Instituto Politécnico Nacional

Programa Académico: Ingeniería en Sistemas Computacionales
Unidad de Aprendizaje: Análisis de Algoritmos
Alumno: Nava Álvarez José Andrés
Fecha: 29/11/2019

Problema de las n-Reinas

Tabla de contenido

Introducción 2

Marco Teórico 2

Problema del Caballo ..... **¡Error! Marcador no definido.**

Programacion Dinamica ..... **¡Error! Marcador no definido.**

Desarrollo 4

Resultados ..... **¡Error! Marcador no definido.**

Conclusiones 5

Introducción

En el presente reporte se muestran los resultados de operar con un programa que resuelve las n-Reinas por medio de algoritmos genéticos. Los objetivos de este ejercicio es utilizar la variación de parámetros al momento de crear configuraciones, para así poder conseguir una solución lo más eficientemente posible.

Marco Teórico

Problema de las N reinas

El problema de las n reinas es un pasatiempo que consiste en poner n reinas en el tablero de ajedrez sin que se amenacen. Fue propuesto por el ajedrecista alemán Max Bezzel en 1848.

En el juego del ajedrez la reina amenaza a aquellas piezas que se encuentren en su misma fila, columna o diagonal. El juego de las n reinas consiste en poner sobre un tablero de ajedrez n reinas sin que estas se amenacen entre ellas.

Algoritmos Genéticos

Un algoritmo genético puede presentar diversas variaciones, dependiendo de cómo se aplican los operadores genéticos (cruzamiento, mutación), de cómo se realiza la selección y

de cómo se decide el reemplazo de los individuos para formar la nueva población. En general, el pseudocódigo consiste de los siguientes pasos:

- **Inicialización:** Se genera aleatoriamente la población inicial, que está constituida por un conjunto de cromosomas los cuales representan las posibles soluciones del problema. En caso de no hacerlo aleatoriamente, es importante garantizar que dentro de la población inicial, se tenga la diversidad estructural de estas soluciones para tener una representación de la mayor parte de la población posible o al menos evitar la convergencia prematura.
- **Evaluación:** A cada uno de los cromosomas de esta población se aplicará la función de aptitud para saber cómo de "buena" es la solución que se está codificando.
- **Condición de término:** El AG se deberá detener cuando se alcance la solución óptima, pero esta generalmente se desconoce, por lo que se deben utilizar otros criterios de detención. Normalmente se usan dos criterios: correr el AG un número máximo de iteraciones (generaciones) o detenerlo cuando no haya cambios en la población. Mientras no se cumpla la condición de término se hace lo siguiente:
    - **Selección:** Después de saber la aptitud de cada cromosoma se procede a elegir los cromosomas que serán cruzados en la siguiente generación. Los cromosomas con mejor aptitud tienen mayor probabilidad de ser seleccionados.
    - **Recombinación o cruzamiento:** La recombinación es el principal operador genético, representa la reproducción sexual, opera sobre dos cromosomas a la vez para generar dos descendientes donde se combinan las características de ambos cromosomas padres.
    - **Mutación:** Modifica al azar parte del cromosoma de los individuos, y permite alcanzar zonas del espacio de búsqueda que no estaban cubiertas por los individuos de la población actual.
    - **Reemplazo:** Una vez aplicados los operadores genéticos, se seleccionan los mejores individuos para conformar la población de la generación siguiente.

Desarrollo

Para el propósito del ejercicio intentaremos encontrar la solución de tableros con dimensiones de 8, 15, 30, 70, 90. Para esto haremos permutaciones en las características de la selección. Las características que tendremos a disposición son las siguientes;

- Numero de Generaciones = 50,000.
- Tamaño de Población = 20.
- Probabilidad de muta = 0.2.
- Probabilidad de muestra = 0.1.

Primero haremos un intento de encontrar la solución de dimensión 8 con la selección predeterminada. El resultado de la ejecución será dado por el número de la generación y el fitness (el número de ataques entre reinas), en caso de encontrar una solución se mostrara en forma de arreglo.

```
g: 49998 f:2 id:1698032342
g: 49999 f:2 id:1698032342
BUILD SUCCESSFUL (total time: 8 seconds)
```

Como podemos apreciar, aun con 50,000 generaciones no le fue posible encontrar una solución. Quedándose atascada en 2 de fitness.

Ahora intentaremos aumentando la población a 50, y veamos si el desempeño aumenta o disminuye.

```
g: 49998 f:4 id:874519230
g: 49999 f:4 id:874519230
BUILD SUCCESSFUL (total time: 7 seconds)
```

Dado que el resultado empeoro, regresaremos la población a su valor original, disminuiremos la probabilidad de muestra a 0.01 y aumentaremos el numero de generaciones a 500,000.

```
g: 7 f:0 id:371739364
g: 7 [1, 4, 6, 0, 2, 7, 5, 3]
BUILD SUCCESSFUL (total time: 1 second)
```

Esta vez no solo obtuvimos una solución sino que solo tardamos 7 generaciones en obtenerla. Utilizaremos estos valores para las siguientes pruebas.

Prueba de 15;

```
g: 23056 f:0 id:874519230
g: 23056 [1, 8, 13, 7, 9, 0, 2, 5, 14, 12, 10, 3, 6, 4, 11]
BUILD SUCCESSFUL (total time: 6 seconds)
```

Prueba de 30;*

```
g: 440 f:0 id:371739364
g: 440 [19, 13, 15, 6, 1, 11, 17, 24, 0, 2, 25, 16, 26, 8, 12, 5,
BUILD SUCCESSFUL (total time: 2 seconds)
```

Prueba de 70;*

```
g: 10190 f:0 id:874519230
g: 10190 [47, 32, 25, 48, 44, 19, 23, 31, 5, 11, 13, 29
BUILD SUCCESSFUL (total time: 20 seconds)
```

Prueba de 90;*

```
g: 10700 f:0 id:874519230
g: 10700 [74, 70, 60, 58, 50, 64, 21, 10, 32, 19, 82, 31,
BUILD SUCCESSFUL (total time: 31 seconds)
```

*La solución es demasiado extensa y no se muestra completa.

Conclusiones

Como pudimos apreciar con las pruebas anteriores con el solo hecho de disminuir la probabilidad de muestra hasta un 10% del valor original, logramos obtener buen resultado. Cabe notar que aunque aumentamos las generaciones hasta 500,000. No era necesario, ya que las pruebas no tomaron más de 25,000 generaciones.