Análisis de Hiperparámetros de un Algoritmo Evolutivo Aplicado al Problema One Max

Andrés Matesanz

Abstract - En este informe se presenta un análisis sobre el rendimiento de un algoritmo evolutivo sencillo sobre el problema *"one max*". Se presenta el mencionado problema y cómo se ha aproximado su solución. Se realizan variaciones en los valores de los hiperparámetros y su relación con el resultado final.

1. Introducción

En matemáticas y ciencias de la computación, un problema de optimización es aquél en el que se trata de encontrar la mejor solución dentro de todas las soluciones factibles a dicho problema.

Los problemas de optimización con múltiples variables discretas (o continuas si se discretiza su espacio) pertenecen a los conocidos como **problemas de optimización combinatoria**. La solución a un problema de optimización es tal que ninguna otra solución dentro del espacio de búsqueda minimiza más la función objetivo. Explorar todo el espacio de búsqueda, en general, es altamente costoso en cuanto a recursos de tiempo y computación, por ello se emplean **algoritmos de aproximación** conocidos como metaheurísticas.

Los **algoritmos evolutivos** son un subconjunto de metaheurísticas, dentro de los problemas de optimización, inspirados en la evolución natural de las especies. Se genera un número determinado de individuos, llamado población, con configuraciones distintas entre sí, cuyo objetivo común es minimizar la función objetivo dentro del espacio de búsqueda. Cada población nace de su predecesora, aplicando una serie de técnicas de cruce y mutación a aquellos individuos que mejores resultados hayan demostrado en cada iteración.

En este caso concreto se va a tratar con el problema ***One Max***. Es uno de los más básicos pero que, aún con todo, nos permite mostrar fácilmente cómo funciona un algoritmo evolutivo. Consiste en crear un conjunto de individuos con un genotipo, llamado también cromosoma, formado por uno o varios vectores unidimensionales de alelos, los cuales pueden tomar únicamente los valores 0 y 1. La **función objetivo** que se trata de minimizar es la de dar con un individuo que tenga en su **cromosoma** algún vector compuesto únicamente por **valores de 1**.

Los alelos, o genes, de los individuos son inicializados aleatoriamente y en cada iteración sufren cruce, con un probabilidad relativamente alta (del 80% para este caso en concreto), y/o mutación con una probabilidad relativamente baja, indirectamente proporcional al número total de genes que posee cada individuo. La función de *fitness* de cada individuo, por el que se valora su rendimiento en la iteración, consiste en determinar el número de valores de 1 que tiene en su genotipo, naturalmente, el *fitness* máximo que puede alcanzar cualquier sujeto de la población es el número total de genes dentro de cualquiera de su vector de alelos.

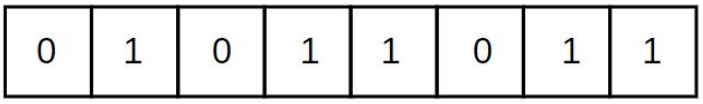

Imagen 1: Vector de alelos

2. Estudio empírico

El objetivo de este documento es analizar cómo se comporta el algoritmo evolutivo en función de los hiperparámetro seleccionados previa resolución del ejercicio. Algunos de ellos han sido ya mencionados previamente: longitud del vector de alelos, número de vectores, tamaño de la población, probabilidad de cruce, probabilidad de mutación o número máximo de iteraciones posible.

Los hiperparámetros para los que vamos a estudiar el comportamiento del algoritmo evolutivo son dos: el **número de genes** y el **tamaño de la población.** Son dos valores que a priori están muy interrelacionados, dado que un mayor número de genes amplía el espacio de búsqueda, una población más grande diversifica más las posibilidades de localizar una solución óptima.

La simulación del algoritmo evolutivo se puede encontrar aquí. Se trata de un modelo escrito en Java que nos devuelve el número de evaluaciones realizadas en cada simulación. Una evaluación consiste en determinar el *fitness* de un único individuo. Para este experimento, sin embargo, se ha tomado **el número de iteraciones** medio tras cada simulación para un tamaño poblacional determinado con una longitud de genes determinada, ya que es más representativo del número de pasos de cruce y mutación que ha sufrido la población. Dicho número de iteraciones se extrae dividiendo el total de evaluaciones entre el total de individuos obtenido tras cada ejercicio.

Cabe mencionar, que a pesar de que la variable **tiempo** no ha sido tomada para este ensayo, a mayor número de individuos se experimentó una duración mayor de cada simulación. Es decir, que un número menor de iteraciones no implica una un menor tiempo en la ejecución del algoritmo.

Para este ensayo se ha tomado un conjunto de configuraciones para la que las mencionadas variables aumentan su valor exponencialmente desde 2 hasta un máximo de 512.

	Gen. 2	Gen. 4	Gen. 8	Gen. 16	Gen. 32	Gen. 64	Gen. 128	Gen. 256	Gen. 512
Pob. 2	3	11	29	69	239	544	1397	3176	6812
Pob. 4	5	10	32	82	207	517	1406	3019	6844
Pob. 8	9	9	33	90	222	553	1401	3428	8140
Pob. 16	17	17	52	128	295	695	1615	3581	8465
Pob. 32	33	33	68	173	420	934	2140	4713	10564
Pob. 64	65	65	105	292	652	1362	2919	6423	14013
Pob. 128	129	129	166	470	1122	2224	4522	9247	20009
Pob. 256	257	257	257	834	1872	3645	7323	15114	30159
Pob. 512	513	513	513	1425	3437	6598	12249	23923	47264

Tabla 1: Mediana de evaluaciones según número de individuos y genes

Cada configuración posible ha sido ejecutada en un total de 50 ocasiones para conseguir relevancia estadística de los datos. La *tabla 1* muestra la distribución de la mediana dado un tamaño poblacional con una longitud de genes determinada, mientras que la *tabla 2* muestra la desviación estándar de las mediciones.

	Gen. 2	Gen. 4	Gen. 8	Gen. 16	Gen. 32	Gen. 64	Gen. 128	Gen. 256	Gen. 512
Pob. 2	4.82	8.81	21.64	38.91	114.42	186.20	381.11	964.89	1645.45
Pob. 4	1.88	9.46	17.64	41.37	101.95	198.62	348.19	783.62	1619.76
Pob. 8	2.30	8.74	18.65	35.02	108.04	240.59	489.15	801.71	1829.82
Pob. 16	0	10.77	18.92	38.88	84.59	177.29	407.66	742.89	1652.43
Pob. 32	0	5.80	24.89	57.10	83.48	186.177	512.85	762.74	1581.51
Pob. 64	0	1.19	42.56	67.48	94.42	203.63	384.55	864.76	1522.29
Pob. 128	0	0	53.78	109.58	173.97	238.17	421.41	915.01	1697.61
Pob. 256	0	0	49.36	186.51	222.345	375.45	502.42	923.95	1900.73
Pob. 512	0	0	15.22	292.12	469.42	506.52	881.72	1.352.54	1507.08

Tabla 2: Desviación estándar de evaluaciones según número de individuos y genes

En la *imagen 2* se puede apreciar mejor la tabla anterior, una vez transformada al número de iteraciones.

La variable *y* muestra la mediana de las iteraciones y su desviación típica para cada uno de los casos, mientras que la variable *x* está determinada por el número de genes total que posee cada individuo de la población.

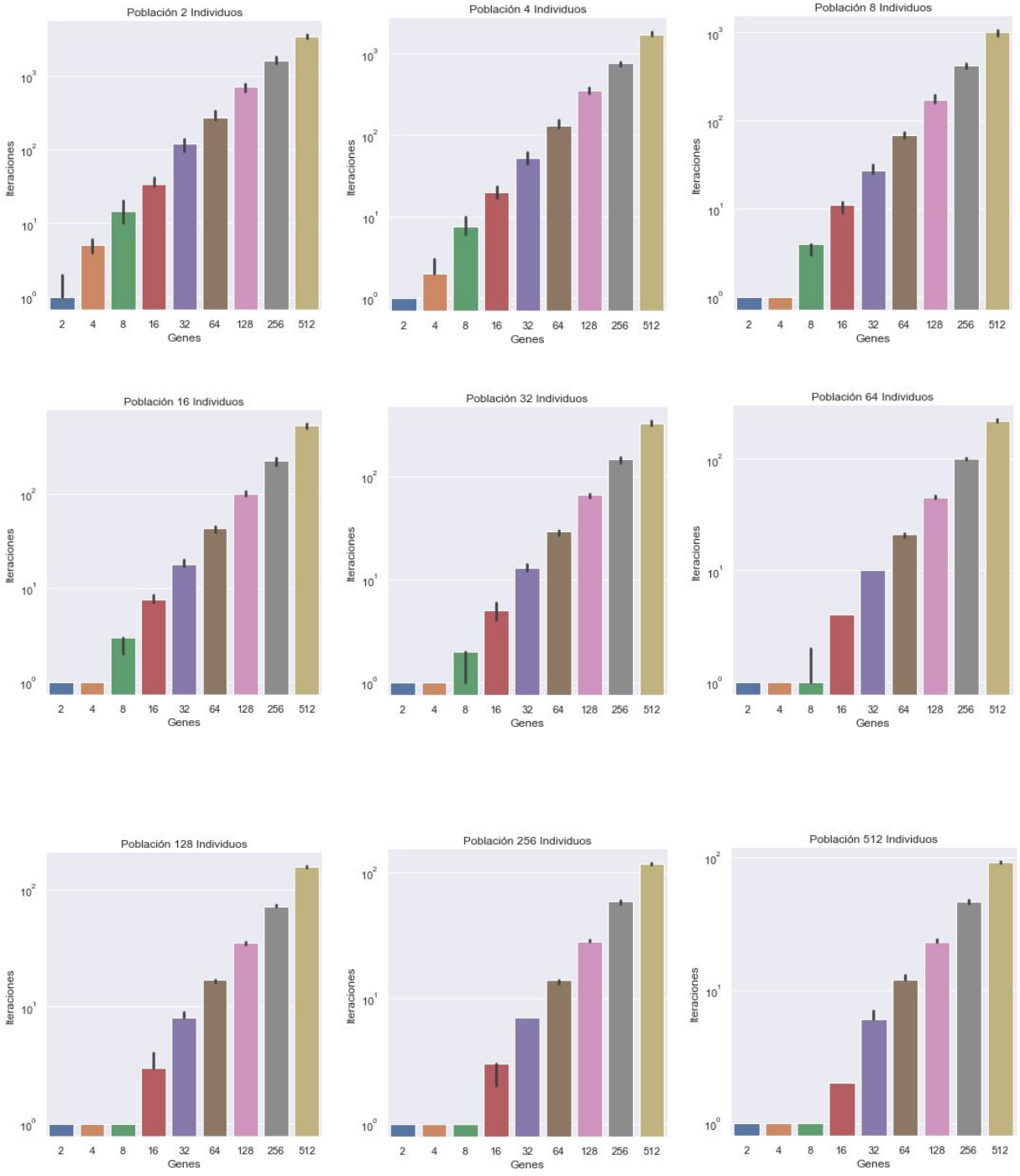

Imagen 2: Distribución de iteraciones según tamaño poblacional y número de genes

Se debe apreciar que tanto una variable, como la otra, están dispuestas en forma logarítmica.

3. Resultados

De estas gráficas se pueden extraer una serie de conclusiones. En primer lugar, como se mencionaba anteriormente, **una población de tamaño menor implica una mayor exploración del espacio de búsqueda**, por ello, en la configuración de 2 individuos con 512 genes se obtiene el número más alto de iteraciones registradas.

En segundo lugar se aprecia que al inicializar aleatoriamente un conjunto poblacional con un problema objetivo de pocos genes se obtienen resoluciones de una única iteración, este evento aumenta para números bajos de genes a medida que aumenta el tamaño de la población. Esto es debido a que el total de individuos inicializado es mayor que el propio espacio de búsqueda, lo que da como consecuencia que, estadísticamente, **al menos uno de esos individuos posea un genotipo completo de valores de 1,** cortando así la simulación tras la evaluación de esa primera generación.

El espacio de búsqueda es de $2^{\text{Num Genes}}$, por lo que para los primeros valores del número de genes se obtienen los siguientes valores:

Gen. 2	Gen. 4	Gen. 8	Gen. 16	Gen. 32
2	16	256	65536	$4.3 \cdot 10^9$

Tabla 3: Tamaño del espacio de búsqueda según número de Genes

Como podemos observar el ratio aumenta exponencialmente, de manera que la situación anteriormente descrita, en la que en la población inicial se obtiene aleatoriamente un candidato con el mayor fitness posible, se da estadísticamente mucho cuando el tamaño de la población supera al espacio de búsqueda.

De hecho, si nos fijamos en las desviaciones estándar reflejadas en la imagen 2 vemos que, a medida que se va dando este caso, van tendiendo a cero, ya que va siendo más y más probable que el problema se solucione en una única iteración.

En tercer lugar recordemos que tanto el eje x como el eje y están dispuestos en escala logarítmica, por lo que a medida que aumenta exponencialmente el número de genes lo hace también el número de iteraciones. Podemos concluir que **hay una relación lineal directa entre el número de iteraciones y el número de genes** así

como una **relación inversa entre el número de iteraciones y el tamaño de la población.**

Por último tenemos la siguiente imagen, que acumula en una sóla todas las gráficas presentadas en la imagen 2, donde se puede ver con mayor claridad los resultados encontrados en este ensayo:

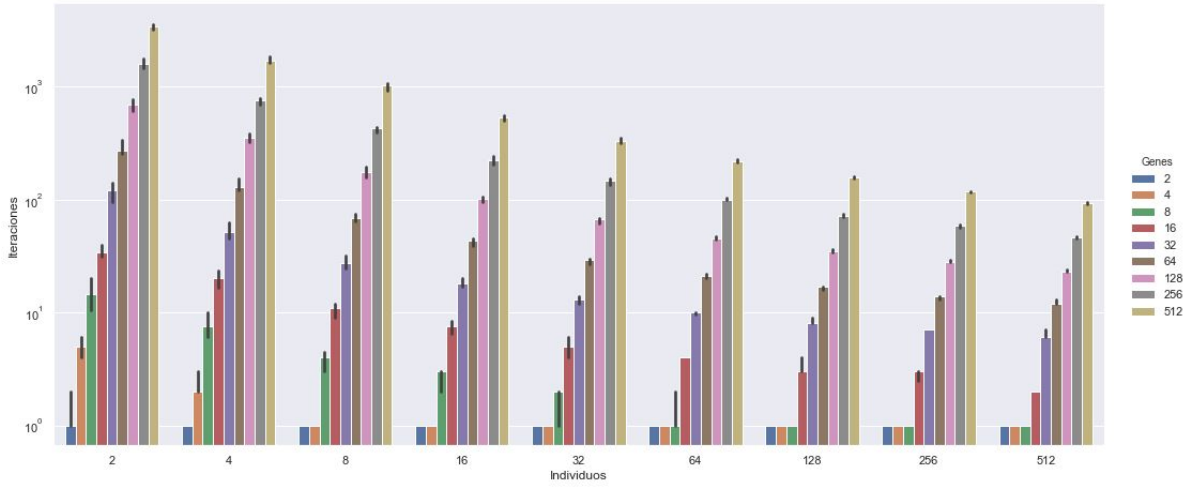

Imagen 3: Comparativa de los Resultados

4. Conclusiones

Para este caso concreto del problema *One Max* se sabe que existe una solución y es conocida. Sin embargo los algoritmos evolutivos pertenecen al conjunto de algoritmos de aproximación dentro de los problemas de optimización combinatoria, para los cuales no siempre se puede determinar que la solución encontrada es la que más minimiza la función objetivo.

Con este tipo de ejemplos podemos visualizar cómo se comportan dichos algoritmos dentro de una casuística controlada, lo que nos ayuda a aplicarlos a problemas mucho más complejos.

5. Referencias

Los resultados obtenidos y presentados en este informe pueden verse recogidos en el siguiente repositorio.