Universidad
Carlos III de Madrid

Grado en Ingeniería Informática

Curso 2021-2022

Algoritmos Genéticos y Evolutivos

Ampliación Práctica 1
Optimización de Sensores en Smart Cities

Jorge Rodríguez Fraile 100405951 Grupo 83 100405951@alumnos.uc3m.es

Índice

Índice de figuras

Índice de tablas

Introducción

En esta ampliación de la práctica 1 vamos a tratar de resolver el problema de las estaciones de medición de la calidad del aire de la Comunidad de Madrid mediante una nueva codificación de los individuos. Se mantienen las 24 estaciones donde cada una mide 16 variables ambientales diferentes. El problema trata sobre determinar que sensores de los presentes en las estaciones deben estar permanentemente o programados para reducir el coste de su mantenimiento e instalación, pero sin perder precisión en las mediciones.

Codificación y Función de fitness

En este caso la codificación pasará de ser binaria, cada individuo un vector de 384 bits de 0 y 1, a una codificación discreta, vector de 384 valores con 3 posibles valores cada uno, los valores que podrá tomar son:

- **0:** No se instala el sensor en la estación.
- **F:** El sensor se instala en la estación y se usa de manera permanente.
- **H:** El sensor se instala en la estación, pero se programa su uso.

Cada uno de los cromosomas estará representado por $24 \cdot 16 = 384$ genes de manera que cada 16 valores sea una estación distinta, se empezará por el primer sensor de la primera estación, después el segundo sensor y así hasta el último de la primera para dar paso a la siguiente estación. En cuanto al sensor que representa cada valor son los siguientes:

1	Dióxido de Azufre	SO_2	$\mu g/m^3$
2	Monóxido de Carbono	CO	$\mu g/m^3$
3	Monóxido de Nitrógeno	NO	$\mu g/m^3$
4	Dióxido de Nitrógeno	NO_2	$\mu g/m^3$
5	Partículas <2.5 µm	PM2.5	$\mu g/m^3$
6	Partículas <10 µm	PM10	$\mu g/m^3$
7	Óxidos de Nitrógeno	NOx	$\mu g/m^3$
8	Ozono	O_3	$\mu g/m^3$
9	Tolueno	TOL	$\mu g/m^3$
10	Benceno	BEN	$\mu g/m^3$
11	Etilbenceno	EBE	$\mu g/m^3$
12	Metaxileno	MXY	$\mu g/m^3$
13	Paraxileno	PXY	$\mu g/m^3$
14	Ortoxileno	OXY	$\mu g/m^3$
15	Hexano (total)	TCH	mg/m^3
16	Hexano (no metánicos)	NMHC	mg/m^3

Tabla 1: Medidas de los sensores

Ejemplo: 00F00F0H00000000 0000000000000000 . . .
Indica que en la primera estación se instalan los sensores de NO permanente, PM10 permanente y O_3 se programará, sin embargo en la segunda estación ni siguiera se instala al no tener sensores.

Aunque la codificación ya la hemos decidido se analizara su conveniencia para los algoritmos genéticos:

- La codificación es completa, todas las posibilidades de sensores para todas las estaciones son codificables.
- Solo se codifican soluciones factibles.
- Cada solución solo tiene una posible solución.
- La descodificación es fácil y rápida, cada 16 valores es una estación y cada uno de esos valores un sensor.
- La codificación es adyacente, es decir, al variar un bit los cambios que experimenta son pequeños, se pone de manera permanente, programada o se quita un sensor de una estación.

El espacio de búsqueda se compone ahora de 3^{384} posibles soluciones, lo que lo hace $4,16 \cdot 10^{67}$ veces más grande que el de codificación binaria. Esto nos hace pensar que emplea algoritmos genéticos en este problema es una buena idea, ya que si no era posible por fuerza bruta en binario este mucho menos.

Cambios

Para realizar esta ampliación ha sido necesario modificar el programa que se había desarrollado, los aspectos cambiados son los que se describen en las siguientes subsecciones, el resto se mantiene igual.

Convergencia y Fitness

La función de fitness que se emplea para esta ampliación se obtiene mediante una petición de get a http://memento.evannai.inf.uc3m.es/age/alfa2?c= seguido por el cromosoma que deseamos evaluar, el cambio es la web a la que consultar. Para obtener la evaluación de una población se utilizará el mismo método que en la parte anterior, *evaluate_population(population, num_evaluations)*.

El criterio de convergencia se ha mantenido igual con respecto a la práctica base, termina si llega a 0 o si llega a una determinada generación máxima.

Población inicial

Se ha modificado el método *initial_population()* que en este caso en vez de generar como base una matriz de 0 con numpy genera una matriz de elementos del tipo carácter, esto nos permitirá poder almacenar los numero y letras (0, H y F). El vector generado conserva las mismas dimensiones, tantas filas como individuos y tantas columnas como genes.

Después para rellenar la matriz con nuestros individuos de manera aleatoria, lo que se ha hecho es generar un número entero entre 0 y 2 incluido. Para cada posición de la matriz se le asigna uno de los posibles valores en función del valor aleatorio que se genera, si sale 0 es 0, 1 es H y 2 es F.

Mutación

El operador de mutación *mutation(population, factor)* ha sido necesario modificarlo dado que ya no vale que si se hace mutación de un gen se calcule como 1-valorAnterior, en este caso cuando se vaya a mutar un gen se calculará un valor entero entre 0 y 1 incluido, de esta manera se podrá elegir aleatoriamente entre los otros 2 valores para el gen.

Pruebas

Los modelos se han nombrado siguiendo una codificación con el siguiente formato XX_YY_ZZ, tal que:

- **XX:** Número de generaciones generadas.
- **YY:** Tamaño del conjunto de candidatos al hacer torneo para la selección.
- **ZZ:** Porcentaje de mutaciones sobre la población.

Las pruebas realizadas en esta ampliación corresponden con los 2 mejores modelos que se obtuvieron en la parte anterior. Se ha probado además una ampliación del mejor modelo poniéndole 2000 ciclos máximos. Los modelos probados son:

- 200 ciclos máximos, 10 individuos por torneo y un porcentaje de mutación del 0,5 %.
- 2000 ciclos máximos, tamaño de torneo 10 y con un porcentaje de realizar mutación del 0,5 %.
- 1000 ciclos máximos, tamaño de torneo 10 y con un porcentaje de realizar mutación del 0,1 %.

Se ha optado por añadir el de 2000 generaciones dado que cambiar los torneos o mutación empeoraba los resultados y dado que ahora hay muchas más posibles soluciones, por lo que ampliar las generaciones máximas era una buena opción para probar.

A continuación se presentan de manera gráfica los resultados obtenidos para los modelos durante 5 ejecuciones y su media en forma de línea, se ha optado por realizar 5 ejecuciones para poder ver como de buenos son los modelos y evitar el sesgo.

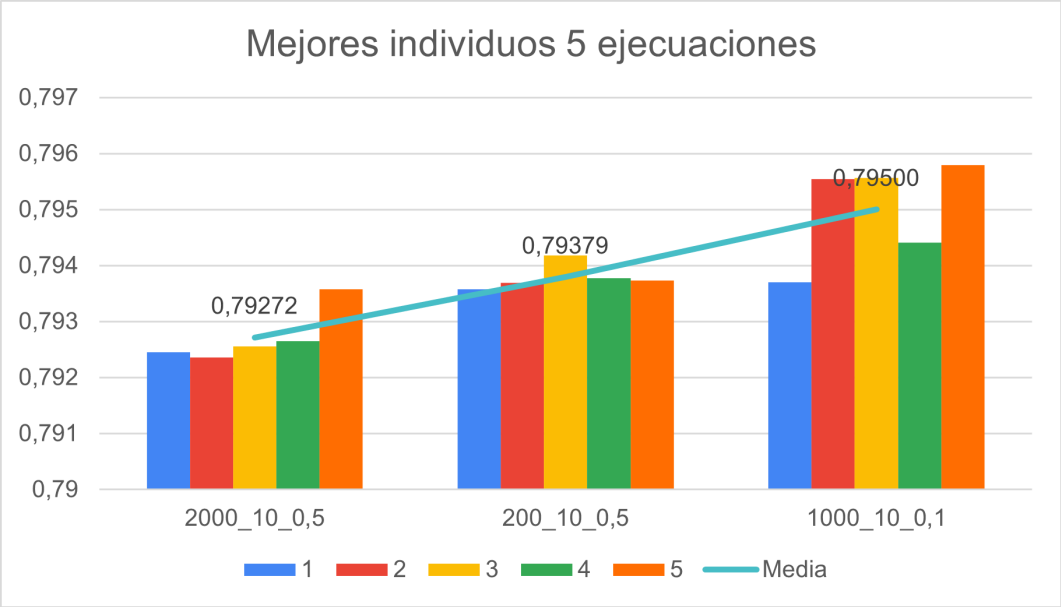

Figura 1: Evaluación de los Modelos probados

Los datos en los que se basa la gráfica son los de la siguiente tabla, en la que se ha marcado para cada uno de los modelos cuál es la mejor prueba en una escala de verde a rojo, siendo verde el mejor y rojo el peor. Se ha elegido el gradiente para cada modelo y no en general para facilitar la distinción de los tonos de colores. Además, se presenta la media de las pruebas, también siguiendo el gradiente de color verde a rojo.

Modelo	1	2	3	4	5	Media
2000_10_0,5	0,792457154	0,792358877	0,792563621	0,792647846	0,793575478	0,792720595
200_10_0,5	0,793575478	0,793694524	0,794183547	0,793778166	0,793740165	0,793794376
1000_10_0,1	0,793702786	0,79554339	0,79556596	0,794407778	0,795797182	0,795003419

Tabla 2: Fitness primeros modelos

Podemos ver que en general dan buenos resultados, como pasaba en la parte anterior de la práctica el mejor modelo ha sido el correspondiente a **tamaño de torneo 10** y **porcentaje de mutación del 0,5 %**. En la siguiente sección se explicará en más detalle la mejor solución.

Resultado

El mejor modelo que se ha obtenido ha sido el **2000_10_0,5**, que corresponde a un máximo de **2000 generaciones** que han sido **200100 evaluaciones**, aunque de media los mejores resultados han necesitado 107320 evaluaciones (1074 generaciones), con un tamaño de **torneo de 10 individuos** y un porcentaje de **mutación del 0,5 %** de la población, el mejor resultado obtenido tiene un **fitness de 0,792358877114** y su cromosoma asociado es:

0000HHHH00H0H0H0 0H00HH0HH0H0H0H0 00H0H00H00H0000H 0H0HHHHHH000H0H0 000H0H00H0H0H0HH 0H00H0H000000HH0 0H0HH0HHHHH0H00H 00HH0HHHHH00HHH0 0H0HHH0HH0H000H0 00H0H0HH0HH000HH 0HH0H00HH0H0H000 0HHHH000H0H0HH0H 0H0HH0H00HH0H0H0 000HH000HH00H00H 0H0HH00HH0H000HH 0H0000H00000H0HH 00H0H000H0H0HHHH 0HHH0H0HHH00H0HH H000000H0H0000HH H000HH0H0HH0HHH0 0H0HH00HH0H000H0 0H0H0HHH000000HH 0H00HH00H0H0HH00 0HHH0H00H0H0H000

Algo que se puede observar en esta solución es que no hay ningún gen con valor F, los sensores no se instalan o se programan, pero no merece la pena dejarlos de manera permanente.

En la siguiente gráfica se muestra cuál ha sido la progresión del fitness a lo largo de las evaluaciones para este modelo.

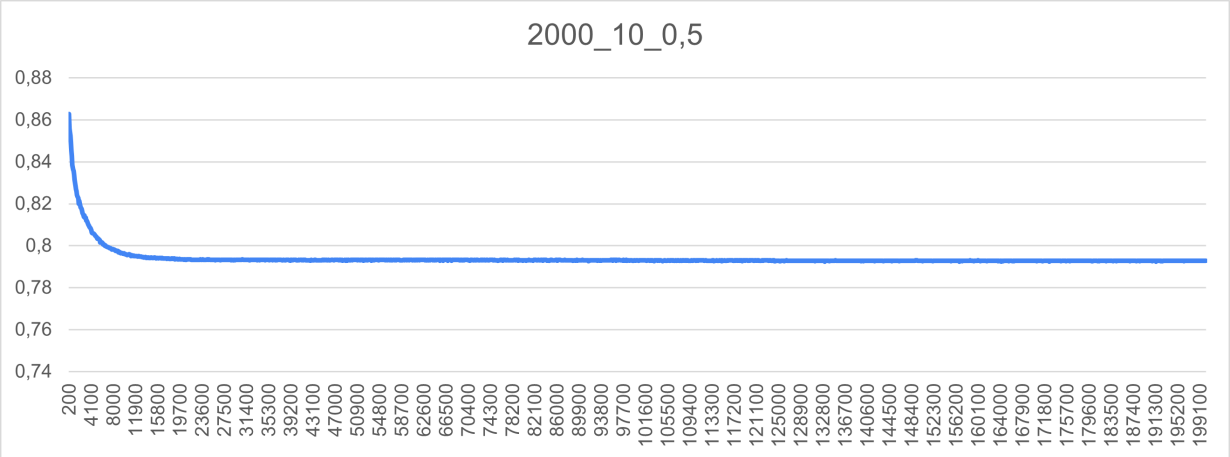

Figura 2: Fitness vs. Evaluaciones 1000_10_0,5

Como se puede ver el fitness al comienzo baja rápidamente y llegado un punto se empieza a estabilizar, hay subidas y bajadas mínimas, pero no hay una tendencia a mejorar o empeorar muy marcada. El mejor resultado está en la evaluación 172600, pero como se puede ver los valores llevaban muchos ciclos oscilando entre 0,793 y 0,794.

El significado del cromosoma que se ha puesto más arriba es que las estaciones tendrán los siguientes sensores programados, dado que no hay ninguno que se instale de manera permanente:

- **Estación 1:** PM2.5, PM10, NOx, O_3, EBE, PXY y TCH.
- **Estación 2:** CO, PM2.5, PM10, O_3, TOL, EBE, PXY y TCH.
- **Estación 3:** NO, PM2.5, O_3, EBE y NMHC.
- **Estación 4:** CO, NO_2, PM2.5, PM10, NOx, O_3, TOL, PXY y TCH.
- **Estación 5:** NO_2, PM10, TOL, EBE, PXY, TCH y NMHC.
- **Estación 6:** CO, PM2.5, NOx, OXY y TCH.
- **Estación 7:** CO, NO_2, PM2.5, NOx, O_3, TOL, BEN, EBE, PXY y NMHC.
- **Estación 8:** NO, NO_2, PM10, NOx, O_3, TOL, BEN, PXY, OXY y TCH.
- **Estación 9:** CO, NO_2, PM2.5, PM10, O_3, TOL, EBE y TCH.
- **Estación 10:** NO, PM2.5, NOx, O_3, BEN, EBE, TCH y NMHC.
- **Estación 11:** CO, NO, PM2.5, O_3, TOL, EBE y PXY.
- **Estación 12:** CO, NO, NO_2, PM2.5, TOL, EBE, PXY, OXY y NMHC.
- **Estación 13:** CO, NO_2, PM2.5, NOx, BEN, EBE, PXY y TCH.
- **Estación 14:** NO_2, PM2.5, TOL, BEN, PXY y NMHC.
- **Estación 15:** CO, NO_2, PM2.5, O_3, TOL, EBE, TCH y NMHC.
- **Estación 16:** CO, NOx, PXY, TCH y NMHC.
- **Estación 17:** NO, PM2.5, TOL, EBE, PXY, OXY, TCH y NMHC.
- **Estación 18:** CO, NO, NO_2, PM10, O_3, TOL, BEN, PXY, TCH y NMHC.
- **Estación 19:** SO_2, O_3, BEN, TCH y NMHC.
- **Estación 20:** SO_2, PM2.5, PM10, O_3, BEN, EBE, PXY, OXY y TCH.
- **Estación 21:** CO, NO_2, PM2.5, O_3, TOL, EBE y TCH.
- **Estación 22:** CO, NO_2, PM10, NOx, O_3, TCH y NMHC.
- **Estación 23:** CO, PM2.5, PM10, TOL, EBE, PXY y OXY.
- **Estación 24:** CO, NO, NO_2, PM10, TOL, EBE y PXY.

Conclusiones

En este proyecto se ha abordado el problema de seleccionar sensores para estaciones de control de calidad del aire en la ciudad Madrid con una nueva codificación. Se ha podido comprobar que esta nueva codificación para el problema es viable y se obtienen buenas soluciones empleando algoritmos genéticos, cosa que no ocurriría con fuerza bruta como vimos en la práctica base. Además, que los modelos que se idearon en la práctica base siguen funcionando para esta codificación.

En cuanto a la codificación se ha podido ver que las soluciones cuando van evolucionando y mejorando nunca codifican soluciones que instalen de manera permanente sensores en las estaciones, al igual que todas las soluciones mantienen las 24 estaciones y no dejan ninguna vacía.