Algoritmos de búsqueda local

Optimización de un servicio de compartición de vehículos mediante Hill Climbing y Simulated Annealing

Antonio J. Cabrera

Paul Gazel-Anthoine

Inteligencia Artificial
Primavera 2018-19

ÍNDICE

INTRODUCCIÓN

Esta práctica consiste en el estudio de los algoritmos de búsqueda local *Hill Climbing* y *Simulated Annealing* proporcionados en la librería AIMA en el contexto de un servicio de compartición de vehículos.

Para realizar el estudio, además de familiarizarnos con los conceptos y la librería, hemos tenido que pensar la representación óptima del estado del problema, la estrategia de generación del estado solución inicial (solución parcial, en este caso), la definición de los operadores para generar estados sucesores, la creación de funciones heurísticas que permitan una correcta evaluación de la calidad de los estados y, evidentemente, la realización de varios experimentos para evaluar nuestra implementación del problema, tomar decisiones y sacar conclusiones.

REPRESENTACIÓN DEL ESTADO

El estado debe ser representado con el mínimo de información que permita conocer la situación del problema en cada instante y que puede variar de un estado a otro.

En nuestro caso, esta implementación la empezamos a hacer con un BitSet de tamaño M*N, donde M son los conductores y N el total de usuarios del servicio. Cada conductor tenía N bits que le permitían marcar a qué usuarios llevaba a su destino. Además, se guardaba un entero para cada conductor, con la distancia mínima necesaria para llevar a todos los usuarios indicados en su parte correspondiente del BitSet.

Sin embargo, con esta representación, no se almacenaba el orden con el que los conductores recogían a sus pasajeros asignados. En su lugar, cada vez que se calculaba el heurístico para un estado, se realizaba un cálculo de distancia mínima para los coches que cambiaban respecto al estado anterior. Pese a ser una representación con mucho ahorro en memoria, el cálculo del heurístico provocaba una lentitud intolerable en la ejecución, por lo que desechamos esta representación del estado.

Finalmente, nuestra implementación es un ArrayList de ArrayLists de Usuarios, es decir, una solución por listas de adyacencia que corresponden a los caminos.

En total, estas listas de adyacencia almacenan 2*N elementos, porque cada usuario debe ser visitado dos veces.

Con esta representación, mantenemos siempre almacenado el camino que hace cada conductor, por lo que el cálculo del heurístico se ha podido simplificar mucho y el tiempo de ejecución se ha disminuido drásticamente.

Esta representación supone aproximadamente 16 Bytes por usuario (para una máquina de 64-bits), lo cual supone unos 3.2 KB para un estado de 200 usuarios, que es el número más habitual con el que se ha experimentado.

Para un PC como el que se ha utilizado para la experimentación, con 16GB de memoria, esto supone que en un momento dado pueden convivir más de 4 millones de estados, por lo que no nos limitará, como veremos a continuación, en la ejecución de los experimentos. Por otro lado, debemos ser conscientes de que para números más grandes (por ejemplo, 2.000 usuarios), el número de estados sucesores sí está restringido por el tamaño de la memoria para una ejecución con *Hill Climbing*. En ese caso, se debería estudiar la posibilidad de limitar el espacio de búsqueda si queremos mantener la representación elegida.

OPERADORES Y GENERACIÓN DE ESTADOS SUCESORES

Los operadores principales que hemos valorado han sido los siguientes:

1. Mover pasajero P a conductor C.
2. Swap de pasajero P1 (con conductor C1) y pasajero p2 (con conductor C2).
3. Mover (y desactivar) conductor C1 a conductor C2.

Por la implementación que hemos hecho, el segundo operador se hace siempre entre pasajeros de dos conductores diferentes. Es decir, nunca hacemos un *swap* interno. Esto es posible gracias a que siempre que introducimos un pasajero (o conductor desactivado) en un coche, siempre lo ponemos en la mejor posición posible, de forma voraz, para minimizar la distancia.

Esto tiene unas limitaciones, por supuesto, ya que pese a introducir el pasajero en la mejor posición posible (que respete el enunciado), es posible que un cambio radical en el recorrido proporcionara una distancia todavía inferior. Sin embargo, sabemos que si

algún pasajero está penalizando gravemente el recorrido, el propio algoritmo se encargará de moverlo a otro vehículo y, si se da el caso, podría regresar al coche en mejor posición.

El operador Swap, en cualquier caso, no es suficiente. Queremos que un coche pueda quedarse sin pasajeros para poder reducir el número de conductores activos. Así pues, surge la necesidad de los operadores 1 y 3.

El operador 1 genera tantos estados sucesores como conductores activos haya (menos uno, donde ya está situado el pasajero) para cada pasajero, para cada usuario si incluímos el operador 3 a este cálculo, podemos asumir un caso peor de N*M posibles sucesores, donde M es el número de conductores. Para el experimento más habitual con N = 200 y M = 100, esto serían aproximadamente 20.000 estados sucesores y alrededor de 64 MB de memoria.

El operador 2 genera tantos estados sucesores como posibles intercambios con pasajeros de otros vehículos. Asumimos un caso peor de N^2 posibles sucesores. Para N = 200 serían aproximadamente 40.000 estados sucesores y unos 128 MB de memoria.

Así pues, para nuestros experimentos más comunes podemos esperar un uso de memoria inferior a 200MB. Sin embargo, podemos prever que hay una limitación muy importante para el algoritmo Hill Climbing por el factor de crecimiento que tiene con respecto al número de usuarios. No así para el *Simulated Annealing*, el cual no necesita generar más de un estado sucesor al mismo tiempo.

GENERADOR DE SOLUCIONES INICIALES

Las estrategias de generación de soluciones iniciales utilizadas son las siguientes:

1. Inicialización aleatoria equitativa
2. Inicialización desequilibrada

Ambas son soluciones parciales, confiando al algoritmo de búsqueda local la reducción de la distancia recorrida de cada conductor que sitúe el estado en el espacio de soluciones.

La primera estrategia distribuye los pasajeros de forma aleatoria entre los conductores,

respetando que cada conductor lleve N pasajeros[1], de forma que:

$$N = \frac{\#\ pasajeros}{\#\ conductores}$$

El comportamiento del algoritmo Hill Climbing con esta solución, como veremos en los experimentos, es realmente bueno para minimizar distancia.

Además, hemos implementado una alternativa que presenta una situación más proclive a minimizar coches, para estudiar si generaba soluciones válidas y a su vez podíamos obtener un buen resultado de minimización de distancia. Esta segunda solución inicial reparte todos los pasajeros entre la mitad de los conductores, dejando el resto sin pasajeros.

Ambos estados iniciales parten de la base de que todos los usuarios del *carsharing* van a ser llevados a su trabajo. Nunca un pasajero o conductor está fuera de un coche. En estas soluciones parciales iniciales, los pasajeros asignados se posicionan siempre dos veces de forma consecutiva en el camino, de manera que nunca hay más de un pasajero en un vehículo (además del conductor).

FUNCIONES HEURÍSTICAS

Para las funciones heurísticas debemos diferenciar los dos criterios de evaluación de calidad de la solución.

En el primer criterio (minimización de la distancia total recorrida), un problema bastante más simple, debemos establecer una penalización por cada unidad de distancia. De esta manera, los algoritmos seleccionarán estados cuya distancia total recorrida sea mínima.

En el segundo criterio (donde también buscamos minimizar el número de conductores), debemos penalizar severamente el número de conductores activos, además de intentar obtener asignaciones desequilibradas de pasajeros, para favorecer la producción de conductores sin pasajeros y que se puedan desactivar conductores.

[1] Asignando un pasajero extra por conductor en el caso de que la proporción no sea entera.

Lo principal que tienen en común estas funciones heurísticas es que deben penalizar de forma excepcional los recorridos de los conductores que superen las 300 manzanas, porque esta debe ser la principal motivación que tienen los algoritmos de avanzar: entrar en el espacio de soluciones.

La primera función heurística consiste en lo siguiente:

a. Para cada conductor activo, coger la distancia de su itinerario: si esta supera 300, añadimos al acumulador esta distancia multiplicada por 500, si no, la añadimos directamente.

$$h(x) = \sum_{i=1}^{M} dist(Ci) * penalty$$

M = Número de conductores activos
penalty = {500: si dist(Ci > 300), 1: en cualquier otro caso}

Con este simple heurístico, el algoritmo nos da solución de forma prácticamente infalible.

b. Para la segunda función heurística, simplemente hemos añadido que, antes de acabar el cálculo del acumulador, lo multiplicamos por el número de conductores activos. De esta forma, el algoritmo premia los sucesores con menos conductores.

$$h(x) = M * (\sum_{i=1}^{M} dist(Ci) * penalty)$$

M = Número de conductores activos
penalty = {500: si dist(Ci > 300), 1: en cualquier otro caso}

Sin embargo, debido a lo fuertemente ligadas que están las variables de distancia total y número de conductores, nos hemos dado cuenta de que el heurístico (b) nos deja el primero totalmente inútil. El segundo heurístico no solo nos minimiza el número de conductores, sino que en el proceso, también es capaz de minimizar todavía más la distancia recorrida.

Cualquier ejecución realizada con el heurístico (a), siempre ha sido superada con el heurístico (b). De forma que los experimentos que mostramos a continuación han sido ejecutados usando la heurística que tiene en cuenta el número de conductores activos,

incluso cuando el objetivo no era reducir el número de conductores.

Por supuesto, también hemos realizado pruebas con diferentes funciones heurísticas que iban en la misma línea, como por ejemplo:

c. $h(x) = penalty * dist(Ci)$

penalty = { $dist(Ci)^2$ *: si dist(Ci > 300),* *1: en cualquier otro caso*}

d. $h(x) = \sum_{i=1}^{M} dist(Ci) * penalty * M$

penalty = {*10: si dist(Ci > 300),* *0.1: en cualquier otro caso*}

M = Número de conductores activos

Sin embargo, aunque en algún caso han dado un resultado aceptable o incluso bastante bueno, especialmente con el caso (d), nos hemos encontrado con mayor inconsistencia de cara a encontrar una solución válida. El heurístico (d), por ejemplo, podría argumentarse que es bastante bueno, proporcionando quizás las soluciones más llamativas de reducción de vehículos, con una precisión menor para encontrar soluciones válidas.

Hemos empezado los experimentos teniendo en cuenta los heurísticos (b) y (d). Sin embargo, rápidamente hemos visto cuál era el defecto de (d). Además, este heurístico funciona mejor con los operadores y la estrategia de generación de soluciones iniciales que en los dos primeros experimentos han funcionado peor para (b), por lo que hemos tenido que tomar una decisión para no tener que desdoblar el estudio en exceso y hemos decidido continuar las experimentaciones con el heurístico **(b),** el más consistente.

EXPERIMENTO 1

Observación	Es posible que el operador Swap no sea rentable si la solución no es mucho mejor y el tiempo de ejecución crece en exceso.
Hipótesis	La ausencia del operador Swap nos dará una solución de calidad similar, pero en la mitad de tiempo.
Método	Estudio de los operadores para una función heurística que optimice la distancia total recorrida. • 200 Usuarios (100 Conductores) mediante *Hill Climbing* • 10 Ejecuciones con semillas diferentes • Solución inicial parcial con asignación aleatoria

Resultados:

MOVE + REMOVE							
Ejecución	**Dist. inicial (km)**	**Dist. final (km) - MR**	**Solución ?**	**Conductores usados (MR)**	**Tiempo (s)**	**% Dist. Reducida**	**% Conduc. Reducidos**
1	1968.9	1171.1	1	53	10.449	40.52%	47.00%
2	1897.3	1162.7	1	54	9.980	38.72%	46.00%
3	1946.3	1174.3	1	52	10.284	39.67%	48.00%
4	2042.7	1238.7	1	55	10.077	39.36%	45.00%
5	1941.2	1204.9	1	54	9.658	37.93%	46.00%
6	2047.8	1207.1	1	51	10.524	41.05%	49.00%
7	1895.7	1150.4	1	52	10.252	39.32%	48.00%
8	1979.8	1161.6	1	52	10.581	41.33%	48.00%
9	2043.1	1192.1	1	57	10.275	41.65%	43.00%
10	1888.6	1125.2	1	52	10.535	40.42%	48.00%
Promedio	**1965.14**	**1178.81**	**1**	**53.2**	**10.261**	**40.00%**	**46.80%**

Tabla 1: Hill Climbing, operadores MOVE y REMOVE, inicialización aleatoria equitativa.

SWAP + MOVE + REMOVE							
Ejecución	**Dist. inicial (km)**	**Dist. final (km) - SMR**	**Solución ?**	**Conductores usados (SMR)**	**Tiempo (s)**	**% Dist. Reducida**	**% Conduc. Reducidos**
1	1968.9	1042.7	1	68	19.022	47.04%	32.00%
2	1897.3	973.2	1	65	20.992	48.71%	35.00%
3	1946.3	1009.3	1	59	21.954	48.14%	41.00%
4	2042.7	1032.3	1	61	23.661	49.46%	39.00%
5	1941.2	944.9	1	65	22.202	51.32%	35.00%
6	2047.8	1039.4	1	62	21.599	49.24%	38.00%
7	1895.7	942.3	1	61	22.391	50.29%	39.00%
8	1979.8	990.6	1	62	24.657	49.96%	38.00%
9	2043.1	982.5	1	61	21.711	51.91%	39.00%
10	1888.6	944.3	1	62	22.302	50.00%	38.00%
Promedio	**1965.14**	**990.15**	**1**	**62.6**	**22.049**	**49.61%**	**37.40%**

Tabla 2: Hill Climbing, operadores SWAP, MOVE y REMOVE, inicialización aleatoria equitativa.

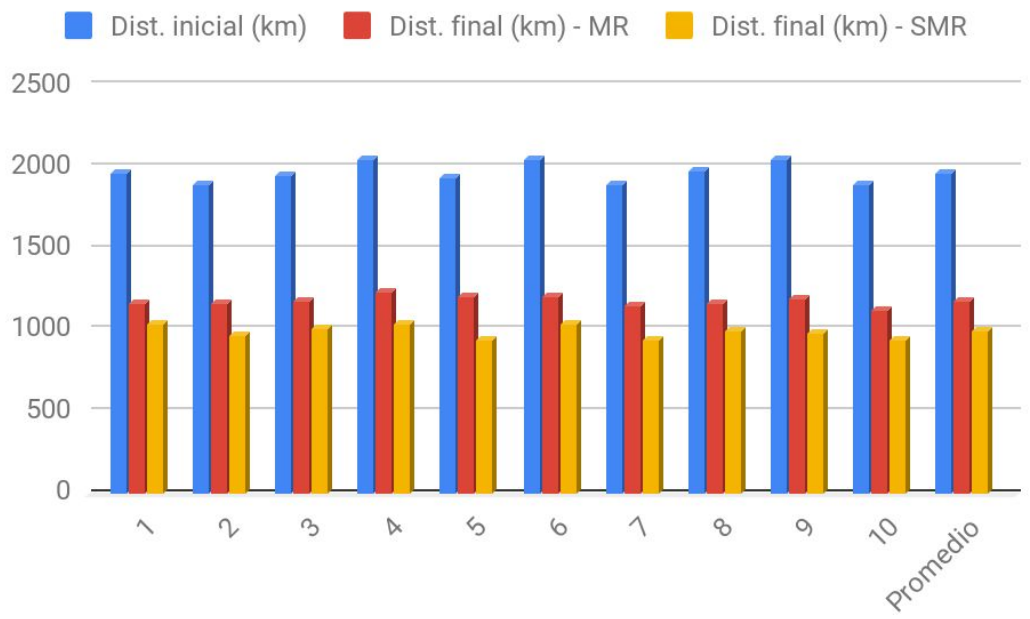

Figura 1: Hill Climbing, inicialización aleatoria equitativa.
Distancia final según operadores: (MR) - MOVE y REMOVE, (SMR) - SWAP, MOVE y REMOVE.

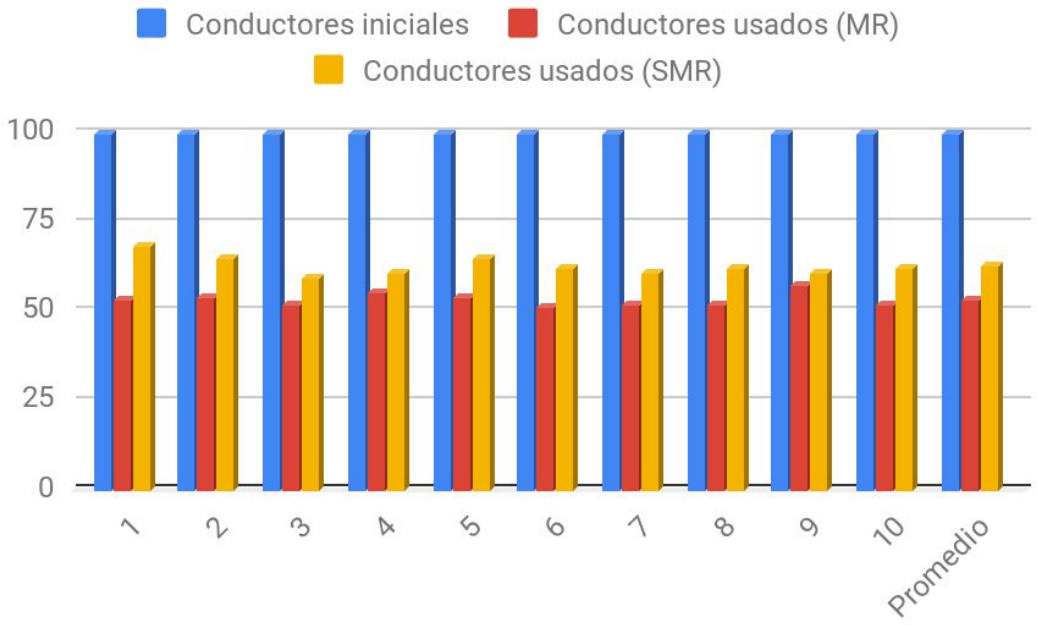

Figura 2: Hill Climbing, inicialización aleatoria equitativa.
Conductores finales según operadores: (MR) - MOVE y REMOVE, (SMR) - SWAP, MOVE y REMOVE.

Comentarios:

Tras ver los resultados del primer experimento, podemos revisar nuestra hipótesis inicial. Por un lado, podemos rechazar la hipótesis nula, ya que, mirando el primer criterio de evaluación de la solución, vemos que en promedio la distancia final obtenida incluyendo el operador Swap es 188.7km menor que cuando no lo incluímos. Por otro, encontramos que utilizando el criterio de minimización de conductores, el resultado no es en absoluto similar, pues con el Swap se utilizan, de media, 9.4 conductores más.

Además de esto, vemos que sí se cumple nuestra previsión en cuanto a tiempo de ejecución. La ejecución sin el operador Swap únicamente requiere un 46.5% del tiempo que la otra.

Así pues, para establecer una configuración ganadora, debemos tener en cuenta nuestras prioridades. Para nosotros, el criterio principal de evaluación siempre va a ser minimizar la distancia. La minimización de conductores nos va a permitir desempatar ante dos soluciones con distancia similar.

Por lo tanto, en este estudio continuaremos utilizando el operador Swap.

EXPERIMENTO 2

Observación	Es posible que creando una situación desequilibrada de repartición de pasajeros, consigamos mantener una buena minimización de distancia a la vez que disminuyen los conductores activos.
Hipótesis	Con la solución inicial desequilibrada (DES), conseguiremos mantener la distancia recorrida en valores similares y disminuir el número de conductores activos respecto a la equitativa (EQ).
Método	Estudio de las estrategias de generación de la solución inicial para la función heurística que optimice la distancia total recorrida. • 200 Usuarios (100 Conductores) mediante *Hill Climbing*. • 10 Ejecuciones con semillas diferentes.

Resultados:

La primera estrategia (EQ) corresponde a la tabla 2 del primer experimento.

Reparto inicial desequilibrado							
Ejecución	**Dist. inicial (km)**	**Dist. final (km) - DES**	**Solución?**	**Conductores usados (DES)**	**Tiempo (s)**	**% Dist. Reducida**	**% Conduc. Reducidos**
1	2071.9	1218.2	1	53	23.252	41.20%	47.00%
2	1972.7	1199.5	1	54	21.806	39.20%	46.00%
3	1966.1	1202.3	1	53	21.229	38.85%	47.00%
4	2048.1	1188.8	1	51	24.540	41.96%	49.00%
5	1914	1141.1	1	50	25.465	40.38%	50.00%
6	2026	1236.3	1	52	20.437	38.98%	48.00%
7	1900.3	1138	1	49	21.756	40.11%	51.00%
8	1971	1142.6	1	49	24.711	42.03%	51.00%
9	1963.7	1202.1	1	51	21.775	38.78%	49.00%
10	1940.6	1119.6	1	49	22.823	42.31%	51.00%
Promedio	**1977.44**	**1178.85**	**1**	**51.1**	**22.779**	**40.38%**	**48.90%**

Tabla 3: Hill Climbing, operadores SWAP, MOVE y REMOVE, inicialización desequilibrada.

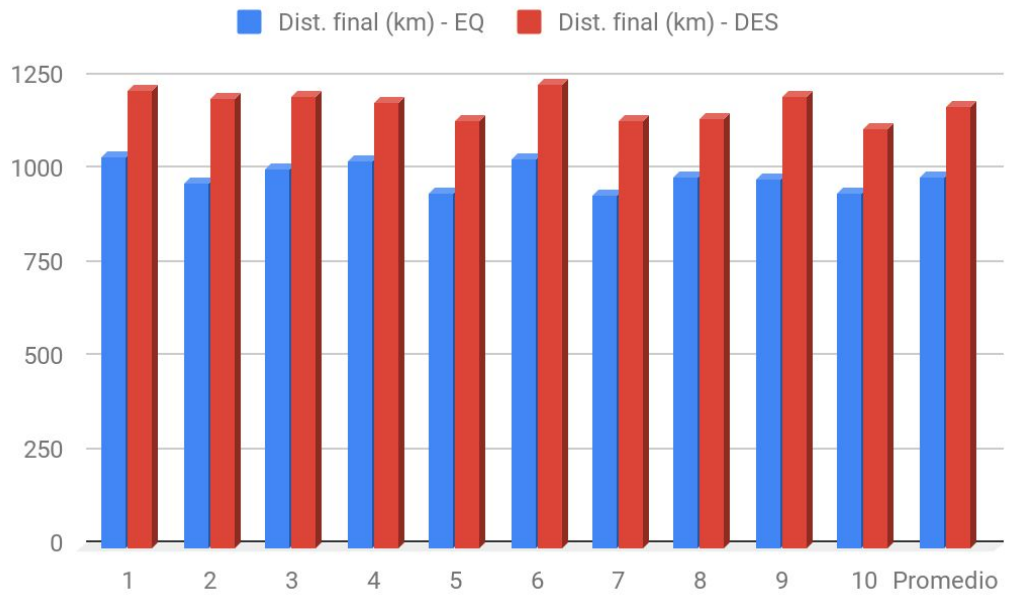

Figura 3: Hill Climbing, operadores (SMR). Distancia final según estrategia de solución inicial.

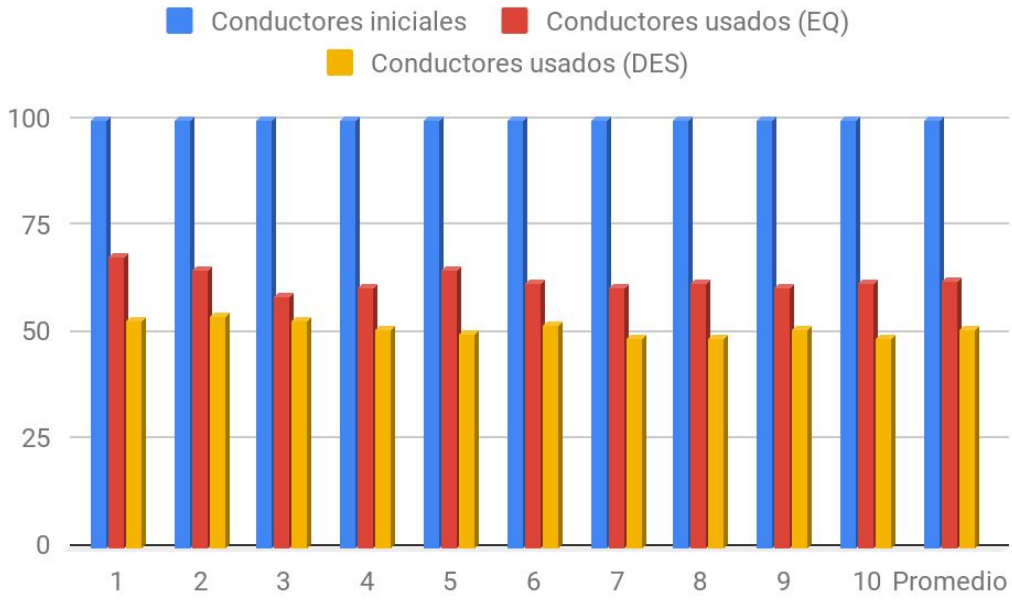

Figura 4: Hill Climbing, inicialización aleatoria equitativa.
Conductores finales según operadores: (MR) - MOVE y REMOVE, (SMR) - SWAP, MOVE y REMOVE.

Comentarios:

Si observamos la figura 4, nos encontramos con un caso muy similar al de la figura 1. La generación de la solución desequilibrada tiende, igual que el desuso del operador Swap, a darnos soluciones peores en cuanto a distancia total, a cambio de una minimización drástica de los vehículos utilizados.

La buena noticia es que en ambos casos obtenemos una solución correcta y utilizable. Sin embargo, la hipótesis nula es rechazada en base a que no somos capaces de mantener una distancia recorrida mínima.

De este experimento concluimos que, para nuestros intereses, va a ser más útil la estrategia de inicializar los vehículos con una carga de pasajeros equilibrada.

EXPERIMENTO 3

Observación	Para realizar futuros experimentos comparando los dos algoritmos, debemos parametrizar correctamente el *Simulated Annealing* para las condiciones iniciales y los operadores propuestos.
Método	Utilizando el estado inicial y los operadores que han dado mejor resultado en los experimentos anteriores, realizamos un estudio de parametrización del *Simulated Annealing*. ● Utilizando 5 escenarios y la misma función heurística. ● Estudio aumentando el número de iteraciones ● Estudio aumentando las iteraciones por paso de temperatura ● Estudio aumentando la variable de temperatura inicial k ● Estudio aumentando la velocidad de enfriamiento con λ

Resultados:

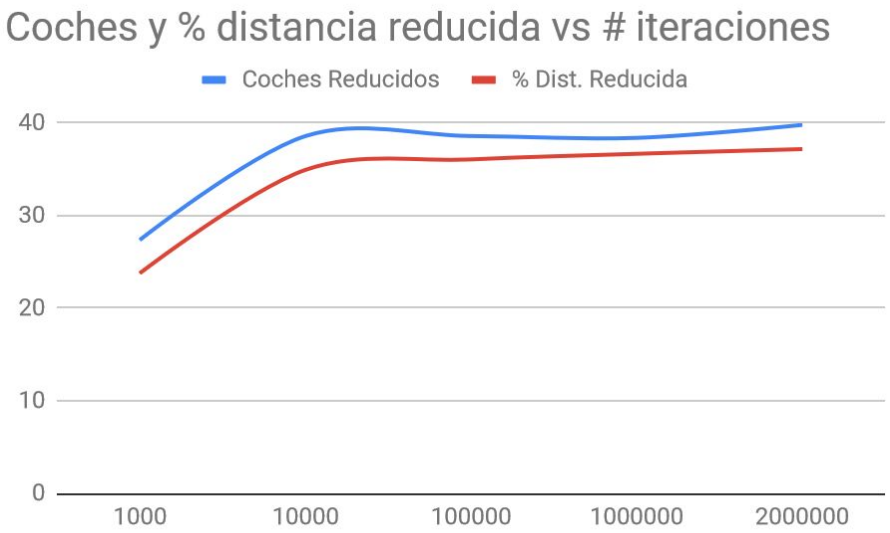

Figura 5: Simulated Annealing, solución inicial EQ y operadores SMR.
Conductores y porcentaje de distancia reducida según número de iteraciones del algoritmo.

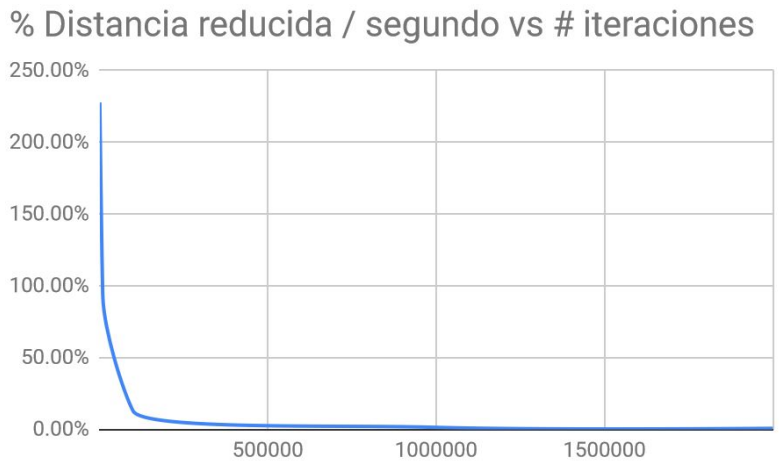

Figura 6: Simulated Annealing, solución inicial EQ y operadores SMR.
Porcentaje de distancia reducida por segundo de ejecución según número de iteraciones.

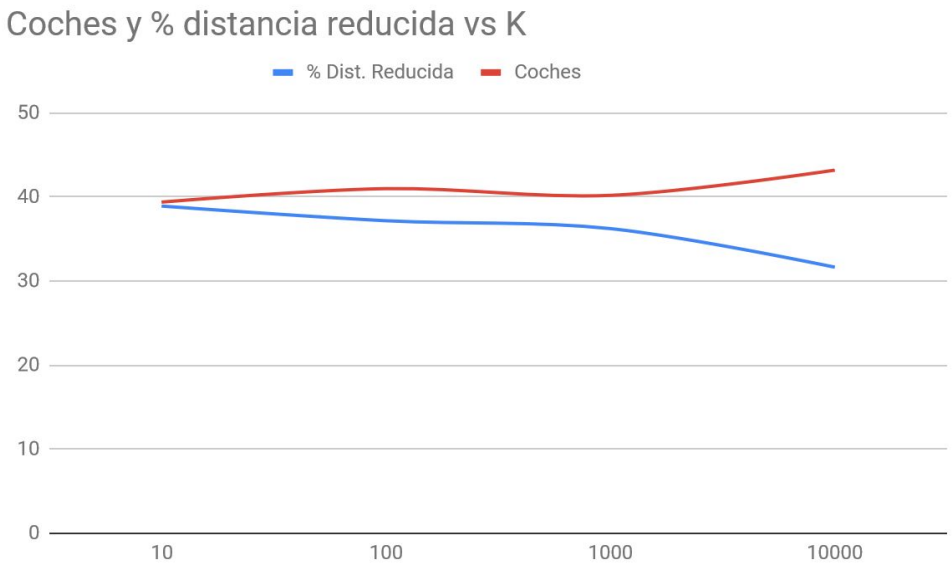

Figura 7: Simulated Annealing, solución inicial EQ y operadores SMR.
Conductores y porcentaje de distancia reducida según el parámetro K (temperatura inicial).

Resúmenes					
# Iteraciones	**Coches Reducidos**	**Tiempo (s)**	**% Dist. Reducida**	**%Dist Red / s**	**Efectividad**
1000	27.4	0.105	23.80	227.46%	80.00%
10000	38.6	0.377	34.95	92.59%	80.00%
100000	38.6	2.823	36.07	12.78%	100.00%
1000000	38.4	25.867	36.68	1.42%	100.00%
2000000	39.8	51.790	37.19	0.72%	80.00%
Iteraciones / paso de Tª	**Coches Reducidos**	**Tiempo (s)**	**% Dist. Reducida**	**%Dist Red / s**	**Efectividad**
100	41	2.851	37.79	13.26%	80.00%
1000	40.4	2.795	36.39	13.02%	100.00%
10000	40.6	2.784	36.49	13.11%	60.00%
K	**Coches Reducidos**	**Tiempo (s)**	**% Dist. Reducida**	**%Dist Red / s**	**Efectividad**
10	39.4	2.801	38.92	13.90%	100.00%
100	41	2.779	37.18	13.38%	80.00%
1000	40.2	2.829	36.24	12.81%	80.00%
10000	43.2	2.888	31.66	10.96%	100.00%
Lambda	**Coches Reducidos**	**Tiempo (s)**	**% Dist. Reducida**	**%Dist Red / s**	**Efectividad**
0.001	39.8	2.769	37.35	13.49%	80.00%
0.0001	40	2.799	36.43	13.01%	80.00%
0.00001	39.2	2.759	36.48	13.22%	60.00%
0.000001	40.4	2.778	37.64	13.55%	80.00%
0.0000001	39.8	2.783	37.21	13.37%	100.00%

Tabla 4: Resumen de resultados (Simulated Annealing), solución inicial (EQ), operadores (SMR). Promedio de ejecuciones con diferente máximo número de iteraciones, iteraciones por paso de temperatura, temperatura inicial y variable de enfriamiento.

Comentarios:

En este experimento la prioridad ha sido encontrar una solución correcta. Además de esto, se ha intentado encontrar el máximo equilibrio entre minimización de distancia y tiempo de ejecución.

Si observamos la figura 5 vamos a ver una evidencia. Si subimos el número de iteraciones, generalmente encontramos mayor minimización, tanto de distancia como de coches usados. Sin embargo, la gráfica se aplana mucho a partir de las 100.000 iteraciones máximas, por lo que vemos que invertir tiempo en una mejor solución no es muy rentable a partir de ese punto.

En la figura 6 nos confirma lo que comentamos en el párrafo anterior. En la gráfica se ve un codo importante hacia las 100.000 iteraciones máximas. Consideraremos 100.000 como el número de iteraciones máximas para nuestras ejecuciones futuras.

En cuanto al número de iteraciones por paso de temperatura, vemos que no hay un gran cambio para diferentes valores. Vamos a escoger el término medio, que además es el que ha dado mejor efectividad de soluciones válidas.

En el estudio del parámetro K nos encontramos con que a medida que hacemos crecer su valor, la gráfica diverge (figura 7). Es decir, empezamos a encontrar menor número de vehículos utilizados a cambio de algo más distancia recorrida.

Manteniendo el criterio seguido hasta ahora, priorizamos que la distancia total sea mínima y que sea solución válida el mayor número de veces, por lo que cogemos el valor 10.

Estudiando la variable de enfriamiento no hemos encontrado grandes diferencias utilizando los valores 100.000, 1.000 y 10 para el resto de parámetros. Por ello, hemos seleccionado el que nos ha ofrecido un 100% de efectividad.

EXPERIMENTO 4

Observación	El número de estados sucesores crece drásticamente cuando incrementamos el número de usuarios. Queremos ver cuánto y hasta qué valores podemos permitirnos trabajar con *Hill Climbing*.
Hipótesis	
Método	Estudio del tiempo de ejecución para *Hill Climbing*, utilizando los operadores y solución inicial con mejores resultados. ● Para tres semillas diferentes ● Empezando con 200 usuarios y añadiendo 100 cada vez. ● Manteniendo la proporción de M = N/2, siendo M los conductores y N los pasajeros totales.

Resultados:

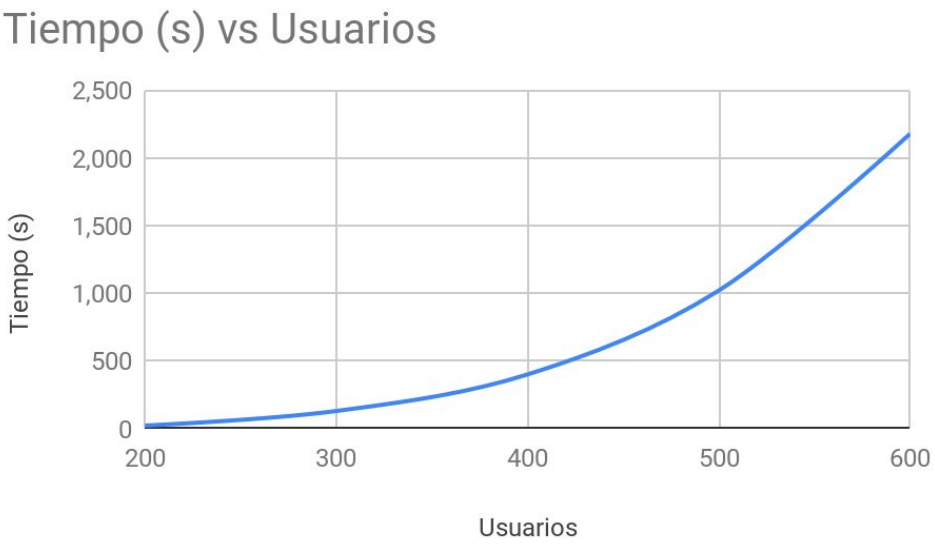

Figura 8: Hill Climbing, solución inicial EQ y operadores SMR.
Tiempo de ejecución según el número de usuarios (con la mitad de conductores).

Usuarios	Tiempo (s)	Tiempo (s) / Usuario
200	21	0.107
300	129	0.430
400	399	0.998
500	1,019	2.038
600	2,178	3.630

Tabla 5: Resumen de medición de tiempo (Hill Climbing), solución inicial (EQ), operadores (SMR). Promedio de ejecuciones por número de usuarios y con proporción 1:1 entre conductores y pasajeros.

Como referencia, para 600 usuarios la ejecución es de 36.3 minutos.

Comentarios:

Como podemos observar en la tabla 5, el tiempo requerido por usuario crece mucho más rápido a medida que incrementamos el número de estos. Es una situación donde el tiempo crece de forma polinómica, pues cada nuevo usuario provoca un crecimiento cuadrático de posibles sucesores, los cuales se evalúan de forma no exponencial.

Esto provoca que, dependiendo del uso previsto para este programa, podría ser una limitación muy importante y podría provocar que el algoritmo *Simulated Annealing* se viera muy favorecido.

Teniendo en cuenta que la ejecución para 600 usuarios supera la media hora y que el enunciado sugiere un uso mensual del cálculo de rutas, haciendo una aproximación se prevé que duplicar los usuarios, hasta 1200, podría dar tiempos superiores a las 10 horas. Según nuestra implementación y considerando el hardware utilizado para el estudio, este algoritmo parece poco viable para un producto exitoso aplicado a un entorno real con miles de personas.

EXPERIMENTO 5

Observación	Queremos ver qué tan rentable es el algoritmo *Hill Climbing* relacionando distancia reducida frente a pasajeros para poder compararlo con el otro algoritmo que estamos estudiando.
Método	Con el mismo escenario del primer experimento, para el *Hill Climbing*, encontrar relación entre la distancia total recorrida y el tiempo de ejecución. • Uso de la heurística que minimiza la distancia total recorrida y que también minimiza el número de conductores.

Resultados:

Hill Climbing							
Ejecución	**Dist. inicial (km)**	**Dist. final (km) - EQ**	**Solución ?**	**Conductores eliminados (EQ)**	**Tiempo (s)**	**% Dist. Reducida**	**(HC) dist. reducida / tiempo**
1	1968.9	1042.7	TRUE	32	18.499	47.04%	50.07
2	2042.7	1032.3	TRUE	39	23.286	49.46%	43.39
3	2053.7	1052.3	TRUE	40	22.007	48.76%	45.50
4	1909.9	945	TRUE	36	22.181	50.52%	43.50
5	1994.7	1041.8	TRUE	37	22.315	47.77%	42.70
6	1898	939.8	TRUE	37	23.006	50.48%	41.65
7	2027.8	1041	TRUE	37	21.172	48.66%	46.61
8	2029.1	956.5	TRUE	39	24.108	52.86%	44.49
9	2010.2	1046.1	TRUE	39	22.823	47.96%	42.24
10	2050.8	978.1	TRUE	39	24.462	52.31%	43.85
Promedio	**1998.58**	**1007.56**	**1**	**37.5**	**22.386**	**49.58%**	**44.40**

Tabla 6: Estudio de distancia reducida y tiempo de ejecución (Hill Climbing), solución inicial (EQ), operadores (SMR).

Comentarios:

Como vemos, para 200 usuarios, obtenemos un promedio de 44.4 km reducidos por cada segundo de ejecución.

Además, estas ejecuciones consiguen reducir aproximadamente 1.7 conductores activos por segundo de ejecución.

Las conclusiones de este experimento se hacen conjuntamente con las del experimento 6 para comparar algoritmos.

EXPERIMENTO 6

Observación	Queremos ver qué tan rentable es el algoritmo *Simulated Annealing* relacionando distancia reducida frente a pasajeros para poder compararlo con los resultados del experimento 5.
Hipótesis	Pese a reducir menos la distancia, el *Simulated Annealing* se ejecuta mucho más rápido.
Método	Con el mismo escenario del primer experimento, para el *Simulated Annealing*, encontrar relación entre la distancia total recorrida y el tiempo de ejecución. Comparar con *Hill Climbing*. • Uso de la heurística que minimiza la distancia total recorrida y que también minimiza el número de conductores.

Resultados:

Simulated Annealing							
Ejecución	**Dist. inicial (km)**	**Dist. final (km) - EQ**	**Solución ?**	**Conductores eliminados (EQ)**	**Tiempo (s)**	**% Dist. Reducida**	**(SA) dist. reducida / tiempo**
1	1968.9	1237.4	TRUE	43	2.791	37.15%	262.11
2	2042.7	1252.5	TRUE	42	2.767	38.68%	285.56
3	2053.7	1274.5	TRUE	41	2.775	37.94%	280.75
4	1909.9	1158.7	TRUE	40	2.765	39.33%	271.68
5	1994.7	1196.7	TRUE	42	2.872	40.01%	277.86
6	1898	1172.8	TRUE	41	2.755	38.21%	263.21
7	2027.8	1259	FALSE	37	2.768	37.91%	277.71
8	2029.1	1168.9	TRUE	42	2.770	42.39%	310.54
9	2010.2	1200	TRUE	40	2.875	40.30%	281.83
10	2050.8	1252.3	TRUE	41	2.829	38.94%	282.29
Promedio	**1998.58**	**1217.28**	**0.9**	**40.9**	**2.797**	**39.09%**	**279.35**

Tabla 7: Estudio de distancia reducida y tiempo de ejecución (Simulated Annealing), solución inicial (EQ), operadores (SMR).

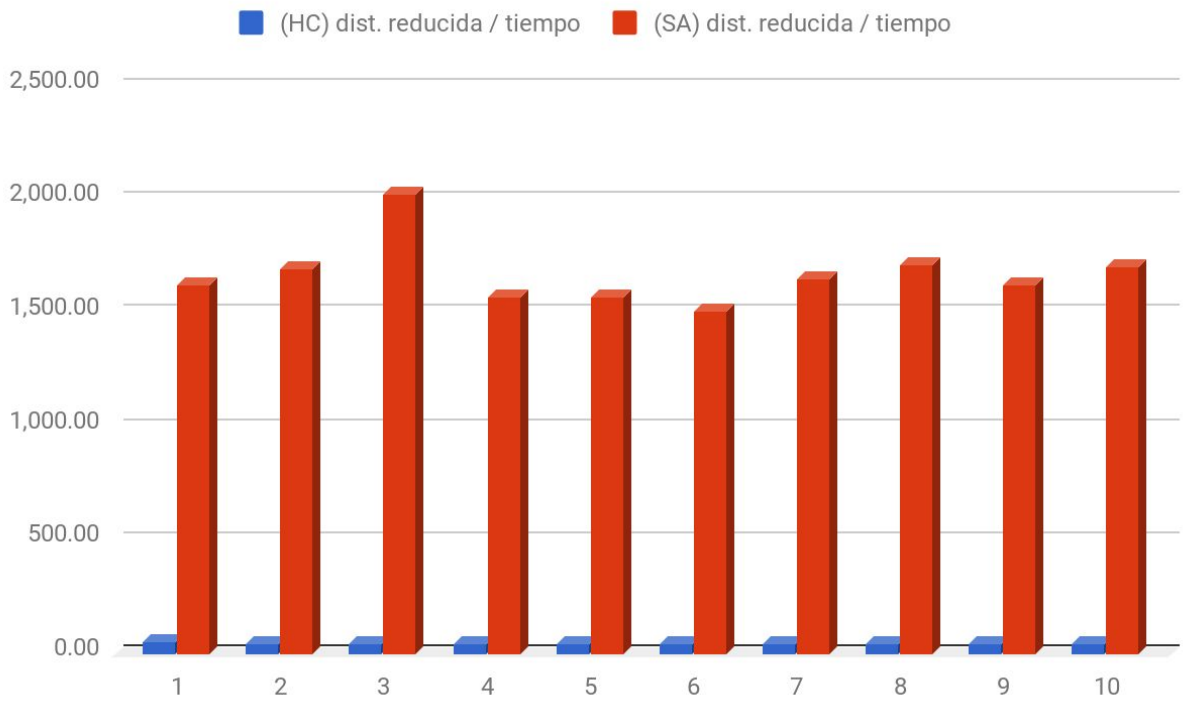

Figura 9: Comparativa distancia reducida según tiempo empleado. Estado inicial EQ y operadores SMR (HC, Hill Climbing) vs (SA, Simulated Annealing)

Comentarios:

En este caso, el algoritmo ha reducido 279.35 km por segundo de ejecución, de media. Además, el número de conductores que ha desactivado es 40.9 de media.

Tras ver los resultados de los dos últimos experimentos, podemos comprobar que el algoritmo *Hill Climbing* reduce aproximadamente 210 km más, de media, que el *Simulated Annealing*. En este sentido, el ganador es claro.

Sin embargo, como hemos visto en el experimento 4, el crecimiento temporal que supone aumentar el número de usuarios nos hace plantearnos la escalabilidad de nuestro programa utilizando *Hill Climbing*.

Buscando soluciones, observamos el experimento 3, donde el *Simulated Annealing* nos permite parametrizar el número de iteraciones a realizar, algo muy útil si queremos hacer el problema más grande. Evidentemente, si incrementamos el número de usuarios, también tendremos que aumentar el número de iteraciones máximas.

Sin embargo, el tiempo de ejecución crece de forma lineal para este algoritmo, y como vemos en la figura 9, la mejora de distancia por tiempo de ejecución es muy superior al *Hill Climbing*. Incluso el número de conductores reducidos lo hace favorable.

De esta forma, podemos ejecutar el programa utilizando *Simulated Annealing* para, por ejemplo, 1.200 usuarios, que sería un problema seis veces más grande y que hemos calculado que sería muy costoso para *Hill Climbing* (del orden de varias horas).

Seguramente el número de iteraciones máximas deba ser superior que para 200 usuarios para obtener un buen resultado. En cualquier caso, unos segundos (tal vez minutos) después, obtenemos un resultado que es la gran mayoría de las veces es solución con una optimización aceptable. Este algoritmo sí parece apto para un producto escalable.

Cabe mencionar que en la tabla 7 vemos un caso donde el *Simulated Annealing* no ha obtenido una solución válida de entre las 10 ejecuciones realizadas. Realmente no le damos demasiada importancia, pues siendo un algoritmo estocástico, ya contamos con que hay veces que no nos va a dar un resultado satisfactorio. Sin embargo, para este experimento nos ha dado un 90% de efectividad. El hecho de que el algoritmo se ejecute de forma tan rápida lo convierte en totalmente rentable. Además, en caso de que se obtenga una solución inválida, podemos volver a ejecutar el problema y nos ofrecerá una salida totalmente diferente.

EXPERIMENTO 7

Observación	Podría ser útil conocer la proporción óptima de conductores para un cierto número de pasajeros. Si obtenemos soluciones válidas, deberíamos poder reducir el número de conductores y obtener una solución de calidad en menor tiempo.
Hipótesis	Si obtenemos una solución reduciendo el número de conductores al que nos sugiere una ejecución previa del algoritmo, el tiempo se verá reducido de forma significativa.
Método	Estudio de la proporción Usuarios/Conductores óptima obtenida en base a la reducción de conductores que realiza *Hill Climbing*. Estudio de la reducción del tiempo de ejecución si bajamos la proporción a ese punto óptimo para ver si se mantiene la calidad de la solución. ● Para 50, 100, 150, 200 y 250 Usuarios. ● Con la proporción M = N/2 (5 ejecuciones diferentes). ● Con los conductores sugeridos en las ejecuciones del punto anterior, volver a ejecutar con la misma semilla todas las ejecuciones.

Resultados:

DIFERENCIAS							
Usuarios	**Dist. inicial (km)**	**Dist. final (km)**	**Conductores iniciales**	**Conductores eliminados**	**Tiempo (s)**	**% Dist. Reducida**	**% Conduc. Reducidos**
50	48.78	34.34	-9.4	-8.4	-0.057	-1.04%	-8.40%
100	93.7	57.72	-18.6	-15.2	-0.131	-0.83%	-15.20%
150	232.5	115.42	-27.8	-22.4	-0.751	0.41%	-22.40%
200	230.96	174.4	-39.8	-33.2	-0.134	-2.71%	-33.20%
250	304.8	130.56	-47.4	-37.4	1.145	0.72%	-37.40%

Tabla 8: Resumen de las diferencias entre promedios de las ejecuciones con proporción N/2 conductores y las que prueban la proporción de conductores que sugiere la ejecución de N/2.

RESUMEN							
Usuarios	**Dist. inicial (km)**	**Dist. final (km)**	**Conductores iniciales**	**Conductores eliminados**	**Tiempo (s)**	**% Dist. Reducida**	**% Conduc. Reducidos**
50	504.88	295.8	25	9.4	0.252	41.25%	9.40%
50	553.66	330.14	15.6	1	0.195	40.21%	1.00%
100	1027.5	541.44	50	18.6	1.615	47.29%	18.60%
100	1121.2	599.16	31.4	3.4	1.484	46.46%	3.40%
150	1463.82	775.94	75	27.8	7.099	46.99%	27.80%
150	1696.32	891.36	47.2	5.4	6.348	47.40%	5.40%
200	1966.72	978.62	100	39.8	22.621	50.21%	39.80%
200	2197.68	1153.02	60.2	6.6	22.487	47.50%	6.60%
250	2466.24	1220.36	125	47.4	56.517	50.52%	47.40%
250	2771.04	1350.92	77.6	10	57.662	51.24%	10.00%

Tabla 9: Promedios de las ejecuciones según proporción de conductores iniciales (N/2 vs proporción sugerida con ejecuciones a N/2)

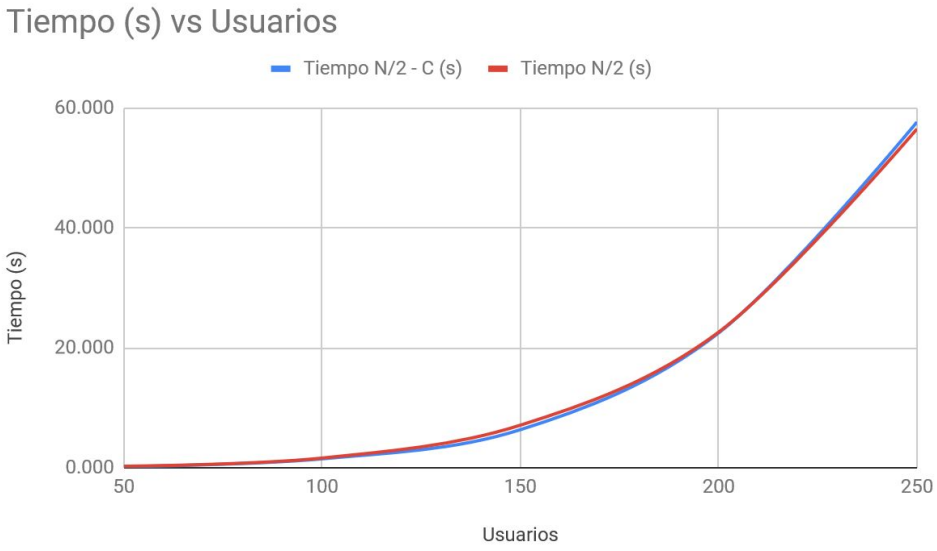

Figura 10: Comparativa entre el tiempo de ejecución con proporción de conductores N/2 vs N/2 menos el número de conductores eliminados en la ejecución de N/2, según número de usuarios.

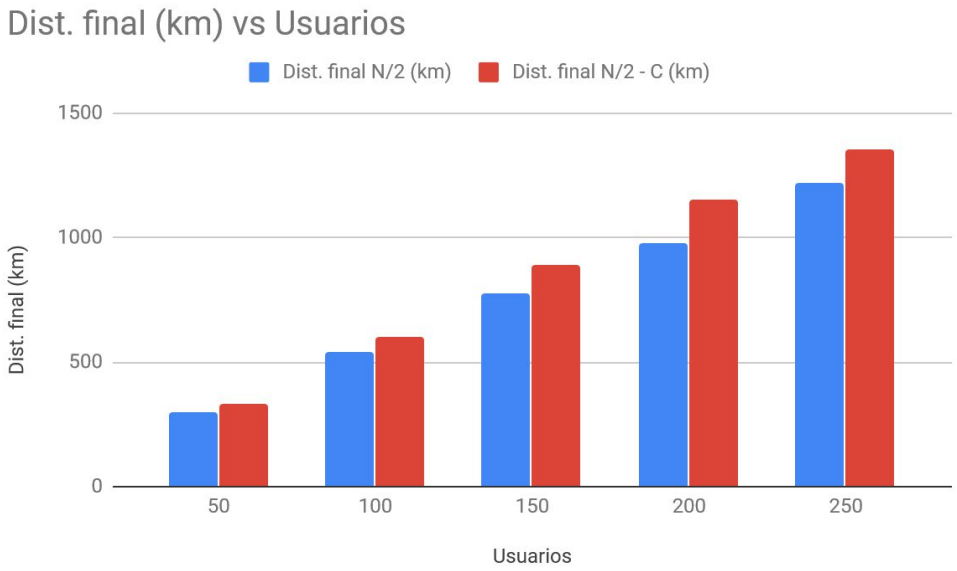

Figura 11: Comparativa entre la distancia final con proporción de conductores N/2 vs N/2 menos el número de conductores eliminados en la ejecución de N/2, según número de usuarios.

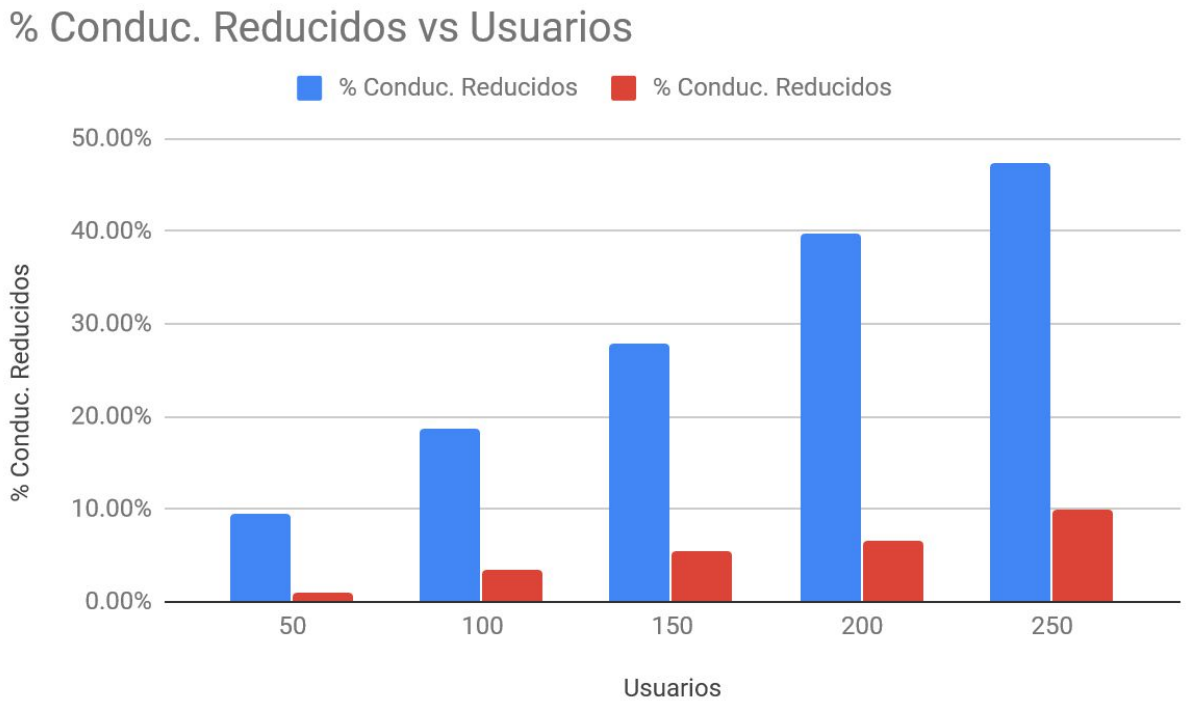

Figura 12: Comparativa entre el porcentaje de coches reducidos con proporción de conductores N/2 vs N/2 menos el número de conductores eliminados en la ejecución de N/2, según número de usuarios.

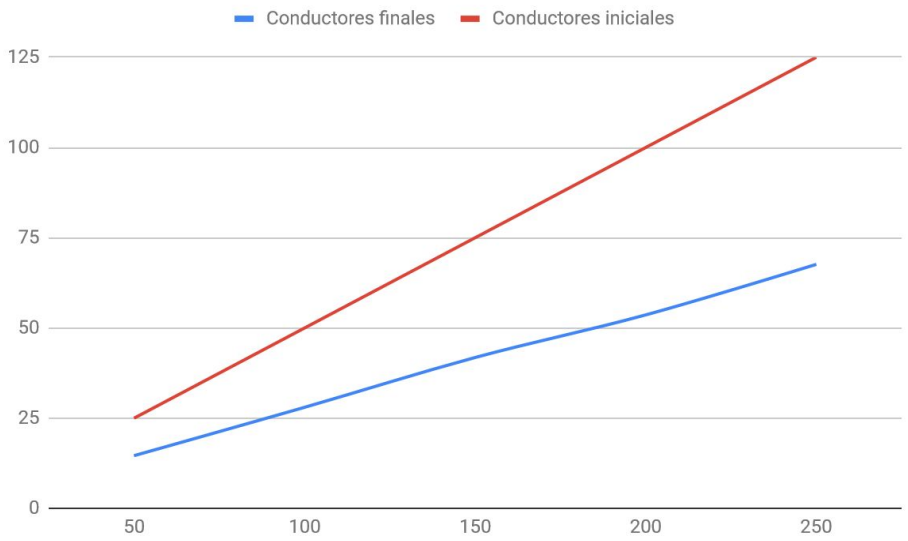

Figura 13: Comparativa entre coches iniciales con proporción N/2 y los conductores finales obtenidos con proporción N/2 menos el número de conductores eliminados en la ejecución de N/2, según número de usuarios.

NOTA: Absolutamente todas las ejecuciones de este experimento han dado soluciones válidas (50 ejecuciones).

Comentarios:

Lo primero que llama la atención de estos resultados es que podemos rechazar la hipótesis, ya que en absoluto obtenemos una mejora significativa del tiempo de ejecución (figura 10). Esto se debe a que el número de estados sucesores generados en cada paso del algoritmo no crece mucho con el número de conductores, sino con el número de usuarios.

Una conclusión muy útil, extraíble de este experimento, es que cuantos más usuarios tiene el problema, menor es el porcentaje de los mismos que deben ser conductores para obtener una solución válida (figura 13). Esto seguramente se debe a que el algoritmo encuentra más usuarios con caminos similares, por probabilidad, y por tanto los conductores hacen caminos más eficientes.

Algo que es también muy llamativo es que cuando volvemos a ejecutar el algoritmo con menos conductores, este es capaz de ir un poco más allá y seguir minimizando conductores (figura 12).

Por contra, un resultado previsible pero no por ello menos negativo es que por lo general el precio a pagar por la minimización de conductores es, como vemos en el experimento 3 (figura 7), recorrer más distancia (figura 11). Esto puede ser útil si el servicio de compartición de coches cuenta con pocos conductores inscritos, pero nuestro principal objetivo no es eliminar conductores si perdemos en distancia total.

EXPERIMENTO 8

Observación	Experimento con las mismas condiciones para todos los estudiantes
Método	Estudio del tiempo de ejecución y el nivel de optimización. Comprobación del resultado de la distancia total recorrida y el número de conductores. ● Utilizando la semilla '1234'. ● Para 200 Usuarios (100 Conductores). ● Para los dos algoritmos de búsqueda.

Resultados:

Simulated Annealing							
Ejecución	**Dist. inicial (km)**	**Dist. final (km) - EQ**	**Solución?**	**Conductores eliminados (EQ)**	**Tiempo (s)**	**% Dist. Reducida**	**% Conduc. Reducidos**
1	2052.6	1207.2	TRUE	40	2.817	41.19%	40.00%
2	2052.6	1184	TRUE	42	2.743	42.32%	42.00%
3	2052.6	1194.8	TRUE	40	2.851	41.79%	40.00%
4	2052.6	1164.3	TRUE	40	2.910	43.28%	40.00%
5	2052.6	1200.9	TRUE	40	2.746	41.49%	40.00%
Promedio	**2052.6**	**1190.24**	**1**	**40.4**	**2.813**	**42.01%**	**40.40%**

Tabla 11: Ejecuciones con semilla 1234 para 200 usuarios y 100 conductores con Simulated Annealing (100.000 - 1.000 - 10 - 0.0000001) y el heurístico b, operadores SMR y solución inicial EQ.

Hill Climbing							
Ejecución	**Dist. inicial (km)**	**Dist. final (km) - EQ**	**Solución?**	**Conductores eliminados (EQ)**	**Tiempo (s)**	**% Dist. Reducida**	**% Conduc. Reducidos**
1	2052.6	996.4	TRUE	37	21.680	51.46%	37.00%
Promedio	**2052.6**	**996.4**	**1**	**37**	**21.680**	**51.46%**	**37.00%**

Tabla 11: Ejecución con semilla 1234 para 200 usuarios y 100 conductores con Hill Climbing y el heurístico b, operadores SMR y solución inicial EQ.

CONCLUSIONES

La conclusión principal que podemos sacar tras la realización de esta práctica es que resolver un problema de esta complejidad no es un camino en línea recta. Este trabajo se ha caracterizado por la propuesta de múltiples enfoques para cada tramo a resolver y la discusión constante de puntos a favor y puntos en contra. Además, conforme íbamos obteniendo resultados de los experimentos, surgían nuevas ideas que podrían (o no) mejorar el resultado. Sin embargo, la práctica no acabaría nunca si no limitamos hasta qué momento nos podemos permitir hacer cambios.

Por ello, sin ninguna duda, de volver a realizar la implementación, tendríamos una visión más rica de las cosas que nos pueden ayudar a optimizar el resultado, que al final es el objetivo de la práctica.

Sobre los algoritmos implementados, hemos visto las limitaciones que tiene *Hill Climbing*, un algoritmo pesado a nivel de memoria y tiempo de ejecución que, sin embargo, nos ha dado los mejores resultados y más consistencia. Por otro lado, hemos descubierto las bondades de la búsqueda local con componente aleatoria de *Simulated Annealing*. Este algoritmo sorprende por la eficiencia temporal y de memoria que proporciona, obteniendo unos resultados muy respetables y que, aun siendo algo más de inconsistente, ejecutarlo otra vez es poco costoso y ofrece soluciones diferentes.

Por la forma en la que hemos implementado los operadores, el espacio de soluciones es visitado en su inmensa mayoría, si bien es cierto que si quisiéramos recorrer absolutamente todas las posibles opciones tendríamos que utilizar más operadores, como podría ser hacer un *swap* entre conductores o el *swap* interno entre pasajeros de un mismo vehículo. Sin embargo, el coste en memoria y en tiempo de ejecución para el *Hill Climbing* crecería muchísimo, así como generar sucesores que no contengan coches con configuración ilegal, lo cual tendría un alto coste de cómputo. Para utilizar este tipo de operadores, nuestra implementación del estado y las funciones heurísticas tendrían que tener un enfoque muy diferente para funcionar correctamente.

Podemos afirmar que esta práctica nos ha ayudado a entender mejor el por qué del uso de un algoritmo de búsqueda local. Este problema es demasiado complejo como para tratar de encontrar un máximo absoluto, la mejor configuración, y una de las formas de obtener una solución *bastante buena* es implementar este tipo de algoritmos que, además, aceptan ciertas parametrizaciones que permiten personalizar el tipo de resultado que van a obtener: qué premiar y qué penalizar.