INTELIGENCIA ARTIFICIAL

Tema: Algoritmos genéticos

Aplicación: Resolución de sudoku

Integrantes: Bearzotti, Kozelnik, Lagares, Rodriguez,Tarnoski

Introducción

En este trabajo exploramos la resolución del juego “Sudoku”, que consiste en un tablero de 9x9 en el que se debe llenar 9 cuadrantes de 3x3 con números del 1 al 9. La dificultad del juego radica en que se debe evitar la repetición de estos números tanto a lo ancho de las filas como columnas del tablero y en cada cuadrante. Resolver el Sudoku con algoritmos genéticos nos pareció una buena idea debido a su capacidad para explorar soluciones y orientarse hacia las mejores posibles. Los algoritmos genéticos se destacan en su capacidad para abordar desafíos complicados y buscar soluciones en diversas direcciones, evitando quedar atrapados en soluciones subóptimas. Esto último cobra especial relevancia en el Sudoku, donde la búsqueda de la solución óptima puede ser un gran desafío.

Definiciones

Explicamos algunos de los conceptos que fueron utilizados durante el desarrollo del trabajo, con el fin de esclarecer los distintos procesos que van a ser demostrados en el mismo:

- Algoritmos geneticos: Los algoritmos genéticos son técnicas de búsqueda y optimización inspiradas en la teoría de la evolución. Estos algoritmos se utilizan para encontrar soluciones aproximadas u óptimas a problemas complejos al imitar procesos biológicos como la selección natural, la reproducción y la mutación.
- Genotipo: El genotipo en un algoritmo genético se refiere a la representación codificada de una solución potencial en forma de una cadena de genes o cromosomas. Estos genes contienen información que se utiliza para construir la solución al problema que se está resolviendo.
- Fenotipo: El fenotipo en un algoritmo genético es la manifestación física o la representación de la solución potencial decodificada a partir del genotipo. Es la solución real o candidata que se evalúa y se compara con otras soluciones en términos de su calidad.
- Función de aptitud (fitness): La función de aptitud es una función que evalúa cuán buena es una solución potencial representada por su fenotipo en relación con el objetivo del problema. Cuanto mayor sea el valor de aptitud, mejor será la solución. La función de aptitud guía la selección de soluciones durante el proceso de evolución.
- Selección y Crossover: La selección es el proceso en el que se eligen soluciones potenciales (fenotipos) de una población actual para formar una nueva población. El crossover (o cruzamiento) es un operador genético que combina información genética de dos soluciones (padres) para crear nuevas

soluciones (hijos) en la nueva población. Estos dos procesos son esenciales en la reproducción y evolución de soluciones.

- Método del dardo: El método del dardo es una técnica de selección utilizada en algoritmos genéticos que se basa en asignar probabilidades a las soluciones de la población actual para ser seleccionadas como padres. Las soluciones con una mayor aptitud tienen una mayor probabilidad de ser seleccionadas, similar al lanzamiento de un dardo hacia un tablero.
- Mutación: La mutación es un operador genético que introduce cambios aleatorios en el genotipo de una solución potencial. Estos cambios aleatorios permiten explorar nuevas regiones del espacio de búsqueda y evitar la convergencia prematura hacia soluciones subóptimas. La tasa de mutación controla la probabilidad de que ocurra una mutación en un gen.

Realización

Se comienza estableciendo una matriz como tablero inicial. Esta matriz representa un tablero de sudoku, con valores iniciales y otros valores faltantes. Los valores faltantes, que deberán ser completados por el algoritmo, se inicializan con el valor 0.

Para la definición de valores decidimos que la población consistirá de 100 individuos, la tasa de mutación de 0,3 y el tope de generaciones será de 10000.

Para cada individuo se calcula su fitness. Este se determina a partir de la función de aptitud, en este caso lineal, la cual corrobora que en las filas, columnas y secciones no haya valores repetidos. Esto lo hace recorriendo cada fila, columna y cuadrante e incrementando en 1 un contador de valores únicos. Este contador es el fitness.

Primero se analizan las filas. Al finalizar, si no hay errores por filas, es decir si cada fila tiene valores únicos, fitness va a tener un valor de 81 (9 valores únicos en cada fila (9 filas)).

Luego pasa a analizar las columnas de la misma manera. Si no hay errores, el fitness se incrementa en 81 nuevamente (9 valores únicos en cada columna (9 columnas)).

Luego pasa a analizar por sectores de la misma manera. Si no hay errores, el fitness se incrementa en 81 nuevamente (9 valores únicos en cada sector (9 sectores)).

De esta manera si no hubiera errores en el tablero, el fitness sería de 243. De otra manera el fitness sería más bajo.

El algoritmo llegará a su fin si encuentra alguna solución que alcanzó ese valor de fitness. Si obtuvo este valor es porque encontramos una posible solución al sudoku. De lo contrario, el algoritmo continúa su ejecución.
Luego de esta evaluación del fitness se crea la próxima generación. Esto se repite hasta alcanzar el tope de generación preestablecido.
Decidimos aplicar el método de dardo como algoritmo de selección debido a que nuestro fitness va a tener como máximo 243, pero depende de cada caso qué valores obtiene. Como la solución tiene valores previamente establecidos que son parte del tablero, no tendrá tantas opciones para variar. Por lo tanto, la mayoría de los tableros estarán cerca de ese número(como por ejemplo 204, 210, 240, 230). Si lo pensamos en una misma escala, como 1/fitness, se obtendría resultados muy pequeños. Como por ejemplo: 1/243 (0,00411) , 1/239 (0,00418), 1/229 (0,0043) etc. Lo cual es una probabilidad muy pequeña y difícil de manejar. Tengo muchos individuos con una baja probabilidad de ocurrencia. Nos pareció que la mejor manera de resolverlo es otorgando a cada individuo una probabilidad con respecto al resto. Quien tenga mayor fitness, tendrá mayor probabilidad. Se compara con respecto al resto. En cambio si hubiéramos usado otro métodos como el mating pool, tendrían un baja probabilidad de ocurrencia, además de que después de cada generación el array mating pool va a ser cada vez más grande, por lo tanto va a utilizar más capacidad de cómputo.
A través de este método se obtienen las probabilidades de cada individuo.
Luego se seleccionan 2 padres a partir de estas probabilidades obtenidas. Se realizará el crossover con estos dos padres. El método de crossover consta de tomar 2 padres, y obtener un número al azar que representará un punto de corte en su matriz, que es una fila determinada. Luego se crearán dos hijos: el primero estará conformado por las filas del primer padre hasta el punto de corte y la segunda parte del segundo padre desde el punto de corte. El otro hijo se creara con las primeras filas del segundo padre hasta el punto de corte y la segunda parte del punto 1 desde el punto de corte.
Una vez obtenidos los hijos, se calculará a través de un número aleatorio si ese hijo recibirá una mutación o no, comparando este número con la tasa de mutación. En el caso de que se determine que se realizará la mutación, se elige al azar una fila y columna donde el valor cambiará. Se comprobará que la posición elegida para la mutación no representa un valor que fuera parte del tablero original, y que el nuevo valor elegido para esa posición sea válido respecto a las reglas. Si se comprueban estas dos condiciones correctamente, se realiza la mutación.
Este proceso de selección de padres y creación de hijos se realizará hasta que se obtengan 100 hijos, y se reemplazará a la población inicial por estos

hijos. Luego se volverá a calcular el fitness para verificar si se ha encontrado la solución. De lo contrario, pasará a la siguiente generación volviéndo a ejecutar el código ya descrito.

Conclusiones

Dado nuestro algoritmo podemos observar que a medida que aumentamos la dificultad de cada sudoku deja de encontrar la solución y se acerca con su valor de fitness a 243 pero nunca llega al mismo debido al límite de generaciones que hemos establecido en el programa para evitar que siga indefinidamente. A partir de que aumente la dificultad podremos variar la población, tasa de mutación y generaciones topes para encontrar la solución.