Sistemas Inteligentes

Practica 1

Pablo José Rocamora Zamora G3.2

9 de Diciembre de 2017

Índice general

Explicación breve y completa de la técnica Algoritmo Genético (AG).

Un algoritmo genético (AG) es una variante de la búsqueda de haz estocástica en la que los estados sucesores se generan combinando a dos estados padres, más que modificar un solo estado.

El algoritmo genético trata de encontrar la mejor solución por comparación de un conjunto de soluciones.

Las soluciones se generan a través del cruzamiento de generaciones o soluciones anteriores; está cruzando generaciones para obtener una nueva generación, de manera que podamos compararla para ver si estamos acercándonos a la solución final [1].

Elementos

Tenemos diferentes formas de codificar una población:

- Codificación binaria
- Codificación entera
- Codificación real
- Codificación en orden

Tenemos distintos operadores genéticos:

- **Selección**: (escoge que individuos se reproducirán y cuales no)
  - **Ruleta**: Se eligen con probabilidad proporcional a su función de idoneidad.
  - **Torneo**: Se establecen k torneos aleatorios entre parejas de individuos y se eligen los que ganan en cada torneo (mejor función idoneidad).
- **Cruce**: (recombina individuos para producir descendencia)
  - **Cruce por un punto**: dos padres a partir de un gen, se intercambian el resto de genes, creando 2 hijos nuevos
  - **Cruce por dos puntos**: dos padres a partir de un rango de genes, se intercambian los genes de ese rango, creando dos hijos nuevos
- **Mutación**: (provoca el cambio de valor de algunos genes del individuo)
  - **Cambio de un gen aleatorio**
  - **Intercambio entre dos genes**

Tenemos que definir la función objetivo

Es necesario crear la función fitness; es una función de adaptación que tiene que retornar: *>0*, es la forma de evaluar la población.

Proceso

1. Codificamos el problema (en nuestro caso en forma de *Integer*).
2. Generanos una población inicial aleatoriamente de k estados.

3. Seleccionamos k individuos a través del operador de selección para crear una nueva población.
4. Elegimos individuos con una probabilidad (p_c) para ser cruzados y crear una nueva población.
5. Con una probabilidad (p_m) mutamos los genes de los individuos de la población actual.

6. Esta nueva población sustituye a la original y forma la nueva población inicial que se usará en la siguiente generación, volvemos al paso 3

Explicación detallada de todas las preguntas realizadas.

Explica de qué manera se están inicializando los individuos en el AG propuesto.

Explica el funcionamiento de los operadores de selección indicados en la sección "Ajuste del Algoritmo Genético".

Tenemos disponibles en este caso 2 operadores de Selección:

- GARouletteWheelSelector: Se asigna una probabilidad de selección proporcional al valor del fitness del cromosoma.
- GATournamentSelector: Escoge al individuo de mejor fitness de entre N_{ts} individuos seleccionados aleatoriamente con reemplazamiento (N_{ts}=2,3,…).

[2]

Explica de qué manera se están cruzando los individuos.

Se hace un cruce por 2 puntos; el primer rango de genes corresponde a *p1* y el segundo rango a *p2*. El tamaño de los rangos es aleatorio, por lo que a veces será mas grande *p1* y otras más pequeño.

Explica de qué manera se están mutando los individuos.

Define y explica la condición de parada que utilizarás.

Tenemos 2 condiciones de parada posibles:

1. La óptima es que el fitness sea 0, lo que significa que el problema ha encontrado la solución y termina correctamente.
2. Se ha llegado al número máximo de generaciones, que en este caso son 12000; aquí llegaremos cuando no lleguemos a un fitness 0 y significará que la configuración que hayamos usado no es correcta.

Diseña y explica la función fitness que utilizarás.

La función fitness se compone de 3 partes: comprobar las filas, comprobar las columnas y comprobar las subcuadrículas

- **Comprobar Filas**: Recorre todas las filas; para cada fila crea un array en el que introduce el número que lee, usando como índice el valor del número, por lo que al acabar de leer la fila tendremos un array de números ordenados en el que sí están todos los números la fila es correcta y si falta algún número habrá un 0 en el array, la cantidad de 0 indican la cantidad de números que hay mal en la fila. Una vez rellenado el array, llama a la función `compruebaHuecosVacios()` para que calcule la cantidad de números que hay mal en la fila y aumentar nuestro contador global de fallos.

- **Comprobar columnas**: Recorre todas las columnas; para cada columna crea un array en el que introduce el número que lee, usando como índice el valor del número, por lo que al acabar de leer la columna tendremos un array de números ordenados en el que si están todos los números la columnas es correcta y si falta algún numero habrá un 0 en el array, la cantidad de 0 indican la cantidad de números que hay mal en la columna. Una vez rellenado el array, llama a la función `compruebaHuecosVacios()` para que calcule la cantidad de números que hay mal en la columna y aumentar nuestro contador global de fallos.
- **Comprobar subcuadriculas**: Recorre todas las subcuadrículas; para cada subcuadrícula crea un array en el que introduce el número que lee, usando como índice el valor del número, por lo que al acabar de leer la subcuadrícula tendremos un array de números ordenados en el que sí están todos los números. La subcuadrícula es correcta, y si falta algún número, habrá un 0 en el array. La cantidad de 0 indican la cantidad de números que hay mal en la subcuadrícula. Una vez rellenado el array, llama a la función `compruebaHuecosVacios()` para que calcule la cantidad de números que hay mal en la subcuadrícula y así aumentar nuestro contador global de fallos.
- **Función compruebaHuecosVacios**: Función que recibe un array con los valores de una fila, columna o subcuadrícula. Lo recorre, y por cada 0 que encuentre, aumenta el contador. Finalmente, lo retorna para aumentar el contador general de la función fitness.

```
/**
 * Metodo que comprueba la cantidad de huecos que tiene el array con un 0
 */
int compruebaHuecosVacios(int *array, int tamSudoku) {
    int huecosVacios = 0;
    for (int i = 1; i <= tamSudoku; i++)
        if (array[i] == 0)
            huecosVacios++;

    return huecosVacios;
}
```

Ejemplo 1: Tenemos una fila con los valores (7, 1, 3, 2, 5, 4, 6, 8, 9); al llamar a `calculaFilas ↪ ()`, obtenemos el array ordenado. Posteriormente, llamamos a `compruebaHuecosVacios()` y comprobamos el número de 0 que haya; en este caso, como no hay repeticiones, tampoco hay 0 y, por tanto, la fila es correcta.

7	1	3	2	5	4	6	8	9
1	2	3	4	5	6	7	8	9

Ejemplo 2: Tenemos una columna con los valores (7, 1, 3, 3, 5, 5, 6, 8, 9); al llamar a `calculaColumnas()` obtenemos el array ordenado. Posteriormente, llamamos a `compruebaHuecosVacios ↪ ()` y comprobamos el número de 0 que hay; en este caso, como están repetidos tanto el 3 como el 5, faltan 2 números que son el 2 y el 4, con lo que el número de fallos son 2.

7	1	3	3	5	5	6	8	9
1	0	3	0	5	6	7	8	9

Tablas con los resultados de las ejecuciones.

Vamos a realizar el ajuste de parámetros de la siguiente manera: Conjunto de casos del problema: Casos 1-5. Métodos/parámetros fijos: Nº generaciones = 12000, operador de cruce y operador de mutación. Métodos/parámetros ajustables: Tam Población, pc, pm, Selector.

Para un mejor ajuste se ha añadido un sudoku extra de una complejidad superior al resto; en la siguiente sección se hablará de él.

	SELECTOR	POBLACION	PROB CRUCE	PROB MUTAC	C1	C2	C3	C4	C5	Especial		Tiempo C1	Tiempo C2	Tiempo C3	Tiempo C4	Tiempo C5	Tiempo Especial	Promedio Tiempos
2	GARouletteWheelSelector	100	0,8	0,01	0	0	3	0	0	4		1,25	0,56	5,45	0,97	0,18	5,5	No valido
3	GARouletteWheelSelector	100	0,8	0,05	0	1	0	0	0	5		1,21	6,47	0,81	2,28	0,11	6,8	No valido
4	GARouletteWheelSelector	100	0,8	0,1	0	1	1	0	0	1		0,43	7,17	7,2	3,13	0,22	7,45	No valido
5	GARouletteWheelSelector	100	0,8	0,125	0	1	0	0	0	3		1,83	7,48	4,02	6,68	1,38	7,78	No valido
6	GARouletteWheelSelector	100	0,8	0,15	0	4	0	0	4	2		3,02	7,75	1,48	0,95	7,83	8,17	No valido
7	GARouletteWheelSelector	100	0,85	0,01	0	1	0	0	3	3		0,24	5,37	0,23	0,62	5,33	5,6	No valido
8	GARouletteWheelSelector	100	0,85	0,05	0	1	3	0	0	2		5,84	6,45	6,46	1,79	0,12	6,7	No valido
9	GARouletteWheelSelector	100	0,85	0,1	4	0	1	0	0	3		7,37	0,44	7,25	6,07	1,23	7,43	No valido
10	GARouletteWheelSelector	100	0,85	0,125	0	4	1	8	0	1		2,43	7,47	7,49	7,67	0,35	7,82	No valido
11	GARouletteWheelSelector	100	0,85	0,15	2	1	1	0	0	7		8,07	7,86	7,88	4,69	0,16	8,15	No valido
12	GARouletteWheelSelector	100	0,9	0,01	0	0	0	2	0	2		0,3	0,22	0,29	5,51	0,39	5,63	No valido
13	GARouletteWheelSelector	100	0,9	0,05	0	4	0	0	0	5		6,59	6,47	3,19	0,84	0,43	6,68	No valido
14	GARouletteWheelSelector	100	0,9	0,1	3	4	1	2	0	2		7,42	7,19	7,3	7,32	0,19	7,41	No valido
15	GARouletteWheelSelector	100	0,9	0,125	8	1	0	0	3	2		7,8	7,47	0,87	0,72	7,58	7,81	No valido
16	GARouletteWheelSelector	100	0,9	0,15	2	1	1	5	0	4		8,1	7,77	7,82	8,04	5,13	8,19	No valido
17	GARouletteWheelSelector	100	0,95	0,01	0	3	0	0	0	2		0,18	5,58	0,16	0,17	0,41	5,82	No valido
18	GARouletteWheelSelector	100	0,95	0,05	4	1	3	0	0	2		6,66	6,7	6,93	1,35	0,42	6,89	No valido
19	GARouletteWheelSelector	100	0,95	0,1	0	1	1	0	0	2		6,21	7,85	7,64	0,98	2,98	8,13	No valido
20	GARouletteWheelSelector	100	0,95	0,125	6	8	0	4	4	5		8,07	7,81	0,08	7,96	7,7	8,09	No valido
21	GARouletteWheelSelector	100	0,95	0,15	7	0	0	2	0	2		8,31	6,6	2,18	8,28	1,2	8,4	No valido
22	GARouletteWheelSelector	150	0,8	0,01	3	0	0	0	0	1		8,49	4,88	0,27	0,11	2,67	8,58	No valido
23	GARouletteWheelSelector	150	0,8	0,05	0	1	0	0	0	6		5,06	10,04	5,09	0,71	9,28	10,45	No valido
24	GARouletteWheelSelector	150	0,8	0,1	0	1	0	0	2	1		3,46	10,92	4,21	9,23	10,84	2,94	No valido
25	GARouletteWheelSelector	150	0,8	0,125	2	3	1	3	0	7		11,74	11,35	11,41	11,62	0,98	11,85	No valido
26	GARouletteWheelSelector	150	0,8	0,15	5	1	1	1	4	2		12,16	11,8	12	12,17	11,83	12,28	No valido
27	GARouletteWheelSelector	150	0,85	0,01	0	4	0	0	2	3		8,14	8,21	0,82	5,24	8,08	8,46	No valido
28	GARouletteWheelSelector	150	0,85	0,05	2	1	0	4	0	6		10,03	9,78	0,31	9,95	0,14	10,29	No valido
29	GARouletteWheelSelector	150	0,85	0,1	6	4	0	0	0	4		11,25	10,84	0,42	10,39	1,09	11,42	No valido
30	GARouletteWheelSelector	150	0,85	0,125	0	0	0	1	0	2		2,95	0,93	3,42	11,75	9,86	11,88	No valido
31	GARouletteWheelSelector	150	0,85	0,15	0	2	1	2	0	2		0,7	11,79	11,89	12,13	1,22	12,32	No valido
32	GARouletteWheelSelector	150	0,9	0,01	0	1	1	3	0	3		0,97	8,39	8,37	8,48	2,69	8,52	No valido
33	GARouletteWheelSelector	150	0,9	0,05	0	0	0	0	0	4		1,21	0,14	1,07	0,7	0,89	10,24	2,375
34	GARouletteWheelSelector	150	0,9	0,1	7	0	0	0	0	2		11,21	0,34	5,28	5,28	4,86	11,27	No valido
35	GARouletteWheelSelector	150	0,9	0,125	0	4	4	2	0	2		0,5	11,48	11,46	12	0,59	12,03	No valido
36	GARouletteWheelSelector	150	0,9	0,15	3	1	1	4	2	2		12,05	11,64	11,61	11,89	11,61	12,18	No valido
37	GARouletteWheelSelector	150	0,95	0,01	0	0	0	0	0	3		2,39	1,01	1,13	8,41	7,8	8,68	4,90333333333333
38	GARouletteWheelSelector	150	0,95	0,05	0	1	0	0	0	2		0,17	9,73	5,01	2,67	0,17	10,16	No valido
39	GARouletteWheelSelector	150	0,95	0,1	0	1	1	0	0	3		6,3	10,88	10,94	7,12	5,44	11,31	No valido
40	GARouletteWheelSelector	150	0,95	0,125	2	4	3	0	0	6		11,68	11,3	11,36	7,94	0,19	11,75	No valido
41	GARouletteWheelSelector	150	0,95	0,15	2	1	4	0	0	8		12,11	11,79	11,85	0,43	0,75	12,33	No valido

Figura 1: Tabla con selector GARouletteWheelSelector para casos de ajuste

Análisis de las pruebas de ajuste.

Para un mejor ajuste se ha añadido un sudoku extra; este sudoku es diferente al de los casos de ajuste porque en vez de tener 30 o 33 números iniciales solo tiene 28, por lo que tiene una complejidad superior. En este análisis primero se observarán los 5 casos de ajuste y, por último, se hará un análisis individual del sudoku extra.

Sudokus de caso de ajuste

Estos sudokus tienen inicialmente 30 y 33 números.

Observando las soluciones vemos que el selector GARouletteWheelSelector consigue en muy pocas ocasiones resolver los 5 casos, por lo que usaremos el selector GATournamentSelector.

En cuanto al tamaño de la población, no se aprecian diferencias significativas, por lo que podríamos usar cualquiera de los dos para comparar.

Mirando la Figura 3 podemos apreciar que el pc de 0.9 es negativo, ya que solo consigue resolver el sudoku del `Caso 1`; por el contrario, 0.8 y 0.95 consiguen resolver todos los casos, por lo que los usaremos para comparar.

	SELECTOR	POBLACION	PROB CRUCE	PROB MUTAC	C1	C2	C3	C4	C5	Especial	Tiempo C1	Tiempo C2	Tiempo C3	Tiempo C4	Tiempo C5	Tiempo Especial	Promedio Tiempo C	Promedio Tiempo Total
2	GATournamentSelector	100	0,8	0,01	5	4	5	5	3	3	5,41	5,11	5,16	5,32	5,14	5,29	No valido	No valido
3	GATournamentSelector	100	0,8	0,05	0	0	0	0	0	1	1,96	4,91	0,03	0,02	0,17	6,13	1,418	No valido
4	GATournamentSelector	100	0,8	0,1	0	0	0	0	0	1	0,09	1,48	1,65	0,1	0,13	7,29	0,69	No valido
5	GATournamentSelector	100	0,8	0,125	0	0	0	0	0	4	0,11	0,13	0,85	0,17	0,03	7,73	0,258	No valido
6	GATournamentSelector	100	0,8	0,15	0	0	0	0	0	3	0,17	0,07	0,31	0,58	0,05	7,97	0,236	No valido
7	GATournamentSelector	100	0,85	0,01	5	3	2	2	3	6	5,44	5,22	5,3	5,26	5,2	5,38	No valido	No valido
8	GATournamentSelector	100	0,85	0,05	3	0	0	0	0	3	6,25	0,18	0,02	0,16	2,98	6,27	No valido	No valido
9	GATournamentSelector	100	0,85	0,1	0	0	0	0	0	3	0,08	5,06	0,08	0,03	0,04	7,12	1,058	No valido
10	GATournamentSelector	100	0,85	0,125	0	0	0	0	0	2	0,16	1,31	2,55	0,87	0,03	7,7	0,984	No valido
11	GATournamentSelector	100	0,85	0,15	0	0	0	0	0	2	0,12	0,41	3,51	0,66	0,13	8,08	0,966	No valido
12	GATournamentSelector	100	0,9	0,01	3	7	8	2	2	4	5,4	5,37	5,43	5,34	5,24	5,53	No valido	No valido
13	GATournamentSelector	100	0,9	0,05	0	1	0	2	0	4	0,44	5,96	0,01	6,14	0,04	6,24	No valido	No valido
14	GATournamentSelector	100	0,9	0,1	0	0	0	0	0	3	0,11	4,05	1,32	0,05	0,03	7,29	1,112	No valido
15	GATournamentSelector	100	0,9	0,125	0	1	0	0	0	3	0,07	7,32	0,16	0,46	0,27	7,69	No valido	No valido
16	GATournamentSelector	100	0,9	0,15	0	0	0	0	0	2	0,65	1,48	1,14	2,6	0,2	7,96	1,214	No valido
17	GATournamentSelector	100	0,95	0,01	5	5	0	0	4	8	5,6	5,47	0,83	5,13	5,41	5,63	No valido	No valido
18	GATournamentSelector	100	0,95	0,05	0	0	0	0	0	1	0,44	0,37	0,01	0,07	0,02	6,24	0,182	No valido
19	GATournamentSelector	100	0,95	0,1	0	0	0	0	0	2	0,02	0,04	0,03	0,28	0,02	7,23	0,078	No valido
20	GATournamentSelector	100	0,95	0,125	0	0	0	0	0	4	0,91	3,4	0,04	0,75	0,03	7,78	1,026	No valido
21	GATournamentSelector	100	0,95	0,15	0	0	0	0	0	0	1,29	1,47	0,1	0,09	0,05	2,46	0,6	0,91
22	GATournamentSelector	150	0,8	0,01	2	1	0	6	4	3	7,79	7,68	0,19	7,85	7,74	7,88	No valido	No valido
23	GATournamentSelector	150	0,8	0,05	0	1	1	0	0	3	1,11	9,1	8,96	0,15	1,69	9,36	No valido	No valido
24	GATournamentSelector	150	0,8	0,1	0	0	0	0	0	3	0,08	0,03	0,95	0,15	0,06	11,03	0,254	No valido
25	GATournamentSelector	150	0,8	0,125	0	0	0	0	0	3	0,08	3,09	0,03	0,08	0,61	11,59	0,778	No valido
26	GATournamentSelector	150	0,8	0,15	0	0	0	0	0	4	0,85	0,59	0,13	0,34	0,21	12,03	0,424	No valido
27	GATournamentSelector	150	0,85	0,01	0	1	1	6	5	2	0,04	7,75	7,84	8,03	7,91	8,09	No valido	No valido
28	GATournamentSelector	150	0,85	0,05	0	1	0	0	0	1	0,04	8,97	0,11	0,04	0,73	9,37	No valido	No valido
29	GATournamentSelector	150	0,85	0,1	0	0	0	0	0	0	0,06	0,06	0,03	0,24	0,12	0,2	0,102	0,118333333333333
30	GATournamentSelector	150	0,85	0,125	0	0	0	0	0	2	0,06	0,65	0,08	0,09	0,06	11,65	0,188	No valido
31	GATournamentSelector	150	0,85	0,15	0	1	0	0	0	3	2,88	11,6	0,24	0,22	0,2	12,24	No valido	No valido
32	GATournamentSelector	150	0,9	0,01	3	4	0	0	0	5	8,25	8	0,14	0,15	0,05	8,31	No valido	No valido
33	GATournamentSelector	150	0,9	0,05	0	3	0	0	0	2	0,07	9,18	0,04	0,34	0,41	9,69	No valido	No valido
34	GATournamentSelector	150	0,9	0,1	0	0	0	0	0	2	0,07	1,88	0,08	0,05	0,03	11,03	0,422	No valido
35	GATournamentSelector	150	0,9	0,125	0	0	0	0	0	4	0,23	2,26	0,06	0,1	0,66	11,81	0,662	No valido
36	GATournamentSelector	150	0,9	0,15	0	0	0	0	0	2	0,29	0,08	0,1	0,15	0,06	12,26	0,136	No valido
37	GATournamentSelector	150	0,95	0,01	2	6	7	2	3	3	8,27	8,19	8,28	8,18	8,53	8,58	No valido	No valido
38	GATournamentSelector	150	0,95	0,05	0	0	1	2	0	1	0,04	0,07	9,29	9,2	0,22	9,44	No valido	No valido
39	GATournamentSelector	150	0,95	0,1	0	0	0	0	0	2	0,04	0,07	0,13	0,22	0,03	10,76	0,098	No valido
40	GATournamentSelector	150	0,95	0,125	0	0	0	0	0	0	0,12	0,19	2,16	0,19	0,13	6,42	0,558	1,535
41	GATournamentSelector	150	0,95	0,15	0	0	0	0	0	3	0,49	1,34	0,13	0,31	0,06	12,5	0,466	No valido

Figura 2: Tabla con selector GATournamentSelector para casos de ajuste

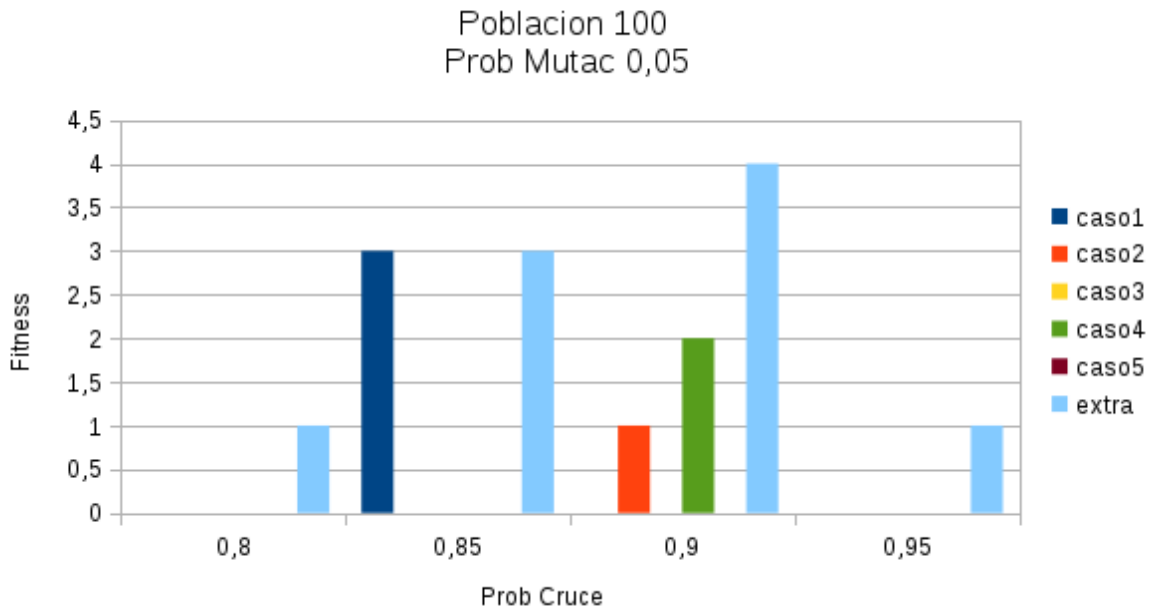

Figura 3: Gráfica comparando Probabilidades de Cruce

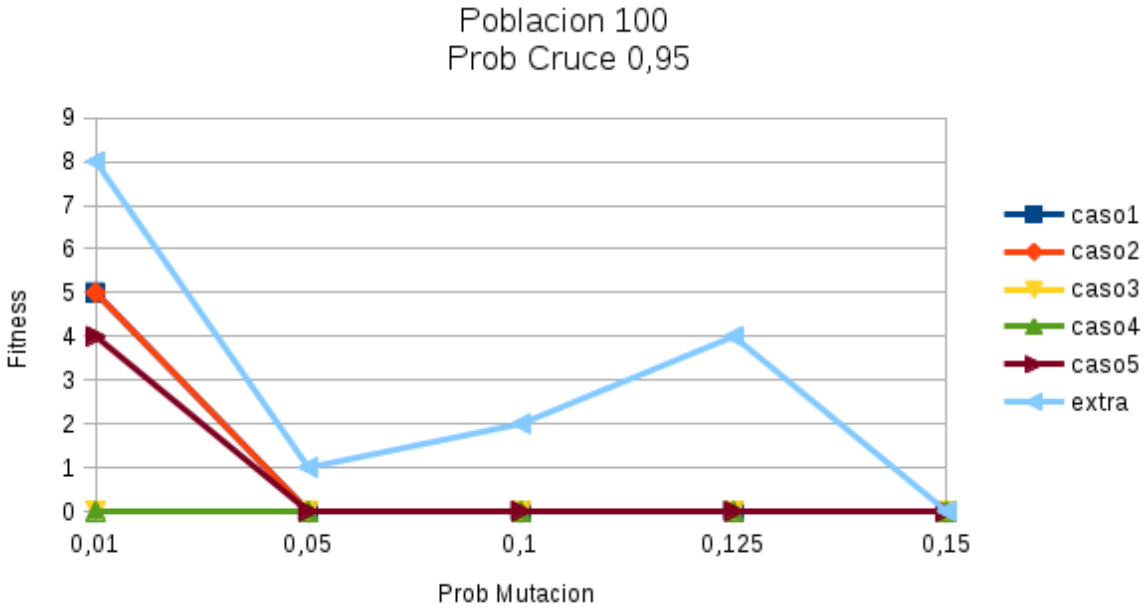

Figura 4: Gráfica comparando Probabilidades de Mutación

Mirando la Figura 3 podemos apreciar que un pm de 0.01 es negativo, ya que solo consigue resolver el `Caso 3` y `Caso 4`; por el contrario, el resto sí consigue resolver todos los sudokus, por lo que usaremos para comparar 0.05, 0.1 y 0.125.

Dado que hay gran cantidad de soluciones válidas para resolver los casos, vamos a añadir también el promedio de los tiempos de ejecución de todos los casos, ya que además de buscar solucionar un sudoku buscamos hacerlo en el menor tiempo posible.

Por lo que al final tenemos:

Selector	Población	pc	pm	Tiempo Ejecución
GATournamentSelector	100	0,8	0,125	0,258
GATournamentSelector	100	0,8	0,15	0,236
GATournamentSelector	100	0,95	0,05	0,182
GATournamentSelector	100	0,95	0,1	0,078
GATournamentSelector	100	0,95	0,125	1,026
GATournamentSelector	100	0,95	0,15	0,6
GATournamentSelector	150	0,8	0,1	0,254
GATournamentSelector	150	0,8	0,125	0,778
GATournamentSelector	150	0,8	0,15	0,424
GATournamentSelector	150	0,95	0,1	0,098
GATournamentSelector	150	0,95	0,125	0,558
GATournamentSelector	150	0,95	0,15	0,466

Tenemos el promedio de los tiempos de ejecución de los casos de ajuste; en ellos, podemos apreciar que un pm de 0.1 suele ofrecer menores tiempos de ejecución. Observando los tiempos de ejecución de cada pc comprobamos que pc=0.95 con población=100 es mucho mas rápido que el resto de configuraciones validas, por lo que nuestra configuración final será:

Selector	Población	pc	pm
GATournamentSelector	100	0.95	0.1

Mirando la configuración válida pero más lenta que tenemos, vemos que tarda 1.214 seg; si la

En esta ocasión, como mejor configuración, tomaremos:

Selector	Población	pc	pm
GATournamentSelector	150	0.85	0.1

Manual-Asignación.

Para sudokus que tengan 30 o mas números iniciales usaremos:

Selector	Población	pc	pm
GATournamentSelector	100	0.95	0.1

Para sudokus de menos de 30 números iniciales usaremos:

Selector	Población	pc	pm
GATournamentSelector	150	0.85	0.1

Los parámetros que hay que pasarle al binario son:

`UM-SSII Caso-X.txt población selector pc pm`

Ejemplo de uso: `UM-SSII Sudoku-1.txt 100 GATournamentSelector 1.95 0.1`

Casos del Usuario

Caso 1

Como tiene 33 números iniciales usaremos:

Selector	Población	pc	pm
GATournamentSelector	100	0.95	0.1

Ejecución: `UM-SSII Sudoku-1.txt 100 GATournamentSelector 1.95 0.1`

Solución:

```
El GA encuentra la solución ( 4 5 8 3 6 7 9 1 2 1 6 3 9 2 5 7 4 8 2 9 7 8 4 1 5 6 3 8 4 5
    ↪ 2 9 6 3 7 1 7 3 2 5 1 8 6 9 4 6 1 9 4 7 3 2 8 5 5 7 1 6 8 2 4 3 9 9 2 6 1 3 4 8 5 7
    ↪  3 8 4 7 5 9 1 2 6 )
fitness: 0
```

Como podemos observar, resuelve el sudoku (fitness 0) y ofrece su solución de forma lineal. Si queremos verlo en un formato mas amigable, tendríamos:

```
4 5 8 3 6 7 9 1 2
1 6 3 9 2 5 7 4 8
2 9 7 8 4 1 5 6 3
```

```
8 4 5 2 9 6 3 7 1
7 3 2 5 1 8 6 9 4
6 1 9 4 7 3 2 8 5
5 7 1 6 8 2 4 3 9
9 2 6 1 3 4 8 5 7
3 8 4 7 5 9 1 2 6
```

Caso 2

Como tiene 33 números iniciales usaremos:

Selector	Población	pc	pm
GATournamentSelector	100	0.95	0.1

Ejecución: `UM-SSII Sudoku-2.txt 100 GATournamentSelector 1.95 0.1`

Solución:

```
El GA encuentra la solución ( 6 7 1 8 9 2 3 5 4 8 5 9 4 1 3 7 2 6 3 2 4 6 5 7 1 8 9 4 3 6
    ↪ 7 8 5 2 9 1 5 1 2 3 6 9 8 4 7 9 8 7 1 2 4 5 6 3 7 6 3 5 4 8 9 1 2 2 4 5 9 7 1 6 3 8
    ↪  1 9 8 2 3 6 4 7 5 )
fitness: 0
```

Como podemos observar, resuelve el sudoku (fitness 0) y ofrece su solución de forma lineal. Si queremos verlo en un formato mas amigable, tendríamos:

```
6 7 1 8 9 2 3 5 4
8 5 9 4 1 3 7 2 6
3 2 4 6 5 7 1 8 9
4 3 6 7 8 5 2 9 1
5 1 2 3 6 9 8 4 7
9 8 7 1 2 4 5 6 3
7 6 3 5 4 8 9 1 2
2 4 5 9 7 1 6 3 8
1 9 8 2 3 6 4 7 5
```

Caso 3

Como tiene 33 números iniciales usaremos:

Selector	Población	pc	pm
GATournamentSelector	100	0.95	0.1

Ejecución: `UM-SSII Sudoku-3.txt 100 GATournamentSelector 1.95 0.1`

Solución:

```
El GA encuentra la solución ( 1 5 9 7 2 4 8 6 3 8 2 4 1 3 6 5 9 7 7 6 3 9 8 5 1 2 4 9 4 2
    ↪ 3 6 8 7 1 5 3 7 8 5 1 9 2 4 6 5 1 6 2 4 7 9 3 8 4 8 7 6 9 1 3 5 2 6 3 1 8 5 2 4 7 9
    ↪  2 9 5 4 7 3 6 8 1 )
fitness: 0
```

Como podemos observar, resuelve el sudoku (fitness 0) y ofrece su solución de forma lineal. Si queremos verlo en un formato mas amigable, tendríamos:

```
1 5 9 7 2 4 8 6 3
8 2 4 1 3 6 5 9 7
7 6 3 9 8 5 1 2 4
9 4 2 3 6 8 7 1 5
3 7 8 5 1 9 2 4 6
5 1 6 2 4 7 9 3 8
4 8 7 6 9 1 3 5 2
6 3 1 8 5 2 4 7 9
2 9 5 4 7 3 6 8 1
```

Bibliografía

La referencia:

- 1: Es un video de youtube que explica el funcionamiento de un algoritmo genético, ha sido usado para explicar como funciona un algoritmo genetico.
- 2: Es un manual antiguo de la asignatura, ha sido usado para explicar los operadores de selección..

[1] J. M. Castillo, «5. Algoritmos genéticos», 2016 [Online]. Disponible en: https://www.youtube.com/watch?v=-rxGSe2ROX4

[2] J. M. C. Figueredo, M. del Carmen Garrido Carrera, R. M. España, y S. P. Moreno, «Material didáctico para la asignatura Sistemas Inteligentes de 3º de Grado en Informática», pp. 65-98, 2016 [Online]. Disponible en: http://ocw.um.es/ingenierias/material-de-practicas-de-sistemas-inteligentes/material-clase/materialssii.pdf