Universidad de Murcia

Sistemas Inteligentes

Grado en Informática: 2018/2019 Curso

Algoritmo Genético - Sudoku

Alumno:

Pardo García, Federico

Grupo:

1.1

Profesor:

Garrido Carrera, María del Carmen

Fecha de entrega:

21 de noviembre de 2018

Índice general

1. ¿Qué es un Algoritmo Genético?

Un algoritmo genético (AG) pertenece a un conjunto algorítmico de búsqueda estocástica basados en la imitación de una evolución biológica. [1] Esto quiere decir que su objetivo es imitar el comportamiento de un conjunto de seres vivos, en el que generación tras generación, solo los mejores sobreviven. Esto se puede traducir como que el programa va mejorándose a sí mismo, obteniendo (supuestamente) soluciones superiores en cada iteración mediante la combinación de diferentes individuos.

Este tipo de algoritmos consisten en un procedimiento que comienza con la selección aleatoria de los datos individuales (genes) que conforman los cromosomas. Estos serán los denominados "generación inicial" y deben ser sometidos al cálculo de su *"fitness"* o función de idoneidad, que no es más que un valor representando cómo de bueno es cada individuo para obtener la solución deseada. El cálculo de la función de idoneidad es lo que diferencia a un AG de otro, además de su representación. Las demás partes pueden ser fácilmente reciclables cuando se requiere implementar un nuevo AG, aunque sea en un contexto completamente distinto. Una vez tenemos el valor, se seleccionan dos de los mejores individuos para combinarlos y crear un nuevo cromosoma. La probabilidad de que un individuo sea seleccionado depende de la *"probabilidad de cruce"* que se le haya indicado al algoritmo. A continuación, se le aplican mutaciones, cambiando valores de forma aleatoria a nivel de gen. La cantidad de elementos que pueden ser sometidos a mutación en cada cromosoma es definida por la *"probabilidad de mutación"*, también indicada al comienzo del programa. Una vez tenemos tantos cromosomas como en la generación anterior, podemos volver a calcular su función de idoneidad y repetir todo el proceso hasta que se cumpla el criterio de parada, el cual puede ser definido como encontrar la solución óptima o llegar a un límite de generaciones en caso de no poder encontrarla.

2. Cuestiones para el diseño e implementación

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

Se comienza generando un array auxiliar inicializado a 0 del tamaño indicado en el encabezado del fichero. Por ejemplo, en el caso de un sudoku 9x9, este tamaño sería de 9. A continuación, rellenamos de forma aleatoria el array con los números que pueden formar parte de la solución. En este ejemplo, valores en el intervalo [1-9].

Una vez llegados a este punto, la función comprueba que los valores del array auxiliar escogidos no hagan conflicto con los números iniciales del sudoku, que no pueden ser modificados a la hora de buscar una solución. Estos son reconocidos por tener valores distintos de 0. Cuando ya hemos comprobado que no existen colisiones, el array auxiliar es volcado al cromosoma, representando la primera fila del sudoku. Por tanto, este proceso hay que repetirlo tantas veces como el tamaño indicado del tablero.

Indica el funcionamiento de los operadores de selección indicados en la sección "Ajustes del Algoritmo Genético"

Para el estudio y resolución de este problema, se nos proponen dos operadores de selección:

- **GARouletteWheelSelector:** Este método de selección escoge un individuo basándose en la magnitud de su fitness en comparación con los demás de su población. A mayor puntuación, más posibilidades tiene de ser escogido. Todos los individuos tienen una probabilidad p de ser escogidos, siendo p igual al fitness del individuo dividido entre la suma de los valores fitness de cada individuo en la población.
- **GATournamentSelector:** El selector de torneo utiliza el método de RouletteWheel para escoger dos individuos y entonces seleccionar el de mejor puntuación. Este selector suele escoger individuos mejor valorados más a menudo que el selector *RouletteWheelSelector*.

Explica de qué manera se están cruzando los individuos

Se está realizando una operación de cruce por un punto, en la cual se generan dos hijos, procedentes de dos padres. Primero se selecciona un punto aleatorio (p_1) en el array de genes y en base a este, se selecciona un segundo (p_2) que es calculado restando el primero al número de genes que componen un individuo. Una vez tenemos ambos puntos, generamos el primer hijo. Esto se consigue copiando desde la posición 0 hasta p_1 de un padre y los elementos restantes del otro (representados por la longitud p_2). De forma análoga se realizaría el proceso para generar el segundo hijo, permutando los padres en el proceso. Este procedimiento es representado de forma gráfica en la figura 1.

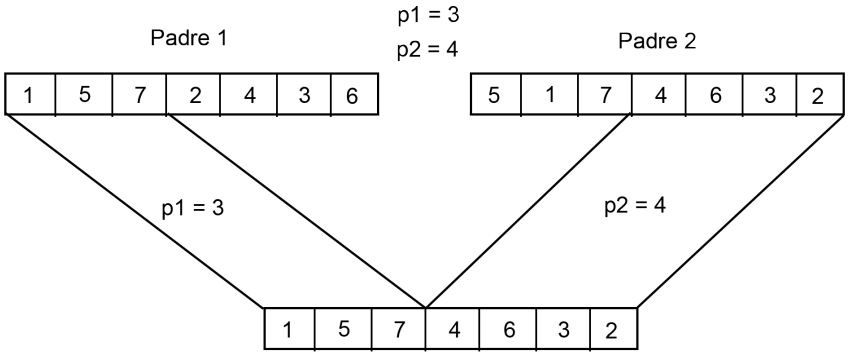

Figura 1: Representación del cruce de individuos

Explica de qué manera se están mutando los individuos

Se recorre el array de genes en cada individuo de la siguiente forma:

Primero se comprueba si el valor que se está analizando es parte de los valores fijos del sudoku original. En caso de no serlo, se modificará o no el valor según la probabilidad de mutación establecida. Si se decide que el valor debe mutar, se determina si mutar valores por filas o columnas con una probabilidad del 50 %.

En el caso de ser por columnas, se comprueba si hay repetidos en la columna actual y si el resultado es negativo, no se ejecuta mutación alguna. Si los hay, se selecciona un número con más de una repetición (v_1) y otro que no aparezca (v_2). Como desconocemos las posiciones con elementos repetidos, debemos buscarlas. Cuando encontremos una, cambiamos su valor por el del número sin apariciones (v_2) seleccionado anteriormente y almacenamos su fila. A continuación, buscamos en esa fila el valor v_2 y si se encuentra, es reemplazado por v_1

En caso de ser por filas, las posiciones de dos valores pertenecientes a la misma fila son permutados.

Define y explica la condición de parada que utilizarás

Con anterioridad, se nos mostró un ejemplo en código de un AG para resolver el problema de las *N-Reinas*. En él, la condición de parada consistía en que algún individuo de la generación actual tenga un valor fitness 0, representando que ha encontrado la solución óptima, o en el caso de no haber podido encontrarla tras un número definido de generaciones, parar el algoritmo y devolver la mejor solución hasta el momento. Dado que el problema es muy similar, he encontrado factible la reutilización de esta función en mi código.

Diseña y explica la función fitness que utilizarás. Recuerda, como se indica en el comienzo de este guión, que una solución del sudoku no puede repetir en una misma fila, columna, o subcuadrícula ninguno de los números

La función fitness diseñada premia los individuos con menor puntuación. El cálculo de este valor se realiza mediante el siguiente algoritmo:

Las siguientes tres fases se realizan tantas veces como el tamaño del tablero indicado. En cada iteración, estaremos comprobando un número en particular.

- **Comprobación de filas:** Es el conteo más sencillo, pues consiste en recorrer el genoma del individuo restableciendo el contador a 0 cada vez que se alcance el tamaño de tablero especificado.
- **Comprobación de columnas:** Mismo principio que en la comprobación de filas, pero recorriendo por columnas, sumando el tamaño del tablero para pasar de una "columna" a otra.
- **Comprobación de subcuadrados:** Una combinación de las dos aproximaciones anteriores. Es el recorrido más abstracto, pues se recorren varios arrays de dos dimensiones en uno lineal. Consiste en aumentar según el tamaño del tablero, el índice que indica la posición actual. Una vez hemos terminado de recorrer un subcuadrado, pasamos al siguiente basándonos en cuantos hemos recorrido ya.

Al finalizar cada una de las subsecciones anteriores, comprobamos si hemos encontrado más de una repetición del valor que estamos comprobando. En caso de cumplir la condición, sumamos el número de repeticiones menos uno a la puntuación obtenida por el individuo, obteniendo así su calificación final.

3. Ajuste del algoritmo genético

3.1. Tabla del selector GARouletteWheel

Selector	Población	P.C	P.M	A1	A2	A3	A4	A5
GARoulette	100	0.8	0.05	4	2	0	2	4
GARoulette	100	0.8	0.1	0	0	2	4	2
GARoulette	100	0.8	0.125	4	4	4	4	4
GARoulette	100	0.8	0.15	5	0	2	4	4
GARoulette	100	0.8	0.175	0	0	4	6	2
GARoulette	100	0.85	0.05	4	2	0	7	2
GARoulette	100	0.85	0.1	0	0	4	4	7
GARoulette	100	0.85	0.125	2	0	6	4	2
GARoulette	100	0.85	0.15	5	0	4	5	2
GARoulette	100	0.85	0.175	7	0	2	7	2
GARoulette	100	0.9	0.05	4	0	5	6	2
GARoulette	100	0.9	0.1	4	0	0	2	4
GARoulette	100	0.9	0.125	5	2	0	6	0
GARoulette	100	0.9	0.15	2	4	4	4	4
GARoulette	100	0.9	0.175	0	0	0	5	6
GARoulette	100	0.95	0.05	0	0	2	0	4
GARoulette	100	0.95	0.1	2	0	4	4	8
GARoulette	100	0.95	0.125	0	2	4	0	8
GARoulette	100	0.95	0.15	0	0	0	4	2
GARoulette	100	0.95	0.175	0	2	5	5	0
GARoulette	150	0.8	0.05	0	0	0	3	4
GARoulette	150	0.8	0.1	0	2	2	5	5
GARoulette	150	0.8	0.125	0	0	4	4	4
GARoulette	150	0.8	0.15	0	0	2	6	4
GARoulette	150	0.8	0.175	0	0	6	4	2
GARoulette	150	0.85	0.05	0	0	0	4	4
GARoulette	150	0.85	0.1	2	0	8	5	5
GARoulette	150	0.85	0.125	4	0	2	6	2
GARoulette	150	0.85	0.15	4	2	4	5	4
GARoulette	150	0.85	0.175	0	0	8	7	4
GARoulette	150	0.9	0.05	0	0	2	0	4
GARoulette	150	0.9	0.1	0	2	2	2	2
GARoulette	150	0.9	0.125	4	0	6	4	4
GARoulette	150	0.9	0.15	0	0	4	6	8
GARoulette	150	0.9	0.175	4	5	4	6	4
GARoulette	150	0.95	0.05	0	2	0	5	2
GARoulette	150	0.95	0.1	0	5	4	4	2
GARoulette	150	0.95	0.125	0	9	2	4	6
GARoulette	150	0.95	0.15	4	0	4	0	7
GARoulette	150	0.95	0.175	4	5	0	8	6

Cuadro 1: Resultados de GARouletteWheel

Como podemos observar, los resultados no son nada brillantes con el selector *GARouletteWheel*. La mayoría de ajustes apenas consiguen resolver algún casos de prueba, lo que indica la poca eficacia del

selector. Sobretodo comparándolo con su otra opción, que vemos a continuación.

3.2. Tabla del selector GATournament

Selector	Población	P.C	P.M	A1	A2	A3	A4	A5
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

Cuadro 2: Resultados de GATournament

A la vista de los resultados, vemos claramente que hay una mejora significativa de *GATournament* respecto al anterior selector. Apenas unas pocas configuraciones no resuelven 4 de los 5 casos de prueba, y la mayoría completa todos.

4. Análisis de las pruebas de ajuste

Tras observar detenidamente ambas tablas y analizar sus resultados, podemos hacer una especificación de los mejores parámetros para resolver el problema dado. Analizaremos la tabla de izquierda a derecha.

- **Operador de selección:** Se puede observar a simple vista que el selector *GARoulette Wheel* es el que peor comportamiento tiene, ya que no consigue resolver con ninguna de las combinaciones dadas para el estudio, los 5 casos simultáneamente. Por tanto, escogeremos el selector *GATournament*.
- **Tamaño de población:** Ambas cantidades consiguen resultados muy parejos. Si bien, el tamaño *100* se impone, con una mayor regularidad y número de aciertos en todas las configuraciones.
- **Probabilidad de cruce (P.C):** Con tamaño de población fijado en *150*, se aprecian valores muy buenos en el intervalo *0.85-0.95*, mientras que con *100*, los resultados son más dispersos, rondando los mejores entre *0.8* y *0.85*.
- **Probabilidad de Mutación (P.M):** El estudio muestra grandes resultados con los valores en el intervalo *0.1-0.15*. Por tanto, podríamos fijar estos parámetros como los más recomendados en la sección *Manual-Asignación*.

Observando estos datos, podemos decir que los parámetros están claramente definidos salvo para la *Probabilidad de Cruce*, cuyo comportamiento nos indica que su valor debe ascender, según lo haga el *Tamaño de Población*.

5. Manual de Asignación

5.1. Ejecución del programa

El programa carece de una interfaz gráfica, y por tanto es necesario ejecutarlo desde la línea de comandos para hacerlo funcionar. Una forma rápida de abrir la consola de Windows es por medio del menú de *Ejecutar*, que se abre mediante la combinación de teclas *Win + R*. Escribimos "powershell" en la ventana que nos aparece y ejecutamos la orden. Una vez aquí, debemos movernos hasta la carpeta que contenga el programa y archivo con el Sudoku que queremos resolver. Esto se consigue mediante el siguiente comando:

$$> \text{cd } \backslash\text{ruta}\backslash\text{hasta}\backslash\text{la}\backslash\text{carpeta}$$

Una forma más sencilla de llegar hasta la carpeta es mediante el *Explorador de Archivos*. Podemos abrir la carpeta donde se encuentre el programa y allí pulsar: *Archivo > Abrir Windows Power_Shell*.

A continuación, ya podemos ejecutar el programa mediante la siguiente sintaxis:

$$> .\backslash\text{sudoku.exe nombre\_fichero.txt parámetro1 parámetro2 parámetro3 parámetro4}$$

El archivo *nombre_fichero.txt* incluye el caso que queremos resolver. En la primera línea únicamente estará un número n con el tamaño del sudoku. A partir de ahí, se esperan n líneas de la misma longitud, separados por espacios con los valores del tablero. Las casillas vacías serán representadas por el valor 0,

5.2. Asignación de parámetros

Para dar opciones al usuario, se proponen 2 configuraciones para la resolución del sudoku. La primera más conservadora que prima encontrar una solución y la segunda más agresiva centrada en la velocidad de ejecución. Ambas descripciones son teóricas. Puede darse el caso de que la más precisa sea en algunos casos la más rápida y viceversa.

Configuración conservadora

- Valor recomendado para el parámetro 1 (Tamaño de población): 150
- Valor recomendado para el parámetro 2 (Operador de selección): 0
- Valor recomendado para el parámetro 3 (Probabilidad de cruce): 0.95
- Valor recomendado para el parámetro 4 (Probabilidad de mutac.): 0.125

Configuración agresiva

- Valor recomendado para el parámetro 1 (Tamaño de población): 100
- Valor recomendado para el parámetro 2 (Operador de selección): 0
- Valor recomendado para el parámetro 3 (Probabilidad de cruce): 0.8
- Valor recomendado para el parámetro 4 (Probabilidad de mutac.): 0.125

Una menor población iterará más rápido, producirá más mutaciones y es más variable. Por otro lado, una población mayor es más lineal y menos agresiva en las mutaciones.

6. Casos de Usuario

Los siguientes casos han sido resueltos por un usuario real, haciendo uso del manual de asignación.

6.1. Sudoku-1

El comando introducido fue el siguiente:

>.\sudoku.exe Sudoku-1.txt 100 12000 0.8 0.125

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 un valor fitness: 0

6.2. Sudoku-2

El comando introducido fue el siguiente:

>.\sudoku.exe Sudoku-2.txt 100 12000 0.8 0.125

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

6.3. Sudoku-3

El comando introducido fue el siguiente:

>.\sudoku.exe Sudoku-3.txt 100 12000 0.8 0.125

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

6.4. Sudoku-4

El comando introducido fue el siguiente:

>.\sudoku.exe Sudoku-4.txt 100 12000 0.8 0.125

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

Con valor fitness:0

7. Bibliografía

[1] M. Negnevitsky (2005). *"Artificial Intelligence - A Guide to Intelligent Systems"*, Second Edition, Section 7.3, pp. 222-223.