Universidad
Carlos III de Madrid

Grado en Ingeniería Informática

Curso 2021-2022

**Algoritmos Genéticos y Evolutivos**

# Práctica 2
# Calibración de motores automática mediante Estrategias Evolutivas

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

# Índice

## Índice de figuras

## Índice de tablas

## Introducción

Este proyecto trata sobre encontrar una aproximación mediante estrategias evolutivas al problema de calibrar motores de un brazo robótico utilizando solo un láser, una cámara y un objeto de referencia. El robot deberá ser preciso en sus movimientos, dado que será utilizado para realizar soldaduras de alta precisión.

Para comenzar se trabajará sobre un brazo compuesto solo por 4 motores para podernos acercar al problema de una manera más gradual, en cuanto se haya comprobado que es factible esta simplificación del problema se pasara a un caso más realista con un brazo de 10 motores.

## Codificación y Función de fitness

Al emplear estrategias evolutivas la codificación de los individuos de este problema consistirá en un vector de codificación compuesto por 4 o 10 números reales (según el número de motores) y un vector de varianzas de la misma longitud que el de codificación.

Los valores del vector de codificación estarán centrados en 0 representando que no se ha realizado ningún giro y podrán rotar tanto como quieran en sentido positivo como negativo, aunque llegado un punto girar mucho en un sentido equivale a movimientos más cortos. Lo ideal es que los valores se muevan entre -180 y 180 grados representando un giro completo de 360 grados, aunque no se restringirán los valores que puede tomar. En la Figura 1 se puede ver un diagrama de cómo es el sistema.

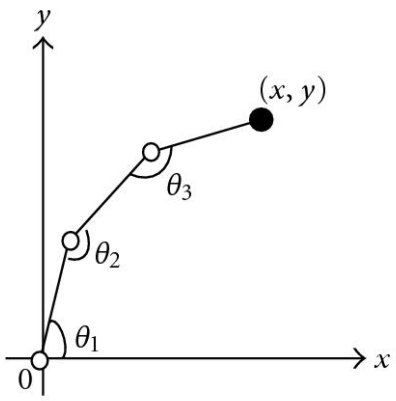


Figura 1: Diagrama brazo robótico

En cuanto a la función de fitness que se empleará para saber si una solución es buena o no, esta tendrá en cuenta el error que comente el soldador en un simulador, según como de dispersos y desviados sean con respecto al centro dará un valor mayor cuanto peor lo haga y menor cuanto más preciso sea. Por lo que el objetivo del problema es minimizar la función de fitness.

Para conocer el error para cada una se las soluciones se hará una llamada get a un servicio que se encuentra en un servidor de la universidad, http://memento.evannai.inf.uc3m.es/age/robot4?, seguido de cX=GradosDeX para cada uno de los 4 motores y separados por un &, en el caso de los 10 motores será igual, aunque con otro servicio http://memento.evannai.inf.uc3m.es/age/robot10? y 10 valores de grados.

## Programación Estrategia Evolutiva (1+1)

Para este proyecto entre las posibles implementaciones de las estrategias evolutivas se ha elegido la (1+1)-EE y a continuación se describirán las características de esta.

El programa se ha desarrollado en el lenguaje Python, el procesamiento principal del programado se encuentra en la función *EE1plus1(c, cycles, test_iteration)*, donde recibirá como parámetros el valor por el que se multiplicaran las varianzas, el número de generaciones máximas del programa y la iteración en la que se encuentra el programa (se incluye para realizar las pruebas de ejecutar múltiples veces).

El proceso que sigue esta función es el siguiente:

1. Generar el individuo inicial.
2. Evaluar el individuo generado de la población inicial.
3. Repetir hasta cumplir el criterio de convergencia:
   *a*) Generar una nueva solución a partir del único individuo de la población, mutando el vector de codificación del descendiente.
   *b*) Evaluar el individuo generado.
   *c*) Eliminar el individuo cuyo valor de adecuación sea mayor, tratamos de minimizar el fitness.
   *d*) Si el individuo que queda es el nuevo, se aumenta la frecuencia de éxitos y si no se disminuye.
   *e*) Mutar el vector de varianzas del individuo elegido de acuerdo con la regla 1/5.

Además de lo relacionado con la generación de individuos, también se ha incluido código que nos permita almacenar la salida de los modelos para poderlos evaluar.

### Operadores genéticos

#### Población inicial

El individuo inicial es generado aleatoriamente, para su vector de codificación se sigue una gaussiana centrada en 0 y con desviación 100, de esta manera los valores serán tanto positivos como negativos y mayoritariamente centrados alrededor de 0, por otro lado el vector de varianzas se genera mediante valores reales positivos entre 5 y 20, aunque se realizarán pruebas con valores mayores.

Este proceso es realizado por la función *initial_individual()*, devolviendo un individuo en forma de matriz con dimensiones NumeroDeMotores x 2.

#### Mutación

Para los algoritmos evolutivos este operador se realiza en dos partes una para el vector de codificación y otra para el de varianzas.

Para el vector de codificación, el nuevo individuo tendrá como valores los del progenitor más un valor de la normal según la varianza de ese valor para el padre, es decir $x_{hijo} = x_{padre} + N_0(\sigma_{padre})$, esta operación se repite para todos los valores del vector.

En cuanto al vector de varianzas, se realizará sobre el individuo seleccionado para la siguiente generación, y consiste en modificar sus varianzas en función de la regla 1/5, que se guía por el porcentaje de mejoras de las últimas generaciones.

Esta regla se aplicará teniendo en cuenta las últimas generaciones, las variaciones vendrán dadas por:

- Si el número de mejoras es superior al 20 %, se aumenta la varianza $\frac{\sigma}{c}$.
- Si el número de mejoras es inferior al 20 %, se reduce la varianza $\sigma \cdot c$.
- Si el número de mejoras es igual al 20 %, se mantiene $\sigma$.

Lo que dice la regla es que si mejora con frecuencia es que todavía estamos lejos de la solución óptima, por lo que aumentaremos la varianza, y si no mejora es que estamos cerca, entonces nos movemos poco a poco.

Esta funcionalidad se recoge en *mutate_codification(individual)* para el vector de codificación, muta al progenitor y devuelve al sucesor, y en *individual_next_generation(individual, son, improvements_counter, iteration, c)* para el vector de varianzas, se encuentran separadas porque hasta que no se conoce el individuo de la siguiente generación no se mutan las varianzas.

#### Inserción y Remplazo

Al tratarse del tipo (1+1) la población está formada por un solo individuo y también se generará un solo individuo, por lo que entre el individuo de la población y el nuevo se elegirá el mejor, en este caso el de menor fitness.

Cuando se elija al descendiente se considera una mejora y se tendrá en cuanta en la frecuencia de mejoras, si no, se considera que no ha mejorado. Tras elegir el individuo que formará la población se mutará su vector de varianzas como se ha definido antes.

La función que recoge la inserción y remplazo, además de la mutación de las varianzas es la que se mencionó antes, *individual_next_generation(individual, son, improvements_counter, iteration, c)*, que evalúa ambos individuos, elige el mejor, actualiza el contador de mejoras y aplica la mutación en función del parámetro c que se le pasa. La función devuelve el individuo elegido, su fitness y el contador de mejoras actualizado.

### Aproximación 4 motores

Comenzaremos realizando pruebas en la versión simplificada del problema, en el que solo se consideraran 4 motores para el brazo soldador.

Estos primeros modelos que se generaran consistirán en la variación del valor c, el que determina cuanto cambiara la varianza de los individuos. Es conocido que el valor debe ser inferior a 1 y que el recomendable es 0,82, pero en busca de mejores resultados se ha probado además con 0,98, 0,92, 0,72 y 0,62. No se han variado más estos valores, porque la selección de valores más bajos provocaría que los valores de varianza se movieran muy rápido.

Todos los modelos se han ejecutado durante 5000 generaciones, que son 10000 evaluaciones, y se han realizado 10 ejecuciones para cada modelo para evitar el sesgo de la aleatoriedad a la hora de inicializar el modelo.

Los modelos han sido nombrados siguiendo el siguiente patrón: evolution_strategy_XX_YY_A-B_C

- **XX:** Indica el número de motores que se consideran y para los que se ajustaran los ángulos.
- **YY:** Indica el valor de c, el cual determina como se modificarán las varianzas.
- **A-B:** Indica el rango de valores entre los que se generan aleatoriamente las varianzas.
- **C:** Indica el número de generaciones que tiene en cuanta para medir la frecuencia de mejora.

En la Figura 2 y Tabla 1 se pueden ver los resultados de fitness obtenidos en las **10 ejecuciones para los 5 modelos**, junto a la media de cada modelo que es la que utilizaremos para comparar.

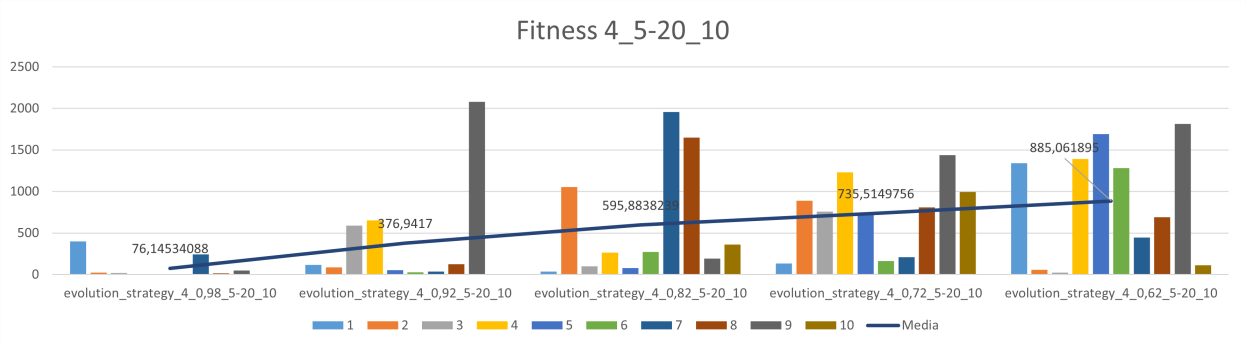


Figura 2: Valores de Fitness: 4_5-20_10

| Modelo | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | Media |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **4_0,98_5-20_10** | 399,8967664 | 21,88903367 | 20,89403404 | 1,989918114 | 2,984877171 | **0,994959057** | 243,1577278 | 13,92939149 | 49,74695184 | 5,969749305 | **76,14534088** |
| **4_0,92_5-20_10** | 115,4108682 | 86,56039686 | 587,9440061 | 654,4753316 | 51,73723573 | 25,86868199 | 35,81824234 | 125,3603064 | 2081,267141 | 4,974790248 | 376,9417 |
| **4_0,82_5-20_10** | 34,8234203 | 1053,037537 | 98,49807057 | 263,6453657 | 79,59564625 | 270,6192328 | 1955,329393 | 1649,132003 | 193,0126789 | 361,1448922 | 595,8838239 |
| **4_0,72_5-20_10** | 134,6082641 | 887,2371193 | 758,3690516 | 1232,534402 | 731,1155627 | 161,1833393 | 208,9351831 | 808,6563643 | 1438,833559 | 993,6769101 | 735,5149756 |
| **4_0,62_5-20_10** | 1342,19067 | 58,71220968 | 24,87415781 | 1391,691525 | 1691,348176 | 1280,418402 | 443,7601042 | 692,0338572 | 1812,167475 | 113,4223726 | 885,061895 |

Tabla 1: Fitness iniciales 4 motores

Se puede ver que los mejores resultados se han obtenido para una c de 0,98 y que según se van reduciendo progresivamente, las evaluaciones van empeorando. En la Figura 2 solo se ve el valor para la media, pero en la Tabla 1 se puede apreciar que los valores normalmente son bajos y los que la suben son unas pocas ejecuciones que dan valores más altos.

Aun con esto, el mejor resultado se ha obtenido en la que da mejor media, que es **c de 0,98** y una ventana de **varianzas entre 5 y 20**. El resultado que nos proporciona el modelo es de **0,994959057 de fitness**, que le lleva **1998 evaluaciones**, que para el caso son 999 generaciones de las 5000 dadas.

**De media este modelo requiere 2670 evaluaciones** para alcanzar su mejor resultado, que es **de media 76,14534088 de fitness**.

En la siguiente página se puede ver en la Figura 3 el progreso del fitness y en la Figura 4 el de los ángulos a lo largo de las 1998 evaluaciones. Al comienzo como los valores de partida son aleatorios el error es muy alto y rápidamente decrece hasta converger, en este caso en torno al 1. En cuanto a los grados se aprecia un comportamiento parecido, se van moviéndose rápidamente hasta llegar a un valor en el que converge en este caso los valores son:

- **Motor 1:** 9,128498158310796
- **Motor 2:** 5,9876539870877865
- **Motor 3:** 3,6666659795960044
- **Motor 4:** 1,7654321185803394

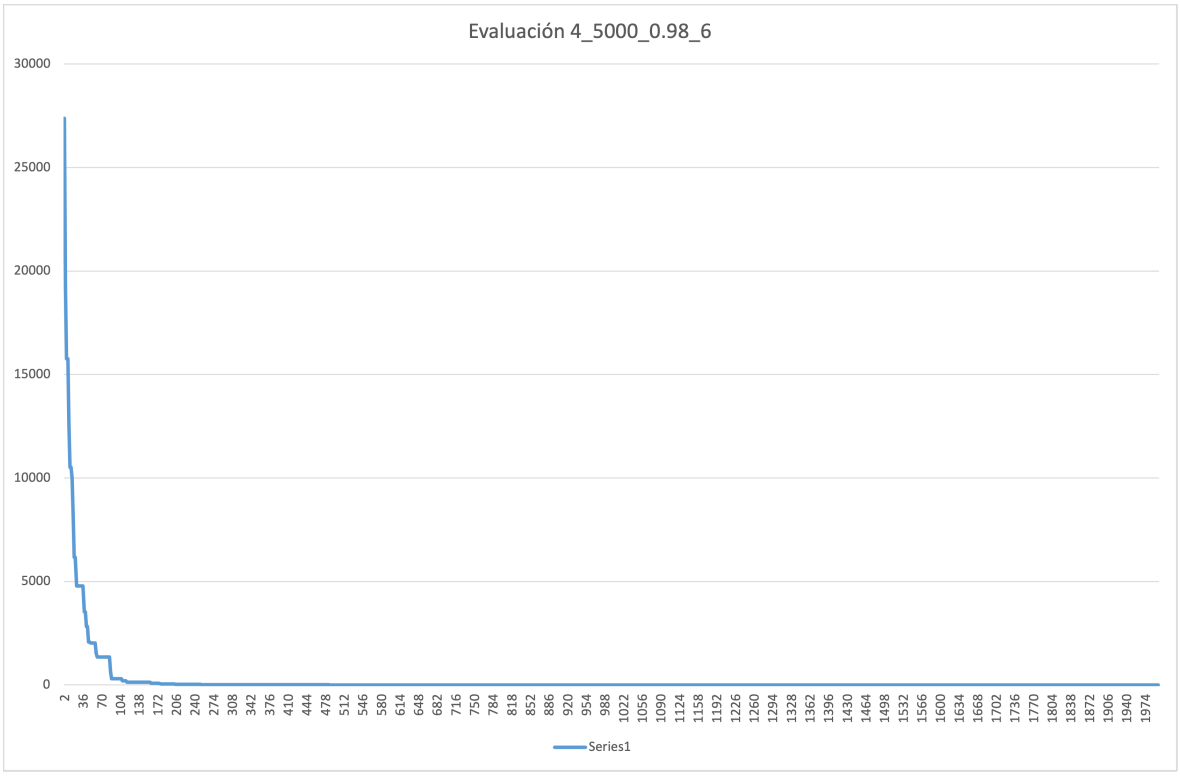


Figura 3: Fitness vs. Evaluaciones: 4_5000_0.98_6

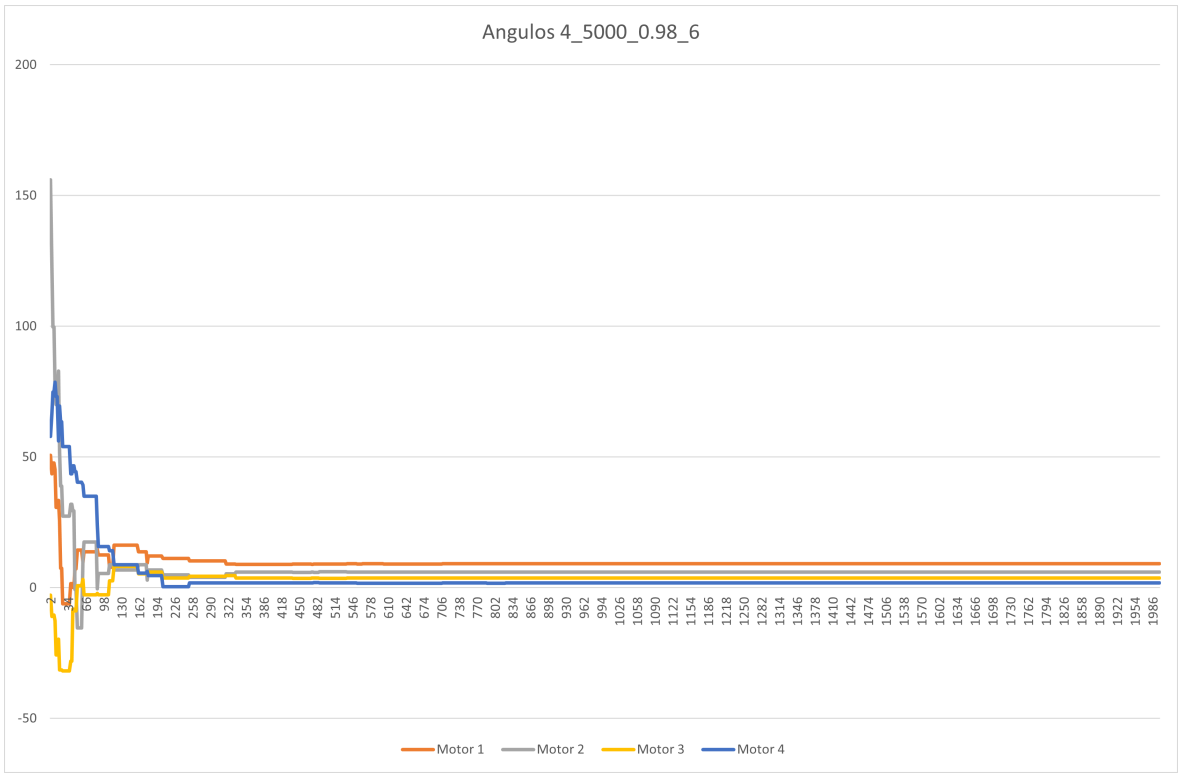


Figura 4: Ángulos: 4_5000_0.98_6

Después de ver esto, pasamos a probar aumentando el número de generaciones que se tienen en cuenta para la regla 1/5, pasaremos de 10 generaciones a 100, de esta manera se intentará suavizar un poco que pase de aumentar a reducir las varianzas. Se repiten de nuevo las pruebas anteriores para ver si este cambio supone una mejora en general en los modelos que se habían considerado.

Los resultados que obtenemos durante **10 ejecuciones para 4 motores, c variable y teniendo en cuenta 100 generaciones** para las varianzas se describen en la Figura 5 y la Tabla 2.

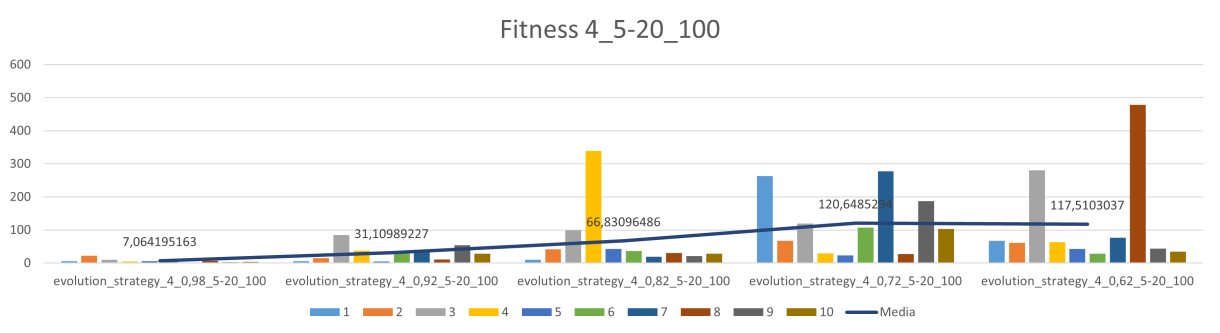


Figura 5: Valores de Fitness: 4_5-20_100

| Modelo | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | Media |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **4_0,98_5-20_100** | 5,969749305 | 21,8889931 | 9,949580495 | 4,974790248 | 5,969749305 | 5,969749305 | **1,989918114** | 6,964708362 | 2,984877171 | 3,979836228 | **7,064195163** |
| **4_0,92_5-20_100** | 5,542532418 | 14,74571547 | 84,98910986 | 37,81101029 | 5,011906134 | 36,82614644 | 34,11110903 | 10,47056077 | 53,73206948 | 27,85876286 | 31,10989227 |
| **4_0,82_5-20_100** | 9,809572078 | 41,36192181 | 99,24012352 | 339,3468103 | 42,84572441 | 36,74907856 | 18,86484429 | 30,76219558 | 20,6740378 | 28,65534023 | 66,83096486 |
| **4_0,72_5-20_100** | 263,4369112 | 67,53778001 | 119,6652832 | 29,2806866 | 22,87188578 | 107,3685088 | 277,6832855 | 27,53812501 | 187,4785023 | 103,624326 | 120,6485294 |
| **4_0,62_5-20_100** | 67,02456414 | 60,89633218 | 280,0429631 | 63,05976061 | 42,89979308 | 28,0932833 | 76,75306751 | 478,2493174 | 43,57174374 | 34,51221151 | 117,5103037 |

Tabla 2: Fitness 4 motores y 100 generaciones modificar varianzas

Se puede ver que los modelos mantienen la misma distribución que en la experimentación previa, es decir que los resultados son mejores cuanto mayor es el c, pero en este caso se reducen en general los valores de fitness.

La media del **fitness es 10 veces más pequeño** que su equivalente de la prueba anterior y esto se repite de la misma manera para los otros modelos. El mejor modelo en este caso tiene de **media 7,064195163 de fitness**.

Aunque de manera general se han reducido las evaluaciones, el mejor resultado obtenido no es mejor que el previo.

El mejor resultado se sigue correspondiendo con una **c de 0,98 valores de varianza iniciales entre 5 y 20**, pero en esta prueba en particular se han tenido en cuenta **100 generaciones para la regla 1/5**. El valor de **fitness de este modelo es 1,989918114**, que lo alcanza **en la evaluación 1990**, al igual que el otro esta evaluación está lejos del límite máximo de evaluaciones y le ha dado tiempo a converger correctamente. De media el mejor resultado para este modelo se obtiene en la evaluación 1962. Esta solución se corresponde con:

- **Motor 1:** 1,98991811419
- **Motor 2:** 9,128498154683596
- **Motor 3:** 4,99269532895493
- **Motor 4:** 1,7654320755166042

Con estas dos pruebas se ha podido ver que este método es factible para el problema y alcanza valores realmente buenos, cercanos al 0. En cuanto a la cantidad de generaciones que considerar para modificar las varianzas parecen mejor 100, pero en ejecuciones puntuales 10 también es buena, por lo que en la siguiente sección con 10 motores se repetirán los experimentos para ver si se puede concluir lo mismo.

### Aproximación 10 motores

Tras haber realizado unas pruebas iniciales con una versión simplificada para ver que es factible emplear esta técnica al problema pasamos al problema real, un brazo robótico de soldadura de 10 motores o articulaciones que se mueven de manera independiente y con el que tendremos que tratar de alcanzar la mejor configuración para que sea preciso.

Se volverá a probar con **5 valores de c (0,98, 0,92, 0,72 y 0,62)** como se hizo en la sección anterior, pero en este caso como se van a configurar **10 motores** se le dejaran **10000 generaciones, 20000 evaluaciones**, para que el de tiempo a converger y se haya obtenido el mejor resultado de la ejecución.

A continuación en la Figura 6 se muestran los resultados de **10 ejecuciones teniendo en cuenta las 10 generaciones previas** para modificar las varianzas, además de la Tabla 3 con sus correspondientes valores para poder consultar mejor las ejecuciones individualmente:

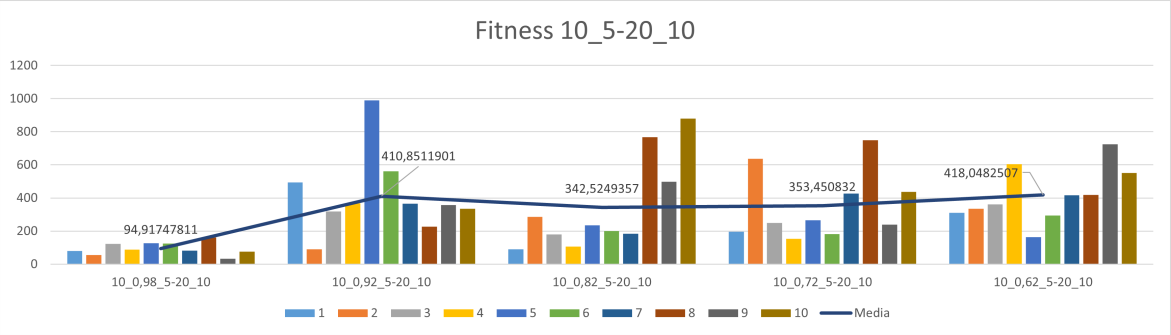


Figura 6: Valores de Fitness: 10_5-20_10

| Modelo | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | Media |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **10_0,98_5-20_10** | 80,59085566 | 54,72242403 | 122,3772289 | 88,55070259 | 126,3554685 | 124,3677632 | 82,57983737 | 162,1761055 | **31,83857359** | 75,61582174 | **94,91747811** |
| **10_0,92_5-20_10** | 493,4360219 | 89,54541129 | 318,3774674 | 368,0959481 | 990,6158079 | 562,034782 | 366,1135179 | 226,8463152 | 358,1536415 | 335,2929881 | 410,8511901 |
| **10_0,82_5-20_10** | 90,54006087 | 286,540584 | 179,0891408 | 105,4647007 | 235,7979006 | 199,9825131 | 183,0700586 | 767,0202184 | 498,4540123 | 879,2901677 | 342,5249357 |
| **10_0,72_5-20_10** | 196,5927933 | 637,7053161 | 249,7252972 | 153,2220596 | 264,6538342 | 181,0785194 | 425,8250794 | 749,1551409 | 239,7793379 | 436,7709423 | 353,450832 |
| **10_0,62_5-20_10** | 310,4180872 | 335,2775147 | 362,1496302 | 604,8171685 | 164,1662712 | 293,4897782 | 416,866646 | 417,8427783 | 725,2770965 | 550,1775361 | 418,0482507 |

Tabla 3: Fitness 10 motores, varianzas 5-20 y c considerando 10 generaciones

Se puede ver que aun aumentando el número de motores a 10, la distribución de resultados sigue siendo parecida, con la excepción del modelo con c de 0,92, que este caso ha empeorado siendo peor que el modelo de 0,82 y 0,72. Se profundizará en la sección *3.4. Resultado* más en estos modelos.

Además, los fitness del **mejor modelo de esta prueba, 10_0,98_5-20_10, son bastante consistentes** entre ejecuciones. En la Figura 7 se observa que el número de evaluaciones que requiere cada modelo para alcanzar su mejor resultado son bastante parecidos, lo que nos permite ver que todos convergen igual de rápido, pero no en los mejores resultados.

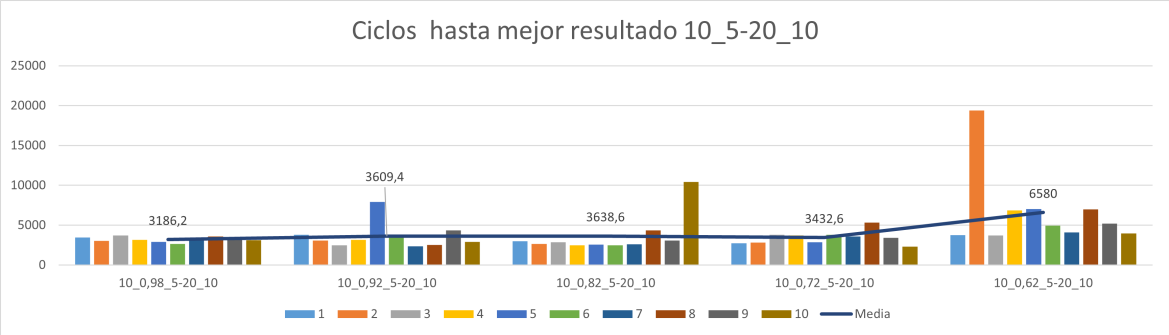


Figura 7: Ciclos hasta mejor resultado: 10_5-20_10

Ahora pasamos a probar si **aumentando el número de generaciones a considerar para modificar las varianzas** mejoran los resultados como ocurría en el caso de los 4 motores.

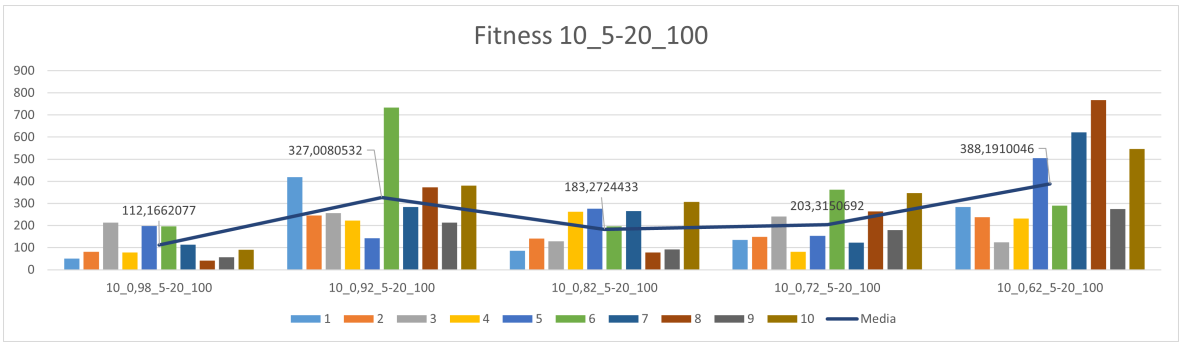


Figura 8: Valores de Fitness: 10_5-20_100

| Modelo | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | Media |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **10_0,98_5-20_100** | 50,74259284 | 81,58612136 | 213,9061234 | 77,60612079 | 197,9927287 | 196,9949114 | 113,4239113 | **41,78817447** | 57,08071544 | 90,54067699 | **112,1662077** |
| **10_0,92_5-20_100** | 418,851877 | 245,7473547 | 255,6940882 | 222,8678002 | 143,2442372 | 733,1081525 | 284,5502886 | 373,0604005 | 212,9140302 | 380,0423028 | 327,0080532 |
| **10_0,82_5-20_100** | 85,56587251 | 141,282387 | 129,4266779 | 261,6562212 | 275,588287 | 196,9962121 | 264,6484531 | 78,60131111 | 91,53563694 | 307,4233739 | 183,2724433 |
| **10_0,72_5-20_100** | 134,3723279 | 148,3272106 | 240,7722142 | 80,59637497 | 153,2277586 | 362,1933594 | 122,4410243 | 263,7641307 | 180,1514111 | 347,3048804 | 203,3150692 |
| **10_0,62_5-20_100** | 284,001316 | 238,3567998 | 123,5151397 | 231,8477615 | 504,8815338 | 289,2721911 | 621,3900092 | 767,7025742 | 275,3068091 | 545,6359119 | 388,1910046 |

Tabla 4: Fitness 10 motores, varianzas 5-20 y c considerando 100 generaciones

Como se esperaba, en general **los resultados han mejorado** con respecto a la prueba anterior, con excepción del modelo de c 0,98 que ha empeorado ligeramente. Además, mantiene la distribución de la prueba previa, siendo mejores cuanto mayor c y empeorando en c 0,92.

Nuestro mejor modelo previo era el de c 0,98 con una media de 94,91747811 de fitness y en este nuevo caso es también el de **c 0,98 con una media de 112,1662077 de fitness**. Aparte de obtener mayor fitness, el nuevo modelo **requiere 3278 evaluaciones de media** para alcanzar el mejor resultado y el otro 3186 evaluaciones.

Hasta este punto se puede concluir que los **mejores modelos se obtienen con una c de 0,98 o 0,82 y considerando 10 generaciones para mutar la varianza**.

Ahora se pasará a probar para estos 2 modelos mejores modelos una **ventana de varianzas de entre 5 y 100**, además se aumentará el número de **generaciones máximas a 20000** para que puedan converger adecuadamente. Los resultados que se han obtenido son mostrados en la Figura 9 y la Tabla 5.

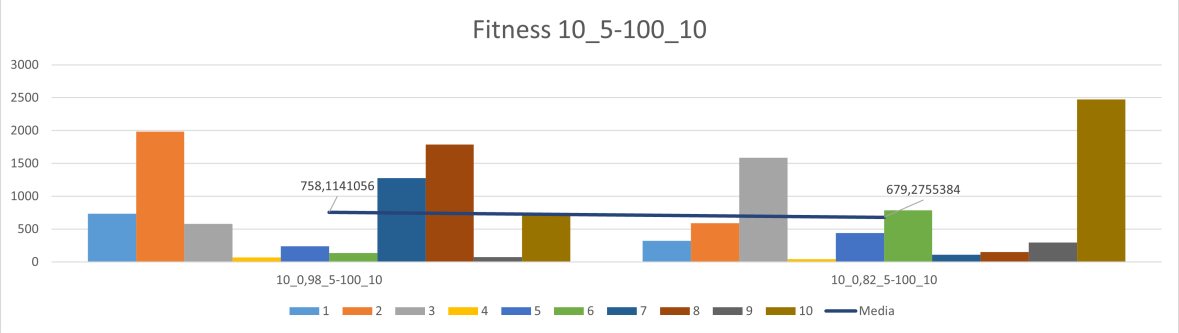


Figura 9: Valores de Fitness: 10_5-20_10

| Modelo | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | Media |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **10_0,98_5-100_10** | 736,0960491 | 1983,36103 | 577,9290908 | 69,64607244 | 236,790299 | 134,3176337 | 1273,989951 | 1787,181803 | 71,63674394 | 710,1923834 | 758,1141056 |
| **10_0,82_5-100_10** | 320,620507 | 589,2224645 | 1587,550479 | **41,78811602** | 442,778636 | 784,7749207 | 108,4492238 | 150,2370648 | 293,4932953 | 2473,840677 | **679,2755384** |

Tabla 5: Fitness 10 motores, varianzas 5-100 y c considerando 10 generaciones

Como se puede ver, realizando esta modificación de los valores con los que se pueden inicializar las varianzas **no produce mejores modelos**. Aun no produciendo modelos mejores en una ejecución particular alcanzó un fitness de 41,7881 que no está muy lejos del mejor modelo para los 10 motores, 31,8385.

Con todo esto se pasa a describir el mejor modelo que se ha obtenido.

### Resultado

Tras todas las pruebas que se han realizado en las secciones *3.3. Aproximación 10 motores* y *3.2. Aproximación 4 motores*, se describirá en esta sección cuál es el mejor modelo que se ha obtenido para resolver el problema de hallar los ángulos para un brazo soldador de 10 motores.

El modelo que mejores fitness de media nos ha proporcionado y de una manera bastante consistente en comparación con el resto ha sido el que tiene los siguientes parámetros:

- **Motores:** 10
- **Valores de inicialización de las varianzas:** Entre 5 y 20
- **c:** 0,98
- **Generaciones consideradas:** 10

El **fitness medio ha sido de 94,91747811**, que se ha alcanzado **en 3186,2 evaluaciones de media de las 20000** que se le ha dejado ejecutando.

En la sección *3.3. Aproximación 10 motores* se puede ver la consistencia del fitness con la Figura 6 y la del número de evaluaciones en la Figura 7.

Analizaremos ahora la mejor ejecución de este modelo, que se corresponde con la novena ejecución y que nos proporciona los siguientes ángulos para el brazo del robot:

- **Motor 1:** 2,2284984784501773
- **Motor 2:** 5,477566084188264
- **Motor 3:** 1,3246628300371095
- **Motor 4:** -0,22448014101449237
- **Motor 5:** 1,4274017184920664
- **Motor 6:** -3,9797837284309523
- **Motor 8:** -0,6517585124073262
- **Motor 7:** -1,055368413073764
- **Motor 9:** -0,7604292115674011
- **Motor 10:** 3,938239892255979

Esta secuencia de ángulos nos da un **fitness de 31,83857359** que se alcanza en **3180 evaluaciones**, que equivalen a 1590 generaciones de las 10000 que se dejaron.

En la Figura 10 se puede ver la progresión del fitness desde la primera evaluación hasta que se encuentra el mejor fitness, de esta manera se evita prolongar la gráfica en exceso para el mismo valor.

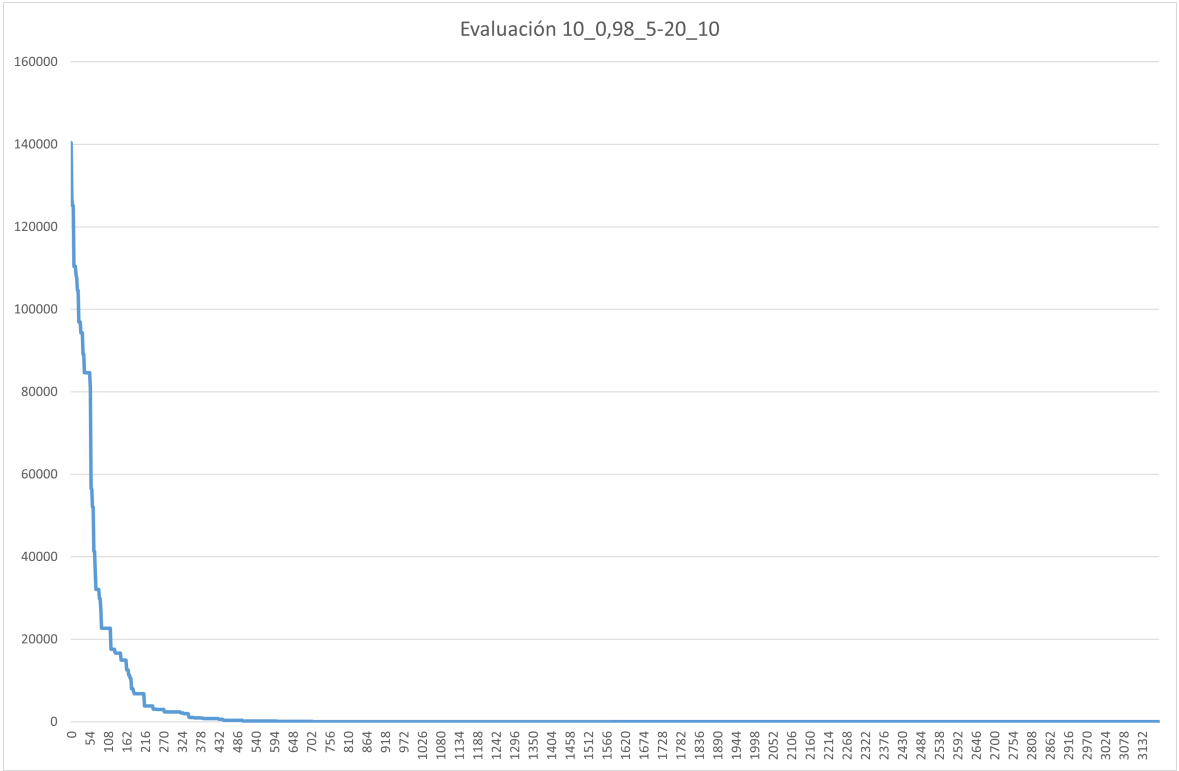


Figura 10: Fitness vs. Evaluaciones: 10_0.98_5-20_10

De la misma manera que en la Figura 10 se puede ver en la Figura 11 como van convergiendo los ángulos al ajustarse y encontrar un valor cuasi-óptimo.

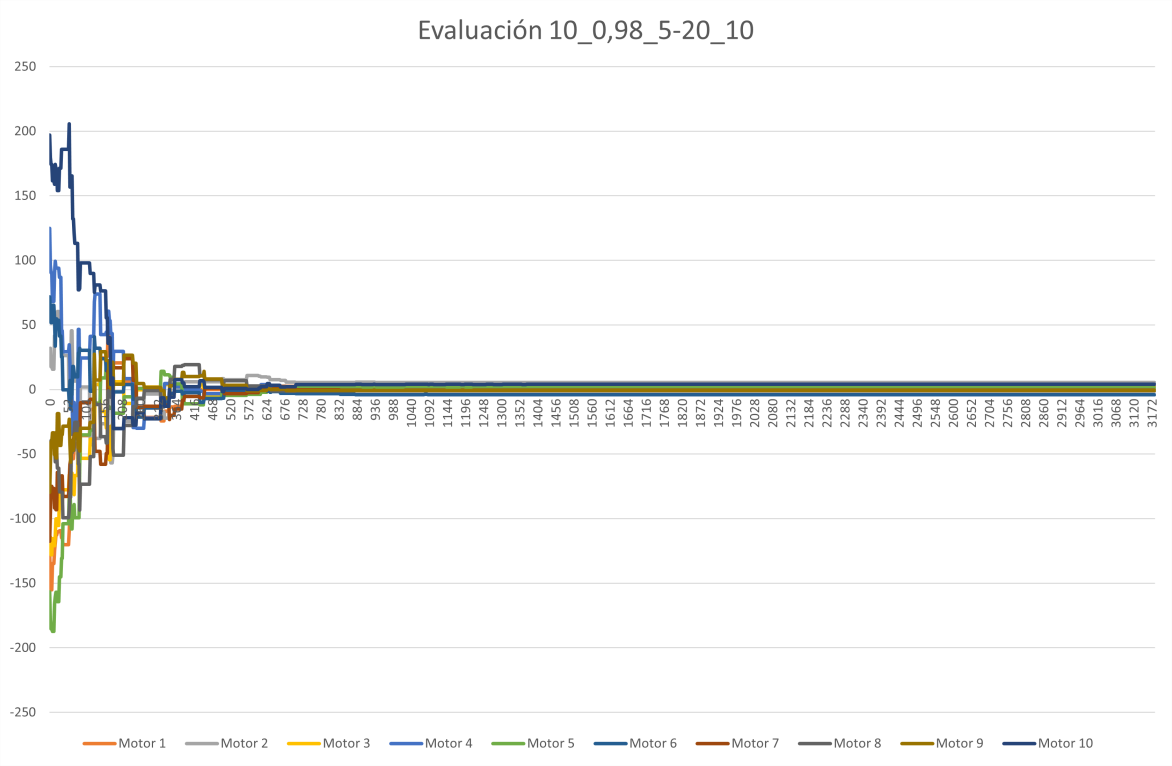


Figura 11: Ángulos: 10_0.98_5-20_10

## Conclusiones

En este proyecto se ha abordado el problema de ajustar los ángulos de un brazo robótico de soldadura para que alcanzara una precisión aceptable para soldadura profesional mediante estrategias evolutivas.

Se comenzó con una aproximación reducida con 4 motores para ver como aproximarnos al problema real y ver si esta técnica era viable para este problema específico, se puedo ver que se alcanzaban valores muy buenos menores que 1 de error.

Tras comprobar que esta aproximación es buena se pasó a problema real, 10 motores, aplicando los conocimientos adquiridos de la versión reducida. Se obtuvieron resultados con más error, cosa que se esperaba dado que cuantas más articulaciones tiene un brazo más difícil es poder calibrarlo, puesto que cualquier cambio afecta a la posición global sobre todo los más alejados de la punta del solador.

Esta práctica ha servido para ver una aplicación en primera mano de las estrategias evolutivas y ver que son una técnica potente para la resolución de problemas.