UNIVERSIDAD NACIONAL DEL CENTRO DE LA PROVINCIA DE BUENOS AIRES

Introducción a la Programación Evolutiva

Trabajo Práctico de cursada

Integrantes:

- Caimmi, Brian. bcaimmi@gmail.com L.U.: 247668
- Vallejos, Sebastián. sebaviru@gmail.com L.U.: 247778

Enunciado:

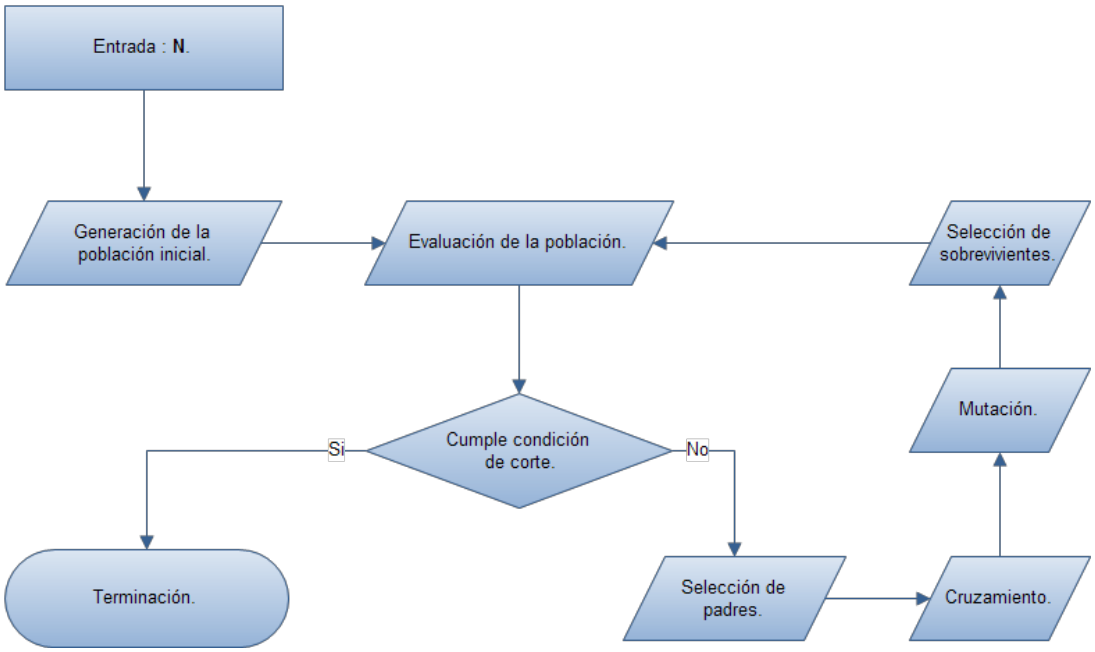
Un controlador aéreo debe ubicar una flota de N aviones en un espacio aéreo predeterminado. Dicho espacio aéreo es virtualmente representado como una grilla de N filas por N columnas. Los N aviones deben ser ubicados en la grilla mencionada de manera tal que no se oculten entre sí. Se considera que dos aviones se ocultan entre sí cuando: se ubican en una misma fila de la grilla, se ubican en una misma columna de la grilla, o se ubican en una misma diagonal de la grilla.

El problema presentado es una variante al problema de las N reinas, un problema de NP completo para el cual no hay algoritmo que aseguren encontrar la solución óptima con las N reinas bien colocadas (solo backtracking).

Tipo de algoritmo evolutivo:

Para la representación de los cromosomas (la cual se verá más adelante) no se necesita ninguna estructura demasiado compleja (como las de árboles). Además, este problema posee la característica de que una configuración muy cercana a la solución ideal (por ejemplo con solo 2 o 3 reinas en conflicto) puede en realidad estar muy lejos de la configuración de la solución ideal. El efecto es que sea difícil diseñar métodos/heurísticas eficientes para la búsqueda local en este problema, por lo que diseñar un algoritmo memético no tiene mucho sentido. Por lo tanto el tipo de algoritmo evolutivo a utilizar para este problema será un **algoritmo genético simple**.

Esquema de ejecución general del algoritmo:



Representación o codificación de las soluciones:

Para el problema presentado, se optará por una permutación de N números naturales entre 1 y N como representación de cada cromosoma. Cada gen representará una de las N columnas y el alelo del mismo representará la fila en la que se encuentra ubicado un avión.

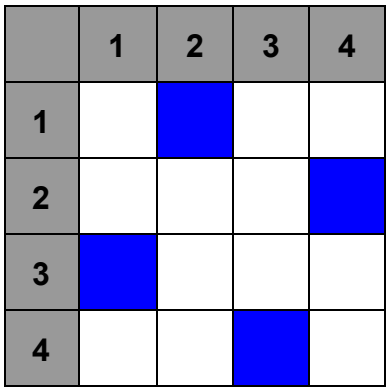
Esta representación se decidió ya que de esta forma no se permite la generación de hijos inválidos (tener más de N aviones, menos de N aviones o en posiciones inexistentes) ni de hijos con aviones que se encuentren en la misma fila o columna (hijos que no serían soluciones ideales del problema).

Proceso de decodificación:

Por ejemplo, el cromosoma (3,1,4,2) representa la posición de 4 aviones:

- Un avión en la columna 1, fila 3. *- Un avión en la columna 2, fila 1.*

- Un avión en la columna 3, fila 4. *- Un avión en la columna 4, fila 2.*



Función de evaluación o de fitness:

Para evaluar el fitness se decidió por una función que retorna valores entre 1 y N, siendo N la solución ideal y 1 la peor solución de todas. Esto permitirá fácilmente verificar si la solución encontrada es la buscada y posee un amplio margen que permitirá diferenciar correctamente las soluciones de forma minuciosa.

La función consiste en proponer un valor ideal N, y a este, restarle cierto valor por cada par de aviones en conflicto. Este valor a restar será 1/N, esto se decidió para que la fórmula siguiente nunca pueda dar menor que 1:

$$Fitness = N - \frac{1}{N}\sum_{1}^{N} Cantidad\ de\ aviones\ en\ conflicto\ con\ el\ avión\ N$$

De esta forma, en el mejor de los casos (sin aviones en conflicto), el resultado será N:

$$Fitness = N - \frac{1}{N}\sum_{1}^{N} Cantidad\ de\ aviones\ en\ conflicto\ con\ el\ avión\ N$$

$$Fitness = N - \frac{1}{N}\sum_{1}^{N} 0$$

$$Fitness = N - 0$$

$$Fitness = N$$

Y en el peor de los casos (con todas las reinas en conflicto):

$$Fitness = N - \frac{1}{N}\sum_{1}^{N} Cantidad\ de\ aviones\ en\ conflicto\ con\ el\ avión\ N$$

$$Fitness = N - \frac{1}{N}\sum_{1}^{N} (N-1)$$

$$Fitness = N - \frac{1}{N} * N * (N-1)$$

$$Fitness = N - (N-1)$$

$$Fitness = N - N + 1$$

$$Fitness = 1$$

Proceso de generación de la población inicial:

Para la generación de la población inicial se generan individuos totalmente aleatorios. Para cada individuo, se genera el cromosoma como la permutación ordenada de los N naturales desde 1 hasta N { 1 , 2 , 3 . . . N }. Luego se genera una permutación al azar del cromosoma, quedando los elementos desordenados.

Proceso de selección de padres:

Debido a las características del problema de que una solución muy bien evaluada pueda en realidad estar muy lejos de ser la solución correcta, se opta por preservar un buen nivel de diversidad. Por lo tanto, se cree que la mejor opción para la elección de padres sería.

- *Selección por torneo:*

  Podría lograr un equilibrio entre el direccionamiento del algoritmo hacia la solución buscada y la diversidad necesaria para llegar a ella.

- *Ruleta:*

  Se elige este método para contrastar a la selección por torneo ya que la ruleta le da mucha más chance a los cromosomas bien evaluados. De esta forma a la hora de implementar se podrá observar si la idea de preservar la diversidad realmente tuvo el efecto positivo que se esperaba

Operadores de cruce genético:

Se implementarán cruces para permutaciones, evitando así la generación de hijos no permutaciones (es decir, con aviones en una misma fila o columna):

- *Cruce en n puntos preservando el orden*.
- *Cruce PMX.*
- *Cruce basado en ciclos.*

Operadores de mutación genética:

Nuevamente se optará por mutaciones aptas para permutaciones, es decir, que el cromosoma mutado siga siendo una permutación de los N elementos.

- *Por inserción.*
- *Por intercambio.*
- *Por inversión.*
- *Por mezcla.*

Proceso de selección de sobrevivientes, o proceso de reinserción:

Se utilizarán los mecanismos más conocidos se selección de sobrevivientes que se sabe, han dado buenos resultados hasta ahora en otros algoritmos evolutivos.

- *Steady-state.*
- *Por ranking.*

Probabilidad de cruce:

La probabilidad de cruce recomendada para problemas de algoritmos genéticos simples ronda entre los valores de 0.6 y 0.9. Por lo tanto, se utilizaran valores entre ese rango, aunque para obtener una mayor precisión sobre el rango óptimo de probabilidad de cruce sólo será posible mediante la ejecución y evaluación del algoritmo.

Probabilidad de mutación:

La probabilidad de mutación recomendada para problemas de algoritmos genéticos simples ronda entre los valores de 0.05 y 0.2. Por lo tanto, se utilizaran valores entre ese rango, aunque para obtener una mayor precisión sobre el rango óptimo de probabilidad de cruce sólo será posible mediante la ejecución y evaluación del algoritmo.

Tamaño de la población inicial:

El tamaño de la población inicial debería ser dependiente del N. Si la población fuera fija, se correría el riesgo de, si el N es muy chico, generar una población inicial demasiado grande y encontrar una solución de forma azarosa, sin dejar que el algoritmo haga su trabajo. Por otro lado, si el N es muy grande, la población quizás sería muy chica y le tomara demasiadas generaciones al algoritmo encontrar la solución buscada. Se optara

por una población de N^2 lo que permitirá que la población escale junto con N, pero no a la velocidad que la diversidad posible lo hace $!N$.

Condición de finalización para la ejecución del algoritmo:

La ejecución del algoritmo finalizará cuando se haya alcanzado la solución ideal al problema (un individuo con fitness N) o, para evitar la posibilidad de que el algoritmo se estanque en un máximo local y no pueda llegar a encontrar la solución ideal, se propondrá un número G de generaciones máximas luego de la cual el algoritmo finalizará su ejecución se haya encontrado o no la solución ideal.