RESOLUCIÓN DE UN PROBLEMA DE REGRESIÓN LINEAL MEDIANTE ALGORITMOS GENÉTICOS

Computación 2023

Carlos Jimeno Miguel

Índice

Introducción y objetivos

Como bien nos indica el enunciado de la práctica, el objetivo es alcanzar una buena comprensión del funcionamiento de los algoritmos genéticos y, al mismo tiempo, desarrollar la capacidad de analizar un problema desde el punto de vista de la Computación Científica. Más específicamente, se trata de analizar el efecto que la variación de los diferentes parámetros y/o técnicas dentro de un algoritmo genético tiene sobre la resolución de un problema concreto.

En nuestro caso se trata de una regresión lineal sobre tres conjuntos de datos distintos, los cuales describiremos a continuación.

Debemos tener en cuenta que para todos los conjuntos de datos se han normalizado los valores en el rango [0, 1] para facilitar la investigación de las diferentes regresiones dentro de nuestro algoritmo genético. Se ha normalizado siguiendo el min-max dado por la siguiente fórmula:

$$\hat{x}_i = \frac{x_i - \min x_i}{\max x_i - \min x_i}$$

Dataset diabetes

En este conjunto de datos observamos las siguientes variables:

Variable	Rango
Edad	[0.9, 15.6]
Déficit	[-29.0, -0.2]
Péptido_C	[3.0, 6.6]

La regresión que se nos presenta para estos datos es determinar la dependencia del péptido C en suero respecto a los factores de edad y déficit de bases en el paciente. La respuesta se expresa en el logaritmo de la concentración del péptido (pmol/ml).

En cuanto a la regresión lineal, observamos que tanto la edad y el déficit corresponderían a las variables independientes, mientras que el péptido sería la variable dependiente de las dos anteriores, nuestro objeto de estudio.

Dataset terremotos

En este conjunto de datos observamos las siguientes variables:

Variable	Rango
Profundidad Focal	[0, 656]
Latitud	[-66.49, 78.15]
Longitud	[-179.96, 180]
Valor Richter	[5.8, 6.9]

Aquí, la regresión que se nos presenta consiste en aproximar la fuerza (en la escala Richter) de un terremoto dadas las tres primeras variables.

En cuanto a la regresión lineal, observamos que la profundidad, latitud y longitud corresponden a las variables independientes, mientras que el valor en la escala Richter es quien depende de las anteriores, el valor a obtener en nuestro experimento.

Dataset laser

En este conjunto de datos observamos las siguientes variables:

Variable	Rango
Input 1	[2.0, 255.0]
Input 2	[2.0, 255.0]
Input 3	[2.0, 255.0]
Input 4	[2.0, 255.0]
Output	[0.0, 255.0]

En este caso, se nos presenta una regresión consistente en un conjunto de valores recogidos de un laser infrarrojo en estado de caos. Este conjunto de datos se ha adaptado para que cada cinco valores, cuatro correspondan a valores de entrada y el último para un valor de salida.

En cuanto a la regresión, los cuatro valores de entrada corresponden a las variables dependientes, por lo que el valor de salida sería la variable dependiente de las anteriores.

Pasos previos al diseño del algoritmo

Antes de comenzar, abordaremos algunos aspectos relacionados con el código para poder asegurar una ejecución correcta de nuestro algoritmo.

En primer lugar, hemos utilizado las librerías de numpy y pandas para poder utilizar sus diferentes métodos. Hemos utilizado numpy para poder utilizar su principal tipo de dato (numpy ndarray), algunas funciones de cálculo sencillas (por ejemplo, la función sum()) y generar tanto distribuciones como números aleatorios.

La librería de pandas únicamente la hemos usado para poder cargar los diferentes datasets y convertirlos en arrays de numpy. De esta parte se encarga la función *carga_datos()*.

Por último, para poder calcular la función de coste para los diferentes cromosomas de la población hemos calculado el valor real de los parámetros correspondientes a cada regresión lineal con la función *obten_solucion(dataset)*, donde pasamos por parámetro el conjunto de datos a estudiar.

Los diferentes parámetros de regresión los hemos calculado gracias a la siguiente fórmula:

$$\beta_n = \frac{\sum\left(x_{n_i} - \bar{x}\right) * (y_i - \bar{y})}{\sum\left(x_{n_i} - \bar{x}\right)^2}$$

$$\beta_0 = \bar{y} - \beta_1\overline{x_1} - \beta_2\overline{x_2} - \cdots - \beta_n\overline{x_n}$$

$$con\ n = n^{\underline{o}}\ variables - 1\ (variable\ dependiente)$$

Diseño del algoritmo genético

En este apartado, explicaremos y justificaremos las decisiones de diseño del propio algoritmo genético planteado.

Codificación de los cromosomas

En primer lugar, al haber normalizado los valores de los diferentes conjuntos de datos en el intervalo [0, 1] para poder mantener sencilla la regresión, hemos optado por una representación real en los cromosomas. Es decir, nuestro cromosoma tendrá la siguiente forma:

$$C_i = (\beta_0, \beta_1, \dots, \beta_n) \; con \; n = nº \; de \; variables - 1$$

Siendo C_i nuestro cromosoma i-ésimo.

Consideramos esta codificación ajustada a los problemas que se nos presentan y representativa a los datos de estos. Al final, cada gen del cromosoma se corresponde con cada uno de los parámetros de la regresión lineal a calcular.

Función de coste

En cuanto a nuestra función de fitness, consideramos a minimizar la fórmula del error cuadrático medio (al que a partir de ahora llamaremos MSE) entre el cromosoma i-ésimo y la solución real.

$$Fit(C_i) = \min\left(MSE(C_i, solución)\right)$$

$$MSE(act, pred) = \frac{1}{n}\sum (act - pred)^2$$

Donde *n* corresponde al número de valores estudiados, *act* al vector de nuestra aproximación y *pred* al vector de la solución exacta.

Utilizamos esta función de coste ya que por cómo hemos codificado los cromosomas, este error nos permite obtener un valor entre $[0, \infty]$ muy representativo de lo alejado que nuestra aproximación se encuentra de la solución exacta, siendo cuanto más cercano a 0 mejor, por esta razón minimizamos nuestra función fitness.

Parámetros de estudio

Hemos definido los siguientes parámetros para ser objeto de estudio:

Tolerancia: Valor del error cuadrático medio observado entre los parámetros de regresión y nuestra aproximación con el cual se da por entendido que el algoritmo converge si el valor obtenido es menor. (Explicación sobre porque usamos este error en el apartado de la función de coste).

Iteraciones máximas: Número máximo de iteraciones necesarias para dar el algoritmo como no convergente.

Desviación estándar: Desviación arbitraria usada para la distribución uniforme de la mutación (explicación en su apartado).

Parámetro alfa: Coeficiente arbitrario (en el rango [0, 1]) usado para determinar el cruzamiento.

Probabilidad de cruce: Probabilidad arbitraria de que ocurra un cruce entre dos progenitores.

Probabilidad de mutación: Probabilidad arbitraria (dentro de su propia función) de que ocurra una mutación en un cromosoma dado.

Número de progenitores: Cantidad de progenitores seleccionados dentro de la población a estudiar.

Selección de progenitores

Hemos decidido implementar dos métodos de selección de progenitores diferentes, para observar como afectan en el proceso de obtener una aproximación. Para añadir el factor de la aleatoriedad, en ambos métodos asignamos probabilidades de poder ser escogidos en los cromosomas.

Método de la ruleta

En este método, la probabilidad de cada uno de los cromosomas se asigna en función de su valor de fitness respecto a la suma de los fitness de todos los cromosomas de la población. Una vez obtenidas las probabilidades, estableceremos intervalos de selección para cada cromosoma en el rango [0, 1] como sigue:

$$\left[\sum P(C_{k-1}), \sum P(C_k)\right]$$

$$Donde, si\ k-1<0 \rightarrow \sum P(C_{k-1}) = 0\ o\ k = tamaño\ población \rightarrow \sum P(C_k) = 1$$

De la asignación de probabilidades se encarga la función *asign_probabilidad(pob_size, fit_array)* donde *pob_size* corresponde al tamaño de la población y *fit_array* a una lista con todos los valores de coste de cada individuo de la población.

Nuestra función *selecciona_prog_ruleta(poblation, model_result, num_prog)* se encarga de para una población *poblation* y una solución exacta *model_result* escoger como progenitores los *num_prog* individuos que más se aproximen al *model_result*.

Método del torneo estocástico

A diferencia del método anterior, aquí no necesitamos conocer el costo de todos los individuos de la población, solo de aquellos que hayan sido escogidos aleatoriamente. En nuestro caso al emplear el torneo estocástico, hemos decidido que sea sin reemplazamiento, para que el algoritmo no se estanque en un mínimo local.

El método que se encarga de este apartado es *torneo_estocastico(actual_poblation, num_contestants, exact_sol)*, donde de una población *actual_poblation*, se escogen aleatoriamente *num_contestants* individuos y se comparan con *exact_sol*. Una vez ordenados los individuos de mejor fitness a peor (de menor a mayor), se genera una probabilidad aleatoria y se le asigna a cada cromosoma una probabilidad siguiendo la siguiente fórmula:

$$P(C_n) = prob * (1 - prob)^n \; con \; n \; \in [0, num\_contestants]$$

Una vez asignadas las probabilidades, la función retorna de manera aleatoria uno de los individuos del torneo de forma aleatoria, teniendo en cuenta las probabilidades anteriores. La función *selecciona_prog_torneo(poblation, model_result, num_prog)* es quien se encarga de llamar a la función anterior *num_prog* veces y guardar los progenitores.

Cruzamiento

Para añadir variabilidad al cruzamiento hemos implementado dos métodos diferentes. Hemos decidido implementar un enfoque aritmético para que los valores obtenidos sean nuevos (siempre dentro del rango de ambos progenitores). Además, este cruzamiento ocurrirá si y solo sí un número aleatorio escogido de una distribución uniforme en el rango [0, 1] es menor que el parámetro probabilidad de cruce establecido. También establecemos que los dos progenitores no pueden ser el mismo individuo, para que la descendencia varíe más.

Aritmético total

En nuestra función *cruzamiento_total(parent_a, parent_b, alpha)* creamos un nuevo cromosoma cuyos genes son obtenidos por la siguiente expresión:

$$New_i = \alpha * x_i + (1 - \alpha) * y_i$$
$$Con \; \alpha = alpha \; \epsilon \; [0,1] \; x_i \; \epsilon \; parent\_a \; e \; y_i \; \epsilon \; parent\_b$$

Simple

Al igual que en el caso anterior, la función *cruzamiento_simple(parent_a, parent_b, alpha)* se rige por la expresión:

$$New_i = (x_i + y_i) * \alpha$$
$$Con \; \alpha = alpha \; \epsilon \; [0,1] \; x_i \; \epsilon \; parent\_a \; e \; y_i \; \epsilon \; parent\_b$$

Mutación

Continuando con este apartado, hemos decidido implementar la mutación no uniforme según una distribución fija. La distribución fija escogida es una distribución gaussiana con media = 0 y desviación típica declarada por el parámetro desviación estándar. Esta mutación siempre ocurre con una probabilidad **probabilidad de mutación**.

La función que se encarga de este apartado es *muta_no_uniforme(cromosome, deviation)*. Aquí, para cada gen de *cromosome* se suma a su valor actual un valor aleatorio de la distribución mencionada antes en el intervalo [0, 1] y se multiplica por *deviation* para ajustarlo según la desviación típica indicada. En el caso de que el valor exceda alguno de los límites del intervalo, se sustituye su valor por el límite más cercano por el que se haya excedido.

Selección de supervivientes

Para mantener sencilla la selección hemos implementado el mismo enfoque que se nos exigió en la práctica número 7. Utilizamos un enfoque elitista en donde para una población Z generamos otros Z descendientes, por lo que de los 2Z individuos seleccionaremos sólo los Z mejores.

La función *select_supers(poblation_fit, childs_fit, population)* une todos los valores de coste de la población actual y los nuevos descendientes. Luego, de la población total *poblation* (población actual + descendientes) ordenada por sus valores de fitness, se escogen los *tamaño(population) – tamaño(childs_fit)* **(2z - z)** primeros individuos. Estos cromosomas coinciden con la función de coste más baja, es decir, los más similares a la solución que buscamos.

Experimentación

Previo a la conclusión, expondremos los diferentes resultados obtenidos para cada set de datos. El conjunto total de los datos obtenidos se encuentra en la hoja de cálculo *experiments.xlsx* adjunta en el archivo .zip de este mismo documento.

Para todos los conjuntos de datos hemos realizado dos tandas de experimentos (cada una 4 veces), variando los parámetros para observar el coste temporal y computacional de alterar estos y de las diferentes combinaciones de métodos de

selección de progenitores y de cruzamiento. El único valor constante es la probabilidad de mutación, la cual mantenemos al valor del 70%.

Diabetes

En este caso, tenemos un conjunto de 43 datos. Al ser una cifra considerablemente pequeña la mayoría de las pruebas de depuración del código se han realizado con este set de datos, así el tiempo de respuesta es relativamente rápido.

El resultado concreto calculado de los parámetros de regresión en este caso es:

$$sol = (0.01111626, 0.33926845, 0.39548496)$$

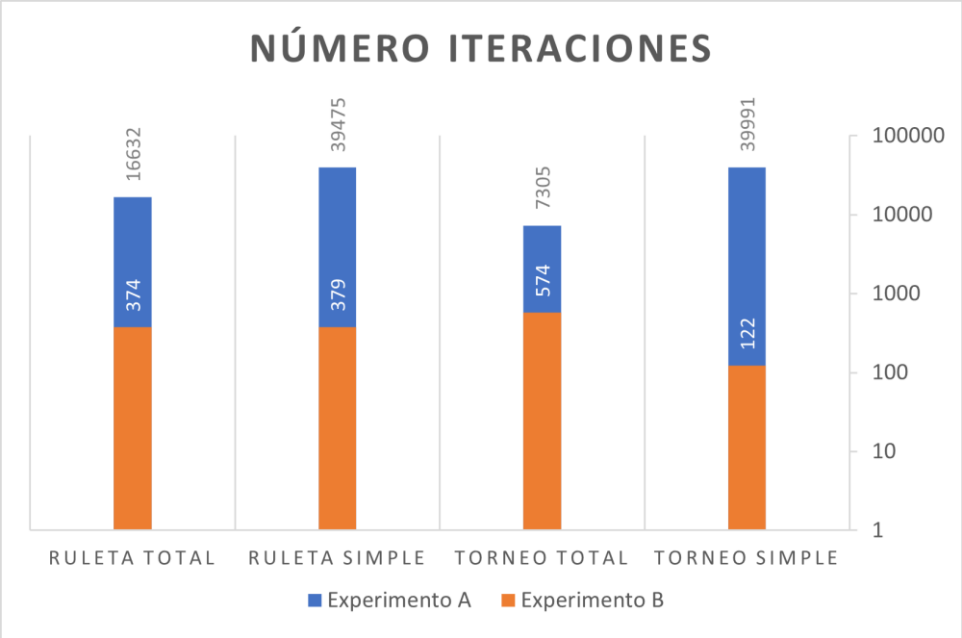
Se han establecido los siguientes parámetros para la primera tanda de pruebas:

- **Tamaño de la población inicial:** 20 individuos.
- **Tolerancia:** MSE < 10^{-7}.
- **Máximo de iteraciones:** 10^7
- **Desviación típica:** 1.5
- **Parámetro alfa:** 0.25
- **Probabilidad de cruzamiento:** 65%
- **Número de progenitores:** 5

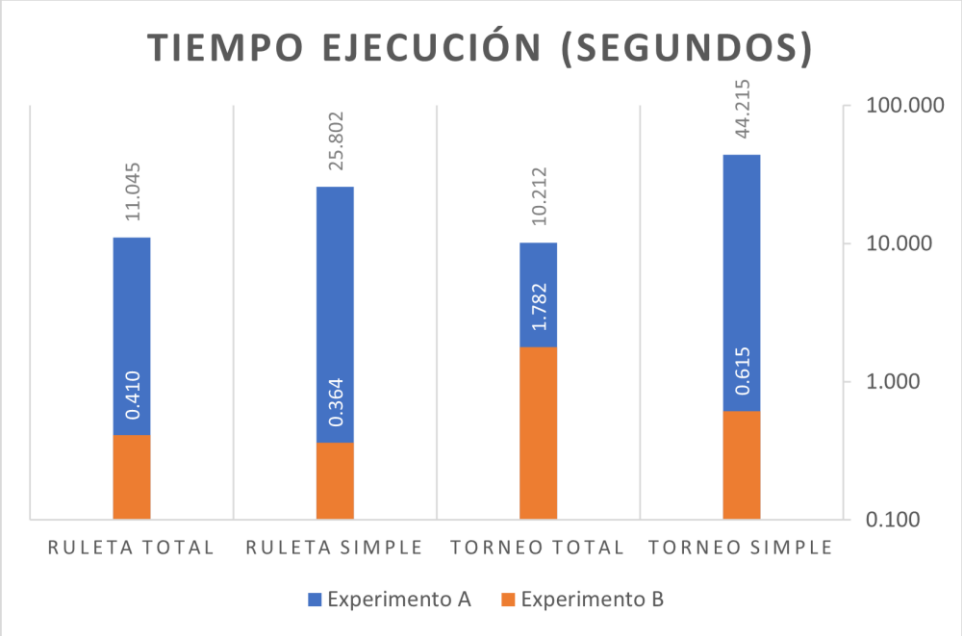
Y para la segunda tanda de pruebas:

- **Tamaño de la población inicial:** 20 individuos.
- **Tolerancia:** MSE < 10^{-5}.
- **Máximo de iteraciones:** 10^4
- **Desviación típica:** 0.75
- **Parámetro alfa:** 0.5
- **Probabilidad de cruzamiento:** 90%
- **Número de progenitores:** 10

En primer lugar, para poder enfrentar las dos pruebas representaremos los diferentes datos de cada par {método de selección de progenitores, método de mutación} con los diferentes parámetros, en $Log_{10}(valor)$, siendo valor bien iteraciones o tiempo de ejecución. A excepción del fitness medio de estos métodos.

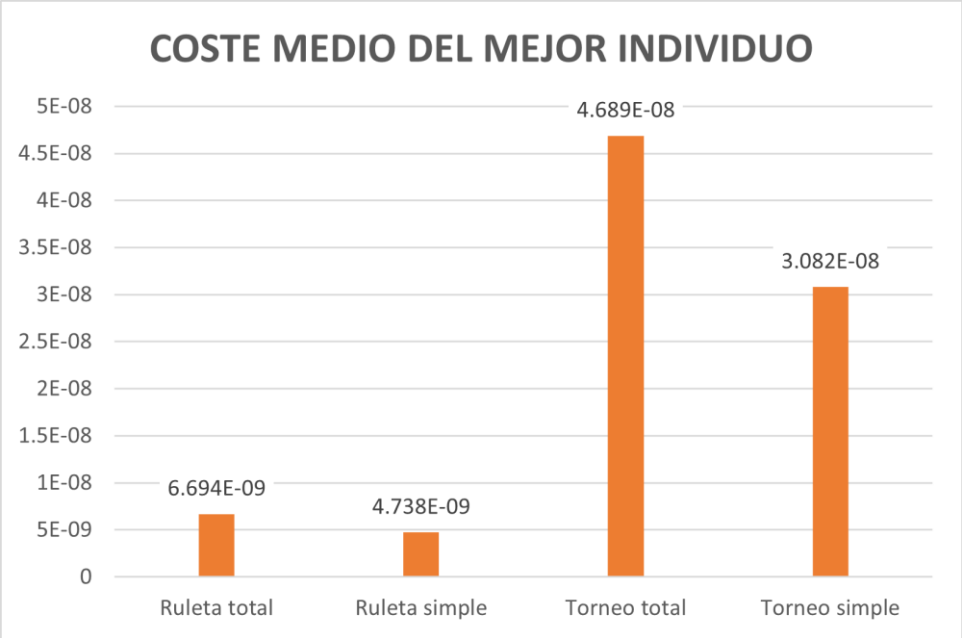


Si estudiamos el coste computacional, siendo más permisivos con la tolerancia observamos un bajón drástico en el número total de iteraciones. En cuanto a las diferentes implementaciones, vemos que el método de la ruleta tiene un mayor coste, seguramente debido al cálculo del ajuste para la población entera, mientras que en el torneo solo se calcula para una muestra. Sin embargo, con una tolerancia más restrictiva, un valor alto en la probabilidad de cruzamiento provoca que el algoritmo converja antes. En el otro caso, aunque incrementemos la probabilidad de cruzamiento si la tolerancia es mayor el cruzamiento simple provee mejores resultados.



En cuanto al coste computacional se observa que, a mayor tolerancia, menor tiempo de ejecución.

Aun así, si nos interesa tener una aproximación más exacta, la mejor opción sería el torneo estocástico con un cruzamiento total entre progenitores. La principal causa que podemos intuir es la misma que en el coste computacional. Si observamos los parámetros, al seleccionar un número mayor de progenitores y tener una desviación menor, podemos inferir que el espacio de soluciones se recorre en un menor tiempo. Como la selección de supervivientes se hace de manera elitista seguramente se converja al mínimo de manera ágil.



Por último, si analizamos como evoluciona el coste del mejor individuo de la población en ambos experimentos, vemos que la solución más próxima al valor real es la que propone la ruleta simple. Es de apreciar que en este conjunto de datos el método de selección de la ruleta presenta aproximaciones de un orden más cercanos al cero que el método del torneo.

Terremotos

En este caso, tenemos un conjunto de 2178 datos. Como esta cifra es notable, hemos realizado varias pruebas hasta obtener un valor bastante acotado para la tolerancia. Ajustar aún más dicho valor provoca que nuestro algoritmo no converja (siendo notificado en el programa el mensaje y el tiempo de ejecución transcurrido).

El resultado concreto calculado de los parámetros de regresión en este caso es:

$$sol = (0.13941657, -0.00984116, 0.05748688, -0.01044945)$$

Si nos fijamos, se presentan valores negativos en el cromosoma solución. Como los genes de nuestros cromosomas están evaluados en el intervalo [0, 1] el valor más próximo al que podemos aspirar es el cero (de hecho, en los cromosomas propuestos como aproximación aparece). Por esta razón no podemos establecer una cota muy restringida, ya que no es capaz de devolver un valor de la función de coste aún más cercano a cero.

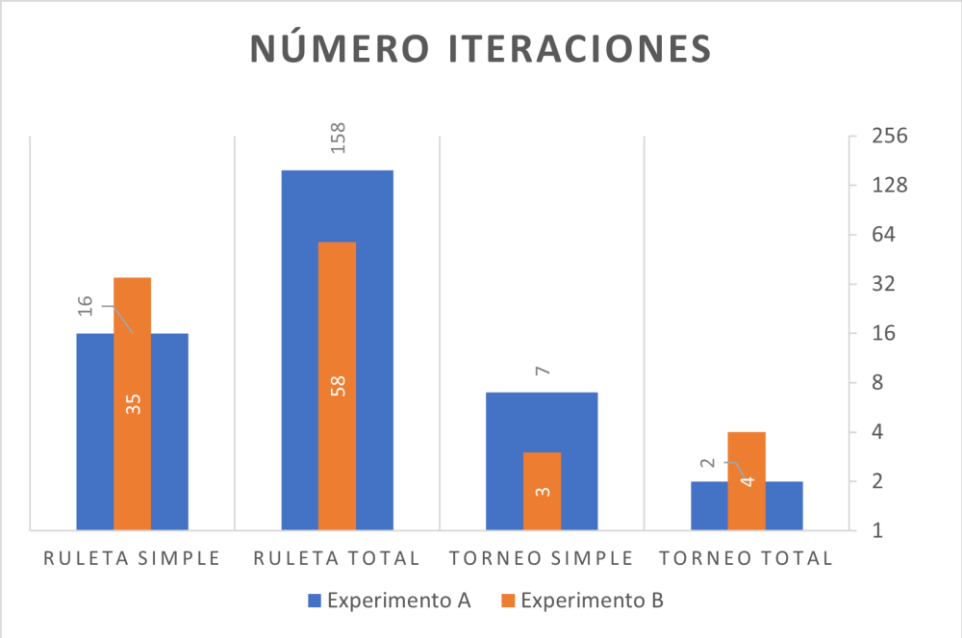
Se han establecido los siguientes parámetros para la primera tanda de pruebas:

- **Tamaño de la población inicial:** 1000 individuos.
- **Tolerancia:** MSE < 10^{-4}.
- **Máximo de iteraciones:** 10^5
- **Desviación típica:** 1.5
- **Parámetro alfa:** 0.25
- **Probabilidad de cruzamiento:** 65%
- **Número de progenitores:** 250

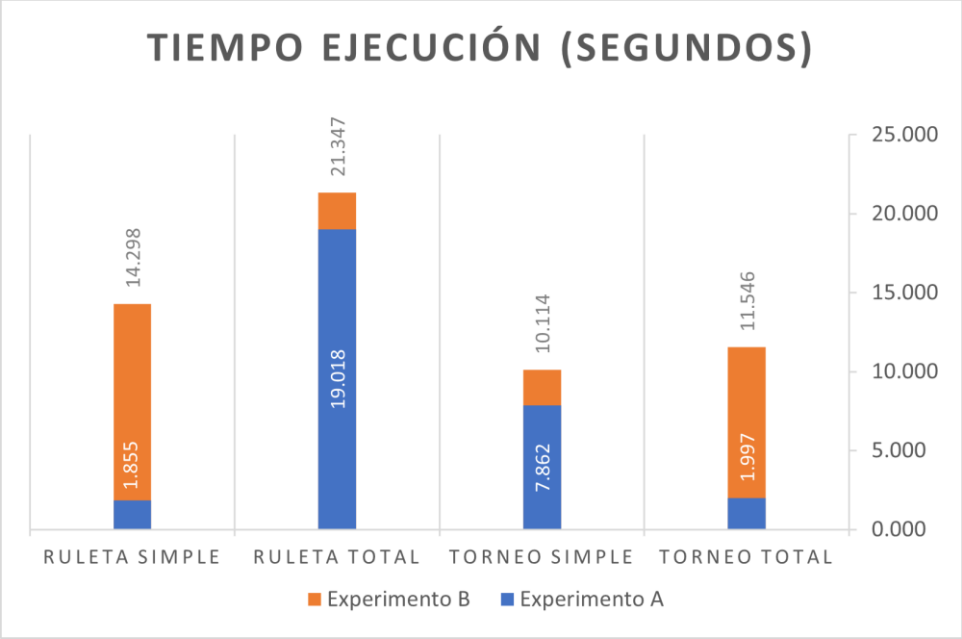
Se han establecido los siguientes parámetros para la segunda tanda de pruebas:

- **Tamaño de la población inicial:** 2000 individuos.
- **Tolerancia:** MSE < $8 * 10^{-5}$.
- **Máximo de iteraciones:** 10^4
- **Desviación típica:** 0.75
- **Parámetro alfa:** 0.5
- **Probabilidad de cruzamiento:** 90%
- **Número de progenitores:** 500

En este caso, para poder enfrentar el coste computacional de ambos experimentos, hemos decidido representar los datos de la gráfica en $Log_2(iteraciones)$.

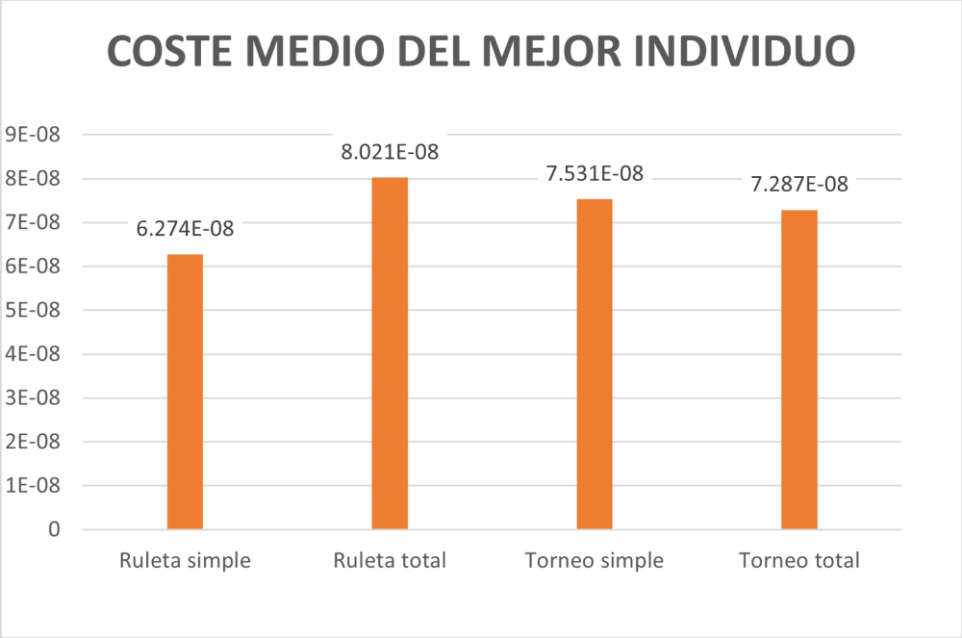


Para este segundo conjunto de datos, apreciamos que siempre el método del torneo es el mejor mecanismo de selección de progenitores. Dependiendo de la probabilidad de cruzamiento y el número de progenitores escogidos, vemos que el cruzamiento aritmético total es mejor cuando ambos son menores y el simple cuando ambos son mayores.



En cuanto al coste temporal, vemos que aumentar el tamaño de la población inicial, el número de progenitores escogidos, restringir más la tolerancia, disminuir la desviación en la mutación y hacer más probable el cruce, provoca en general, un aumento bastante notable en el tiempo de ejecución del algoritmo.

Por el otro lado, si todos estos parámetros los definimos de la manera contraria, el método de la ruleta con cruzamiento aritmético simple y el del torneo con cruzamiento aritmético total son las mejores opciones.



Para un mayor volumen de datos apreciamos que la diferencia entre métodos es casi insignificante. Si que obtenemos el mejor resultado con el método de la ruleta con cruzamiento simple. Aun así, al tener el mismo orden en el fitness de todos los individuos, la mejora de un método respecto a otros no es muy notable.

Laser

Al igual que en el caso anterior, al disponer de un conjunto de 992 datos, hemos realizado varias pruebas para acotar una tolerancia lo suficientemente ajustada. Excedernos en este valor provoca que el programa devuelva la respuesta de no convergencia y el tiempo de ejecución transcurrido.

El resultado concreto calculado de los parámetros de regresión en este caso es:

$$sol = (0.12638563, 0.77068876, -0.63004538, -0.19529967, 0.52759662)$$

Al igual que antes, observamos valores negativos, por lo que la mejor aproximación en los genes tercero y cuarto es el cero. Como la función de fitness no es capaz de encontrar un valor para el MSE más cercano a cero, acotar más la tolerancia provoca que el algoritmo no pueda converger.

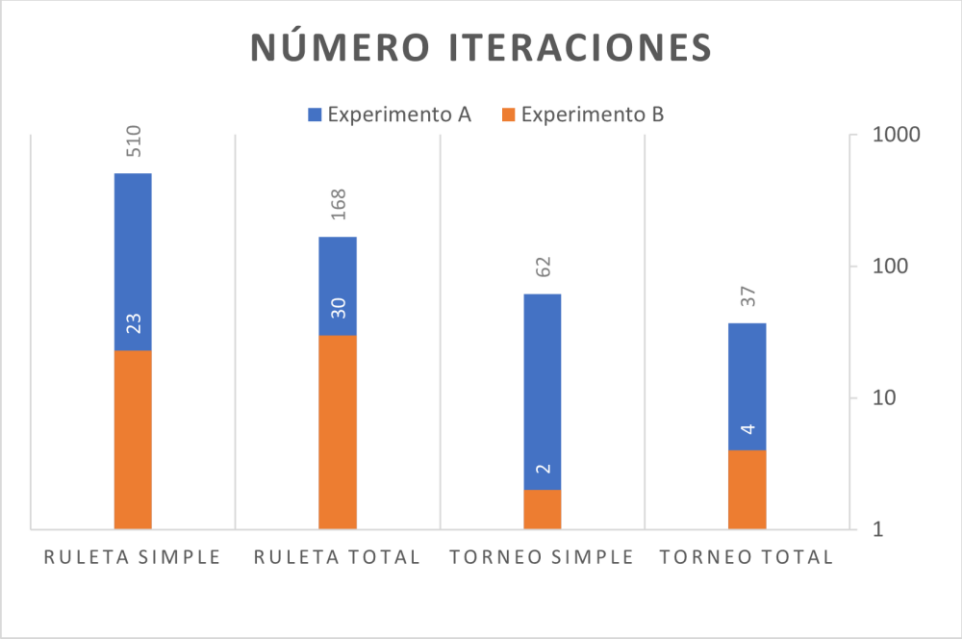
Se han establecido los siguientes parámetros para la primera tanda de pruebas:

- **Tamaño de la población inicial:** 100 individuos.
- **Tolerancia:** MSE < $8.75 * 10^{-2}$.
- **Máximo de iteraciones:** 10^4
- **Desviación típica:** 1.5
- **Parámetro alfa:** 0.25
- **Probabilidad de cruzamiento:** 65%
- **Número de progenitores:** 50

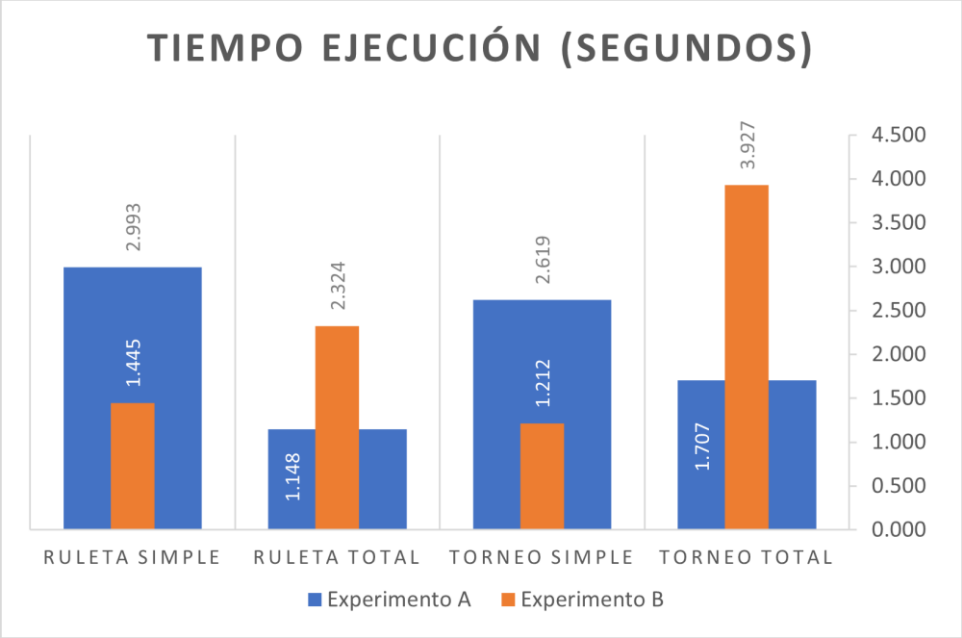
Se han establecido los siguientes parámetros para la segunda tanda de pruebas:

- **Tamaño de la población inicial:** 500 individuos.
- **Tolerancia:** MSE < $8.75 * 10^{-2}$.
- **Máximo de iteraciones:** 10^4
- **Desviación típica:** 0.75
- **Parámetro alfa:** 0.5
- **Probabilidad de cruzamiento:** 90%
- **Número de progenitores:** 250

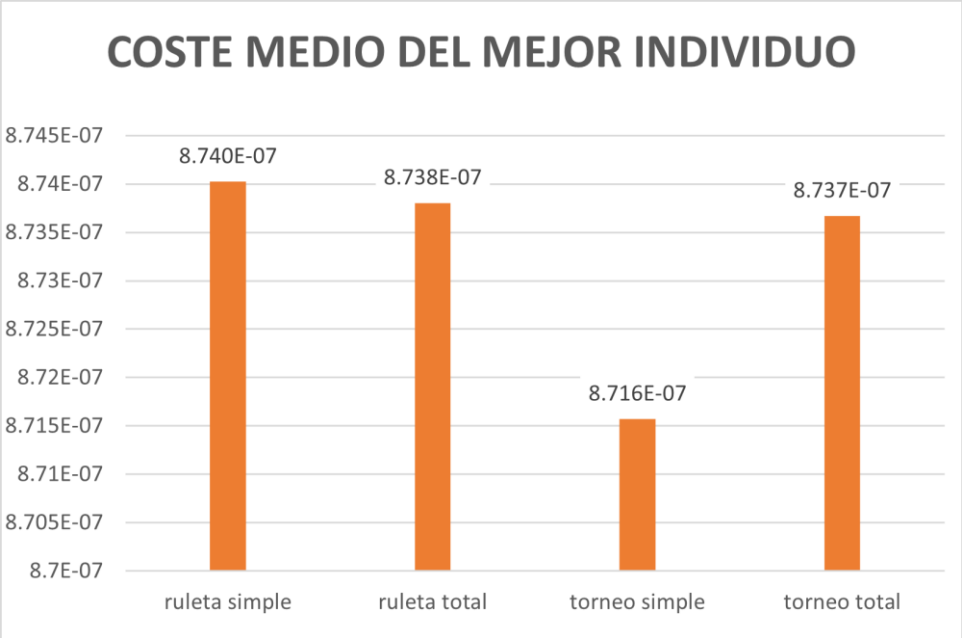
Para nuestro último conjunto de datos, el único caso donde representamos los valores en función de $Log_{10}(iteraciones)$ para estudiar el coste computacional de ambos experimentos.



En el primer experimento observamos que, a una menor población inicial, menor número de progenitores, mayor desviación y menor probabilidad de cruzamiento provoca que el coste aumente considerablemente. Si aumentamos los parámetros (excepto la desviación) el número de iteraciones se reduce mucho. En ambos casos cualquier método de cruzamiento con la selección por medio del torneo estocástico presenta los mejores resultados.



Sin embargo, en cuanto al coste temporal, el método de cruzamiento total bien con el método de selección del torneo o de la ruleta, presenta el menor tiempo cuando la cantidad de datos no es excesiva. Si aumentamos los datos a procesar las tornas cambian. Los mejores resultados los obtenemos en el cruzamiento aritmético simple con cualquiera de los dos métodos de selección de progenitores.



Al igual que en el conjunto de datos anterior, al tener un volumen de información considerable los diferentes costes obtenidos en cada método no difieren demasiado, todos tienen el mismo orden. Sin embargo, el mejor valor, el más pequeño de todos y por tanto la aproximación mas acertada, lo presenta el método de selección mediante torneo estocástico con cruzamiento aritmético simple.

Conclusiones

Para cualquier volumen de datos la mejor implementación para el cruzamiento en el algoritmo es el aritmético simple, presentando siempre el menor valor de coste.

En cuanto al método de selección de progenitores, si la cantidad de datos a procesar no es elevada, la mejor opción es el método de la ruleta. Sin embargo, si el volumen crece considerablemente habría que probar ambos y ver cual proporciona una aproximación más ajustada, ya que en el tercer conjunto de datos parece ser que el torneo estocástico funciona mejor, pero para el segundo caso el método de la ruleta da mejores resultados. Debemos tener en cuenta que la diferencia del valor de la función de fitness del mejor individuo en ambos casos no presenta una diferencia drástica respecto a los otros métodos, el orden del valor sigue siendo el mismo.

Si lo que nos interesa no es acotar de la mejor forma posible el resultado sino el tiempo y coste de ejecución. Dentro de estos requisitos si el volumen de datos es bajo, los resultados siempre van a ser bastante mejores cuando iniciemos la población con más individuos y seleccionemos una cantidad importante de

progenitores. Aquí el método de selección por ruleta y el cruzamiento aritmético simple sugieren una opción interesante

Defendemos esta opción por los siguientes motivos. En primer lugar, descartamos el método del torneo con el cruzamiento sencillo, ya que con las condiciones anteriores siempre presenta el peor resultado. Si ahora estudiamos el crecimiento del tiempo para cada par de métodos obtenemos los siguientes resultados:

$$\Delta T_{\frac{ruleta\ total}{ruleta\ simple}} = \frac{0.410}{0.364} = 1.1264 \rightarrow Crece\ un\ 12{,}64\%$$

$$\Delta T_{\frac{torneo\ simple}{ruleta\ simple}} = \frac{0.615}{0.364} = 1.6896 \rightarrow Crece\ un\ 68{,}96\%$$

$$\Delta T_{\frac{torneo\ simple}{ruleta\ total}} = \frac{0.615}{0.410} = 1.5 \rightarrow Crece\ un\ 50\%$$

Si realizamos las mismas apreciaciones en estos pares para el número de iteraciones:

$$\Delta Iter_{\frac{ruleta\ simple}{ruleta\ total}} = \frac{379}{374} = 1.0134 \rightarrow Crece\ un\ 1{,}34\%$$

$$\Delta Iter_{\frac{ruleta\ simple}{torneo\ simple}} = \frac{379}{122} = 3.1066 \rightarrow Crece\ un\ 210{,}66\%$$

$$\Delta Iter_{\frac{ruleta\ total}{torneo\ simple}} = \frac{374}{122} = 3.0656 \rightarrow Crece\ un\ 206{,}56\%$$

Como podemos ver, el menor incremento siempre lo presenta la ruleta con el cruzamiento total respecto al simple. Como el tiempo de ejecución es bajo y el impacto de escoger este método frente al total significan solo un 1.34% más de iteraciones, consideramos que es un crecimiento despreciable frente al buen tiempo de respuesta que ofrece.

Para los requisitos especificados al principio si el volumen de datos es mayor, deberíamos estudiar cuanto mayor es ese volumen. Si el volumen supera los 1000 datos, es mejor inicializar una población más reducida, escoger menos progenitores y ser mas tolerantes con la aproximación. Si no supera esta cifra, obtendremos mejores resultados si la población inicial es mayor, escogemos más progenitores y establecemos una alta probabilidad de cruce.

Para el segundo caso con tan solo analizar los resultados obtenemos que la mejor implementación corresponde al método del torneo estocástico con cruzamiento aritmético simple.

Sin embargo, si queremos obtener la mejor aproximación con un volumen de datos aún mayor tenemos que estudiar más detenidamente los resultados.

En primer lugar, podemos descartar directamente el método de la ruleta con cruzamiento total ya que presenta unos resultados excesivamente altos y malos. Siendo críticos, el método del torneo con cruzamiento simple también lo desecharemos ya que posee un tiempo de ejecución alto para las pocas iteraciones que realiza, además, el método de la ruleta con ese mismo cruzamiento realiza más del doble de iteraciones y tarda casi cuatro veces menos. No es un buen resultado.

Por descarte resta estudiar el crecimiento del tiempo de ejecución y las iteraciones de las otras dos combinaciones:

$$\Delta T_{\frac{torneo\ total}{ruleta\ simple}} = \frac{1.997}{1.855} = 1.0765 \rightarrow Crece\ un\ 7{,}65\%$$

$$\Delta Iter_{\frac{ruleta\ simple}{torneo\ total}} = \frac{16}{2} = 8 \rightarrow Crece\ un\ 700\%$$

El crecimiento de las iteraciones es muy agudo en la ruleta con cruzamiento simple, sin embargo, el valor absoluto de las mismas es muy bajo (16 y 2) y como el tiempo de respuesta es el menor obtenido, esta sería la mejor técnica.

Exposición de los resultados

En resumen. Para nuestro algoritmo genético tenemos diferentes casos en los que poder hacer regresión lineal:

- Si el conjunto de datos es pequeño.
- Si el conjunto de datos es mayor pero no supera los 1000 valores.
- Si el conjunto de datos es considerablemente superior a 1000 valores.

Además, podemos evaluar que opción es mejor según qué objetivo tengamos:

- Obtener la aproximación más exacta posible.
- Obtener una aproximación en un costo temporal y computacional razonable.

Si evaluamos el primer objetivo, para cualquiera de los tres casos usaremos un cruzamiento aritmético simple. Para el primer caso utilizaremos el método de selección mediante la ruleta, y para los otros dos casos hay que estudiar el problema detenidamente. Ocasionalmente para el segundo caso funciona mejor el torneo estocástico y para el tercero la ruleta (aunque la diferencia no es notable).

Si evaluamos el segundo objetivo, para el primer y segundo caso deberemos inicializar una población relativamente grande y seleccionar una cantidad considerable de progenitores. Aquí el método de selección por ruleta junto al cruzamiento simple es el recomendado. También lo es para el tercer caso, solo que debemos establecer las condiciones contrarias. Para el caso que nos queda, además debemos establecer una probabilidad de cruce alta y el mejor resultado nos lo proporciona el método del torneo con un cruzamiento simple.