Curso
2018/19

Sistemas Inteligentes

JOSÉ MIGUEL SÁNCHEZ ALMAGRO
SUBGRUPO 2.2

PRÁCTICA 1 | Fecha: 08/11/2018

Índice

Memoria del Algoritmo Genético

¿Qué es un Algoritmo Genético?

Para poder entender qué es un Algoritmo Genético, primero hay que explicar que es una Búsqueda por haz local, y su derivada, búsqueda de haz estocástica. Si ya lo sabes, puedes saltar la explicación de ambos tipos de búsqueda.

Búsqueda por haz local. En un primer lugar genera estados aleatoriamente, y a cada paso, selecciona los *k* mejores sucesores y repite el proceso. El problema de este tipo de búsqueda es que, al elegir siempre a los *k* mejores, puede haber momentos en los que necesite más diversidad. Es decir, elegir por un momento sucesores malos para alcanzar en un futuro sucesores mejores que los que teníamos anteriormente.

Búsqueda de haz estocástica. Con el objetivo de solucionar el problema de la búsqueda anterior, en lugar de escoger a cada paso los *k* mejores candidatos, escoge *k* sucesores de manera aleatoria.

Algoritmo genético (AG). Es una variante de la búsqueda de haz estocástica, con la diferencia de que, a cada paso en lugar de generar un sucesor a partir de un candidato, mezcla dos candidatos para conseguir dos sucesores. Esto se puede ver como una analogía a la reproducción sexual. De esta manera podemos conseguir que, al **cruzar** dos individuos con buenas características de la generación anterior, obtengamos un individuo que tenga las mejores características de cada predecesor. Los primeros *k* estados generados aleatoriamente son llamados **población**, y cada estado también es llamado **individuo**. Este algoritmo además incluye un pequeño añadido: una vez que se ha producido el cruce entre dos individuos, existe una probabilidad de que se produzca una **mutación** en un individuo, esto es, que se cambie el valor de alguna posición del individuo. Existen muchos tipos de selección de los individuos de una generación a la hora de ser mezclados. El algoritmo termina su ejecución cuando se ha llegado al límite de generaciones, ha pasado mucho tiempo, o un individuo es suficientemente bueno. [1]

[1] La referencia corresponde a un libro, en el cual se ha basado la explicación de todo este apartado.

Cuestiones para el diseño e implementación

Pregunta 3.

InicioSudoku(), dado un Sudoku, rellena ese Sudoku (siempre y cuando tenga posiciones vacías) para que no quede ninguna posición vacía en el Sudoku. Pero lo hace de tal manera que las filas en el nuevo Sudoku tendrán fitness 0 (siempre y cuando el usuario no haya introducido un Sudoku erróneo). Por lo tanto, en las casillas donde antes había huecos (ceros), ahora hay números colocados de manera aleatoria. Si el usuario introduce un Sudoku completo, esta función no realiza ninguna modificación en el Sudoku.

Pregunta 4.

- Selección por Ruleta. En función de los idóneo que sea cada individuo, se le asigna un espacio en una ruleta (probabilidad proporcional), a mayor idoneidad, mayor espacio. Una vez tengamos la ruleta completa con la probabilidad de todos los individuos, se pone en marcha y se elige al individuo a que la ruleta haya seleccionado. Esto puede ocasionar que por azar seleccionemos a peores individuos. [2]
- Selección por Torneo. Se van escogiendo parejas de individuos al azar de la población. Una vez elegidos ambos individuos compiten entre sí, y el ganador será el seleccionado. Al igual que en el método anterior, los ganadores se escogen según su función de idoneidad. [3]

Pregunta 7.

Los individuos se están cruzando de manera aleatoria. Cada vez que se produce un cruce se escoge una posición al azar en ambos individuos, y se cruzan en a partir de esa posición.

Pregunta 9.

Si se encuentra una posición cuyo gen es modificable, se muta con una probabilidad introducida con el parámetro *float pmut*. Con una probabilidad del 50% se realiza una mutación intercambiando dos genes de una misma columna. Y con una probabilidad del 50% se realiza una mutación aleatoria.

Si la probabilidad de mutación es cero no se realiza ningún cambio.

Pregunta 10.

La condición de parada es si se ha conseguido una generación con un Sudoku de *fitness* cero. En caso afirmativo, el algoritmo dejará de evolucionar.

Pregunta 12.

Utilizamos un array auxiliar de enteros con la longitud de cada fila/columna/cuadrícula del Sudoku. Si por ejemplo encontramos en número 3, vamos a la posición 3 del array, y comprobamos que valor hay:

- Si hay un cero, colocamos un uno y seguimos avanzando.
- Si hay un uno, es que no es la primera vez que aparece este número en la fila/columna/cuadrícula actual, por lo que aumentamos el fitness.

Una vez entendido el funcionamiento de *aux*, es fácil entender cómo funciona el método. Para las filas y las columnas usamos el mismo procedimiento. Vamos avanzando fila/columna a fila/columna, y dentro, posición a posición, y ayudándonos de *aux* vamos incrementando (o no) el *fitness*.

En las cajas el procedimiento es el mismo. Vamos avanzando caja a caja, y dentro de cada caja comprobamos todas las posiciones/casillas/celdas. Lo que ocurre es que en las cajas no hay una manera inmediata y sencilla de ir avanzando por las celdas, por lo que se ha creado una variable auxiliar llamada *posAbs* que irá almacenando la posición absoluta (en el Sudoku) de la casilla en la que estamos actualmente. Vamos realizando operaciones de suma

[2] La referencia se refiere a una página web, y la definición de este documento se ha basado en el contenido de la referencia.
[3] La referencia se refiere a una página web, y la definición de este documento se ha basado en el contenido de la referencia.

y resta sobre esta posición para ir moviéndonos por el Sudoku, y así proceder de la misma manera que en las filas y en las columnas.

Tabla de análisis

Una vez implementado el código, se han introducido cinco casos de usuario, variando algunos parámetros del algoritmo genético, con el objetivo de encontrar la configuración de mejor rendimiento.

Los parámetros que se han dejado con valores fijos son:

- **Número de generaciones**: 12000.
- **Operador de cruce** (Implementado).
- **Operador de mutación** (Implementado).

Los parámetros variables, con sus respectivos posibles valores, son:

- **Operador de selección**: GARouletteWheelSelector y GATournamentSelector.
- **Tamaño de la población**: 100 y 150.
- **Probabilidad de cruce**: 0.8, 0.85, 0.9, 0.95.
- **Probabilidad de mutación**: 0.05, 0.1, 0.125, 0.15, 0.175.

A continuación, se expone una tabla en la que podremos observar el fitness que se ha obtenido con cada caso, y sus respectivos parámetros. Es decir, tenemos TODOS los posibles casos (ejecuciones) que podríamos tener dadas las variables y los casos.

Selector	TP	pc	pm	Caso-A1	Caso-A2	Caso-A3	Caso-A4	Caso-A5
GARouletteWheel	100	0.8	0.05	4	2	0	2	4
GARouletteWheel	100	0.8	0.1	0	0	2	4	2
GARouletteWheel	100	0.8	0.125	4	4	4	4	4
GARouletteWheel	100	0.8	0.15	5	0	2	4	4
GARouletteWheel	100	0.8	0.175	0	0	4	6	2
GARouletteWheel	100	0.85	0.05	4	2	0	7	2
GARouletteWheel	100	0.85	0.1	0	0	4	4	7
GARouletteWheel	100	0.85	0.125	2	0	6	4	2
GARouletteWheel	100	0.85	0.15	5	0	4	5	2
GARouletteWheel	100	0.85	0.175	7	0	2	7	2
GARouletteWheel	100	0.9	0.05	4	0	5	6	2
GARouletteWheel	100	0.9	0.1	4	0	0	2	4
GARouletteWheel	100	0.9	0.125	5	2	0	6	0
GARouletteWheel	100	0.9	0.15	2	4	4	4	4
GARouletteWheel	100	0.9	0.175	0	0	0	5	6
GARouletteWheel	100	0.95	0.05	0	0	2	0	4
GARouletteWheel	100	0.95	0.1	2	0	4	4	8
GARouletteWheel	100	0.95	0.125	0	2	4	0	8
GARouletteWheel	100	0.95	0.15	0	0	0	4	2
GARouletteWheel	100	0.95	0.175	0	2	5	5	0
GARouletteWheel	150	0.8	0.05	0	0	0	3	4
GARouletteWheel	150	0.8	0.1	0	2	2	5	5
GARouletteWheel	150	0.8	0.125	0	0	4	4	4
GARouletteWheel	150	0.8	0.15	0	0	2	6	4
GARouletteWheel	150	0.8	0.175	0	0	6	4	2
GARouletteWheel	150	0.85	0.05	0	0	0	4	4
GARouletteWheel	150	0.85	0.1	2	0	8	5	5
GARouletteWheel	150	0.85	0.125	4	0	2	6	2
GARouletteWheel	150	0.85	0.15	4	2	4	5	4
GARouletteWheel	150	0.85	0.175	0	0	8	7	4

GARouletteWheel	150	0.9	0.05	0	0	2	0	4
GARouletteWheel	150	0.9	0.1	0	2	2	2	2
GARouletteWheel	150	0.9	0.125	4	0	6	4	4
GARouletteWheel	150	0.9	0.15	0	0	4	6	8
GARouletteWheel	150	0.9	0.175	4	5	4	6	4
GARouletteWheel	150	0.95	0.05	0	2	0	5	2
GARouletteWheel	150	0.95	0.1	0	5	4	4	2
GARouletteWheel	150	0.95	0.125	0	9	2	4	6
GARouletteWheel	150	0.95	0.15	4	0	4	0	7
GARouletteWheel	150	0.95	0.175	4	5	0	8	6
GATournament	100	0.8	0.05	0	0	2	3	4
GATournament	100	0.8	0.1	0	0	0	0	0
GATournament	100	0.8	0.125	0	0	0	0	0
GATournament	100	0.8	0.15	0	0	0	0	0
GATournament	100	0.8	0.175	0	0	0	0	5
GATournament	100	0.85	0.05	0	2	0	0	2
GATournament	100	0.85	0.1	0	0	0	3	4
GATournament	100	0.85	0.125	0	0	0	0	0
GATournament	100	0.85	0.15	0	0	0	0	0
GATournament	100	0.85	0.175	0	0	0	0	0
GATournament	100	0.9	0.05	0	0	0	2	6
GATournament	100	0.9	0.1	0	0	2	4	0
GATournament	100	0.9	0.125	0	0	0	0	0
GATournament	100	0.9	0.15	0	0	0	0	0
GATournament	100	0.9	0.175	0	0	2	3	0
GATournament	100	0.95	0.05	0	0	2	0	0
GATournament	100	0.95	0.1	0	0	0	0	0
GATournament	100	0.95	0.125	0	0	0	0	0
GATournament	100	0.95	0.15	0	0	4	0	0
GATournament	100	0.95	0.175	0	0	2	0	4
GATournament	150	0.8	0.05	4	2	2	4	0
GATournament	150	0.8	0.1	0	0	0	0	0
GATournament	150	0.8	0.125	0	0	0	3	0
GATournament	150	0.8	0.15	0	0	0	0	0
GATournament	150	0.8	0.175	0	0	0	2	2
GATournament	150	0.85	0.05	2	0	0	4	0
GATournament	150	0.85	0.1	0	0	0	0	0
GATournament	150	0.85	0.125	0	0	0	0	0
GATournament	150	0.85	0.15	0	0	0	3	0
GATournament	150	0.85	0.175	0	0	0	0	2
GATournament	150	0.9	0.05	0	0	2	4	4
GATournament	150	0.9	0.1	0	0	0	3	0
GATournament	150	0.9	0.125	0	0	0	0	0
GATournament	150	0.9	0.15	0	0	0	0	0
GATournament	150	0.9	0.175	0	0	0	0	2
GATournament	150	0.95	0.05	0	0	2	4	5
GATournament	150	0.95	0.1	0	0	0	0	0
GATournament	150	0.95	0.125	0	0	0	0	0
GATournament	150	0.95	0.15	0	0	0	4	0
GATournament	150	0.95	0.175	0	0	0	0	0

Análisis de las pruebas de ajuste

Nuestro objetivo es encontrar en qué condiciones (con qué valores) se comporta mejor el algoritmo genético, para ello, vamos a realizar un análisis sobre la tabla del apartado anterior.

Algunas filas se han subrayado con colores. Esto sirve de filtro para observar los mejores y peores resultados. Se han marcado de color verde las filas que tienen en todos los casos de prueba un valor de *fitness* igual a cero. Se han marcado de amarillo las filas que tienen en 4 de los 5 casos de prueba un valor *fitness* cero, y en otro caso de prueba, un *fitness* de 2 o menor. Por último, se han marcado de rojo las filas que no tienen un valor *fitness* cero para ningún caso.

Selector

Los resultados arrojados por el Torneo y la Ruleta son completamente distintos. Cuando se ejecuta el algoritmo genético con la selección por **Torneo**, el *fitness* de las pruebas desciende considerablemente. Además, con la selección por torneo ya no tenemos filas marcadas en rojo, y tenemos filas amarillas y verdes. Con la selección por **Ruleta** no conseguimos tener ni una fila ni verde ni amarilla, y si varias rojas, por lo que independientemente del resto de parámetros, el resultado en todos los casos no será bueno: podremos conseguir en algunos casos *fitness* cero para algunas pruebas, pero no para todas ni casi todas. Por lo tanto, el selector se puede clasificar como un factor diferencial en los resultados del algoritmo.

Tamaño de la población

Los posibles valores que hay en la tabla son 100 y 150. En el caso de la selección por torneo, el número de filas rojas es igual cuando tenemos una población de **100** que cuando tenemos una población de **150** individuos. En la selección por torneo encontramos con **100** individuos, diez filas verdes y una amarilla; y con **150**, nueve filas verdes y dos amarillas. La diferencia es prácticamente nula, y el valor de este parámetro puede variarse sin sufrir una gran pérdida de rendimiento.

Probabilidad de cruce

En la selección por torneo y con una población de 100 individuos, es mejor tener un valor bajo, **0.8** y **0.85**, ya que son los únicos que tienen cuatro filas en verde. Sin embargo, con 150, es mejor usar el valor **0.95**, ya que es el único que tiene tres filas en verde. En la selección por ruleta los resultados son más aleatorios, sin aparentar grandes diferencias. Además, estamos interesados en obtener los mejores resultados, por lo tanto, no nos fijaremos en este tipo de selección.

Probabilidad de mutación

Como se ha indicado anteriormente, no es de nuestro interés la selección por ruleta, por lo que solo analizaremos los resultados pertenecientes a la selección por torneo.

- Con una probabilidad de **0.05**, no tenemos ninguna fila con todos los valores de *fitness* a cero, es decir, ninguna fila de color verde. Podemos concluir que no nos interesa en ningún caso utilizar este valor.
- Con los valores **0.1** y **0.15**, tenemos 5 filas verdes. Ambos son buenos valores, ya que en múltiples ocasiones con diferentes condiciones da buenos resultados.
- Con **0.125** de probabilidad, tenemos 7 filas verdes. Es aún mejor que el anterior, y podemos adelantar que es el mejor de todos. Este será por lo tanto el que utilizaremos.
- Finalmente, con una probabilidad de **0.175**, contamos 2 filas verdes y 2 amarillas. Tampoco estaremos interesados en usar este valor de probabilidad.

Valorando todo lo expuesto en este apartado, obtenemos las siguientes configuraciones:

Selección: **Torneo**

Tamaño de la población: **100** / **150**

Probabilidad de cruce: **0.8** o **0.85** / **0.95**

Probabilidad de mutación: **0.125**

Elegiremos la configuración: **Torneo**, **100**, **0.8**, **0.125**. ¿Por qué? En un principio nos es indiferente, pero a menor número de individuos, el tiempo y la memoria necesarios por el algoritmo será menor. La probabilidad de cruce ha sido elegida de manera arbitraria.

Manual de asignación

Una vez elegida la mejor configuración, explicaremos como ejecutar el algoritmo.

Se le proporcionará al usuario un programa con extensión *.exe*. Este debe ejecutarse en la terminal de Windows, en una instrucción, a la que se le deben pasar seis parámetros:

```
Sudoku.exe MiSudoku.txt Tamaño_población 12000 Probabilidad_cruce
Probabilidad_mutación Selección
```

MiSudoku.txt será el archivo donde se encuentre el Sudoku que el usuario desea resolver (este archivo debe tener una extensión *.txt*, y se debe encontrar en el mismo directorio/carpeta que el archivo *.exe*).

Partiendo de la mejor configuración elegida anteriormete, la orden la ejecutar es la siguiente:

```
Sudoku.exe MiSudoku.txt 100 12000 0.8 0.125 1
```

La salida devolverá al principio el tamaño del Sudoku y la configuración que hemos elegido para resolverlo. Finalmente, se devolverá el Sudoku que ha podido resolver el Algoritmo Genético con su valor fitness, es decir, cuantas veces ha repetido algún número cometiendo un error.

Si el valor fitness obtenido con la primera ejecución no es cero, se recomienda al usuario probar con las otras configuraciones que también dieron buenos resultados (expuestas en el apartado de análisis):

```
Sudoku.exe MiSudoku.txt 100 12000 0.85 0.125 1
```

```
Sudoku.exe MiSudoku.txt 150 12000 0.95 0.125 1
```

Casos de usuario

Sudoku-1

Comando: `Sudoku.exe Sudoku-1.txt 100 12000 0.8 0.125 1`

Resultado:

```
Sudoku de tamaño 9x9

Parametros:    - Tamano poblacion: 100
               - Numero de generaciones: 12000
               - Probabilidad cruce: 0.8
               - Probabilidad mutacion: 0.125

El GA encuentra la solucion

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

con valor fitness 0
```

Sudoku-2

Comando: `Sudoku.exe Sudoku-2.txt 100 12000 0.8 0.125 1`

```
Sudoku de tamaño 9x9

Parametros:    - Tamano poblacion: 100
               - Numero de generaciones: 12000
               - Probabilidad cruce: 0.8
               - Probabilidad mutacion: 0.125

El GA encuentra la solucion

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

con valor fitness 0
```

Sudoku-3

Comando: `Sudoku.exe Sudoku-3.txt 100 12000 0.8 0.125 1`

```
Sudoku de tamaño 9x9

Parametros:    - Tamano poblacion: 100
```

```
                - Numero de generaciones: 12000
                - Probabilidad cruce: 0.8
                - Probabilidad mutacion: 0.125

El GA encuentra la solucion

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

con valor fitness 0
```

Sudoku-4

Comando: `Sudoku.exe Sudoku-4.txt 100 12000 0.8 0.125 1`

```
Sudoku de tamaño 4x4

Parametros:     - Tamano poblacion: 100
                - Numero de generaciones: 12000
                - Probabilidad cruce: 0.8
                - Probabilidad mutacion: 0.125

El GA encuentra la solucion

1 2 4 3
3 4 2 1
2 1 3 4
4 3 1 2

con valor fitness 0
```

Bibliografía

REFERENCIAS

[1] S. Russell, P. Norvig (2004). “Inteligencia Artificial (2ª ed.), Un enfoque moderno”. Pearson Prentice Hall, section 4.3, pp. 131--135.

[2] M. Gestal (2006). Introducción a los Algoritmos Genéticos [Online]. Available: http://sabia.tic.udc.es/mgestal/cv/AAGGtutorial/

[3] Wikipedia (2018). Selección por torneos [Online]. Available: https://es.wikipedia.org/wiki/Selección_por_torneos