Modelos Bioinspirados y Heurísticas de Búsqueda

PRÁCTICA 2

ANÁLISIS DE ALGORITMOS HEURÍSTICOS NO CONSTRUCTIVOS BÚSQUEDA VNS Y GENÉTICOS

Pablo Cordón Hidalgo

ÍNDICE

INTRODUCCIÓN

En este documento se verán y analizarán los resultados de la aplicación del algoritmo Búsqueda Local, Búsqueda VNS y algoritmos genéticos Básico, CHC y Multimodal para resolver un problema de optimización de uso de una red de bicicletas en la ciudad de Santander, al igual que en la primera práctica.

Para ello partimos de una red de 16 estaciones de bicicletas con una capacidad máxima total de 220 bicicletas, donde se ha extraído el comportamiento medio a lo largo de un día capturando el movimiento de bicicletas en cada estación cada 5 minutos desde las 8:00 am hasta las 7:00 am del día siguiente, es decir , el número de bicicletas que cada cinco minutos en un día se han tomado o dejado en cada estación.

Ejecutaremos cada algoritmo 2 veces, cada una de ellas con una semilla diferente, para comparar resultados. Las semillas usadas en esta práctica son: [7777, 8888]

El dataset usado para esta práctica es también el mismo que en la primera práctica de esta asignatura.

NOTA: En las gráficas hemos usado el color naranja para el mejor coste y el azul para la media de costes

ALGORITMO DE BÚSQUEDA LOCAL, EL PRIMER MEJOR VECINO

Este algoritmo analiza N vecinos de la solución inicial en cada iteración. 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.

Algoritmo	Ev. Mejor	Ev. Media	Ev. Desv	Kms. mejor	Kms. Media	Kms. Desv	Tiempo
BL Primer	267	269	2	393.600	406.398	12.797	4.734

Los Algoritmos de Búsqueda Local están planteados de forma que siguen generando vecinos hasta que no haya ningún vecino que mejore la solución actual, es decir, no paran hasta llegar a un óptimo local.

ALGORITMO DE BÚSQUEDA VNS

La Búsqueda de Entorno Variable (VNS: Variable Neighborhood Search) es una metaheurística para resolver problemas de optimización cuya idea básica es el cambio sistemático de entorno dentro de una búsqueda local (aumentando/alterando el tamaño cuando la búsqueda no avanza).

Algoritmo	Ev. Mejor	Ev. Media	Ev. Desv	Kms. mejor	Kms. Media	Kms. Desv	Tiempo
Búsqueda VNS	1773	2155	382	381.139	400.212	19.073	37.608

Si bien este algoritmo de búsqueda mejora los resultados con respecto a la Búsqueda Local, esto se ve reflejado en un gran aumento de llamadas a la función de evaluación respecto a este, y por consecuencia un aumento del tiempo de ejecución, debido a que este debe calcular el tamaño de la solución a mutar en cada iteración.

ALGORITMO GENÉTICO BÁSICO

El algoritmo genético básico empieza su ejecución generando una población aleatoria de individuos (posibles soluciones del problema) y, en cada iteración, intenta mejorar esta población de individuos combinándola con otra población de hijos y descartando los peores individuos. Esta población de hijos se genera a partir del cruce y mutación de individuos de la población base de la iteración.

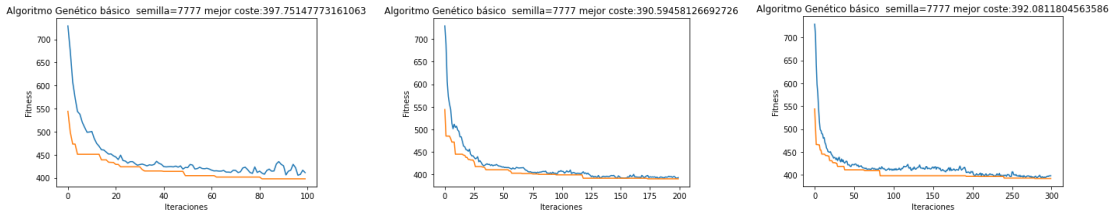
Los resultados presentados en la tabla se han obtenido con 200 generaciones, probabilidad de mutación del 10%, granularidad = 3 y K = 0.1. El tamaño de la pob. inicial será de 30 individuos

Algoritmo	Ev. Mejor	Ev. Media	Ev. Desv	Fitness. mejor	Fitness. Media	Fitness. Desv	Tiempo
G.Básico	5025	5025	0	380.905	384.672	3.767	77.683

ESTUDIO

Se ha realizado un estudio sobre los parámetros numéricos de este algoritmo comparando la evolución del mejor individuo y la media de los individuos

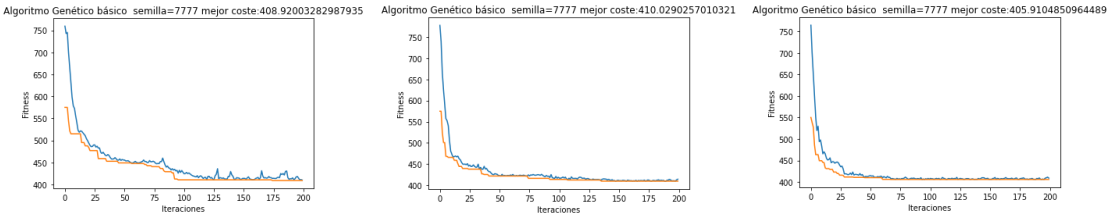
- **Estudio del número de generaciones**



100 generaciones 200 generaciones 300 generaciones

Como podemos observar, a medida que el número de generaciones aumenta, mejoran los resultados. Sin embargo, una vez que el algoritmo ha convergido, la mejora es mínima, o incluso inexistente, por lo que es más rentable buscar un número óptimo de generaciones.

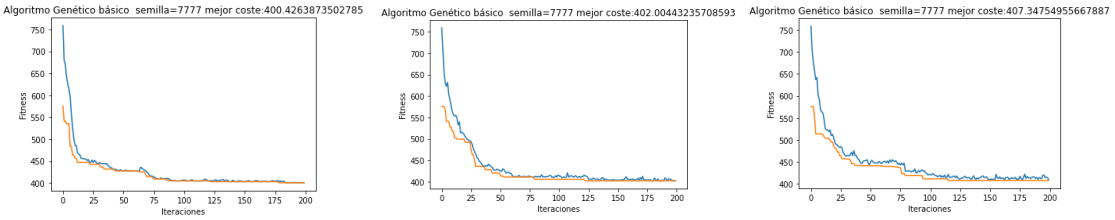
- **Estudio del tamaño de la población**



15 individuos 23 individuos 30 individuos

El tamaño de población es un número [15-30] que establece el tamaño de la población a usar. Se observa, como era de esperar, que según aumenta la población mejora el resultado final obtenido. Pero, en estas gráficas se observa también que la mejor solución no varía mucho entre una población de 15 individuos y una de 23 individuos.

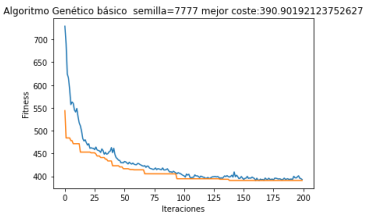
- **Estudio de la probabilidad de mutación**



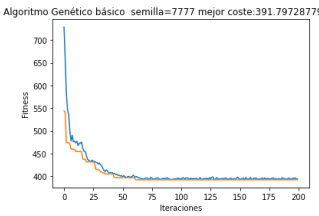
prob 0.05 prob 0.1 prob 0.2

La probabilidad de mutación se define como un porcentaje normalizado [0 - 1], como se puede observar en las gráficas, al elegir una probabilidad de mutación grande se consiguen los resultados varían más. Esto es debido a que se permite al algoritmo explorar más con una probabilidad alta de mutación antes de terminar la ejecución. Para una ejecución óptima, la probabilidad de mutación debería de ser alta al principio e ir disminuyendo en función al número de iteraciones.
En nuestro caso, al tener que ser este constante y con un valor de entre 5% y 20%, la exploración del algoritmo no es especialmente alta.

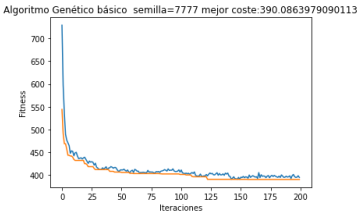
- **Estudio del parámetro k de torneo**



K 0.1



K 0.2



K 0.3

El parámetro K define el porcentaje de la población que se utilizara para la selección de dos individuos a cruzar mediante torneo. Se observa que se obtienen los mejores resultados globales (ambas semillas) con un 10% de los individuos. Pero, para una ejecución óptima, este parámetro debería de empezar con un valor pequeño permitiendo así una mayor exploración, e ir creciendo en el tiempo, para ofrecer una mayor explotación.

ALGORITMO CHC

Este algoritmo, a diferencia del algoritmo genético básico, en cada iteración desordena la población de individuos y genera una población de hijos de cada par de individuos en las que su distancia de Hamming sea superior a la distancia calculada en la iteración. Esta distancia de Hamming se inicializará como la longitud del tamaño de población dividido entre 4 y en cada iteración en la que la población no se altere, se reducirá en 1. Una vez que esta distancia llegue a 0, el cromosoma que represente la mejor solución hasta ese momento se utilizará como patrón para generar la nueva población (copiándose), y el resto se inicializarán de forma aleatoria. La condición de parada de este algoritmo es un número de reinicios fijo.

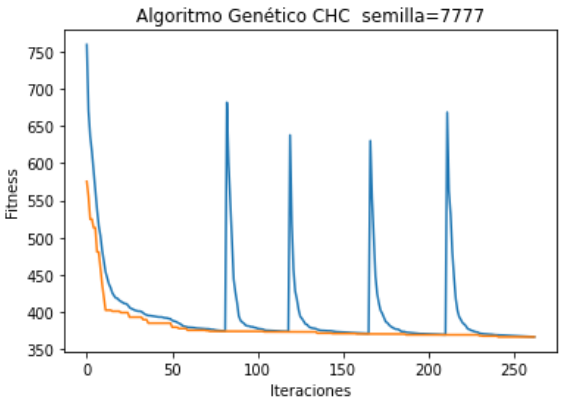
En este caso, para obtener buenos resultados, se ha establecido una población de 30 individuos, 4 reinicios y un umbral de 16/4 = 4

Algoritmo	Ev. Mejor	Ev. Media	Ev. Desv	Fitness. mejor	Fitness. Media	Fitness. Desv	Tiempo
CHC	20259	29195	8936	367.694	367.956	0.261	509.36

Como se puede observar, este algoritmo genético mejora considerablemente los resultados anteriores. También es, con diferencia, el que más veces llama a la función de evaluación, y mayor desviación típica sobre este valor tiene. Esto se debe a la condición de parada, en la que reinicializaremos cada vez que converja.

Sin embargo, la desviación típica de la función fitness es muy baja, lo cual quiere decir que la explotación y exploración del espacio de búsqueda han llegado a la misma región (o regiones muy parecidas).

La cantidad de tiempo que este algoritmo tarda en ejecutarse es directamente proporcional al número de llamadas a la función fitness.



ALGORITMO GENÉTICO MULTIMODAL

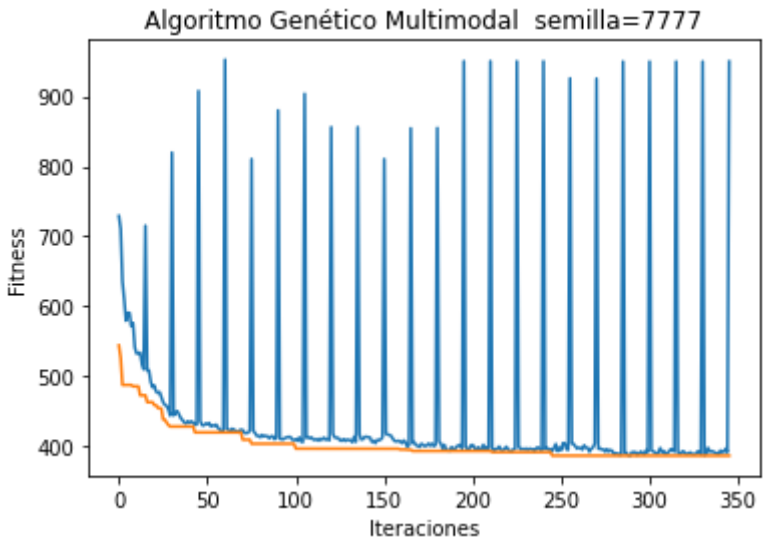
El algoritmo genético multimodal es un algoritmo que utiliza como base el algoritmo genético generacional básico y, la única diferencia que se encuentra frente a este algoritmo es que realiza un proceso de aclarado antes de generar la población de hijos. Este proceso de aclarado sirve para eliminar las soluciones repetidas o muy próximas. De esta forma se consigue una mayor capacidad de exploración.

Para eliminar las soluciones repetidas o muy próximas, les daremos un valor muy alto, haciendo que el algoritmo las deseche rápidamente. Es por eso que en la gráfica cada x generaciones vemos picos en la media de soluciones.

Algoritmo	Ev. Mejor	Ev. Media	Ev. Desv	Fitness. mejor	Fitness. Media	Fitness. Desv	Tiempo
G. Multimodal	5.900	7.212	1.312	382.519	386.633	4.115	115.790

Se observa, como era de esperar, un comportamiento muy parecido al que se ha obtenido en el algoritmo genético básico. Esto es debido a que este algoritmo parte del funcionamiento del algoritmo genético básico y en lo único en lo que cambia es en la distribución de soluciones gracias al proceso de aclarado, sin llegar a mejorar ligeramente los resultados de este.

En este caso, se ha establecido como radio de nicho (**σ**) un 3% del cromosoma y 5 individuos por nicho (**kappa**).

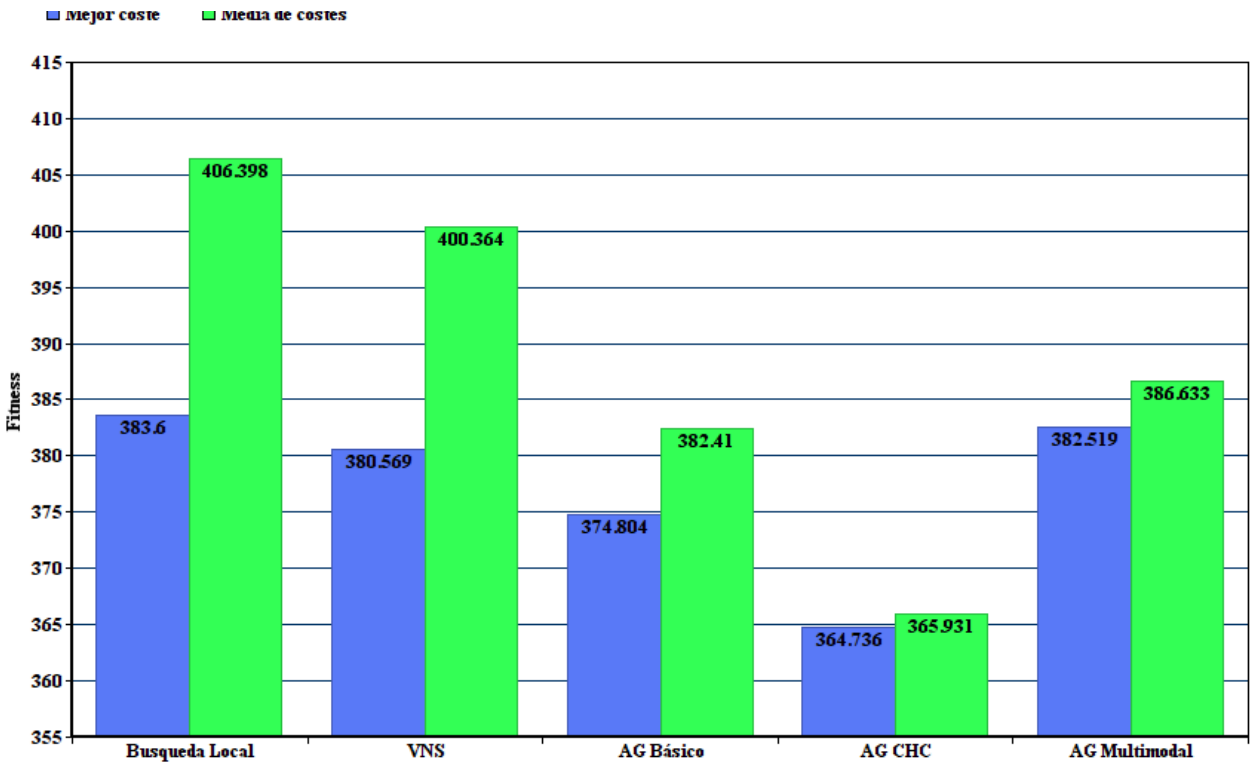


COMPARACIÓN DE ALGORITMOS

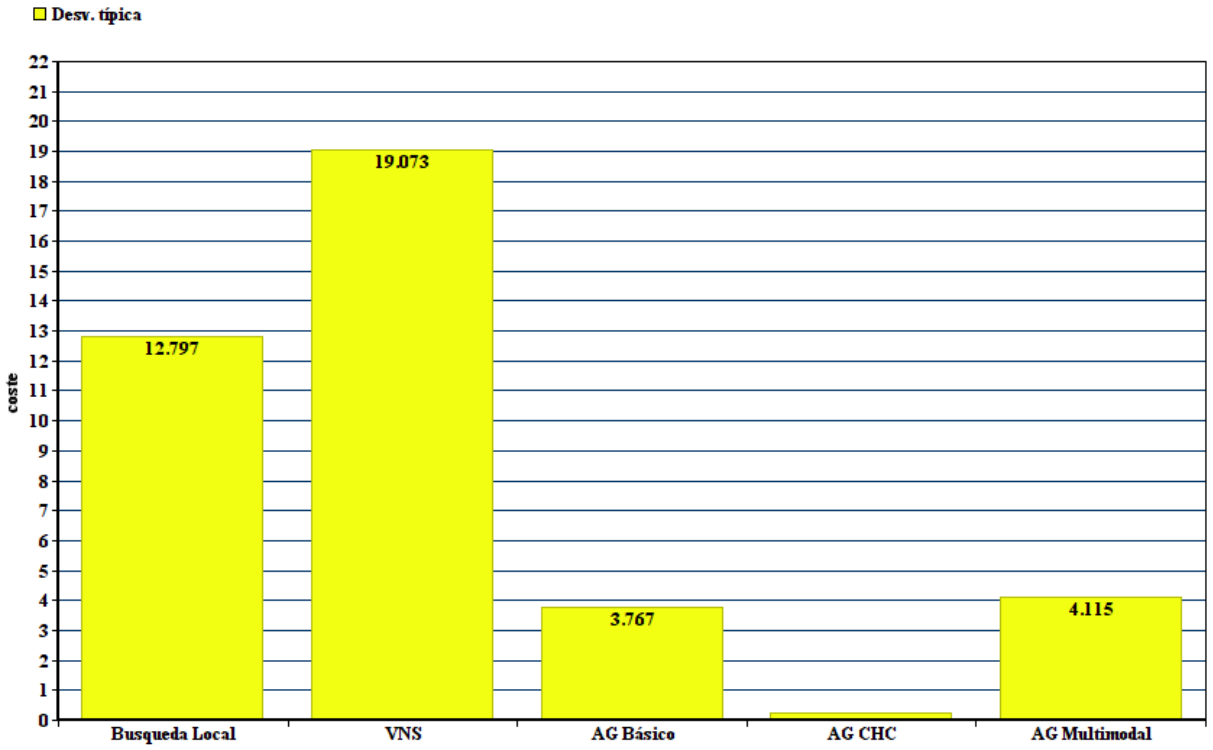
- **Tabla comparativa**

Algoritmo	Ev. Mejor	Ev. Media	Ev. Desv	Fitness. mejor	Fitness. Media	Fitness. Desv	Tiempo
BL Primer	267	269	2	393.600	406.398	12.797	4.734
Búsqueda VNS	1773	2155	382	381.139	400.212	19.073	37.608
G.Básico	5025	5025	0	380.905	384.672	3.767	77.683
CHC	20259	29195	8936	367.694	367.956	0.261	509.36
G. Multimodal	5900	7212.5	1312.5	382.519	386.633	4.115	115.790

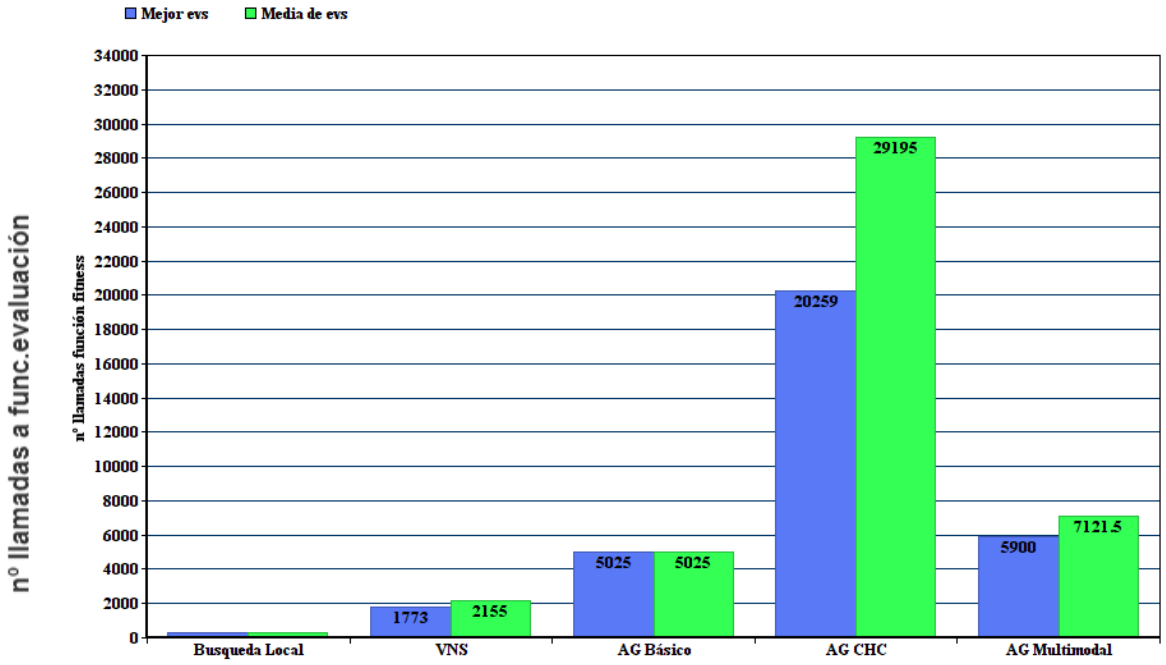
- **Gráfica de comparación de costes**



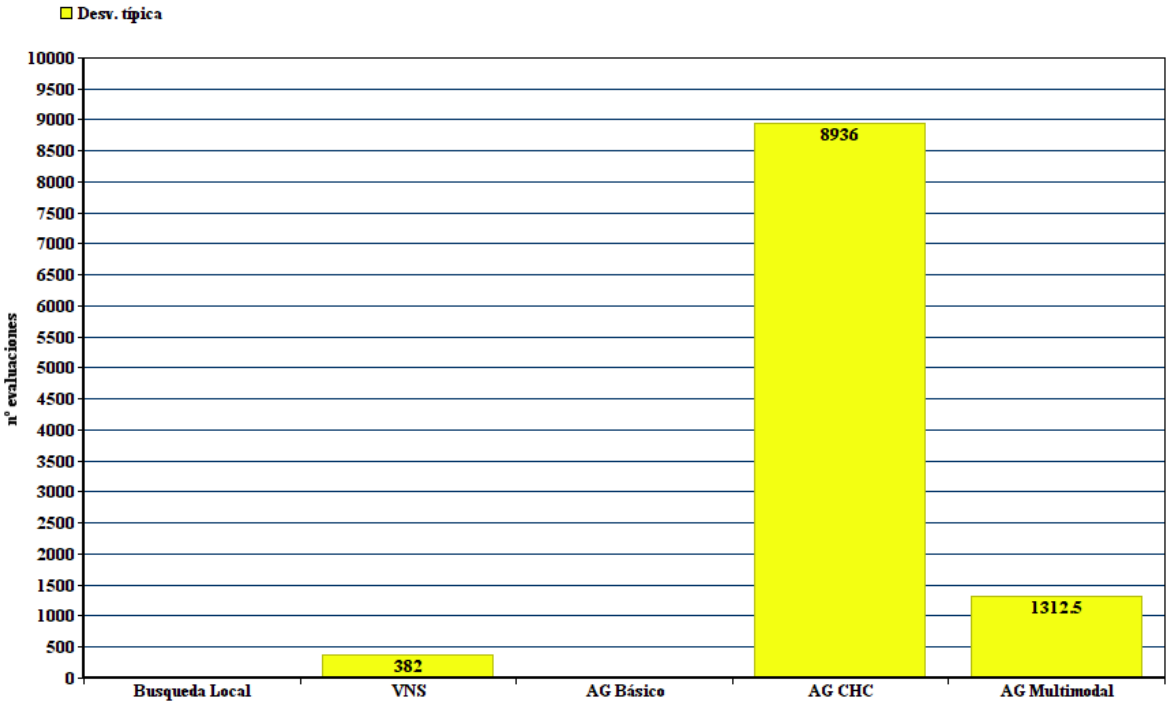
- **Gráfica de comparación de desviación típica de los costes**



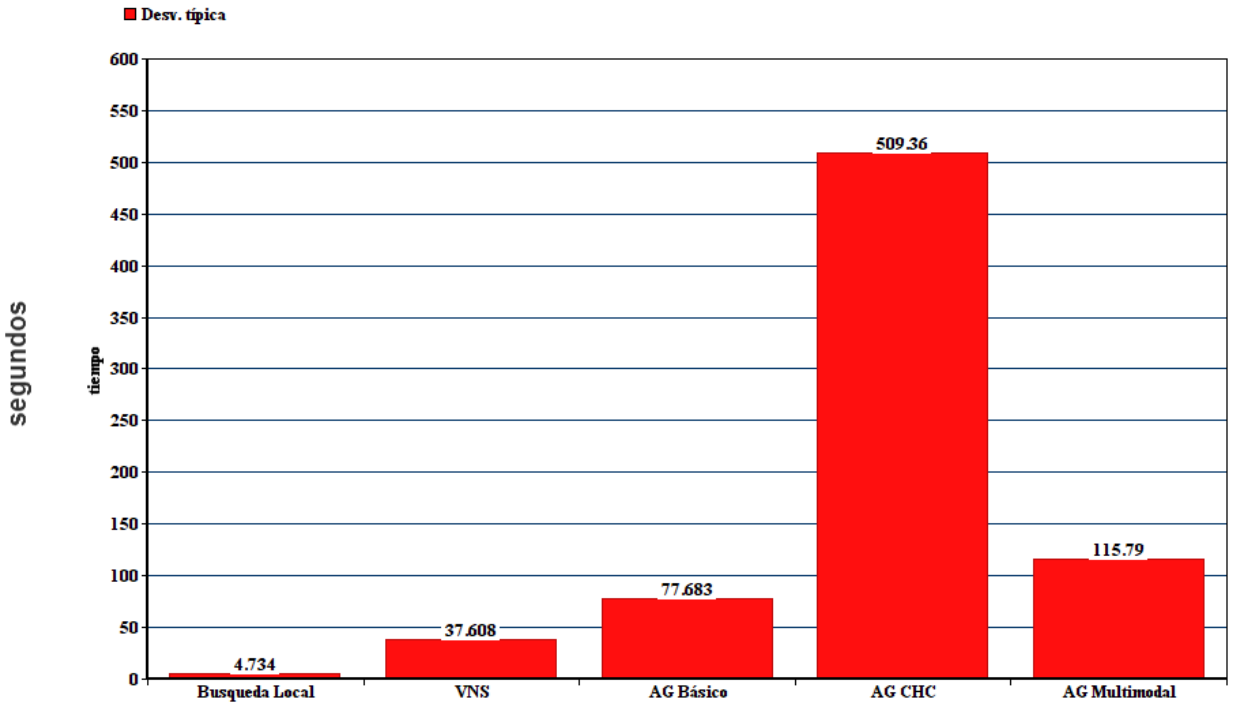
- **Gráfica de comparación de número de llamadas a la función de evaluación**



- **Gráfica de comparación de desviación típica en el número de llamadas a la función de evaluación**



- **Gráfica de comparación de tiempos**



OBSERVACIONES

- Debido a que el Algoritmo Genético Básico Generacional siempre realiza el mismo número de iteraciones, la desviación típica del número de llamadas a la función de evaluación es 0

- El tiempo de ejecución del algoritmo y el número de llamadas a la función de evaluación son directamente proporcionales, pues lo más costoso de todos estos algoritmos son las llamadas a la función de evaluación.

- Los picos que podemos observar en las gráficas de coste medio de los AG CHC y Multimodal corresponden a cada vez que en el código se realiza un reinicio o un clearing, respectivamente.

- La mayor desviación típica respecto al número de llamadas a la función fitness la tiene el algoritmo Genético CHC. Esto puede deberse al valor aleatorio de la población para cada una de las semillas, al igual que el uso de esta función de evaluación en cada reinicio y en funciones auxiliares como *check_pob_igual().* Al mismo tiempo, este algoritmo también tiene la menor desviación típica respecto al fitness.

- Por otro lado, se observa que el algoritmo más inconsistente con la generación solución es el algoritmo “VNS”, lo cual era de esperar, ya que este aumenta el tamaño del entorno cuando la búsqueda no avanza.

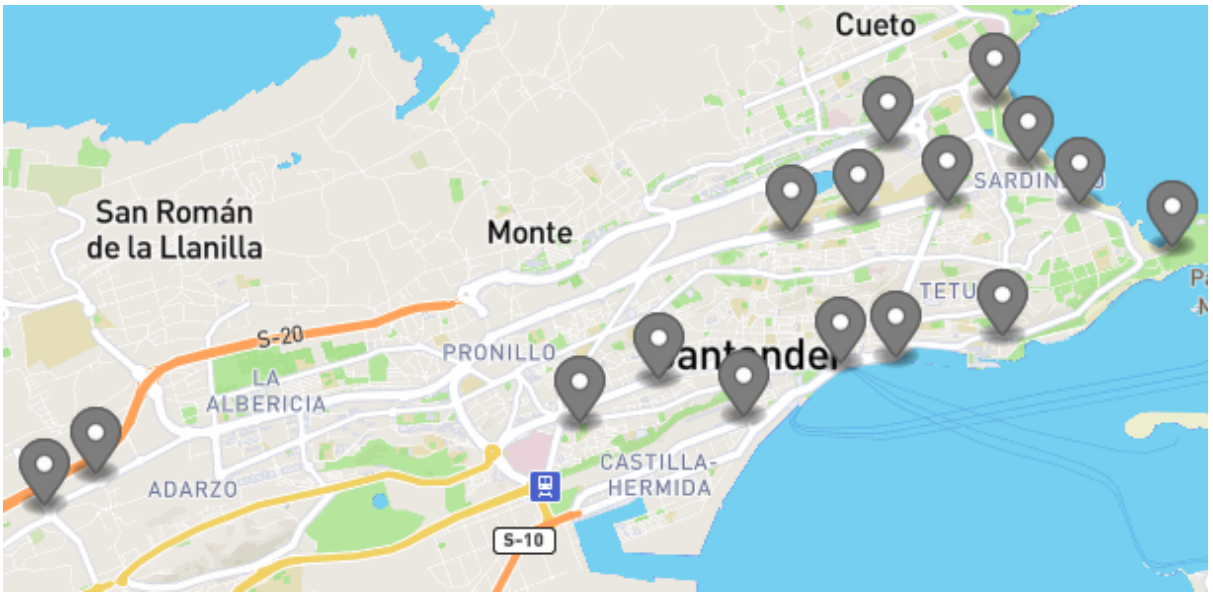
CONCLUSIONES

Si bien las mejores soluciones las aporta el Algoritmo Genético CHC, este es también la que tarda mayor tiempo, por lo que en el caso de tener un equipo con menor capacidad computacional , o un dataset mayor, el tiempo de evaluación crecería exponencialmente, pudiendo llegar a tiempos inalcanzables.

Por lo general, la desviación típica del coste no es muy grande en este dataset, pero teniendo en cuenta la robustez de los algoritmos, el más robusto teniendo en cuenta coste y llamadas a la función fitness es el AG Básico Generacional, ya que su número de iteraciones y parámetros numéricos son constantes.

REPRESENTACIÓN DE LAS ESTACIONES EN EL MAPA

Para representar la solución final en un mapa hemos usado el repositorio interactivo de github “geojson” (http://geojson.io/#map=13/43.4836/-3.8353) , que nos permite indicar puntos en un mapa con un json dadas unas coordenadas y definir para cada uno de estos puntos ciertos elementos, en nuestro caso el id, la posición en la que se encuentra en el dataframe, el número de bicicletas en la solución inicial, y el número de bicicletas en la mejor solución de todas las estudiadas




```
{
  "type": "Feature",
  "geometry": {
    "type": "Point",
    "coordinates": [
      -3.817978245,
      43.46072554
    ]
  },
  "properties": {
    "id": 13,
    "pos_dataframe": 9,
    "sol_ini": 10,
    "sol_final": 15
  }
```