listings

Algoritmos Genéticos y Evolutivos

Práctica 1: N Reinas.

Autor:

Daniel Fernández: daniel.f.rico@alumnos.uc3m.es

10 de octubre de 2017

Índice

1. Contexto.

1.1. Problema de las 8 reinas.

El problema de las ocho reinas consiste en poner ocho reinas en un tablero de ajedrez (de 8 casillas) sin que se amenacen

En ajedrez, las reinas pueden moverse tantas casillas como quieran en cualquier dirección. Dado un tablero de dimensiones NxN, queremos colocar N reinas sin que se ataquen entre ellas

El problema original data de 1848 y fue propuesto por el ajedrecista Max Bezzel. Durante años, muchos matemáticos (incluyendo a Gauss), han trabajado en él y lo han generalizado a N-reinas. Las primeras soluciones se plantean en 1850.

Edsger Dijkstra usó este problema en 1972 para ilustrar el poder de la llamada programación estructurada. Publicó una descripción muy detallada del desarrollo del algoritmo de backtracking, "depth-first".

1.2. Problema planteado.

Se plantea resolver el problema de las N-reinas para un valor de N muy inferior a 1000 sin necesidad de hallar todas las soluciones, mediante la utilización de un Algoritmo Genético

2. Solucionar las N-reinas empleando fuerza bruta.

2.1. ¿Cuánto se tarda en obtener la primera solución?

2.2. ¿Cuánto se tardan en obtener soluciones subsecuentes?

3. Solucionar las N-reinas con Algoritmos Genéticos.

3.1. Codificación.

La codificación actual se trata de un vector de tamaño N donde cada valor corresponde a la posición x de una reina sobre la fila y del tablero, siendo y el índice del elemento x en el vector de reinas.

3.1.1. ¿Por qué es mejor que otras alternativas?

Esta codificación es mucho más eficiente que las codificaciones anteriormente propuestas (descritas abajo), ya que ocupa mucho menos espacio en memoria (un vector de ints ocupa $K * N$ bits, siendo $K = 4$ para una arquitectura estándar de 32-bit o 64-bit, frente a la peor solución anteriormente propuesta, un vector $N * N$ binario, que ocuparía $N * N$ bits).

Además, esta codificación lleva implícitas las restricciones de fila y columna para cada reina. La restricción fila va implícita en la propia estructura de datos (no puede haber dos elementos en la misma posición de la lista), y la restricción columna va implícita en el hecho de que los valores del vector son permutaciones de N (no se repite ningún valor de x).

Esto nos elimina automáticamente muchas soluciones inválidas de la población, agilizando el algoritmo y permitiendo encontrar una solución mucho más rápidamente.

3.1.2. ¿Qué otras alternativas se han barajado y descartado?

En primer lugar, se planteó una codificación del tablero como vector binario de tamaño NxN, de tal modo que cada individuo de la población fuese un vector de 0s y 1s, donde un 1 equivale a una reina en la posición equivalente del tablero, y un 0 la ausencia de ésta.

En segundo lugar, se planteó un vector de reinas de tamaño N, donde sólo se acumulasen las reinas que se encontraban en el tablero. Dicho vector estaría compuesto de tuplas (x, y) correspondientes a la posición de cada reina en el tablero.

Al darnos cuenta de que los números x e y han de ser excluyentes (restricción para evitar ataques en la misma fila y columna), se planteó la codificación actual que hace uso del índice de la lista como posición y de la reina (unívoca).

3.2. Función de fitness.

3.3. Métodos

- Mutacion: 1. Primera aproximación: se mutan los dos hijos con probabilidad p_m. Aproximación clásica: por cada bit del individuo, se evalúa su probabilidad de mutación, y se cambia el bit (0-1). Al no ser codificación binaria, lo cambiamos por valores 0-N.

2. Segunda aproximación: con (1) obtenemos resultados repetidos, nuestra codificación es una permutación de columnas (valores 0-N). Por lo tanto, evaluamos la probabilidad del individuo entero de mutar, y si es positiva hacemos un reordenamiento de sus valores (shuffle).

3. Tercera aproximación: con (2) obtenemos valores demasiado aleatorios con una probabilidad de mutación elevada. Nuestra aproximación esta vez consiste de nuevo en evaluar la probabilidad de mutación bit a bit, y en caso de ser positiva, intercambiando dicho bit con otro bit del individuo de manera aleatoria. De esta forma, en caso de mutar 1 bit, el individuo sólo diferirá del original en 2 bits.

Nota: la probabilidad de mutar depende de N, ya que el número de bits a mutar depende también del tamaño del individuo, así que para hacerlo más constante tomamos la probabilidad de 1/N y la multiplicamos por la probabilidad por individuo.

- Cruce:

3.4. Parametrización

Parámetros propuestos, e impacto de los mismos en los resultados, incluyendo gráficas y tablas que sean pertinentes

- Capas: **3**.
- Momentum: **0.2455**.
- Número de ciclos: **400**.

- El conjunto de datos, con un número reducido de patrones de entrada, requiere un modelo con una tasa de aprendizaje alta. Se fijó este valor en **0.61** y sin decaer durante el entrenamiento.

La elección de dichos parámetros se basó en una comparativa de los distintos valores posibles, como podemos ver en la siguiente figura:

Figura 1: Evolución del acierto en la clasificación según los parámetros escogidos para el entrenamiento

3.5. Resultados

Resultados obtenidos y número de evaluaciones necesarias

4. Conclusiones, Problemas Encontrados y Opiniones Personales

5. Anexo