Introducción a la Programación Evolutiva

Marcelo Rodríguez

N° leg: 245.221

mailto:mrodriguez@alumnos.exa.unicen.edu.ar

Profesores: Doctor Ing. Analía Amandi

Doctor Ing. Virginia Yannibelli

Resumen

En este documento se exponen brevemente las decisiones de diseño para la implementación de un algoritmo evolutivo aplicado a una instancia del problema de las N reinas. Se hace énfasis en la definición de sus componentes o etapas y parámetros de ejecución.

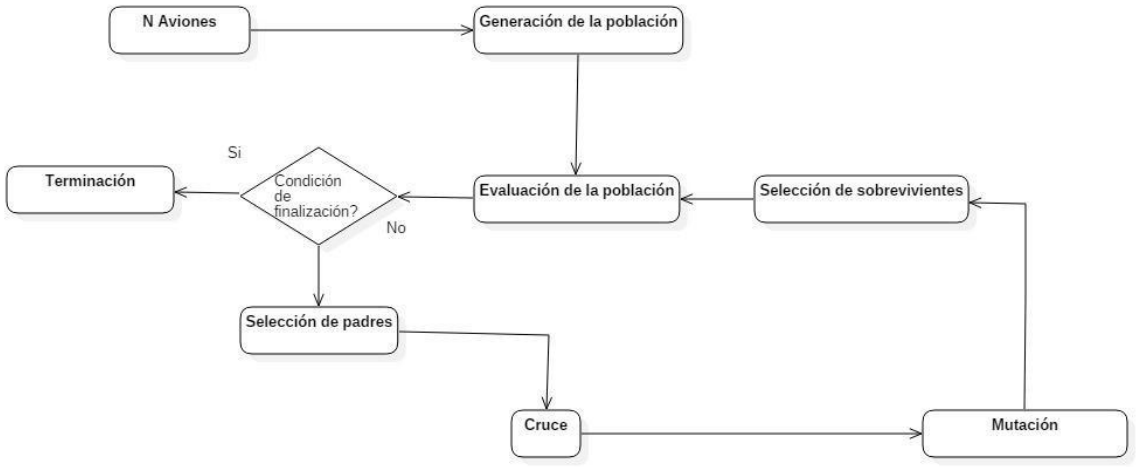
1. Enunciado del problema a resolver.

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 si. Se considera que dos aviones se ocultan entre si 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.

2. Descripción del algoritmo evolutivo.

2.1. Esquema de ejecución general del algoritmo.

En este problema en particular, sucede que obtener una solución de muy pocos aviones en conflicto, puede considerarse buena solución, pero en realidad dista mucho de la solución ideal que puede obtenerse mediante un algoritmo especializado como backtracking. Esto conlleva a complicar el diseño de métodos heurísticos eficientes para una búsqueda local y no da sentido a utilizar un algoritmo memético. Por ende, el algoritmo a utilizar será un algoritmo genético simple.



2.2. Representación o codificación de las soluciones.

Una representación directa sería utilizar matrices y definir los correspondientes operadores de variación. En cambio, la propuesta definitiva se implementará con permutaciones; en estas, está implícita la columna y en cada componente sólo será necesaria la fila. Adicionalmente, esta representación satisface una de las restricciones del problema, que no puede haber dos aviones en una misma columna.

2.3. Proceso de decodificación.

Para obtener el fenotipo a partir de un genotipo (o cromosoma) el proceso de decodificación es directo. El tablero solución se obtiene por la lectura del cromosoma, cada coordenada y componente se obtiene a partir del gen y su alelo.

2.4. Función de evaluación o Fitness

Al observar el fenotipo buscado podemos establecer una función de calidad y decidir una relación para maximizarla. Cuantos menos aviones en conflicto tengamos, un individuo tendrá mayor calidad. La relación propuesta será.

$$F(N) = N - \sum \text{cantidad de aviones en conflicto}$$

Por lo tanto, si no hay aviones en conflicto la función retorna un valor máximo y en el peor caso, 0.

2.5. Proceso de generación de la población inicial.

En este paso se propone generar una población de individuos al azar. Luego, para cada individuo se genera aleatoriamente su cromosoma como una permutación ordenada según la cantidad de aviones N.

2.6. Proceso de selección de padres.

En este problema en particular, puede suceder que una solución bien evaluada no sea precisamente correcta. En tales condiciones se propone tener diversidad en la población. Se propone como método de selección de padres la **Tournament Selection**, de esta forma se considera la diversidad y relación entre individuos al tiempo que sesga a la población a padres con alto Fitness. En la etapa de implementación, también se incorporó el método **Wheel Selection** a fin de contrastar a Tournament Selection para intentar dar más chance a los cromosomas bien evaluados.

2.7. Operadores de cruce genético.

Los operadores de cruce genético deben satisfacer la restricción de que la descendencia generada debe pertenecer al espacio genotipo siendo permutaciones válidas. Durante el diseño conceptual del algoritmo se propusieron inicialmente los siguientes métodos:

- **Partially mapped Crossover**
- **Order Crossover.**

Y finalmente, se añade **Cycle Crossover**. Por lo que la implementación del algoritmo quedará con:

- **Partially mapped Crossover**
- **Order Crossover.**
- **Cycle Crossover**

2.8. Operadores de mutación genética.

El aspecto importante de aplicar estos operadores es la intención de alterar un cromosoma. Los operadores propuestos deben advertir que el cambio de un gen no es independiente del resto. En consecuencia, se propone utilizar:

- **Swap Mutation**.
- **Insert Mutation**
- **Scramble Mutation.**
- **Inversion Mutation**

2.9. Proceso de selección de sobrevivientes o proceso de reinserción.

Se utilizará el modelo de población **Steady State**. En este, los viejos individuos de la población son reemplazados por un porcentaje de los nuevos teniendo en cuenta el **Ranking** de su Fitness.

2.10. Probabilidad de cruce.

La probabilidad de **crossover**, para determinar si pasar o no material genético a los hijos estará dada en un rango de [0.6, 0.9]; en consecuencia, en momento de implementación se determinará el valor más efectivo.

2.11. Probabilidad de mutación.

La probabilidad de aplicar operadores de mutación sobre un genotipo se recomienda entre [0.05, 0.2] Al igual que la probabilidad de cruce, será necesario observar el comportamiento del algoritmo para determinar este parámetro más precisamente.

2.12. Tamaño de la población inicial.

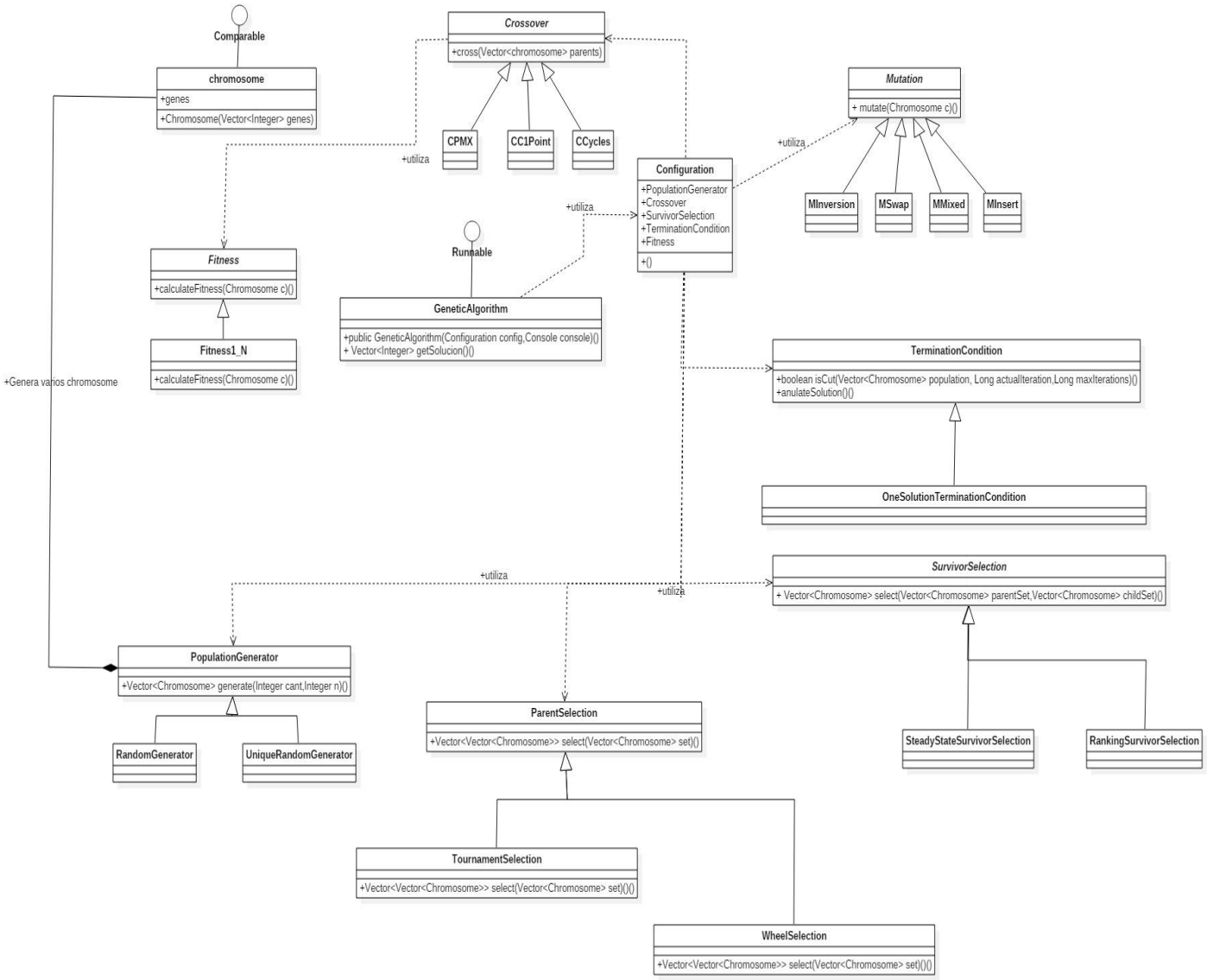
Se propone un tamaño de población en función del número N de aviones, para permitir que el algoritmo realice su trabajo sin encontrar una solución ni demasiado aleatoria ni requerir de demasiadas generaciones.

2.13. Condición de finalización de ejecución.

Se propone parametrizar este aspecto dando en el peor caso un número máximo de generaciones permitidas a fin de que el algoritmo no estanque en un óptimo local, si es que no se encuentra la solución ideal.

3. Diagrama de clases de la aplicación.

En el siguiente diagrama de clases se exponen sólo aquellas que implementan en conjunto el algoritmo genético (la lógica del mismo) y sus relaciones. Las clases correspondientes a la interface gráfica no están presentes en este diagrama, así como también clases cuya cohesión es pobre por reunir métodos de funcionalidad general (p.ej: una clase **Utils**).
Los nombres de las clases se escogieron de forma tal que auto describan la funcionalidad los componentes del algoritmo propuesto en los apartados anteriores. Vale destacar que **GeneticAlgorithm**, que realiza todo el trabajo de cálculo, acondiciona su funcionamiento según **Configuration**; esta última recibe instancias de los objetos cuyos parámetros han sido seleccionados por el usuario: función de Fitness, método de selección de sobrevivientes y padres, operadores de mutación, probabilidades de cruce, etc.

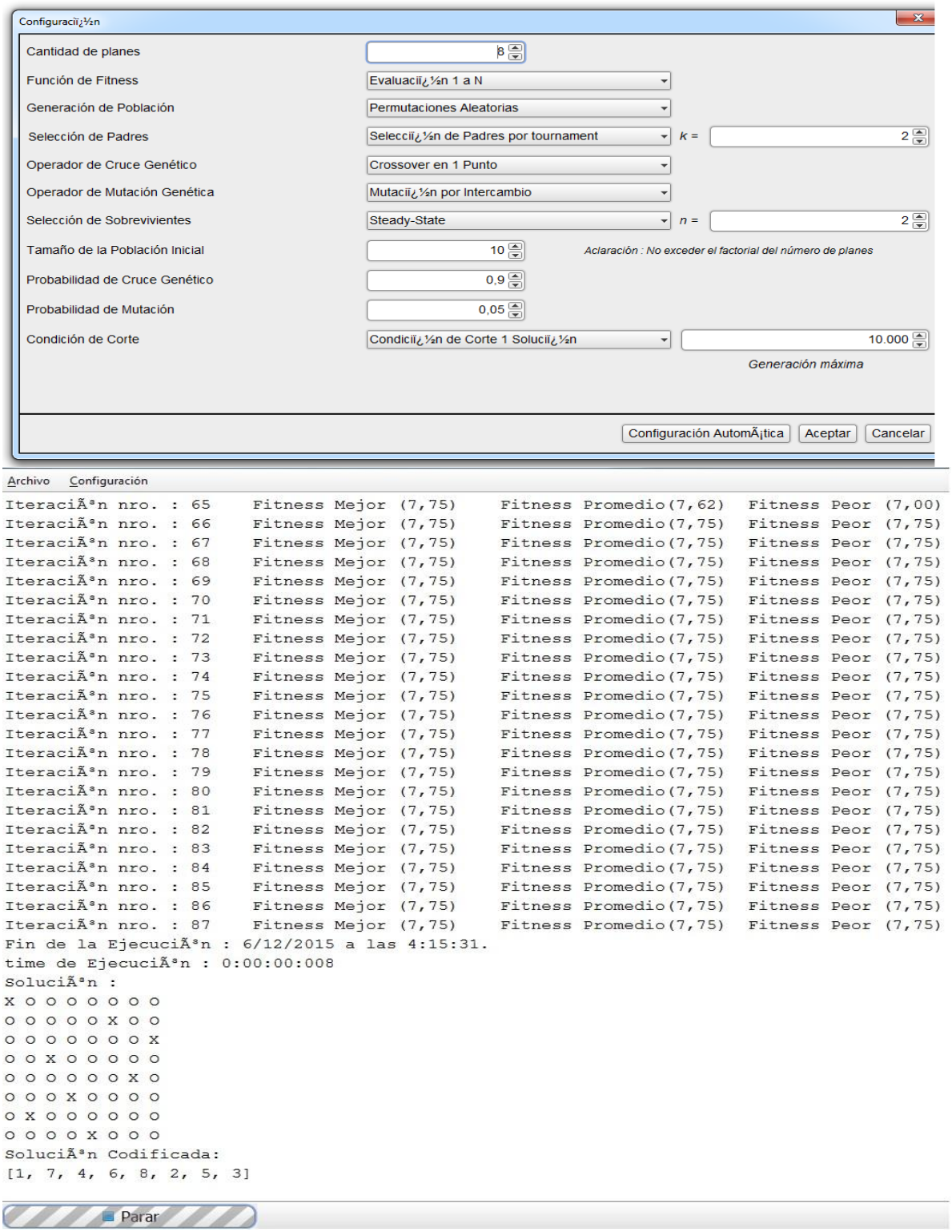


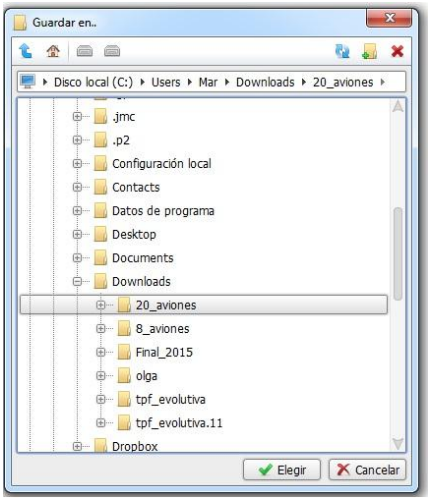
4. Conclusiones

Se ha desarrollado un Algoritmo Genético para un problema estilo N-Reinas. A partir de observar el comportamiento del mismo se pueden inferir los siguientes comentarios:
Dentro de las mejores soluciones, es decir aquellas con mayor efectividad, se puede extraer un breve análisis para determinar la configuración de la ejecución más precisa del algoritmo genético. Evidentemente, las ejecuciones más veloces se obtienen sobre una población de individuos sin repetición; en tales, el método de selección de padres por torneo converge a la solución más rápido. También, una configuración con operadores de mutación por mezcla y selección de sobrevivientes Steady State, se mantiene en promedio mejor que otras configuraciones. En este contexto, conforme el parámetro N crece, se requieren más ciclos evolutivos en orden de miles para llegar a una solución. Para un N = 8, casi es 100% seguro obtener una solución en 100000 ciclos; y para N = 20, será necesario llegar a 200000, aún así, el 50% de las ejecuciones no llega a obtener una solución. Ante configuraciones idénticas del algoritmo, el operador PMX consigue soluciones en un 10% más rápido qué otros operadores de Crossover como Basado en ciclos o crossover en un punto.

5. Apéndice. Interface de Usuario y uso de la aplicación.

La pantalla configuración permite establecer los diferentes parámetros para acondicionar el algoritmo genético. Para extraer las conclusiones del punto **4**, la aplicación dispone de un botón "Configuración Automática": esta funcionalidad, ejecuta el algoritmo con las probabilidades fijas, pero variando los operadores de mutación, selección de padres y sobrevivientes; lo que permite jugar el comportamiento del algoritmo en forma global.
Durante la ejecución, la aplicación muestra los resultados obtenidos en la pantalla.





Así mismo, es posible almacenar la salida de la ejecución en un directorio. Cada solución encontrada se almacena en un archivo identificado con la fecha y tiempo de la ejecución.