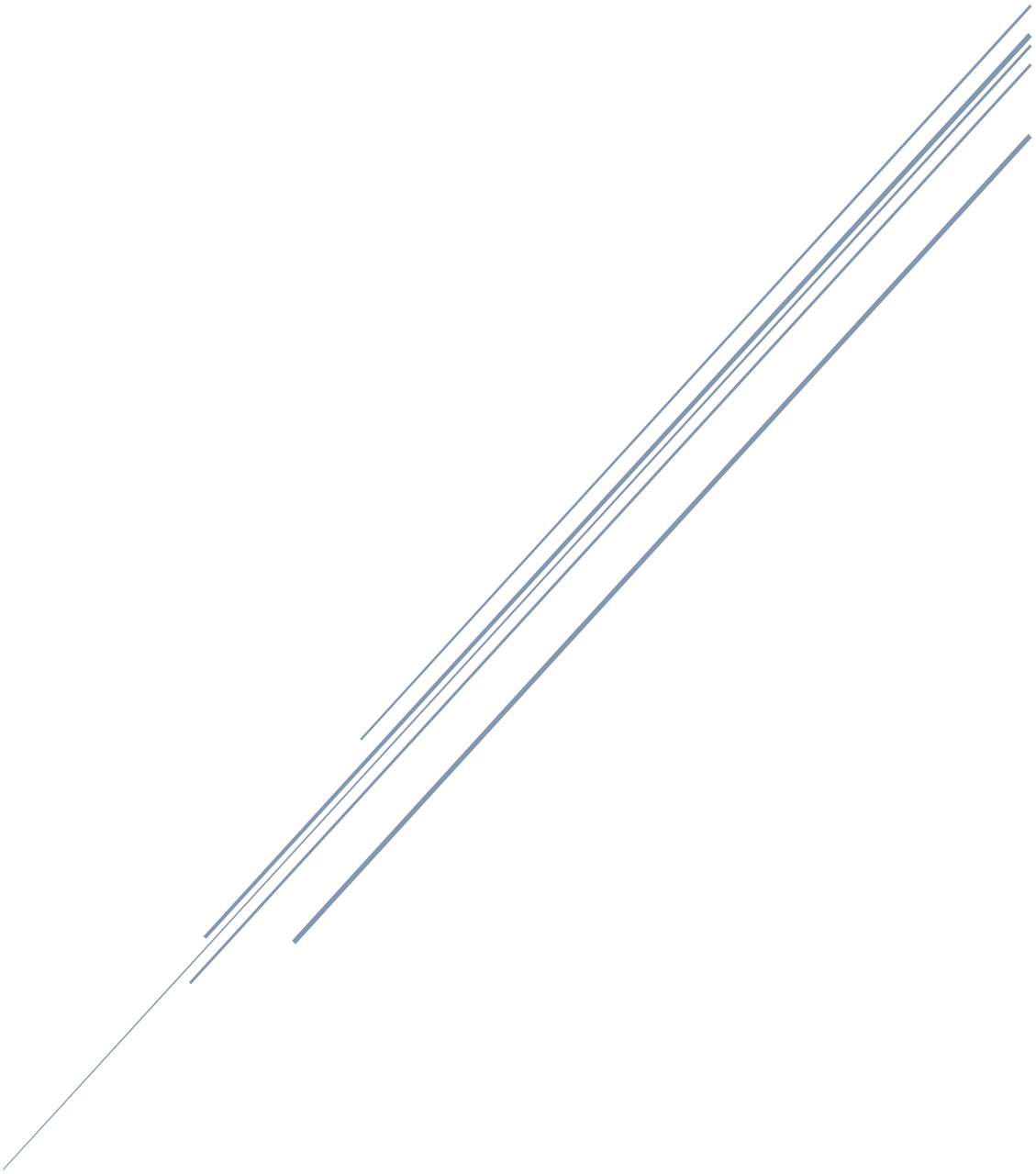
ANALISIS DE ALGORITMOS DE BÚSQUEDA BASADOS EN ENTORNOS Y TRAYECTORIAS

Ihar Myshkevich



Universidad de Huelva

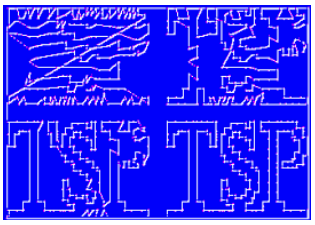
ETSI
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA
uhu.es

Índice

Introducción 2
Mejores resultados de cada dataset 2
Resultados Voraz (Greedy) 2
Comparación de costes 3
Algoritmo Aleatorio 4
Comparación de costes 5
Comparación de desviaciones típicas de coste 6
Búsqueda local, el mejor. 7
Comparación de costes 8
Comparación de desviaciones típicas de coste 9
Búsqueda local, el primer mejor vecino. 10
Comparación de costes 11
Comparación de desviaciones típicas de coste 12
Enfriamiento simulado 13
Comparación de costes 14
Comparación de desviaciones típicas de coste 15
Búsqueda tabú 16
Comparación de costes 17
Comparación de desviaciones típicas de coste 18
Comparación de algoritmos 19
Comparación de costes 20
Comparación evaluaciones 22
Comparación de desviación típica de costes 23
Conclusiones 24

Introducción

En este documento se verán y analizarán los resultados de la aplicación de los algoritmos *Voraz (Greedy)*, *Búsqueda Aleatoria*, *Búsqueda local, el mejor*, *Búsqueda local, el primer mejor vecino*, *Enfriamiento Simulado* y *Búsqueda Tabú* a los problemas del viajante (TSP).



Para ello, se han usado los datasets St70, Ch130, A280, Pa654, Vm1084, y Vm1748 de TSPLib.

Mejores resultados de cada dataset

El mejor resultado posible para cada dataset es:

	St70	**Ch130**	**A280**	**Pa654**	**Vm1084**	**Vm1748**
Mejor Coste	675	6110	2579	34643	239297	336556

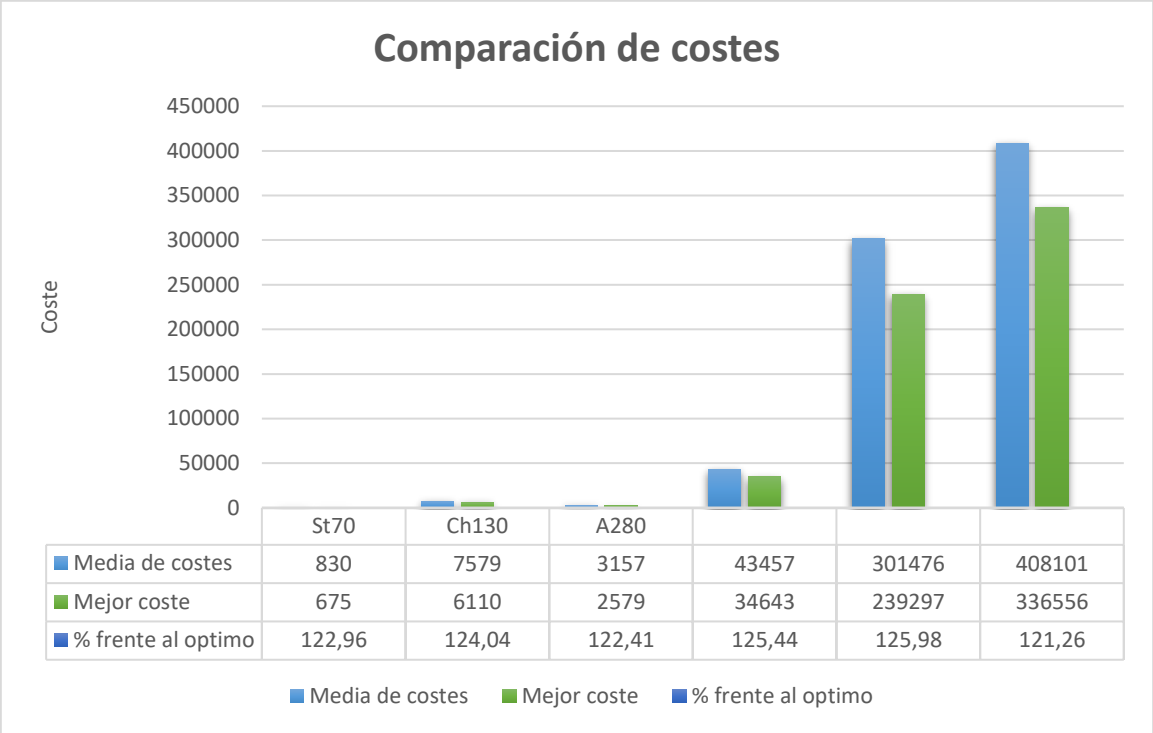
Resultados Voraz (Greedy)

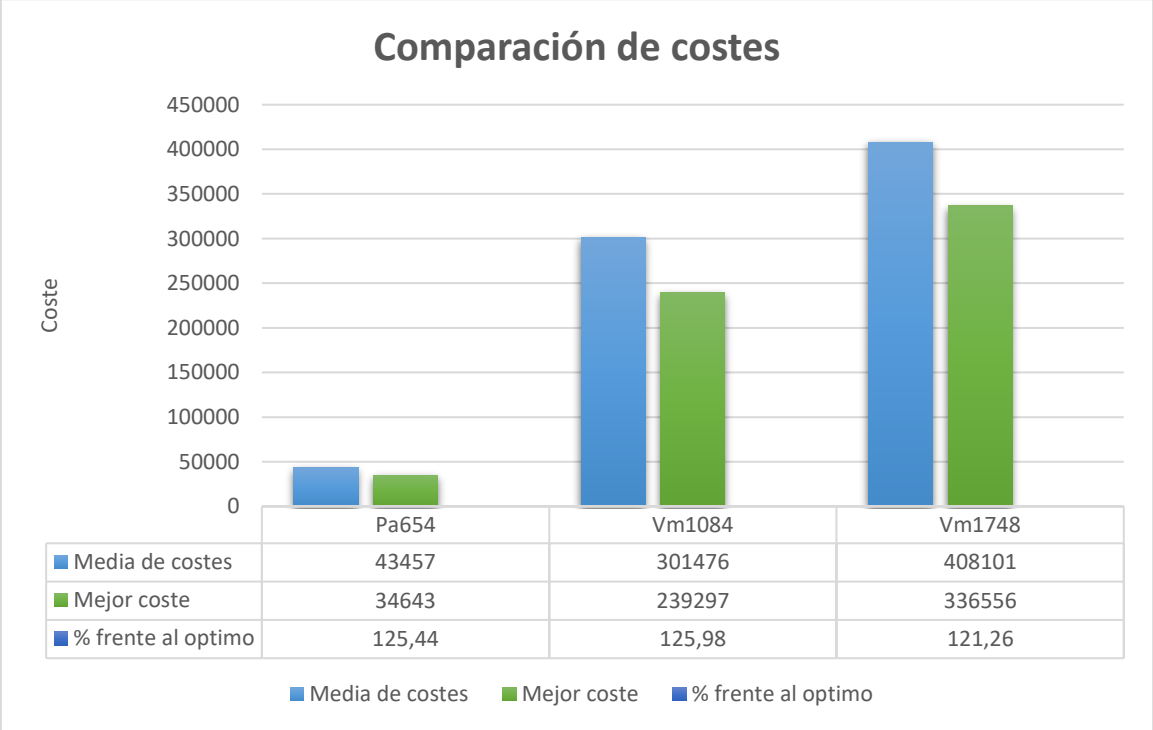
El algoritmo voraz (Greedy) es un algoritmo constructivo que empieza por la primera ciudad y, a partir de ella, construye una solución. Este algoritmo siempre genera una solución en un mínimo. Pero no asegura que este sea un mínimo global.

	St70		Ch130		A280		Pa654		Vm1084		Vm1748	
	Coste	#Ev	Coste	#Ev	Coste	#Ev	Coste	#Ev	Coste	#Ev	Coste	#Ev
Ejecución 1	830	1	7579	1	3157	1	43457	1	301476	1	408101	1
Media	830	1	7579	1	3157	1.0	43457	1	301476	1	408101	1
Des. Tip. (s)	0	0	0	0	0	0	0	0	0	0	0	0

Debido a que este algoritmo se ha ejecutado una vez, no se estudiara ni las columnas de evaluaciones ni la fila de desviaciones típicas al ser estas constantes.

Comparación de costes





Se observa que, con este simple y rápido algoritmo, se obtienen unos resultados muy cercanos al óptimo. Con una subida máxima de 25 % del coste óptimo.

Algoritmo Aleatorio

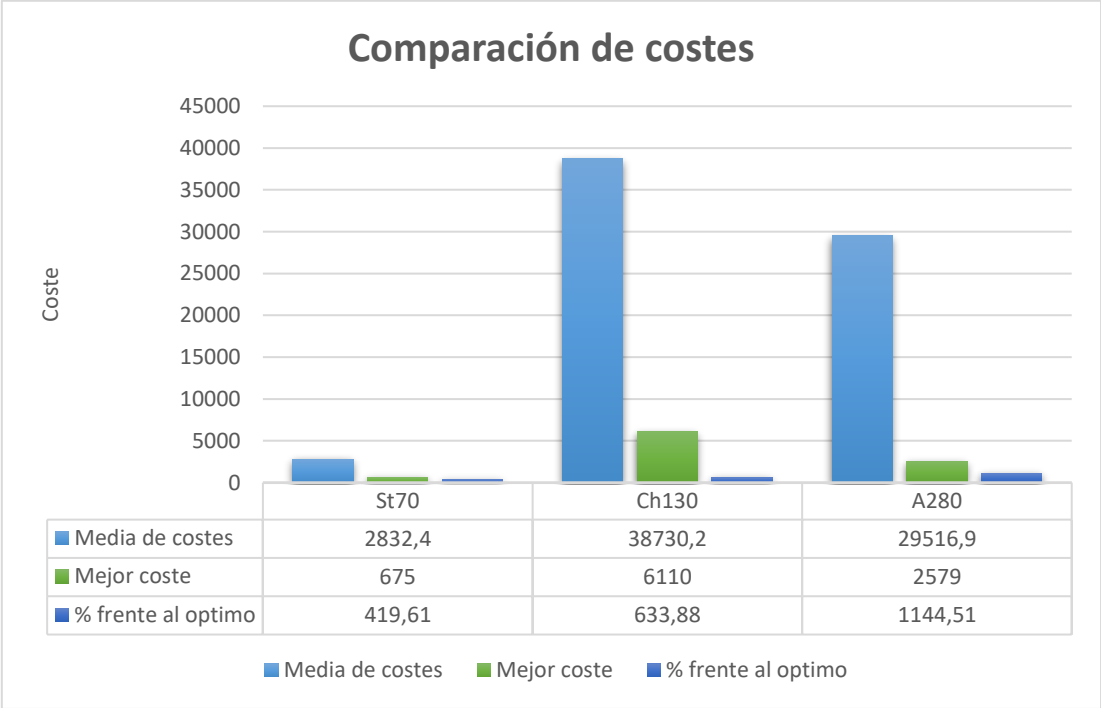
El algoritmo aleatorio genera una solución completamente aleatoria en cada iteración. Devolviendo al final la mejor encontrada.

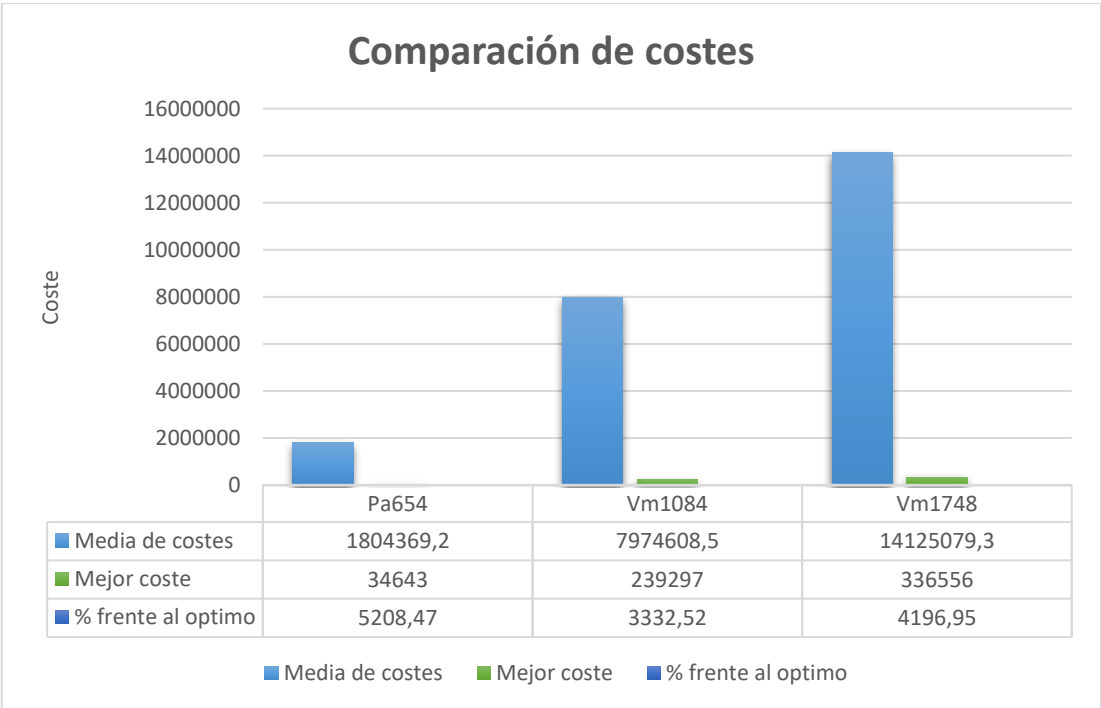
	St70		Ch130		A280	
	Coste	#Ev	Coste	#Ev	Coste	#Ev
Ejecución 1	2899	112000	38446.0	208000	29732.0	448000
Ejecución 2	2780	112000	38830.0	208000	29790.0	448000
Ejecución 3	2837	112000	39013.0	208000	29428.0	448000
Ejecución 4	2870	112000	38251.0	208000	29076.0	448000
Ejecución 5	2836	112000	38654.0	208000	29754.0	448000
Ejecución 6	2845	112000	39077.0	208000	29636.0	448000
Ejecución 7	2838	112000	39229.0	208000	30064.0	448000
Ejecución 8	2834	112000	38961.0	208000	29796.0	448000
Ejecución 9	2810	112000	38041.0	208000	29131.0	448000
Ejecución 10	2775	112000	38800.0	208000	28762.0	448000
Media	2832.4	112000	38730.2	208000	29516.9	448000
Des. Tip. (s)	37.44	0	381.27	0	406.82	0

	Pa654		Vm1084		Vm1748	
	Coste	#Ev	Coste	#Ev	Coste	#Ev
Ejecución 1	1789857.0	1046400	7993816.0	1734400	14122987.0	2796800
Ejecución 2	1814094.0	1046400	7944549.0	1734400	14116575.0	2796800
Ejecución 3	1797975.0	1046400	7994081.0	1734400	14138041.0	2796800
Ejecución 4	1819499.0	1046400	7963697.0	1734400	14143659.0	2796800
Ejecución 5	1798806.0	1046400	7977197.0	1734400	14169420.0	2796800
Ejecución 6	1808572.0	1046400	7950698.0	1734400	14176894.0	2796800
Ejecución 7	1807236.0	1046400	7996856.0	1734400	14120624.0	2796800
Ejecución 8	1811747.0	1046400	7959511.0	1734400	14069521.0	2796800
Ejecución 9	1783474.0	1046400	8005380.0	1734400	14112588.0	2796800
Ejecución 10	1812432.0	1046400	7960300.0	1734400	14080484.0	2796800
Media	1804369.2	1046400	7974608.5	1734400	14125079.3	2796800
Des. Tip. (s)	11488.41	0	21663.22	0	34143.68	0

En este algoritmo el número de evaluaciones es constante. Debido a esto no se analizarán las columnas referentes a estos datos.

Comparación de costes

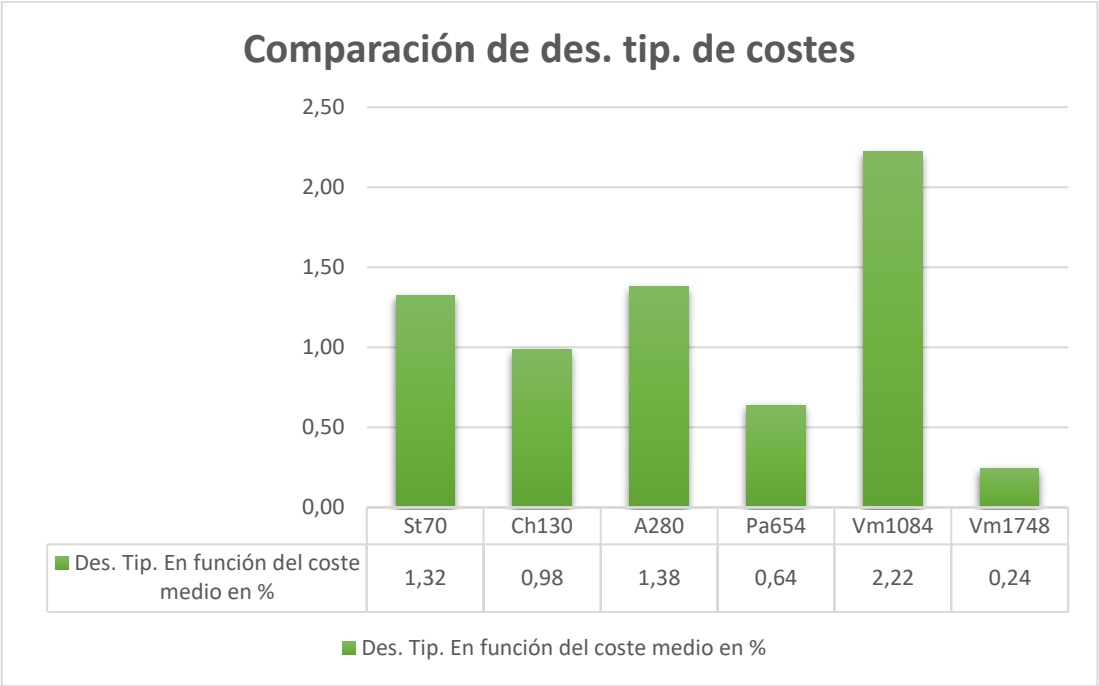




Se observa que este algoritmo, aun siendo rápido, genera soluciones muy malas. Esto, es debido a:

- El algoritmo selecciona soluciones al azar sin criterio ninguno y, la probabilidad de alcanzar la solución óptima es $\frac{1}{tamaño\ del\ dataset!}$.
- El incremento de posibles soluciones crece de manera factorial mientras el incremento de evaluaciones se amplía linealmente.

Comparación de desviaciones típicas de coste



Además, como era de esperar, se observa que la desviación típica no varía mucho en función al coste medio. Esto es debido, como vimos antes en la gráfica de costes, a que el algoritmo genera soluciones sin ningún criterio.

Búsqueda local, el mejor.

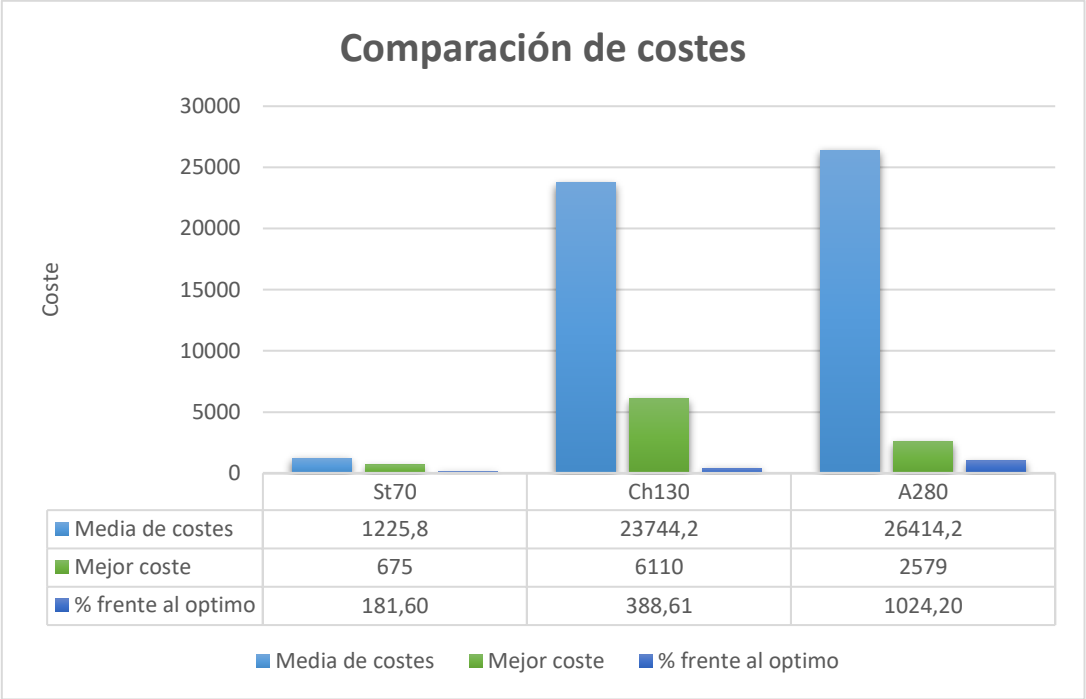
Este algoritmo analiza N vecinos en cada iteración, seleccionando el mejor. En el caso de que alcance un mínimo, terminara su ejecución. Este algoritmo pertenece a los algoritmos de búsqueda en anchura.

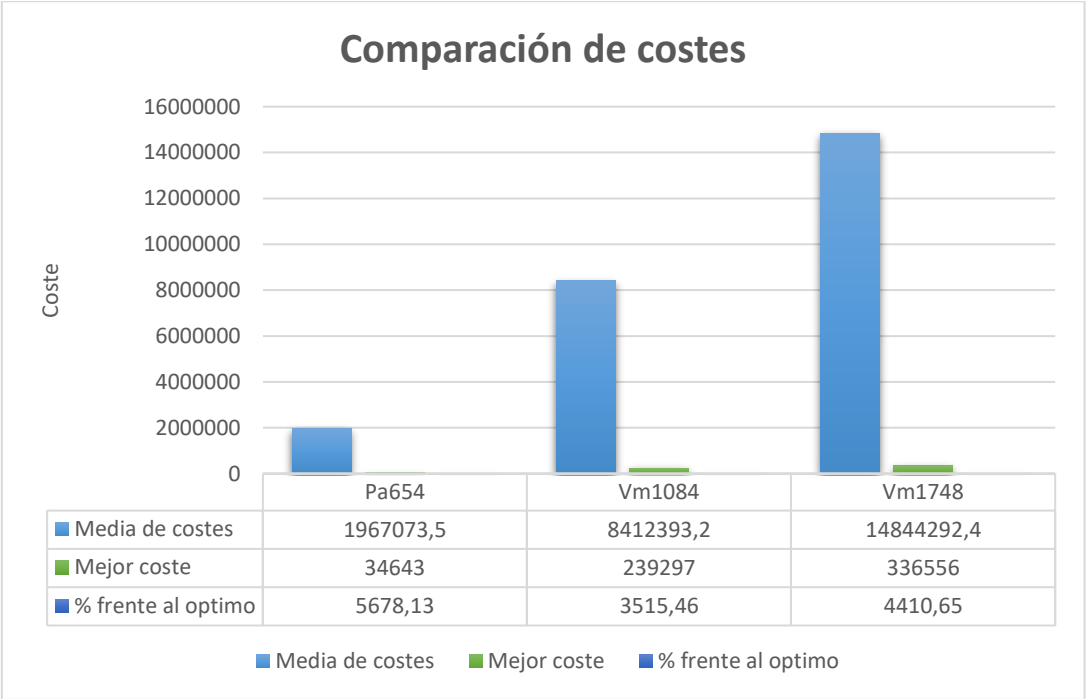
	St70		Ch130		A280	
	Coste	#Ev	Coste	#Ev	Coste	#Ev
Ejecución 1	1343.0	112000	23549.0	208000	27283.0	448000
Ejecución 2	1120.0	112000	24947.0	208000	26558.0	448000
Ejecución 3	1298.0	112000	23166.0	208000	25863.0	448000
Ejecución 4	1071.0	112000	24656.0	208000	25367.0	448000
Ejecución 5	1144.0	112000	22211.0	208000	27055.0	448000
Ejecución 6	1102.0	112000	23846.0	208000	26820.0	448000
Ejecución 7	1512.0	112000	23937.0	208000	27333.0	448000
Ejecución 8	1198.0	112000	24717.0	208000	26324.0	448000
Ejecución 9	1237.0	112000	22495.0	208000	25139.0	448000
Ejecución 10	1233.0	112000	23918.0	208000	26400.0	448000
Media	1225.8	112000.0	23744.2	208000.0	26414.2	448000.0
Des. Tip. (s)	132.90	0.0	918.07	0.0	762.29	0.0

	Pa654		Vm1084		Vm1748	
	Coste	#Ev	Coste	#Ev	Coste	#Ev
Ejecución 1	1982334.0	1046400	8322839.0	1734400	15051918.0	2796800
Ejecución 2	1829036.0	1046400	8278559.0	1734400	14588539.0	2796800
Ejecución 3	1997397.0	1046400	8330949.0	1734400	14771671.0	2796800
Ejecución 4	2025072.0	1046400	8325040.0	1734400	14940328.0	2796800
Ejecución 5	2010331.0	1046400	8551875.0	1734400	14907335.0	2796800
Ejecución 6	1963249.0	1046400	8515591.0	1734400	14980121.0	2796800
Ejecución 7	1985424.0	1046400	8559312.0	1734400	14950542.0	2796800
Ejecución 8	2015600.0	1046400	8438387.0	1734400	14615445.0	2796800
Ejecución 9	1956213.0	1046400	8527299.0	1734400	14912724.0	2796800
Ejecución 10	1906079.0	1046400	8274081.0	1734400	14724301.0	2796800
Media	1967073.5	1046400.0	8412393.2	1734400.0	14844292.4	2796800.0
Des. Tip. (s)	59615.67	0.0	117801.46	0.0	159227.16	0.0

Se observa que, en todas las ejecuciones de este algoritmo, al igual que en el anterior, no ha alcanzado el mínimo y siempre ha salido por el número de iteraciones. Debido a esto, no se estudiarán las tablas referentes a las evaluaciones.

Comparación de costes

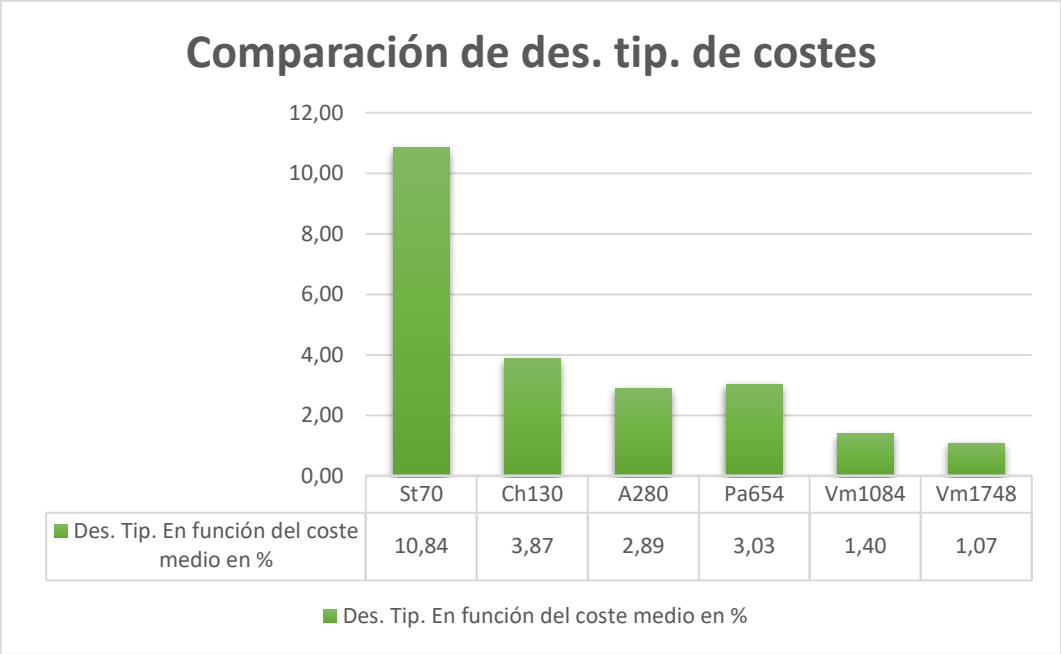




Se observa que este algoritmo, con la configuración preestablecida, genera peores resultados en función del tamaño del dataset. Esto es debido, como se ha dicho antes, a que el algoritmo no alcanza el mínimo y termina por el número de iteraciones. Esto es debido a que este algoritmo analiza todos los vecinos en cada iteración y este crece de manera factorial mientras el número de evaluaciones crece linealmente.

En el caso de eliminar esa característica se obtendrían unos resultados muy parecidos al algoritmo Greedy con el inconveniente de aumentar significativamente coste computacional.

Comparación de desviaciones típicas de coste



Por otro lado, se observa que la desviación típica en función del coste medio se reduce según aumenta el dataset. Esto es debido al tamaño del vecindario de cada solución. Permitiendo al algoritmo profundizar más en dataset pequeños al tener un vecindario más reducido y menos en datasets grandes con un vecindario mayor.

Búsqueda local, el primer mejor vecino.

Este algoritmo tiene el mismo funcionamiento que el algoritmo anterior, pero con la diferencia de que una vez que se encuentre un mejor vecino, se actualiza la solución actual y se reinician los vecinos a analizar. Este algoritmo pertenece al grupo de algoritmos de búsqueda en profundidad.

	St70		Ch130		A280	
	Coste	#Ev	Coste	#Ev	Coste	#Ev
Ejecución 1	1198.0	66622	15764.0	208000	10204.0	448000
Ejecución 2	1067.0	112000	15018.0	208000	7901.0	448000
Ejecución 3	1034.0	89396	12473.0	208000	8912.0	448000
Ejecución 4	1097.0	100103	15251.0	208000	10038.0	448000
Ejecución 5	998.0	112000	14735.0	208000	9997.0	448000
Ejecución 6	1039.0	80213	12624.0	208000	8449.0	448000
Ejecución 7	1084.0	64762	13635.0	208000	13353.0	448000
Ejecución 8	1083.0	112000	13045.0	208000	7517.0	448000
Ejecución 9	1285.0	112000	16471.0	208000	10536.0	448000
Ejecución 10	1018.0	104536	13091.0	208000	9643.0	448000
Media	1090.3	95363.2	14210.7	208000.0	9655.0	448000.0
Des. Tip. (s)	88.02	18977.66	1414.67	0.0	1655.65	0.0

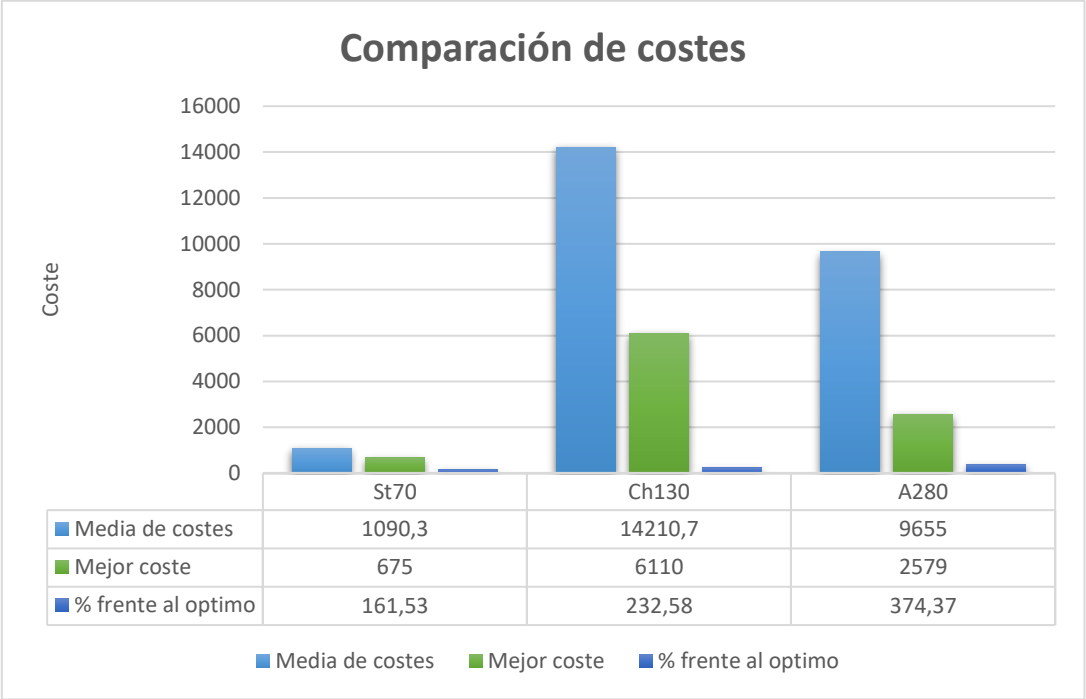
	Pa654		Vm1084		Vm1748	
	Coste	#Ev	Coste	#Ev	Coste	#Ev
Ejecución 1	525826.0	1046400	3234770.0	1734400	8200932.0	2796800
Ejecución 2	664775.0	1046400	3355581.0	1734400	7774568.0	2796800
Ejecución 3	432282.0	1046400	4009915.0	1734400	7481105.0	2796800
Ejecución 4	567343.0	1046400	3705448.0	1734400	7712646.0	2796800
Ejecución 5	487967.0	1046400	3014575.0	1734400	8416668.0	2796800
Ejecución 6	678666.0	1046400	3856392.0	1734400	7971610.0	2796800
Ejecución 7	496012.0	1046400	3390939.0	1734400	7742555.0	2796800
Ejecución 8	465582.0	1046400	4928702.0	1734400	7615341.0	2796800
Ejecución 9	544637.0	1046400	3229159.0	1734400	7850176.0	2796800
Ejecución 10	675403.0	1046400	3215409.0	1734400	7812913.0	2796800
Media	553849.3	1046400.0	3594089.0	1734400.0	7857851.4	2796800.0
Des. Tip. (s)	90656.02	0.0	565192.38	0.0	276297.50	0.0

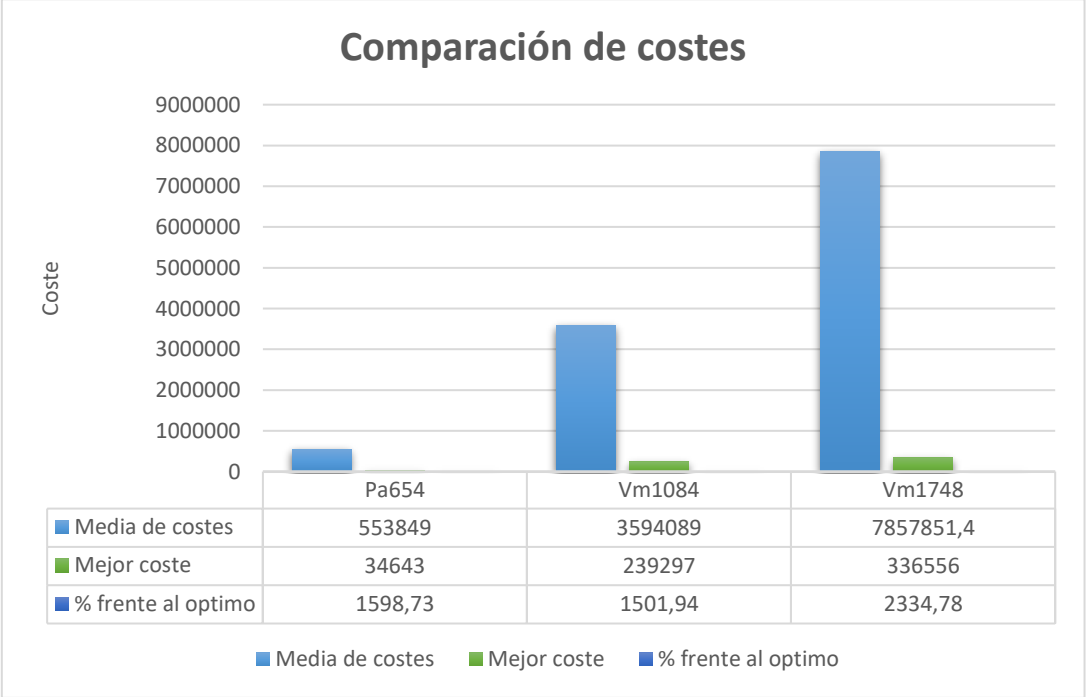
Al igual que con el algoritmo anterior, se observa que el número de evaluaciones es un factor limitante en este algoritmo.

La única diferencia que se puede apreciar, en este apartado, es que este algoritmo con el dataset St70 es capaz de alcanzar mínimos. Esto es debido, a como se indicó en la descripción del algoritmo, el echo de que este algoritmo pertenece al grupo de algoritmos de búsqueda en profundidad. Pero, solo teniendo datos de un dataset, no es un apartado relevante de estudio.

Por lo que no se analizaran estas columnas. Por otro lado, si se quisiese mejorar este algoritmo, este debería de ser la primera característica por estudiar.

Comparación de costes

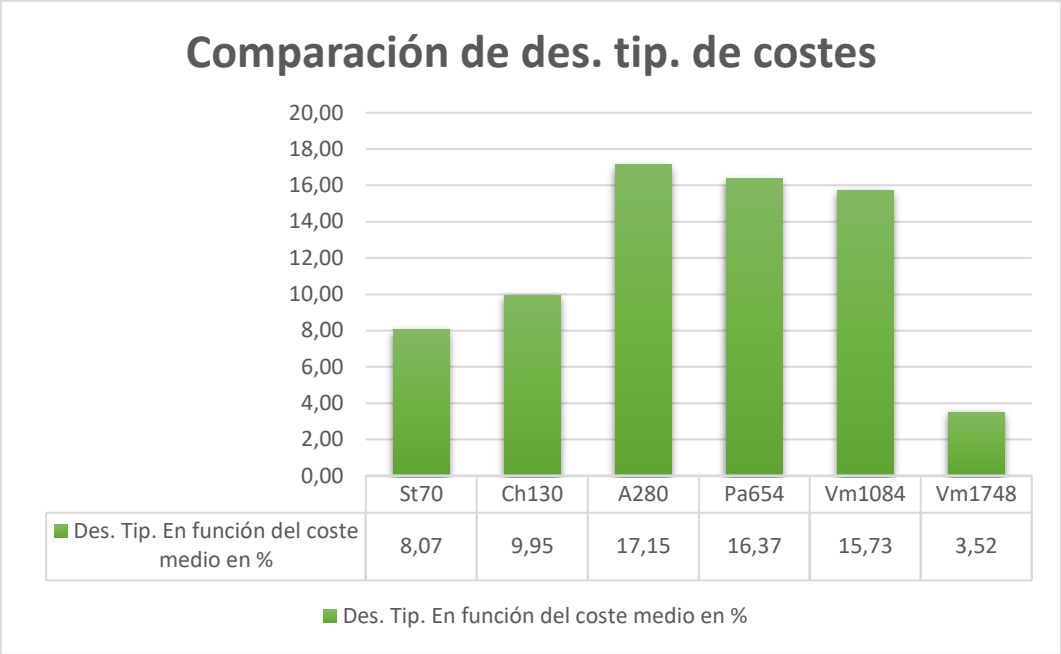




Se observa que este algoritmo, al igual que en el anterior, se obtienen peores resultados según aumenta el dataset. Pero, en comparación al algoritmo anterior, este obtiene mejores resultados gracias a que profundiza más en la búsqueda de la mejor solución.

En el caso de eliminar el límite de evaluaciones, se obtendrían unos resultados muy parecidos al algoritmo "Greedy" con el inconveniente de aumentar significativamente coste computacional.

Comparación de desviaciones típicas de coste



En este caso, al ser un algoritmo de búsqueda en profundidad, se observa una clara diferencia de desviación típica frente al algoritmo anterior, siendo esta más irregular.

Enfriamiento simulado

Este algoritmo realiza una búsqueda en profundidad permitiendo, en función de la temperatura, aceptar peores soluciones con el objetivo de buscar un mejor mínimo.

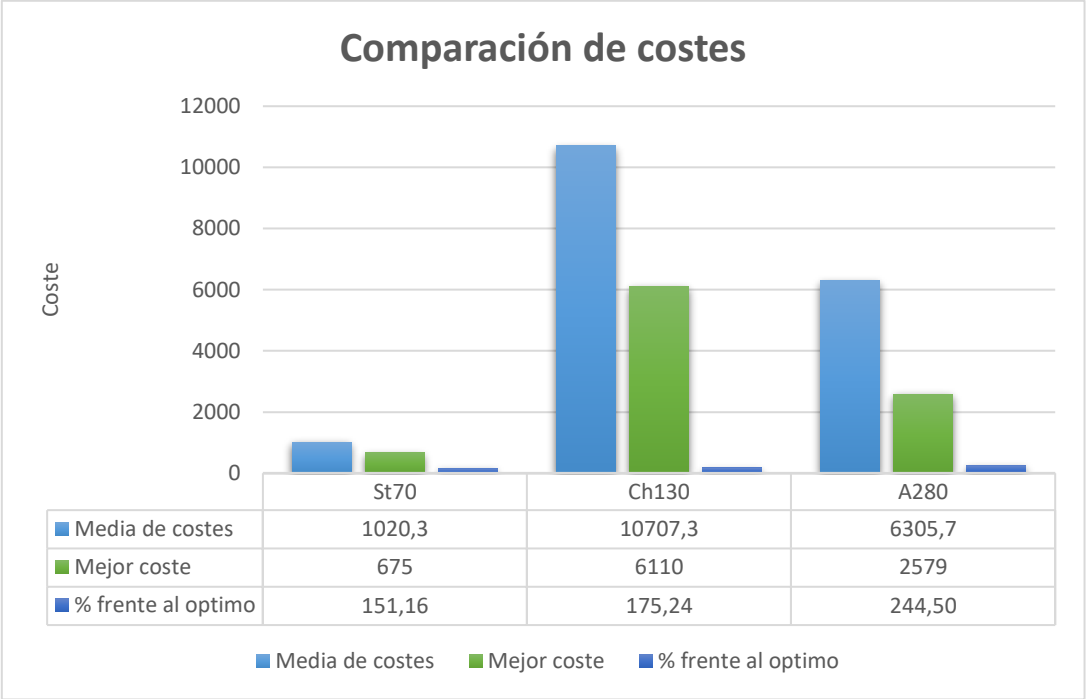
	St70		Ch130		A280	
	Coste	#Ev	Coste	#Ev	Coste	#Ev
Ejecución 1	971.0	112001	10006.0	208001	6752.0	448001
Ejecución 2	931.0	112001	11416.0	208001	6306.0	448001
Ejecución 3	1129.0	112001	10553.0	208001	6386.0	448001
Ejecución 4	1012.0	112001	10173.0	208001	6056.0	448001
Ejecución 5	1094.0	112001	10396.0	208001	6451.0	448001
Ejecución 6	1005.0	112001	11612.0	208001	6179.0	448001
Ejecución 7	1013.0	112001	11576.0	208001	6304.0	448001
Ejecución 8	978.0	112001	10932.0	208001	6757.0	448001
Ejecución 9	1097.0	112001	9577.0	208001	5849.0	448001
Ejecución 10	973.0	112001	10832.0	208001	6017.0	448001
Media	1020.3	112001.0	10707.3	208001.0	6305.7	448001.0
Des. Tip. (s)	64.90	0.0	692.58	0.0	298.28	0.0

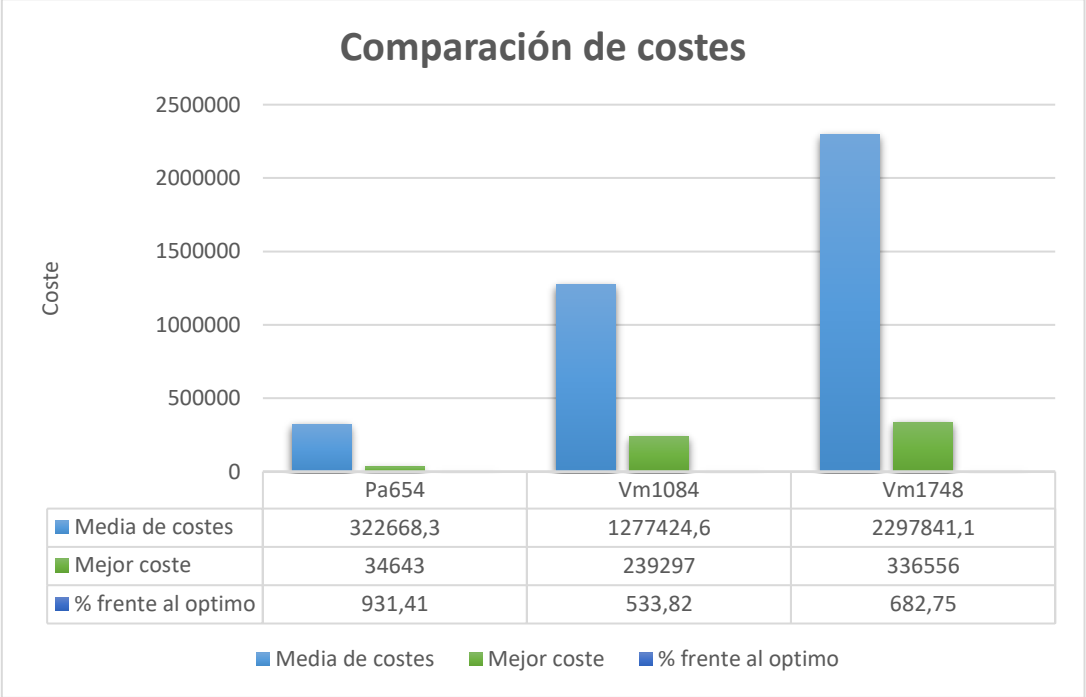
	Pa654		Vm1084		Vm1748	
	Coste	#Ev	Coste	#Ev	Coste	#Ev
Ejecución 1	360825.0	1046401	1315865.0	1734401	2384911.0	2796801
Ejecución 2	343347.0	1046401	1277981.0	1734401	2417620.0	2796801
Ejecución 3	323467.0	1046401	1264055.0	1734401	2370424.0	2796801
Ejecución 4	347536.0	1046401	1258736.0	1734401	2206825.0	2796801
Ejecución 5	330375.0	1046401	1295593.0	1734401	2174382.0	2796801
Ejecución 6	328241.0	1046401	1274642.0	1734401	2366712.0	2796801
Ejecución 7	282705.0	1046401	1301120.0	1734401	2272068.0	2796801
Ejecución 8	313166.0	1046401	1330433.0	1734401	2311700.0	2796801
Ejecución 9	303061.0	1046401	1177841.0	1734401	2218595.0	2796801
Ejecución 10	293960.0	1046401	1277980.0	1734401	2255174.0	2796801
Media	322668.3	1046401.0	1277424.6	1734401.0	2297841.1	2796801.0
Des. Tip. (s)	24695.93	0.0	41676.50	0.0	84605.26	0.0

Al igual que con los algoritmos anteriores, se observa que el número de evaluaciones es un factor limitante en este algoritmo. Por lo que no se analizaran estas columnas.

Por otro lado, si se quisiese mejorar este algoritmo, este debería de ser la primera característica por estudiar.

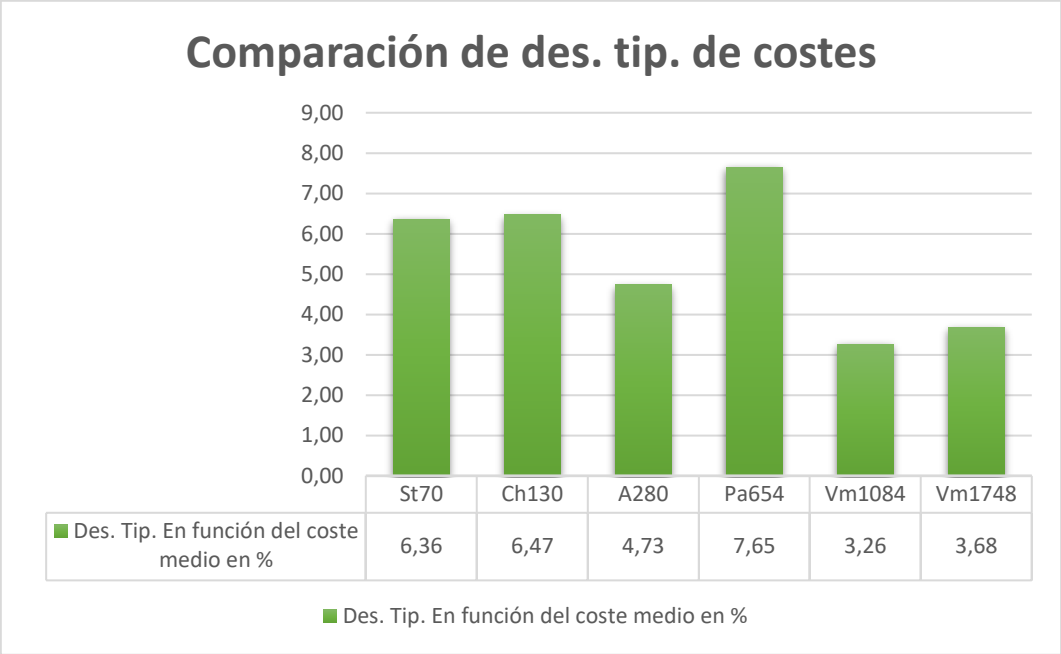
Comparación de costes





Se observa que este algoritmo, con la configuración preestablecida, nunca alcanza un mínimo. Debido a esto, los resultados son bastante malos. Y, empeoran en función del tamaño del dataset. Esto es debido a que este algoritmo analiza todos los vecinos en cada iteración y este crece de manera factorial mientras el número de evaluaciones crece linealmente.

Comparación de desviaciones típicas de coste



Se observa una cierta constancia en la desviación típica en el coste. Esto demuestra la constancia de este algoritmo a la hora de buscar la mejor solución.

Búsqueda tabú

Este algoritmo busca una mejor solución en descartando movimientos repetidos a corto plazo a menos que mejore la solución. Y, cada cierto número de iteraciones reinicia la búsqueda a partir de la mejor solución encontrada intensificando el coste de esta, una solución aleatoria o en función de la memoria a largo plazo para explorar zonas no visitadas.

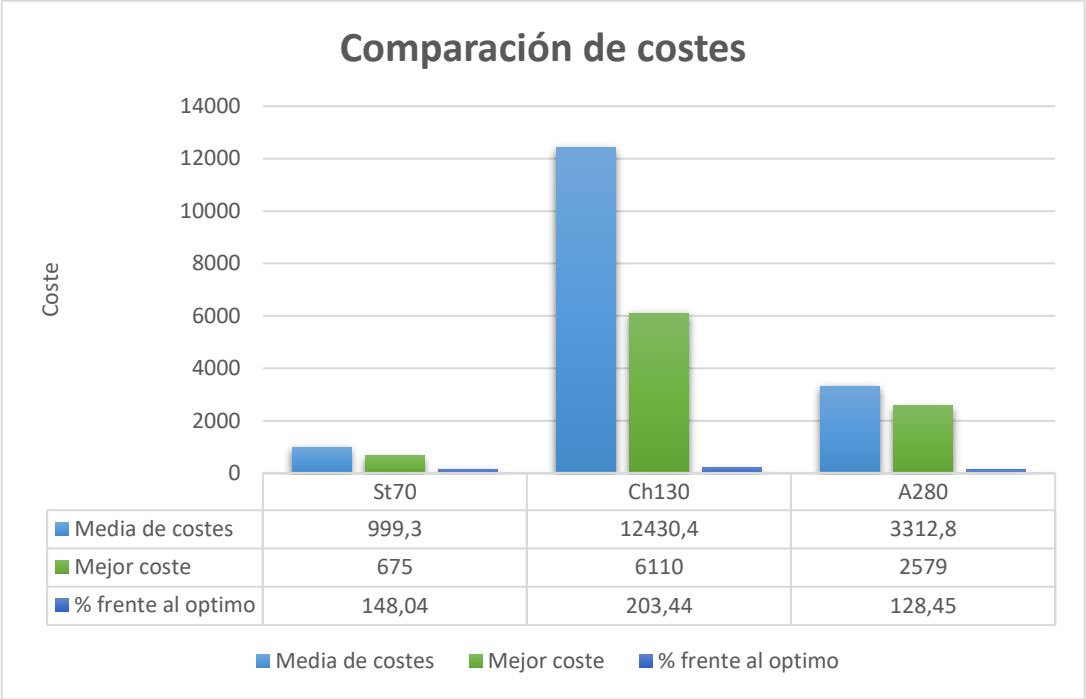
	St70		Ch130		A280	
	Coste	#Ev	Coste	#Ev	Coste	#Ev
Ejecución 1	936.0	112001	12708.0	208001	4123.0	448001
Ejecución 2	1073.0	112001	12965.0	208001	3462.0	448001
Ejecución 3	991.0	112001	12399.0	208001	3250.0	448001
Ejecución 4	1011.0	112001	12593.0	208001	3218.0	448001
Ejecución 5	1075.0	112001	12612.0	208001	2964.0	448001
Ejecución 6	1010.0	112001	12412.0	208001	3400.0	448001
Ejecución 7	1017.0	112001	12753.0	208001	3224.0	448001
Ejecución 8	1005.0	112001	12313.0	208001	3299.0	448001
Ejecución 9	964.0	112001	11568.0	208001	3183.0	448001
Ejecución 10	911.0	112001	11981.0	208001	3005.0	448001
Media	999.3	112001.0	12430.4	208001.0	3312.8	448001.0
Des. Tip. (s)	52.57	0.0	406.49	0.0	323.27	0.0

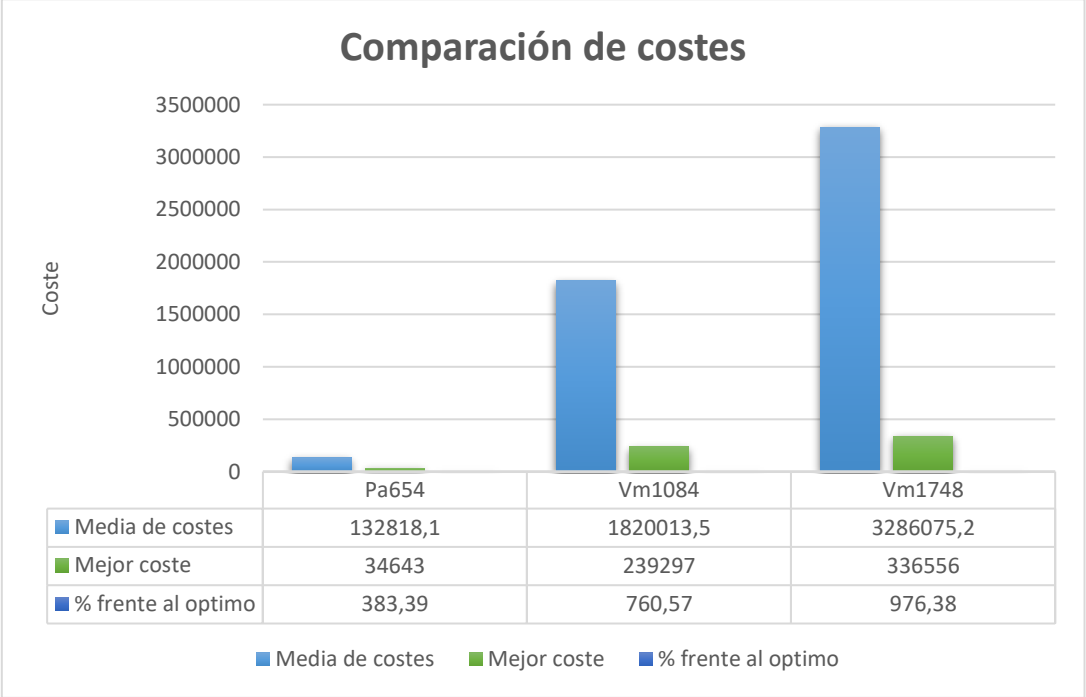
	Pa654		Vm1084		Vm1748	
	Coste	#Ev	Coste	#Ev	Coste	#Ev
Ejecución 1	108566.0	1046401	1795905.0	1734401	3235236.0	2796801
Ejecución 2	112408.0	1046401	1918289.0	1734401	3413694.0	2796801
Ejecución 3	136613.0	1046401	1824882.0	1734401	3272731.0	2796801
Ejecución 4	107542.0	1046401	1819819.0	1734401	3270034.0	2796801
Ejecución 5	110473.0	1046401	1957831.0	1734401	3133754.0	2796801
Ejecución 6	114844.0	1046401	1703828.0	1734401	3278107.0	2796801
Ejecución 7	108905.0	1046401	1759032.0	1734401	3300488.0	2796801
Ejecución 8	119205.0	1046401	1815570.0	1734401	3198248.0	2796801
Ejecución 9	300989.0	1046401	1739881.0	1734401	3364568.0	2796801
Ejecución 10	108636.0	1046401	1865098.0	1734401	3393892.0	2796801
Media	132818.1	1046401.0	1820013.5	1734401.0	3286075.2	2796801.0
Des. Tip. (s)	59725.08	0.0	78136.69	0.0	87100.66	0.0

Al igual que con los algoritmos anteriores, se observa que el número de evaluaciones es un factor limitante en este algoritmo. Por lo que no se analizaran estas columnas.

Por otro lado, si se quisiese mejorar este algoritmo, este debería de ser la primera característica por estudiar.

Comparación de costes





Se observa que este algoritmo, con la configuración preestablecida, nunca alcanza un mínimo. Debido a esto, los resultados son bastante malos según aumenta el dataset. Esto es debido a que este algoritmo analiza todos los vecinos en cada iteración y este crece de manera factorial mientras el número de evaluaciones crece linealmente.

En el caso de eliminar esa característica se obtendrían unos resultados muy parecidos al algoritmo Greedy con el inconveniente de aumentar significativamente coste computacional.

Por otro lado, se observa una subida brusca de coste en el dataset Ch130 frente al dataset St80. Esto es debido al propio dataset. Ya que este tiene las ciudades más separadas.

Comparación de desviaciones típicas de coste

En este caso no se puede apreciar ninguna tendencia en la desviación típica. Debido a esto se puede asegurar que el algoritmo es bastante constante a la hora de buscar la mejor solución.

Comparación de algoritmos

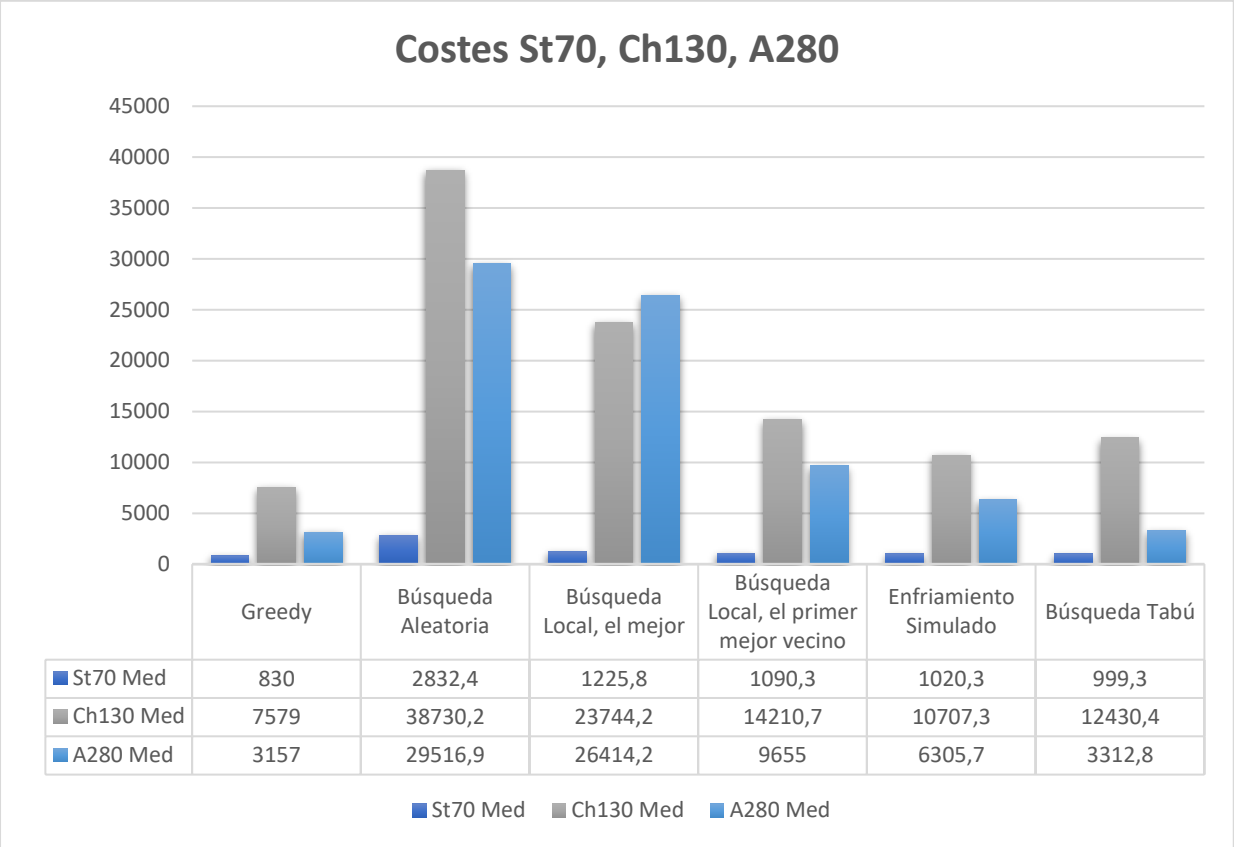
A continuación, se verá una tabla que compara todos los algoritmos desarrollados.

	St70				Ch130				A280			
	Med	Mej	S	Ev	Med	Mej	S	Ev	Med	Mej	S	Ev
Greedy	830	830	0	1	7579	7579	0	1	3157	3157	0	1
Búsqueda Aleatoria	2832.4	2775	37.44	112000	38730.2	38041	381.27	208000	29516.9	28762	406.82	448000
Búsqueda Local: el mejor	1225.8	1071	132.90	112000	23744.2	22211	918.07	208000	26414.2	25139	762.29	448000
Búsqueda Local: el primer mejor vecino	1090.3	998	88.02	95363.2	14210.7	12473	1414.67	208000	9655	7517	1655.65	448000
Enfriamiento Simulado	1020.3	931	64.90	112001	10707.3	9577	692.58	208001	6305.7	5849	298.28	448001
Búsqueda Tabú	999.3	911	52.57	112001	12430.4	11568	406.49	208001	3312.8	2964	323.27	448001

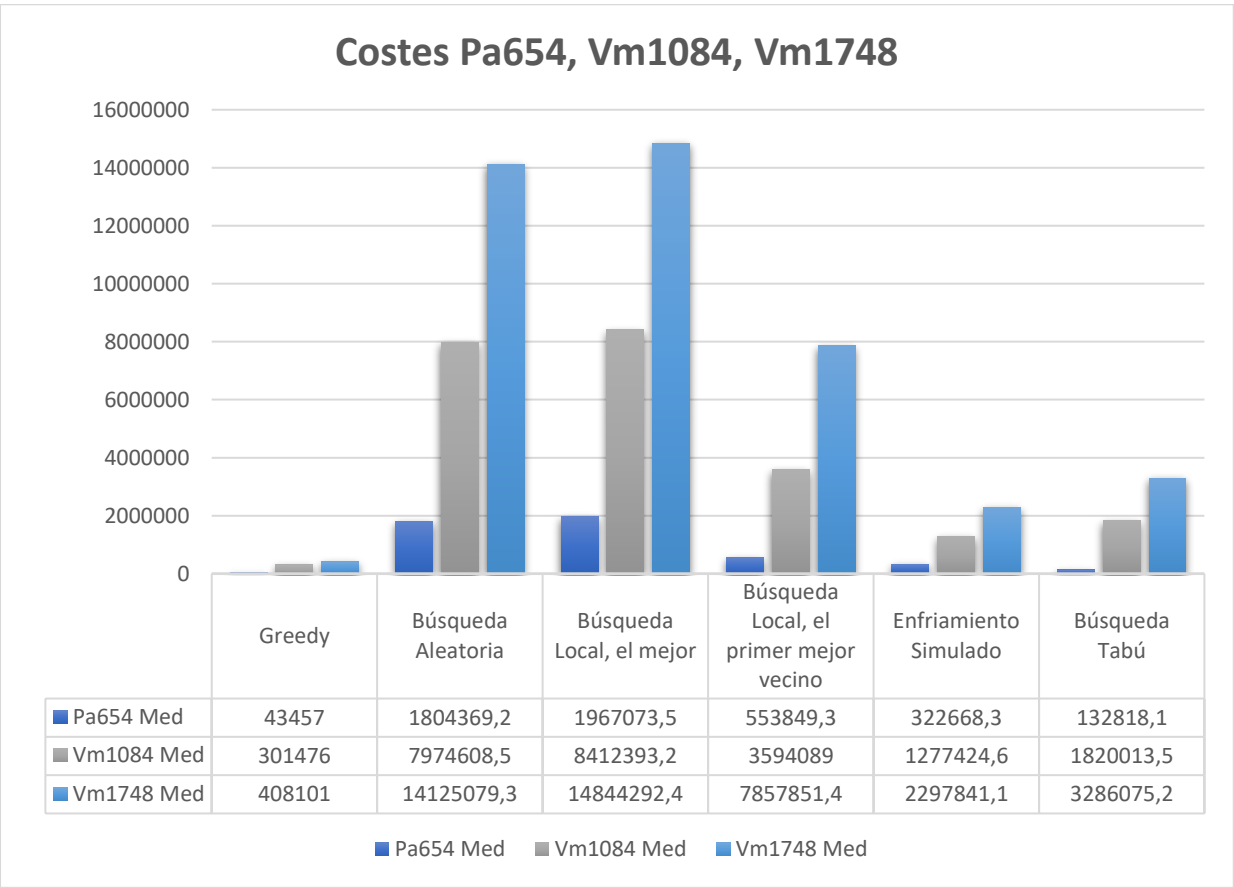
	Pa654				Vm1084				Vm1748			
	Med	Mej	S	Ev	Med	Mej	S	Ev	Med	Mej	S	Ev
Greedy	43457	43457	0	1	301476	301476	0	1	408101	408101	0	1
Búsqueda Aleatoria	1804369.2	1783474	11488.41	104640	7974608.5	7944549	21663.22	173440	14125079.3	14069521	34143.68	279680
"Búsqueda Local: el mejor"	1967073.5	1829036	59615.67	104640	8412393.2	8274081	117801.46	173440	14844292.4	14588539	159227.16	279680
"Búsqueda Local: el primer mejor vecino"	553849.3	432282	90656.02	104640	3594089	3014575	565192.38	173440	7857851.4	7481105	276297.50	279680
Enfriamiento Simulado	322668.3	282705.0	24695.93	104641.0	1277424.6	1177841	41676.50	173441	2297841.1	2174382	84605.26	279681
Búsqueda Tabú	132818.1	107542	59725.08	104641	1820013.5	1703828	78136.69	173441	3286075.2	3133754	87100.66	279681

Para mejorar la comparación de datos, se ha dividido su representación en dos gráficos. Por un lado, se analizarán dataset inferiores a 500 elementos y por otro lado superiores a 500 elementos.

Comparación de costes



	Greedy	Búsqueda Aleatoria	Búsqueda Local, el mejor	Búsqueda Local, el primer mejor vecino	Enfriamiento Simulado	Búsqueda Tabú
St70 Med	830	2832,4	1225,8	1090,3	1020,3	999,3
Ch130 Med	7579	38730,2	23744,2	14210,7	10707,3	12430,4
A280 Med	3157	29516,9	26414,2	9655	6305,7	3312,8



	Greedy	Búsqueda Aleatoria	Búsqueda Local, el mejor	Búsqueda Local, el primer mejor vecino	Enfriamiento Simulado	Búsqueda Tabú
Pa654 Med	43457	1804369,2	1967073,5	553849,3	322668,3	132818,1
Vm1084 Med	301476	7974608,5	8412393,2	3594089	1277424,6	1820013,5
Vm1748 Med	408101	14125079,3	14844292,4	7857851,4	2297841,1	3286075,2

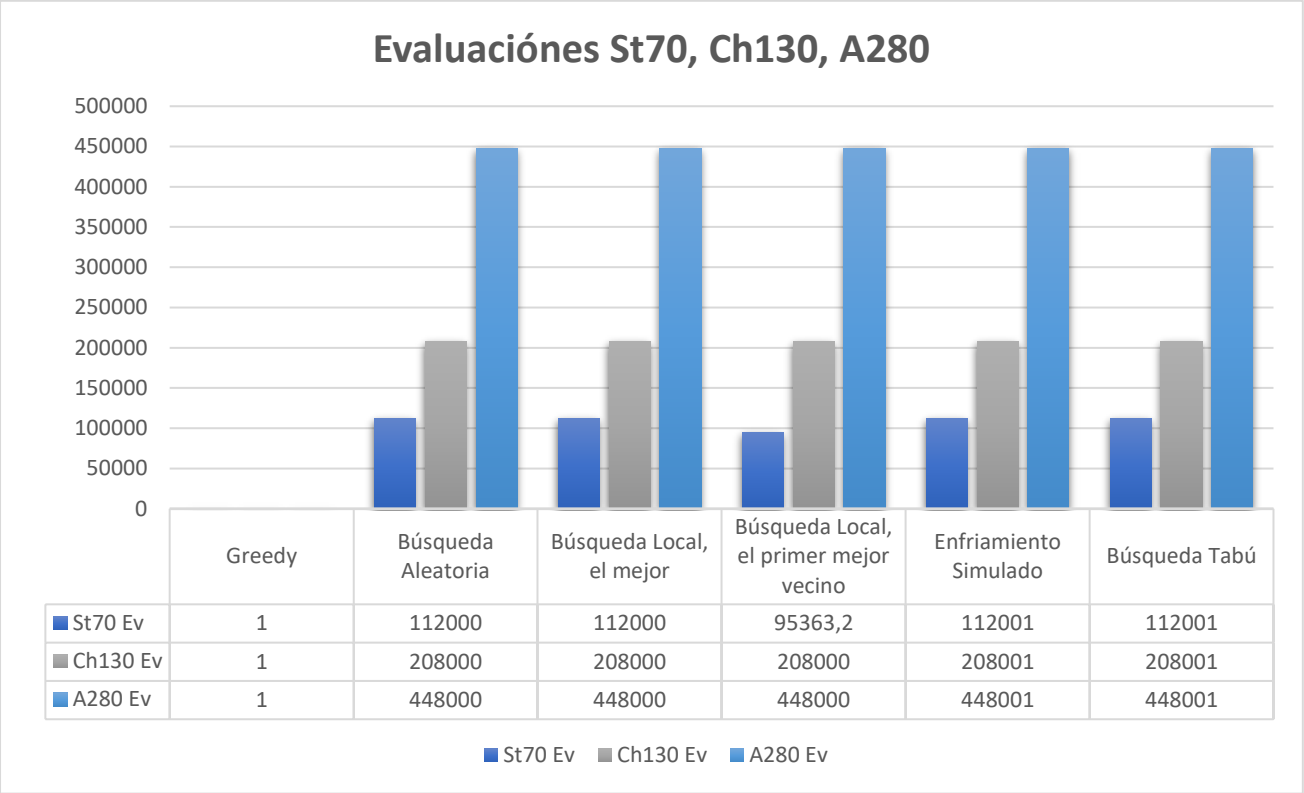
Observaciones:

- Se observa que el algoritmo que ofrece los mejores resultados es el “Greedy”. Pero, debido a que este algoritmo siempre empieza por la primera ciudad y el resto de los algoritmos no alcanzan un mínimo, no se puede decir que este algoritmo sea mejor que los demás.
    - Si lanzásemos este algoritmo con semillas aleatorias, observaríamos una subida del coste.
    - Por otro lado, si delimitásemos el resto de los algoritmos, teniendo en cuenta su capacidad de exploración, tendrían más posibilidades de encontrar el mínimo global.

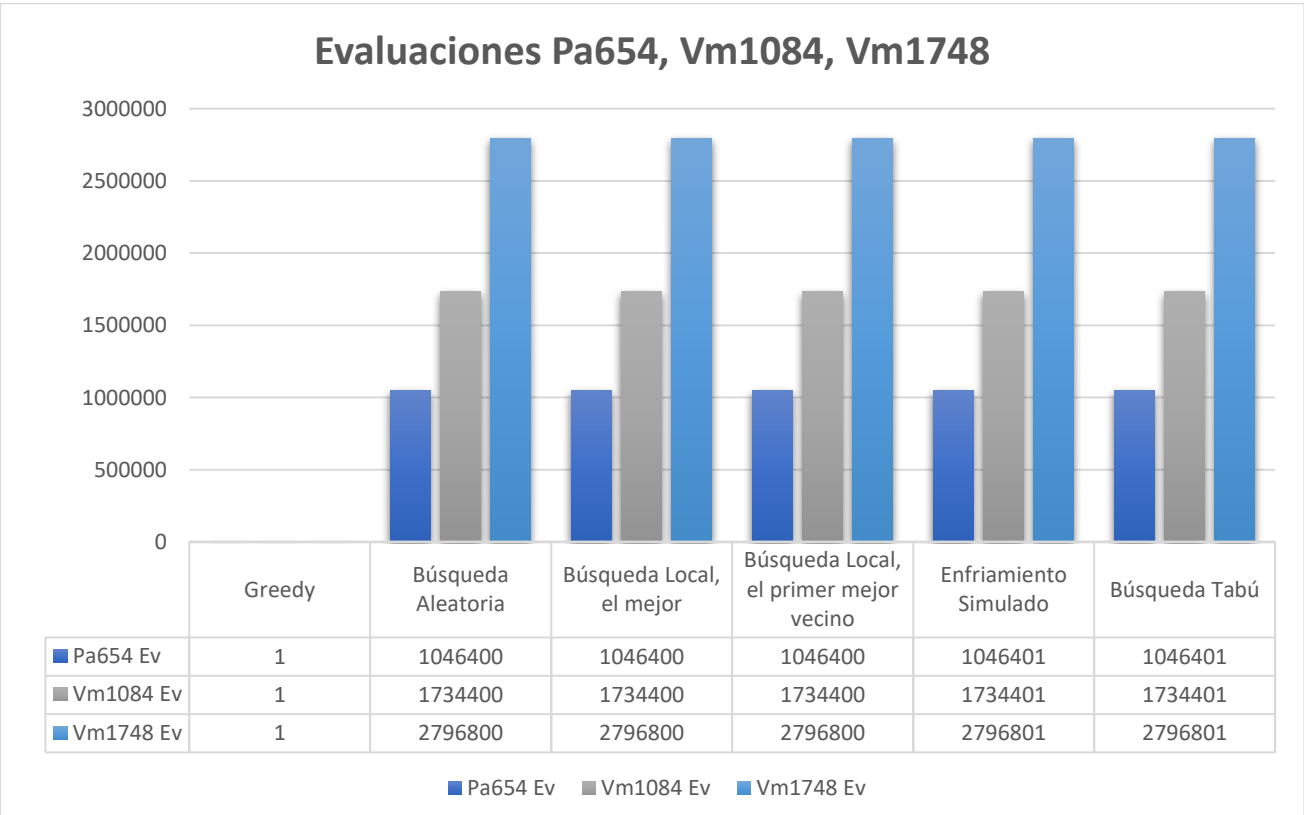
- Por otro lado, se observa que, con conjuntos de datos inferiores a 500 elementos el algoritmo que peor resultado genera es el aleatorio. En cambio, con trazas grandes, se observa que es el algoritmo “Búsqueda local: el mejor”. Esto es debido al límite de evaluaciones establecido, que se alcanza en las primeras iteraciones del algoritmo.

- Por otro lado, se observa que tanto con datasets grandes como con dataset pequeños sin contar el algoritmo “Greedy”, los mejores algoritmos son “Enfriamiento simulado” y “Búsqueda tabú”. Esto es debido a que estos algoritmos poseen una mayor capacidad de exploración.

Comparación evaluaciones



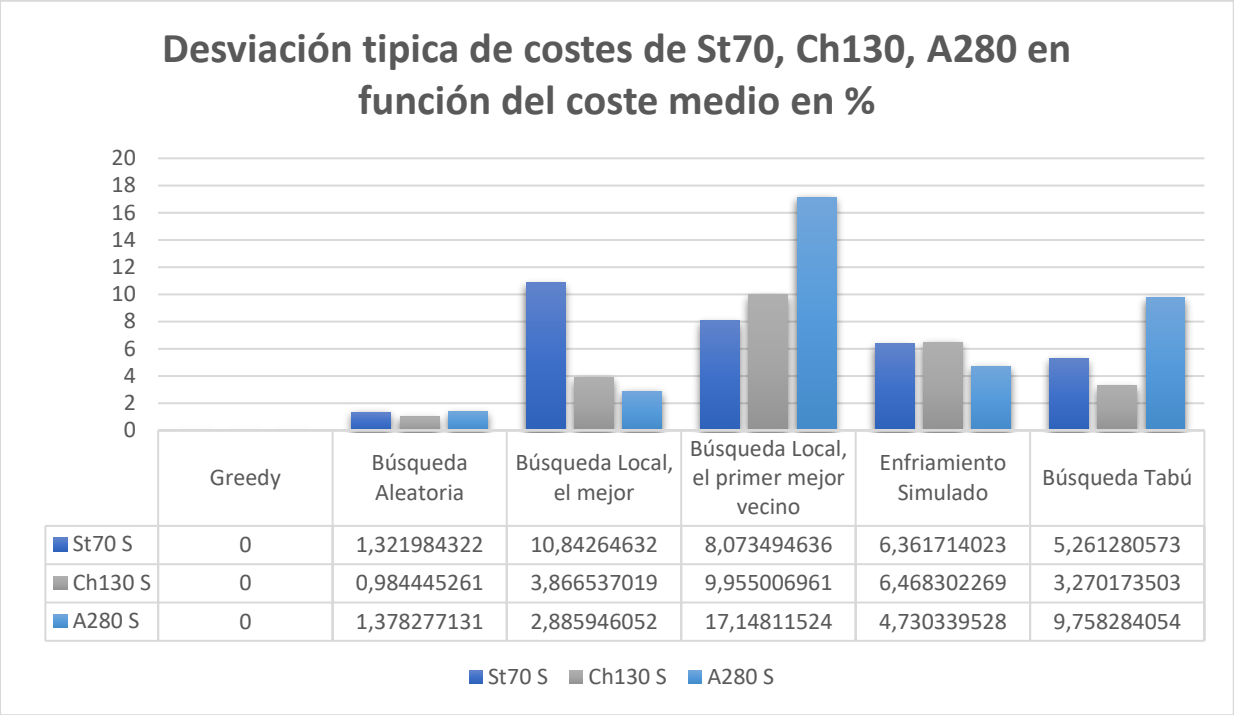
	Greedy	Búsqueda Aleatoria	Búsqueda Local, el mejor	Búsqueda Local, el primer mejor vecino	Enfriamiento Simulado	Búsqueda Tabú
St70 Ev	1	112000	112000	95363,2	112001	112001
Ch130 Ev	1	208000	208000	208000	208001	208001
A280 Ev	1	448000	448000	448000	448001	448001



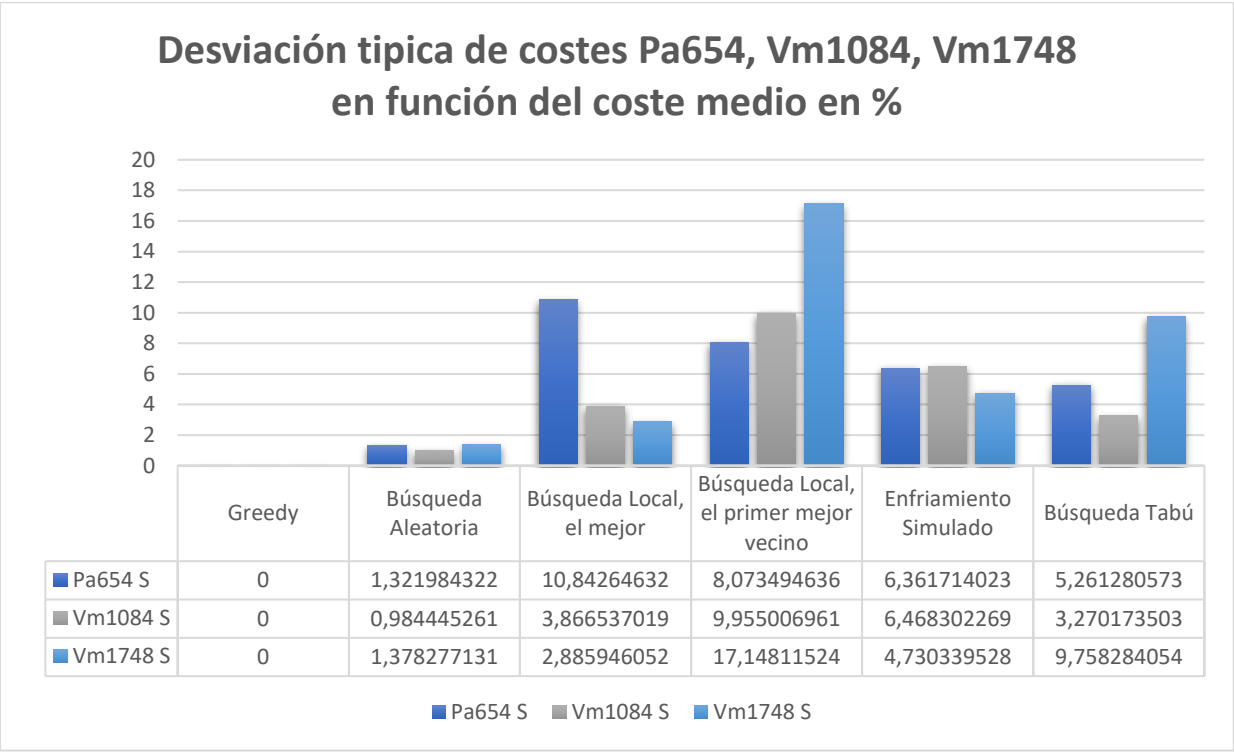
	Greedy	Búsqueda Aleatoria	Búsqueda Local, el mejor	Búsqueda Local, el primer mejor vecino	Enfriamiento Simulado	Búsqueda Tabú
Pa654 Ev	1	1046400	1046400	1046400	1046401	1046401
Vm1084 Ev	1	1734400	1734400	1734400	1734401	1734401
Vm1748 Ev	1	2796800	2796800	2796800	2796801	2796801

Podemos observar que los costes entre los diferentes algoritmos son constantes. Teniendo esto en cuenta, las gráficas de costes anteriores y el funcionamiento del algoritmo “Greedy”. Se reafirma que los mejores algoritmos son “Enfriamiento simulado” y “Búsqueda tabú”.

Comparación de desviación típica de costes



	Greedy	Búsqueda Aleatoria	Búsqueda Local, el mejor	Búsqueda Local, el primer mejor vecino	Enfriamiento Simulado	Búsqueda Tabú
St70 S	0	1,321984322	10,84264632	8,073494636	6,361714023	5,261280573
Ch130 S	0	0,984445261	3,866537019	9,955006961	6,468302269	3,270173503
A280 S	0	1,378277131	2,885946052	17,14811524	4,730339528	9,758284054



	Greedy	Búsqueda Aleatoria	Búsqueda Local, el mejor	Búsqueda Local, el primer mejor vecino	Enfriamiento Simulado	Búsqueda Tabú
Pa654 S	0	1,321984322	10,84264632	8,073494636	6,361714023	5,261280573
Vm1084 S	0	0,984445261	3,866537019	9,955006961	6,468302269	3,270173503
Vm1748 S	0	1,378277131	2,885946052	17,14811524	4,730339528	9,758284054

Observaciones:

- Debido a que el algoritmo “Greedy” siempre genera la misma solución su desviación típica es 0.
- Por otro lado, se observa que el algoritmo más inconsistente con la generación solución es el algoritmo “Búsqueda local: el primer mejor vecino”. Esto es debido a que este algoritmo, partiendo de una solución, no se centra tanto en explorar el entorno sino en buscar el primer mejor vecino para profundizar en la búsqueda de la solución.

Conclusiones

Habiendo analizado las tablas anteriores, se observa que el algoritmo que mejores resultados genera, con la configuración por defecto, es el Greedy. Por consiguiente, si en la vida real tuviésemos limitado el número de evaluaciones, este algoritmo seria el mas óptimo.

Pero, si tuviésemos un equipo con mayor capacidad de computacional, habría más posibilidades de obtener mejores resultados usando algoritmos que exploraran el entorno. Entre los estudiados, los mejores serían el Enfriamiento Simulado y Búsqueda Tabú. Debido a que estos, con las limitaciones establecidas, consiguen encontrar las mejores soluciones el espacio de soluciones frente a los demás. Y, cabe destacar también, la robustez de estos algoritmos a la hora de encontrar estas soluciones. Teniendo una desviación típica de coste inferior al 10% frente al coste medio.