	11.214	16.3
7	8353	25835	10718	16118	669	42	313	239.7	14.805	106.6
8	2572	57013	10858	23522	297	88	136	109	28.802	48.7
9	3646	49725	9526	17885	231	95	221	68.8	25.527	67
10	3063	30662	11700	16948	427	36	247	118	10.863	73.4

Cuadro 3: Resultados del experimento 2

En las gráficas que vienen a continuación en el eje horizontal se marcan los métodos usados como 0, 1 y 2. En este caso són los generadores de solución inicial *low quality*, *less time needed* y *less time occupied*, respectivamente.

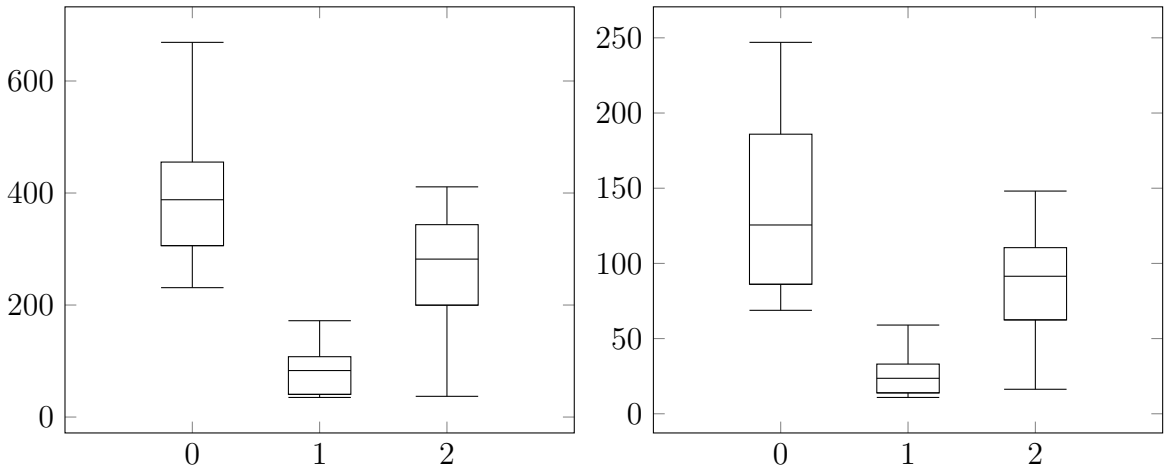
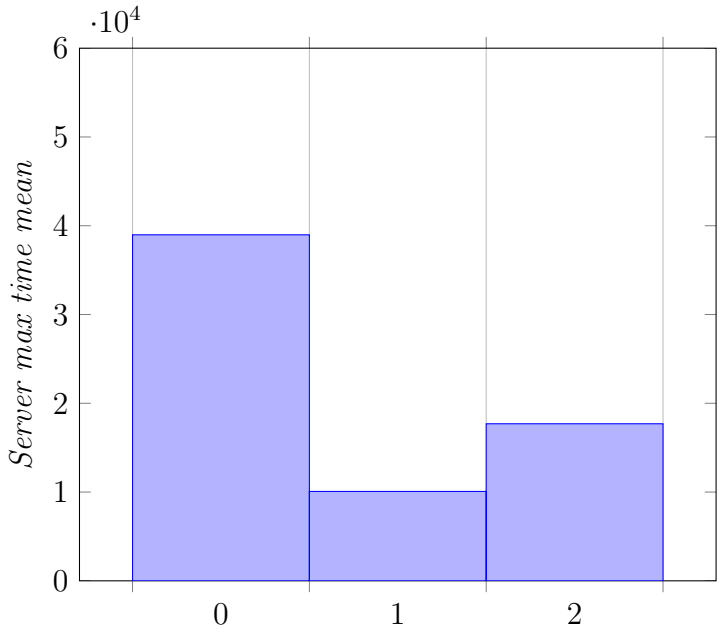


Figura 2: Distribución del número pasos y del tiempo de ejecución

En la figura 2 se puede apreciar que el que menor número de pasos y tiempo necesita es el generador *less time needed* con bastante diferencia. Además los resultados obtenidos con este no són muy dispersos.

Aun así se ha querido comprobar la calidad de la solución. Para ello se ha realizado la media de la medida *Tiempo máximo* como se puede ver en la siguiente gráfica.



Esta última gráfica está escalada, como se puede observar arriba a la izquierda, a -10^4.

Como se puede observar en la gráfica anterior la calidad de solución basándonos en el primer criterio del generador *less time needed* es mucho mejor que la de los otros. Dados los resultados se ha elegido que la estrategia de generación de la solución inicial será está.

Habiéndose analizado los resultado se puede afirmar que la hipótesis inicial (H0) planteada era errónea y por lo tanto hay generadores de la solución inicial mejores que otros.

Experimento 3

En este experimento se pide trabajar con el algoritmo *Simulated Annealing*. Este no funciona igual que el *Hill Climbing*, ya que tanto los parámetros como su funcionamiento interno son diferentes, es por eso que requiere de una serie de experimentos y pruebas para encontrar los valores adecuados de los parámetros para ejecutarlo. Estos son:

- Número total de iteraciones, el cual depende de k y λ y del problema y se ha de determinar experimentalmente
- Iteraciones por cada cambio de temperatura
- Parámetro k de la función de aceptación de estados
- Parámetro λ de la función de aceptación de estados

En este caso lo primero que se va a determinar serán la k y la λ. Para hacer eso, se tiene que estudiar como es la función que se utiliza para la aceptación de estados. Esta es:

$$\mathcal{F}(T) = k \cdot e^{-\lambda \cdot T}$$

Y la función que determina la probabilidad de aceptación de un estado peor es la siguiente:

$$\mathcal{P}(estado) = e^{(\frac{\Delta E}{\mathcal{F}(T)})}$$

Esta última tiene un rango de [0,1] que varia con la temperatura(T), que en esta implemetación representa el número de iteración actual, y la diferencia de energía (ΔE) entre el estado actual y el siguiente estado.

Estudiando su comportamiento se puede ver que cuanto mayor es la k, más tarda en comenzar a decrecer y cuanto mayor es λ más rápido desciende. Si vamos variando λ, la velocidad a la que esa probabilidad desciende va aumentando cuanto mayor es, disminuyendo además el número de iteraciones que hacen falta para llegar a probabilidad 0.

Como su comportamiento no varía con el problema se tendrá que estudiar para el problema en concreto cuantas iteraciones harán falta para que la exploración pueda sobrepasar el punto en que la probabilidad se hace 0, que es cuando el algoritmo solo puede mejorar la solución actual.

Con todo este conocimiento se hará el experimento para una instancia particular del problema. En este caso el escenario será: el número de usuarios que piden ficheros será 200, el número máximo de peticiones por usuario será 5, el número de servidores será 50, el número mínimo de replicaciones será 5, la función heurística será la *max server time*, como en anteriores experimentos se ha definido se usaran los operadores de mover e intercambiar peticiones y se usará también como antes se ha determinado el generador de soluciones iniciales *less time needed*.

Como se puede observar el comportamiento de los dos algoritmos en las figuras 3 y 4 son totalmente diferentes. El primera disminuye de manera monótona y el segundo va incrementando y disminuyendo su coste. Esto último puede ser debido a que el número de iteraciones es insuficiente. El resultado de aumentar el número de iteraciones se puede ver en la figura 5.

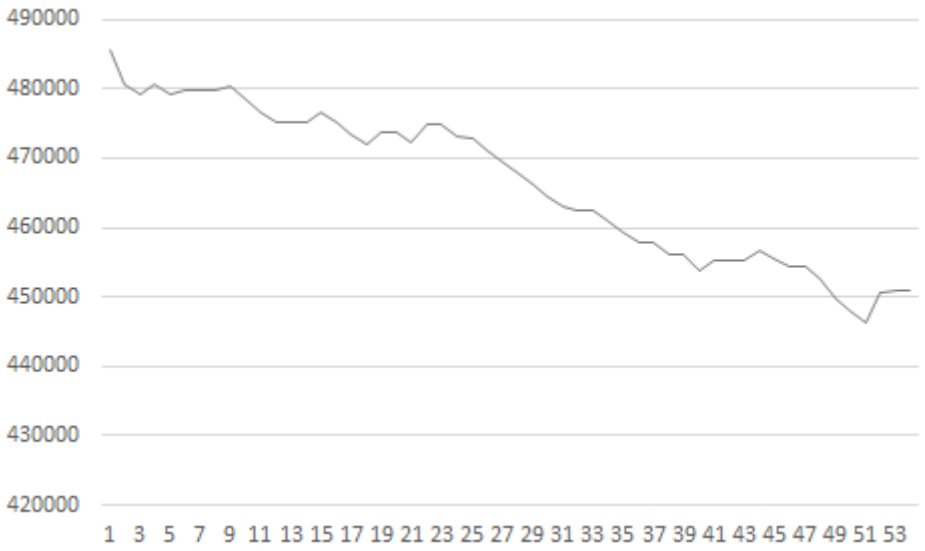


Figura 3: Evolución del coste (HC) con las condiciones descritas

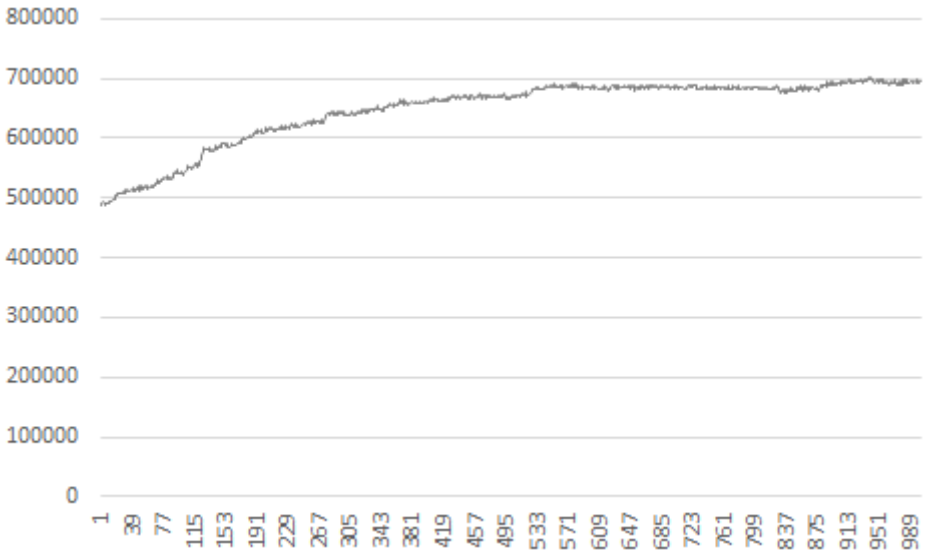


Figura 4: Evolución del coste (SA/It= 1000 k=5 λ=0.01) con las condiciones descritas

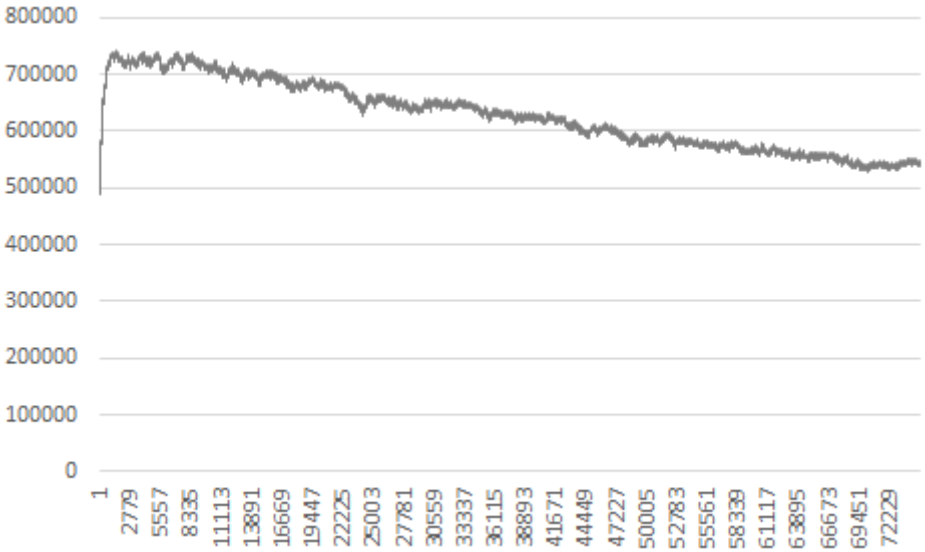


Figura 5: Evolución del coste (SA/It= 100000 k=5 λ=0.01) con las condiciones descritas

Es por eso que es esencial que los parámetros sean ajustados adecuadamente.

Para ello se van a hacer una serie de experimentos para averiguar cuales serían los mejores valores de k y λ. Para ello usaremos un número de iteraciones grande para asegurarnos que llegue a converger y probaremos los valores para $k = 5, 25, 135$ y para $\lambda =$ 0.1, 0.001, 0.0001.

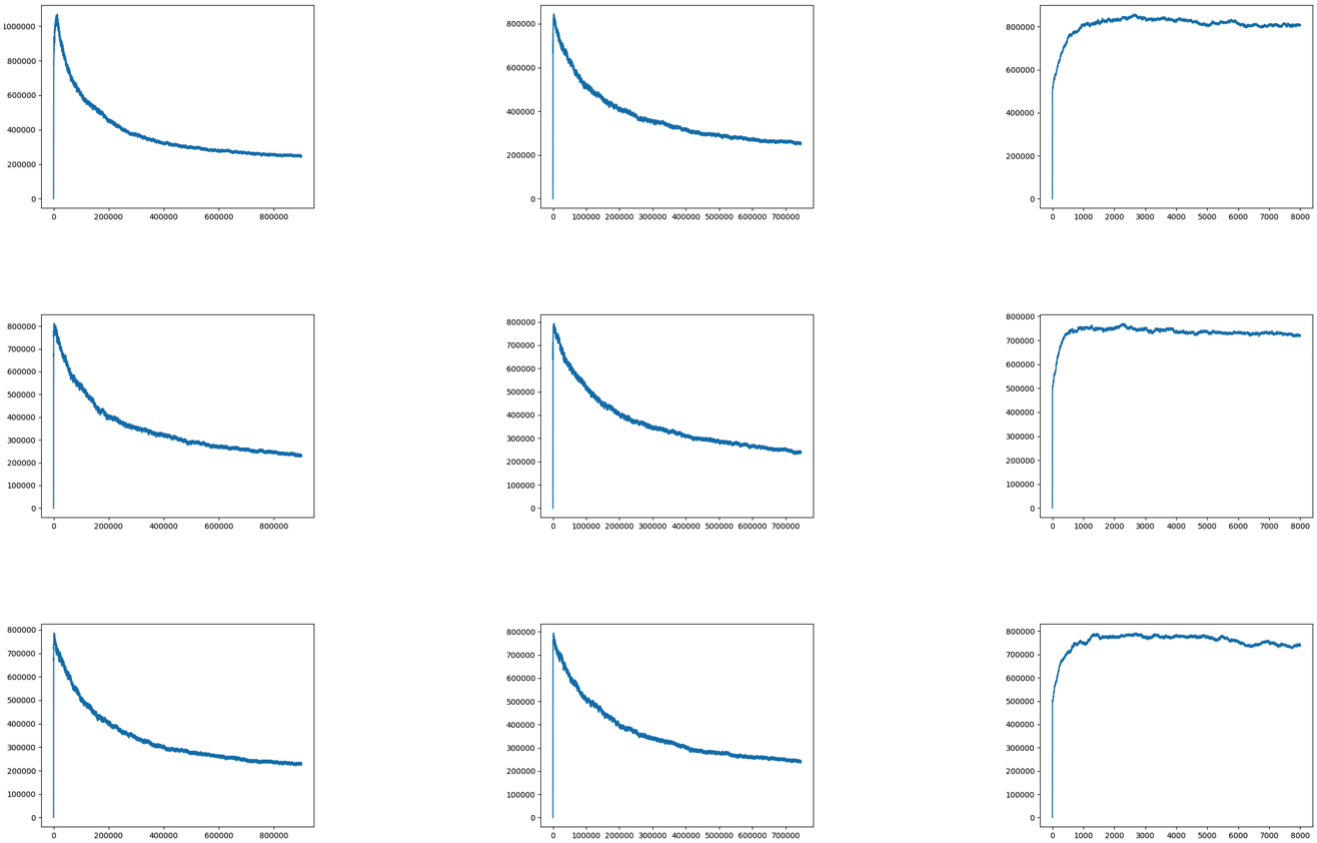


Figura 6: Resultados obtenidos para las diferentes combinaciones de k y *lambda*.

En la figura 6 los valores de k y *lambda* se reparten de la forma siguiente: por cada fila mantenemos el valor de k y por cada columna el de *lambda*. De esta forma, de fila superior a inferior los valores de k son 125, 25 y 5 y de la columna izquierda a derecha los valores de *lambda* son 0.0001, 0.001, 0.1.

De esta manera se puede ver que los cambios más significativos se muestran al variar la *lambda* con una k fijada (filas) y no escogiendo una *lambda* y variando la k (columnas).

Sabiendo esto se puede ver que los parámetros més idóneos para que la convergencia del algoritmo sea más rápida a la hora de encontrar una solución mejor que usando *Hill Climbing* es $k = 125$ y $lambda = 0.0001$ necesitando así entre 300000 y 400000 iteraciones.

Experimento 4

En anteriores experimentos se ha trabajado con un escenario fijo para así poder analizar un parámetro en concreto y poder determinar de las opciones que se tenían para él cuál era la mejor. En este experimento pero, se pide que usando los parámetros determinados anteriormente se modifique el escenario para ver como evoluciona el tiempo de ejecución.

Para poder ver dicha evolución se va a realizar un estudio incrementando los siguientes dos parámetros:

- Número de usuarios que piden ficheros
- Número de servidores

El número de peticiones por usuario se va a mantener en 5 al igual que el número de repeticiones. Se usará la función de evaluación *server max time* y el conjunto de operadores y el generador de la solución inicial determinados por los anteriores experimentos, mover e intercambiar peticiones y el generador less time needed respectivamente.

Es importante denotar que el número de usuarios y el número de servidores van directamente ligados al tamaño de los vectores de peticiones y de servidores, motivo por el cual se van a generar muchos más sucesores que van a tener que ser evaluados.

Para evitar dispersiones en casos concretos, se usarán 10 semillas aleatorias y se generarán 20 ejecuciones con cada una. En diez de ellas se incrementarán el número de usuarios desde el 100 hasta 1000 fijando el número de servidores en 10. En las otras 10 se incrementarán el número de servidores desde 50 hasta 500 fijando el número de usuarios en 200.

Observación	Cómo afectan el número de usuarios y el número de servidores a nuestro problema
Planteamiento	Se realizan diversas ejecuciones con diferentes cantidades de usuarios y servidores y se observan los resultados
Hipótesis	Todas las variaciones en la cantidad de usuarios y servidores generan la misma complejidad(H0) o no
Método	▪ Se elegirán 10 semillas aleatorias, una por cada réplica ▪ Se ejecutará 1 experimento por cada semilla, cambiando sólo la cantidad de usuarios ▪ Se ejecutará 1 experimento por cada semilla, cambiando sólo la cantidad de servidores ▪ Se experimentará con los parámetros usando los siguientes valores fijos: • Número máximo de peticiones: 5 • Número de replicaciones: 5 ▪ Se usarán los operadores de mover e intercambiar combinados ▪ Se usará el generador de solución inicial *less time needed* ▪ Se usará la heurística *server max time* ▪ Se usará el algoritmo *Hill Climbing* ▪ Se medirán diferentes parámetros y resultados para compararlos

Los resultados obtenidos se pueden observar en las siguientes gráficas, los números de la leyenda son las semillas usadas:

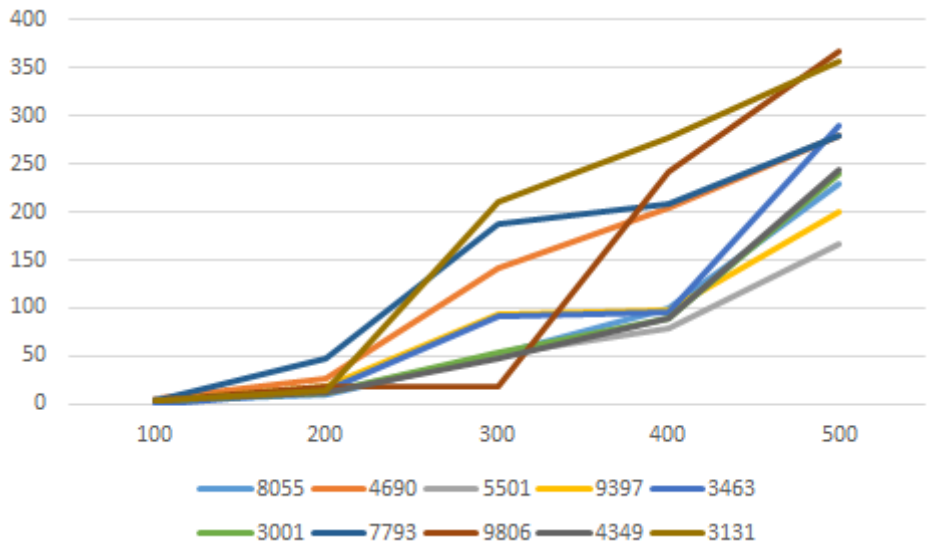


Figura 7: Resultados manteniendo el número de servidores

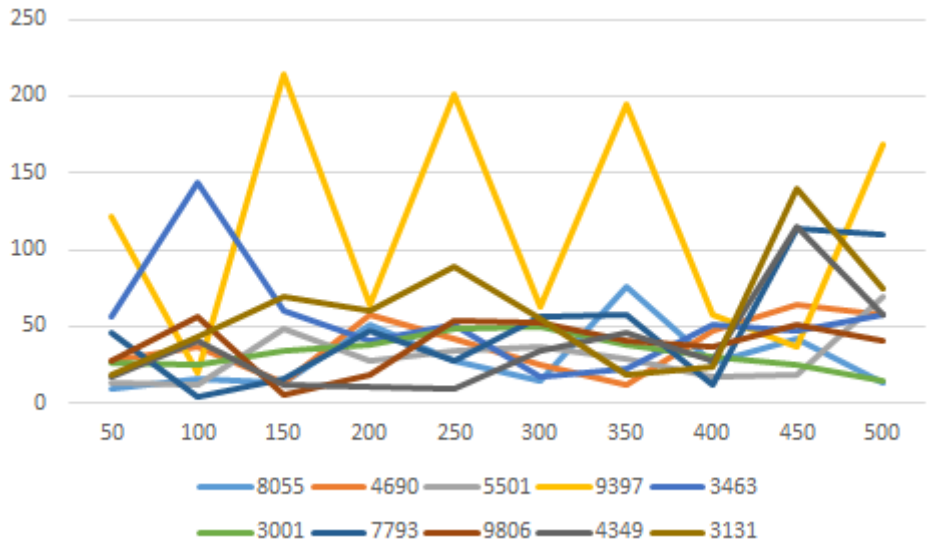


Figura 8: Resultados manteniendo el número de usuarios

En los gráficos anteriores se puede ver como evoluciona la relación entre la complejidad del problema con los parámetros de cantidad de usuarios y cantidad de servidores. Como se puede apreciar, estos parámetros están estrechamente ligados a la complejidad.

En la figura 6 se puede ver que el tiempo de ejecución del programa aumenta exponencialmente a medida que aumenta la cantidad de usuarios, ya que existirán más peticiones que gestionar y por lo tanto más sucesores que crear y evaluar.

En la figura 7 se puede ver que el parámetro número de servidores disponibles no parece presentar relación con el tiempo de ejecución. Intuitivamente, se puede pensar que el número de réplicas que existen para un archivo concreto se mantiene en un mínimo de 5, por lo tanto, el número de servidores donde estará disponible un archivo concreto, no aumentará del mismo modo que aumenta el número de servidores totales.

Experimento 5

Lo que se quiere obtener en este experimento es una estimación de la diferencia entre el tiempo total de transmisión y el tiempo que se ha tardado en encontrar la solución. Para realizarlo se emplearán las dos heurísticas que se encargan de minimizar los dos criterios indicados en el enunciado.

El algoritmo usado para el desarrollo del experimento será *Hill Climbing* y la heurística empleada para el cumplimiento del primer criterio será la utilizada en los experimentos anteriores (*server max time*). Para el segundo criterio, se utilizarán dos heurísticas distintas: la *Varianza* y la *Entropia*.

Así pues, dado que para el cumplimiento del segundo criterio existen dos heurísticas, antes de poder hacer las pruebas para el cálculo de la estimación, se deberá determinar experimentalmente cuál de las dos heurísticas propuestas produce mejores resultados. Para ello, se realizarán diez réplicas con el mismo escenario que en los experimentos anteriores (200 usuarios, 5 peticiones máximo por usuario, 50 servidores y 5 replicaciones mínimo por fichero) pero escogiendo de forma aleatoria una semilla para cada una de las replicaciones. En cada replicación se aplicaran tanto la *Varianza* como la *Entropía*. De esta manera aseguramos que los resultados obtenidos por cada heurística se han producido bajo las mismas circunstancias y por lo tanto pueden ser comparados.

Una vez se ha discernido que heurística es mejor para el segundo criterio, con los valores obtenidos para esta y la heurística del primer criterio, se genererará para cada una la media del tiempo de transmisión total y la media del tiempo para obtener la solución.

En la siguiente tabla están resumidas las características de este experimento:

Observación	Encontrar la estimación de la diferencia entre el tiempo total de transmisión y el tiempo necesario para encontrar la solución
Planteamiento	Se determina experimentalmente cuál es la mejor heurística para satisfacer el segundo criterio, y una vez determinada se procede a la determinación de la estimación
Hipótesis	-
Método	▪ Se elegirán 10 semillas aleatorias, una por cada réplica ▪ Se ejecutará 1 experimento por cada semilla aleatoria solo para la heurística Varianza ▪ Se ejecutará 1 experimento por cada semilla aleatoria solo para la heurística Entropía ▪ Se ejecutará 1 experimento por cada semilla aleatoria solo para la heurística *Server max time* ▪ Se experimentará con los parámetros usando los siguientes valores fijos: • Número de usuarios: 200 • Número máximo de peticiones: 5 • Número de servidores: 50 • Número de replicaciones: 5 ▪ Se usará el generador *less time needed* ▪ Se usarán los operadores de mover e intercambiar combinados ▪ Se usará el algoritmo *Hill Climbing* ▪ Se medirán diferentes parámetros y resultados para compararlos

Los resultados obtenidos se pueden observar en la siguiente tabla:

		Tiempo Total Transmision			Tiempo			Pasos		
Réplica	Semilla	SMT	Entr	Var	SMT	Entr	Var	SMT	Entr	Var
1	9397	363430	189749	465356	30.98	275.944	41.348	110	585	107
2	3463	417283	182199	505528	22.593	314.888	40.146	67	604	110
3	3001	380798	171899	461460	21.653	233.61	35.555	84	551	117
4	7793	473460	193499	570230	35.088	468.769	52.936	86	724	110
5	9806	447891	187199	534179	22.57	369.192	44.477	70	689	121
6	4349	462285	187647	532868	15.099	371.164	43.167	45	688	114
7	3131	446017	168399	513324	15.439	379.126	47.346	43	678	124
8	8055	435313	178099	483400	15.462	314.985	31.998	53	660	102
9	4690	417194	203449	558023	65.968	410.943	50.117	141	650	119
10	5501	446239	206399	499486	23.827	438.289	56.436	61	678	117

Cuadro 4: Resultados del experimento 5

En las gráficas que vienen a continuación en el eje horizontal se marcan los métodos usados como 0, 1 y 2. En este caso són las heurísticas *server max time*, la entropía y la varianza, respectivamente.

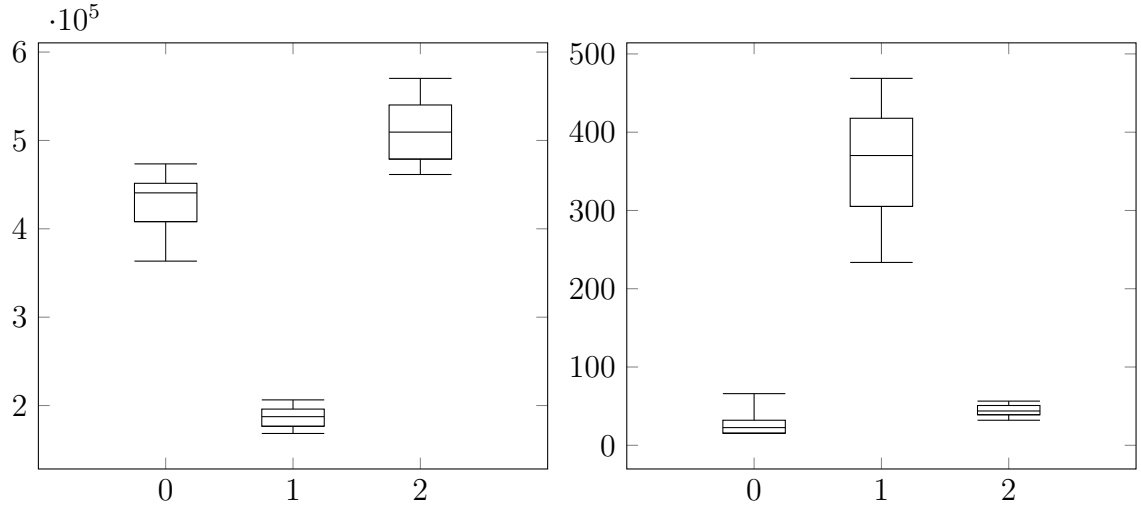


Figura 9: Distribución del tiempo total de transmisión y del tiempo de ejecución

En los gráficos de la figura 6 se puede observar la gran diferencia existente entre las dos heurísticas encargadas de regular el segundo criterio. Como bien se muestra, la entropía llega a soluciones de calidad muy superior respecto a la varianza, pero esto se consigue a cambio de expandir muchos más nodos y por lo tanto, una mayor cantidad de tiempo invertido para encontrar la solución. Aun así, se ha considerado que la diferencia en la calidad de la solución prevalece sobre la diferencia en el tiempo de ejecución, por lo tanto para el segundo criterio se usará la entropía.

Así pues, para estimar la diferencia entre el tiempo total de transmisión y el tiempo para hallar la solución, se calculará la diferencia entre la media de los valores recogidos en el experimento para esas dos variables. En concreto serán los valores de *server max time* para el primer criterio y los de la entropía para el segundo. De esta forma los resultados son los siguientes:

- *Estimación con el primer criterio* = abs(428.991 - 26.8679) = 402.1231 segundos
- *Estimación con el segundo criterio* = abs(186.854 - 357.691) = 170.837 segundos

El primer valor de la resta pertenece a la media del tiempo de transmisión total, pasado a segundos, y el segundo a la media del tiempo total para encontrar la solución (Columna Tiempo en la tabla), el cual ya estaba en segundos.

Experimento 6

Lo que se quiere obtener en este experimento es como en el anterior una estimación de la diferencia entre el tiempo total de transmisión y el tiempo que se ha tardado en encontrar la solución. Para realizarlo se emplearán las dos heurísticas que se encargan de minimizar los dos criterios indicados en el enunciado.

El algoritmo usado para el desarrollo del experimento será *Simulated Annealing* y la heurística empleada para el cumplimiento del primer criterio será la utilizada en los experimentos anteriores (*server max time*). Para el segundo criterio y como ya se ha demostrado se utilizará la *Entropia*.

Con los valores obtenidos para esta y la heurística del primer criterio, se genererará para cada una la media del tiempo de transmisión total y la media del tiempo para obtener la solución.

En la siguiente tabla están resumidas las características de este experimento:

Observación	Encontrar la estimación de la diferencia entre el tiempo total de transmisión y el tiempo necesario para encontrar la solución
Planteamiento	Se determinará la estimación pedida
Hipótesis	-
Método	▪ Se elegirán 10 semillas aleatorias, una por cada réplica ▪ Se ejecutará 1 experimento por cada semilla aleatoria solo para la heurística Entropía ▪ Se ejecutará 1 experimento por cada semilla aleatoria solo para la heurística *Server max time* ▪ Se experimentará con los parámetros usando los siguientes valores fijos: • Número de usuarios: 200 • Número máximo de peticiones: 5 • Número de servidores: 50 • Número de replicaciones: 5 ▪ Se usará el generador *less time needed* ▪ Se usarán los operadores de mover e intercambiar combinados ▪ Se usará el algoritmo *Simulated Annealing* con la k= 125 y la λ= 0.0001 ▪ Se medirán diferentes parámetros y resultados para compararlos

Los resultados obtenidos se pueden observar en la siguiente tabla:

		Tiempo Total Transmision		Tiempo	
Réplica	Semilla	SMT	Entr	SMT	Entr
1	9397	288573	226637	2.198	1.735
2	3463	316725	227747	2.291	1.967
3	3001	284811	201591	2.441	1.967
4	7793	367632	245899	2.908	1.733
5	9806	298158	213595	2.844	1.708
6	4349	309153	221775	2.758	1.683
7	3131	328862	216918	2.365	1.694
8	8055	310314	206936	2.382	1.719
9	4690	348177	242191	2.414	1.731
10	5501	326571	238584	2.355	1.758

Cuadro 5: Resultados del experimento 6

En las gráficas que vienen a continuación en el eje horizontal se marcan los métodos usados como 0 y 1. En este caso són las heurísticas *server max time* y la entropía.

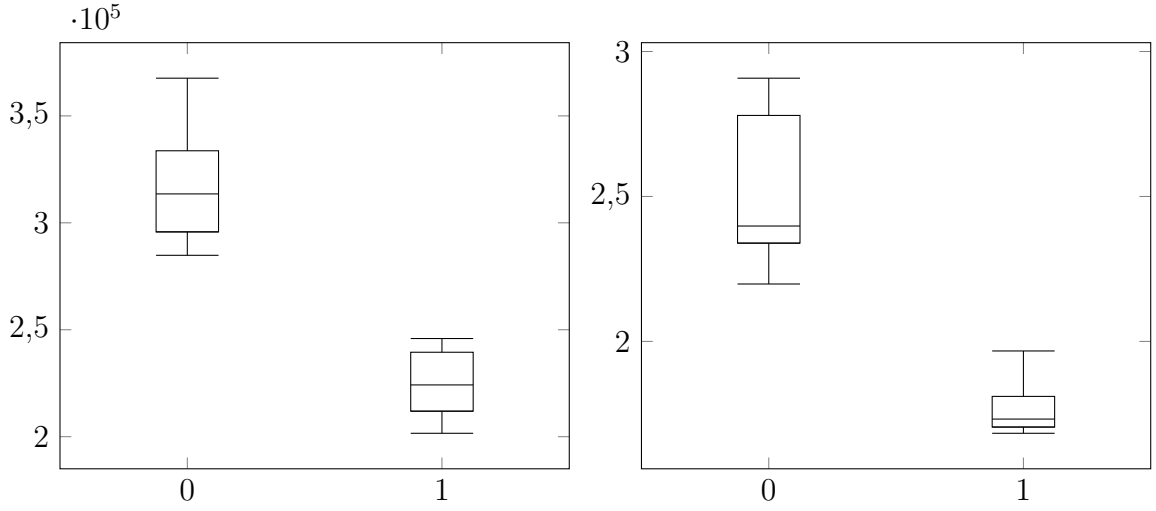


Figura 10: Distribución del tiempo total de transmisión y del tiempo de ejecución

Los resultados después de comparar la diferencia entre los valores de *server max time* para el primer criterio y los de la entropía para el segundo son los siguientes:

- *Estimación con el primer criterio* = abs(317.9 - 2.49) = 315.41 segundos
- *Estimación con el segundo criterio* = abs(224.2 - 1.9) = 222.3 segundos

Igual que en el anterior experimento el primer valor de la resta pertenece a la media del tiempo de transmisión total, pasado a segundos, y el segundo a la media del tiempo total para encontrar la solución (Columna Tiempo en la tabla), el cual ya estaba en segundos.

Experimento 7

En este experimento se plantea la relevancia del parámetro *número mínimo de replicaciones de un fichero*, ya que intuitivamente, cuanta más réplicas tenga un fichero en los servidores, más fácil será encontrar un servidor que tenga ese archivo y por lo tanto existirá un mayor rango de posibles asignaciones para cada petición.

Además, el experimento también plantea la realización de mediciones para las distintas heurísticas, ya que interesa conocer la relación en el comportamiento que existe entre estas y el parámetro en cuestión.

Para evitar desviaciones de los resultados, se trabajará con cinco semillas aleatorias y dos funciones de evaluación, generando para cada permutación cinco valores del parámetro. Las características del experimento se resumen en la siguiente tabla:

Observación	Relevancia del parámetro *número mínimo de replicaciones de un fichero*
Planteamiento	Se incrementará el número mínimo de réplicas de cada archivo de 5 a 25 en incrementos de 5. Se evaluarán los resultados generados
Hipótesis	El parámetro *Número mínimo de replicaciones de un archivo* afecta al tiempo total de transmisión (H0) o no
Método	▪ Se elegirán 5 semillas aleatorias, una por cada réplica ▪ Se ejecutarán 5 experimentos por cada semilla, y esto se hará 2 veces, una por cada heurística ▪ Se experimentará con los parámetros usando los siguientes valores fijos: • Número de usuarios: 200 • Número máximo de peticiones: 5 • Número de servidores: 50 ▪ Se usarán los operadores de mover e intercambiar combinados ▪ Se usará el generador de solución inicial *less time needed* ▪ Se usará la función heurística *Server max time* para el primer criterio ▪ Se usará la función heurística *Entropía* para el segundo criterio ▪ Se usará el algoritmo *Hill Climbing* ▪ Se medirán diferentes parámetros y resultados para compararlos

Los resultados obtenidos se pueden observar en las siguientes tablas y gráficos:

		Tiempo Total Transmisión					Tiempo				
Réplica	Semilla	5	10	15	20	25	5	10	15	20	25
1	9397	363430	285933	221324	170995	158425	17.757	16.411	5.039	1.956	5.252
2	3463	417283	292462	232595	196498	161434	11.2	2.52	17.228	22.832	6.678
3	3001	380798	260808	208851	186513	159787	11.873	13.714	7.063	13.331	11.902
4	7793	473460	305725	253579	212972	200950	40.932	48.921	11.848	19.1	34.079
5	9806	447891	293908	244035	180625	164350	13.322	10.659	13.074	4.764	8.237

Cuadro 6: Resultados del experimento 7 para *Server max time*

		Tiempo Total Transmisión					Tiempo				
Réplica	Semilla	5	10	15	20	25	5	10	15	20	25
1	9397	189749	187199	163999	145999	156749	193.941	165.966	116.304	127.937	95.937
2	3463	182199	180249	166249	180149	164199	203.725	166.288	181.485	145.047	119.756
3	3001	171899	171249	173499	181399	154999	154.971	129.905	105.324	122.578	92.409
4	7793	193499	202449	189399	180899	180499	492.071	419.663	331.836	257.59	315.525
5	9806	187199	187099	211699	211299	168049	239.957	173.516	197.61	184.028	133.123

Cuadro 7: Resultados del experimento 7 para *Entropía*

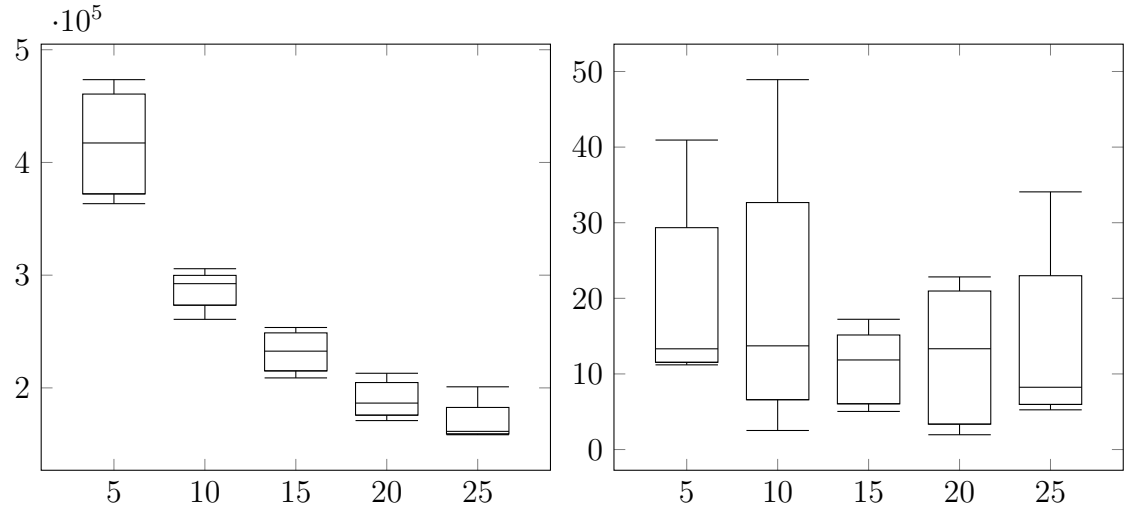


Figura 11: Distribución del tiempo total de transmisión y del tiempo de ejecución de *Server Max Time*

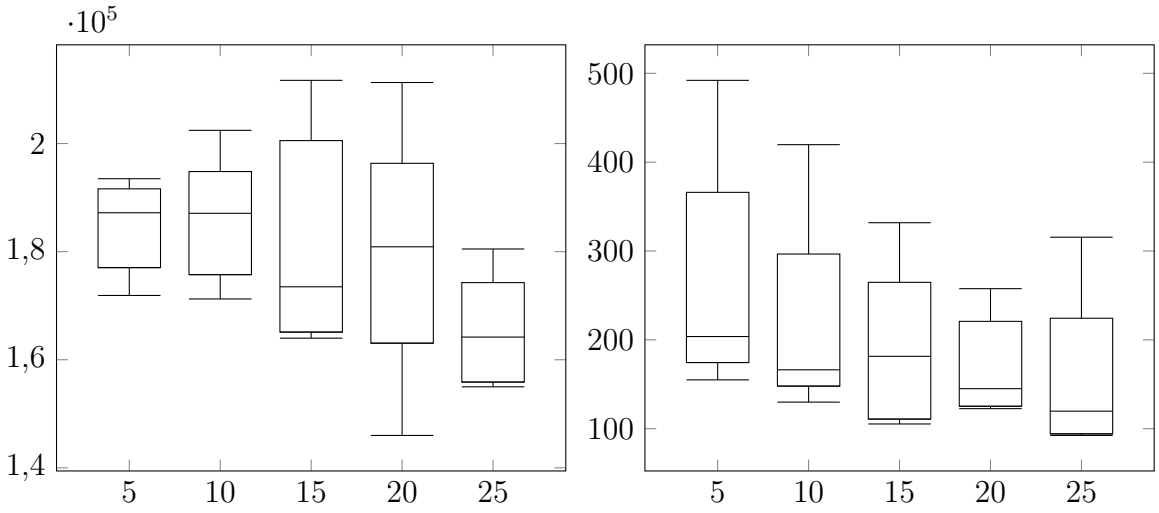


Figura 12: Distribución del tiempo total de transmisión y del tiempo de ejecución de *Entropía*

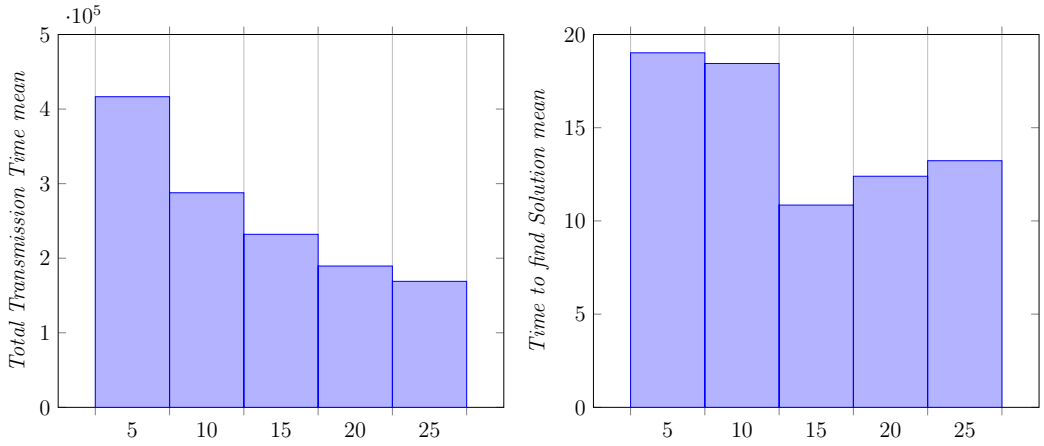


Figura 13: Medias del *Total Transmission Time* y del *Time to find solution* para los diferentes valores de replicación mínima de ficheros (*Server Max Time*).

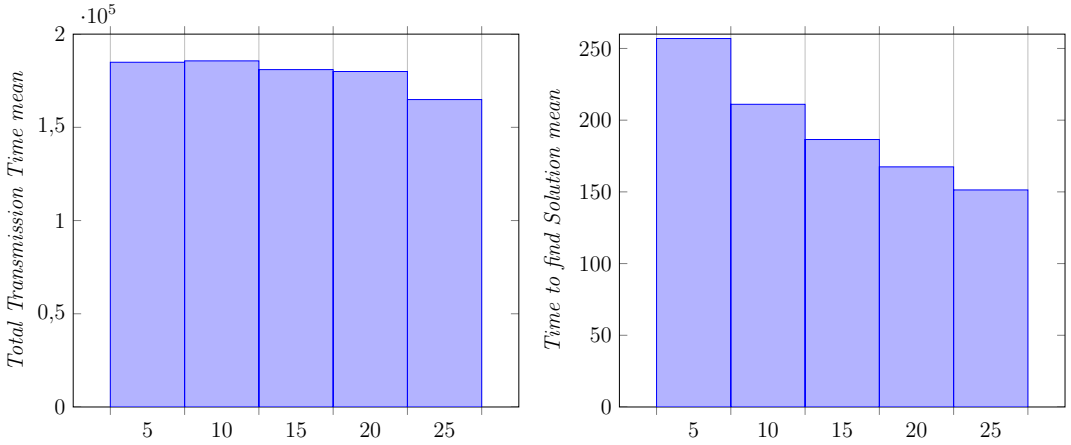


Figura 14: Medias del *Total Transmission Time* y del *Time to find solution* para los diferentes valores de replicación mínima de ficheros (*Entropía*).

En los gráficos anteriores se pueden ver las distribuciones y los valores de las medias de las características que se quieren observar en este experimento, es decir, el tiempo total de transmisión y el tiempo necesario para encontrar la solución. Esto se puede ver para ambas heurísticas, *Server max time* y *Entropía*.

En el caso de la heurística para el primer criterio, podemos observar que a medida que el número de replicaciones mínimas de los ficheros aumenta, el tiempo total de transmisión medio mejora, tal y como se podía intuir, habiendo una diferencia significativa entre las 5 y las 25 replicaciones. Aun así, se puede ver que en el caso del tiempo necesario para encontrar la solución esto no se cumple siempre, ya que como se puede observar, entre las 10 y 15 replicaciones hay una bajada significativa de este tiempo, pero este comienza a aumentar progresivamente con 20 y 25 replicaciones mínimas. Podría ser interesante ver si este fenómeno se propaga o es un repunte puntual.

Para la heurística del segundo criterio, se puede ver que la media del tiempo total de transmisión también va disminuyendo a medida que aumentan las replicaciones, pero en este caso el descenso es mínimo y no hay una gran diferencia entre el caso mínimo de 5 y el máximo de 25. Además, con este heurístico el tiempo para encontrar la solución si que disminuye progresivamente a medida que aumentan las replicaciones.

De esta forma se puede concluir que, en ambas heurísticas, al aumentar el número de replicaciones de un fichero se favorece a la disminución del tiempo total de transmisión como se había planteado en la hipótesis inicial (H0).

Conclusiones

En lo referente al primer criterio, se muestra claramente la diferencia entre los dos algoritmos. Se puede observar que *Hill Climbing* produce soluciones de menor calidad en un tiempo más elevado que el que necesita *Simulated Annealing* para producir soluciones de mayor calidad. Por lo que es indiscutible que *Simulated Annealing* es mejor a la hora de producir soluciones teniendo en cuenta este criterio.

En lo referente al segundo criterio, la función de entropía genera una curva 'suave', con ausencia de máximos y mínimos locales, pero que sin embargo crece lentamente. Es por este motivo que funciona tan bien para el algoritmo de *Hill Climbing*, cuyo peor contratiempo son los máximos o mínimos locales donde se puede quedar estancado. Contrariamente, en el caso de *Simulated Annealing*, nos encontramos con un algoritmo con un cierto margen de maniobra para sortear estos máximos y mínimos locales, pero que por contrario necesita la expansión de más nodos para lograr su objetivo. Es por este motivo que cuando le aplicamos la función heurística de entropía, es de algún modo demasiado floja. *Simulated Annealing* ya cuenta como algoritmo con las facilidades que proporciona la función de evaluación de entropía.

Expuestas las observaciones anteriores, dependiendo del criterio que se quiera usar es preferible que se use uno u otro algoritmo. Aún así y aunque para el segundo criterio el *Simulated Annealing* sea relativamente peor en cuestión de tiempo de transmisión, su velocidad en comparación al Hill Climbing puede hacer que la balanza caiga a su favor ya que su desempeño en general teniendo en cuenta ese factor es mejor.