Universidad de Chile
Facultad de Ciencias Físicas y Matemáticas
Departamento de Ciencias de la Computación

Tarea 2

Algoritmos Genéticos

Integrantes: Guillermo Martínez
Profesor: Alexandre Bergel
Auxiliar: Ignacio Slater

Fecha de realización: 26 de noviembre de 2020
Fecha de entrega: 26 de noviembre de 2020
Santiago, Chile

Índice de Contenidos

Índice de Figuras

1. Utilización de un algoritmo genético para adivinar una palabra

1.1. Descripción del problema y solución

El problema para adivinar una palabra mediante un algoritmo genético consiste en encontrar una palabra partiendo con la generación de palabras aleatorias del mismo largo.

Cada individuo corresponde a una palabra del mismo largo de la palabra a adivinar, en donde las letras son guardadas dentro de un arreglo, es decir, el arreglo con las letras corresponde al individuo. Con esto, una población se crea mediante la generación de palabras del largo de la palabra objetivo utilizando letras del alfabeto de manera aleatoria.

La función *fitness* que evalúa el desempeño de una palabra entrega la cantidad de letras bien posicionadas dentro de la palabra, por lo que el máximo *fitness* corresponde a la cantidad de letras de la palabra a encontrar.

Por otro lado, el problema específico consiste en adivinar la palabra "paralelepipedo". Con esto, en cuanto a las características del algoritmo, se tiene lo siguiente:

- Cantidad de genes por individuo: Largo de la palabra a adivinar (14).
- Tamaño de la población: 100 individuos.
- Tasa de mutación: 0.2.
- Estrategia de selección: Torneo de Tenis.
- Estrategia de crossover: Crossover de un solo punto.
- Estrategia de mutación: Se genera un número aleatorio entre 0 y 1, y si el resultado es menor a la tasa de mutación, se muta el gen aleatorio del individuo asegurándose que la mutación cambie efectivamente el gen.
- Condición de término: El algoritmo termina cuando se encuentra un individuo con un *fitness* igual a la cantidad de caracteres de la palabra a adivinar, o cuando se alcanzan las 200 generaciones.

1.2. Resultados

En la figura 1 presentada a continuación se puede observar un gráfico mostrando cómo evoluciona el *fitness* para el problema de adivinar la palabra "paralelepipedo" durante las generaciones, ya sea tanto el mejor *fitness*, como el promedio y el peor.

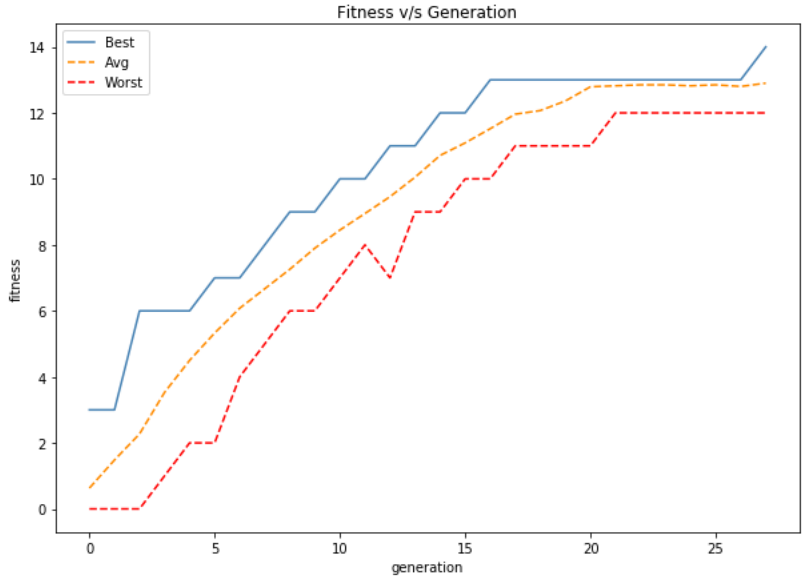

Figura 1: Evolución del *fitness* durante las generaciones para el problema de adivinar la palabra.

Luego de correr el algoritmo reiteradas veces se puede concluir que este problema siempre es resuelto por el algoritmo implementado sin complicaciones, y tarda entre 15 a 25 generaciones con las características ya mencionadas.

2. Utilización de un algoritmo genético para convertir números a binario

2.1. Descripción del problema y la solución

El problema consiste en encontrar la representación en número binario de un número decimal arbitrario.

Cada individuo corresponde a un numero binario aleatorio dentro de un arreglo con la misma cantidad de bits que la representación binaria del número a encontrar. La población se crea mediante la generación de números binarios del largo ya mencionado asignando los bits de manera aleatoria.

La función *fitness* para este problema transforma el número binario a representación decimal y calcula la diferencia absoluta negativa entre dicho número y el número al cual se le quiere encontrar la representación binaria, con lo que el máximo *fitness* corresponde al valor 0.

Como problema específico se decide encontrar la representación binaria del número “157492”, con lo que se definen las siguientes características del algoritmo:

- Cantidad de genes por individuo: Largo de la representación binaria del número a encontrar (18).

- Tamaño de la población: 100 individuos.
- Tasa de mutación: 0.4.
- Estrategia de selección: Torneo de Tenis.
- Estrategia de crossover: Crossover de un solo punto.
- Estrategia de mutación: Se genera un número aleatorio entre 0 y 1, y si el resultado es menor a la tasa de mutación, se muta el gen aleatorio del individuo asegurándose que la mutación cambie efectivamente el gen.
- Condición de término: El algoritmo termina cuando se encuentra un individuo con un *fitness* igual a 0, o cuando se alcanzan las 200 generaciones.

2.2. Resultados

El siguiente gráfico mostrado en la figura 2 es uno de los obtenidos luego de correr el algoritmo en reiteradas ocasiones:

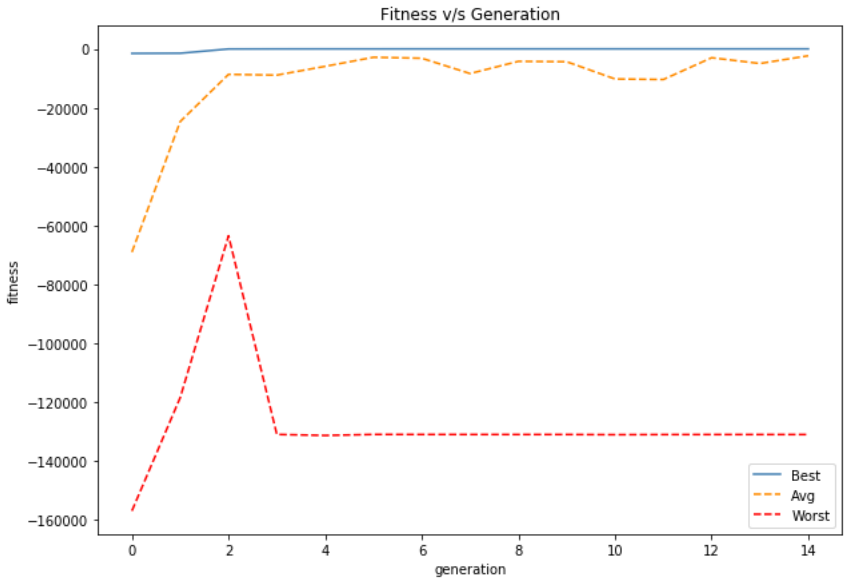

Figura 2: Evolución del *fitness* durante las generaciones para el problema de convertir un número a binario.

El problema de convertir un número a binario en general es resuelto sin problemas por el algoritmo cuando se tienen individuos con una cantidad menor de genes (10 por ejemplo). Sin embargo, cuando la cantidad de genes aumenta, el algoritmo tiende a encontrar máximos locales de los cuales no puede salir. Es por esto que, para este caso particular del problema, se decide utilizar una tasa de mutación más alta, de manera que si el algoritmo está atascado en un máximo local, una mutación pueda solucionarlo.

3. Problema electo: Problema de las N reinas

3.1. Descripción del problema y solución

El problema de las N reinas consiste en poner N reinas en un tablero de ajedrez de NxN de manera que no se puedan atacar entre ellas.

Cada individuo consiste de un arreglo con número enteros del 0 al $N-1$ en donde cada posición del arreglo corresponde a una columna del tablero de ajedrez, y el entero en dicha posición corresponde a la fila del tablero. La población se crea mediante la generación de N números enteros aleatorios en un arreglo.

La función *fitness* verifica que para cada reina no haya otra hacia los lados, ya sea arriba o abajo, y en diagonal, y suma un punto de colisión por cada reina que encuentra en las posiciones mencionadas. Con esto, el máximo *fitness* corresponde al valor 0, es decir, cuando existen 0 colisiones dentro del tablero.

Para probar el algoritmo se decide utilizar 8 reinas en un tablero de $8x8$, en donde las características son:

- Cantidad de genes por individuo: Cantidad de reinas a poner en el tablero (8).
- Tamaño de la población: 100 individuos.
- Tasa de mutación: 0.2.
- Estrategia de selección: Torneo de Tenis.
- Estrategia de crossover: Crossover utilizando dos puntos.
- Estrategia de mutación: Se genera un número aleatorio entre 0 y 1, y si el resultado es menor a la tasa de mutación, se muta el gen aleatorio del individuo asegurándose que la mutación cambie efectivamente el gen.
- Condición de término: El algoritmo termina cuando se encuentra un individuo con un *fitness* igual a 0, o cuando se alcanzan las 500 generaciones.

3.2. Resultados

A continuación, en la figura 3, se puede apreciar el desempeño de los individuos a lo largo de las generaciones para el problema de las N reinas, donde $N = 8$:

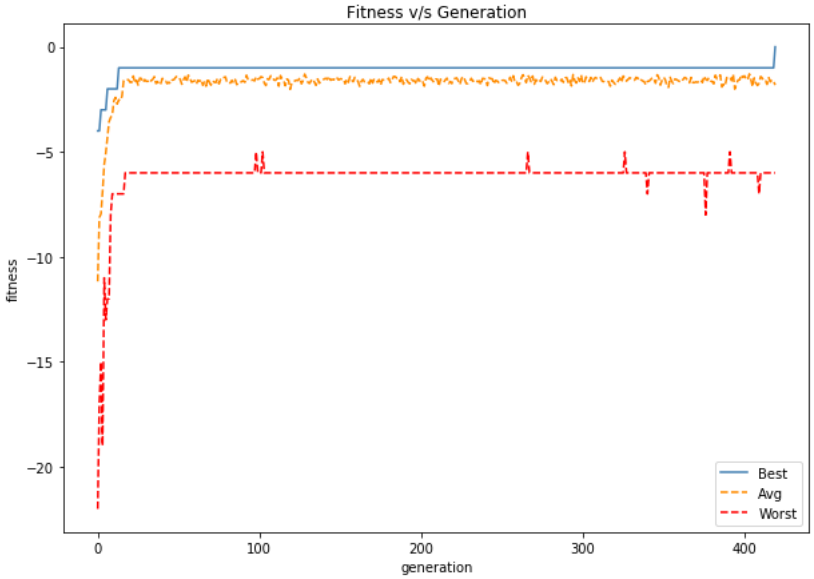

Figura 3: Evolución del *fitness* durante las generaciones para el problema de las N reinas.

También se creó una función que permite visualizar el tablero con las reinas posicionadas, dando como resultado lo observado en la figura 4:

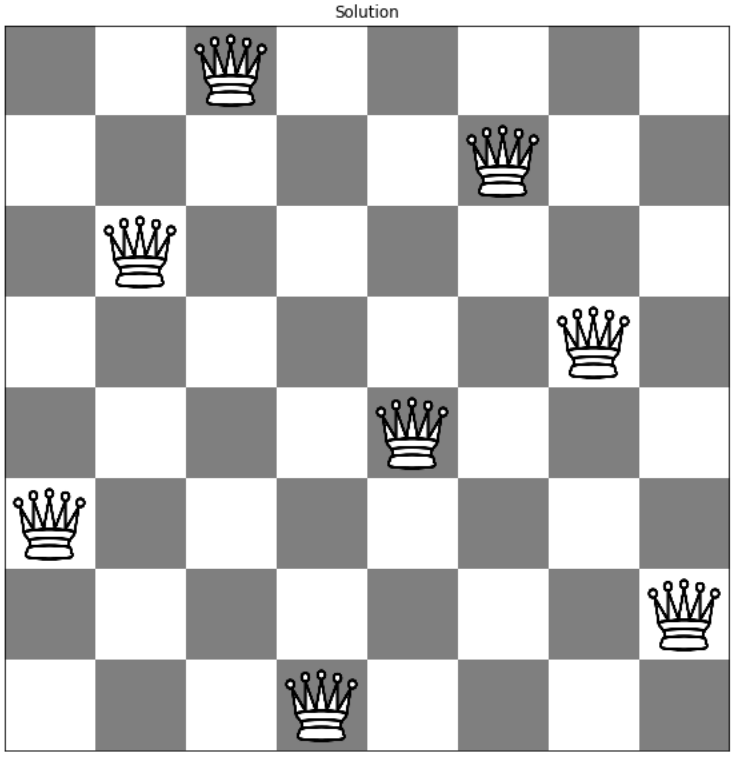

Figura 4: Tablero con la solución encontrada para el problemas de las N reinas.

Este problema, al igual que el de convertir un número a binario, tiende a encontrar máximos locales cuando la cantidad de reinas aumenta. En ocasiones encuentra el máximo *fitness* luego de pasar una gran cantidad de generaciones en un máximo local, tal como se puede observar en el gráfico de la figura 3, razón por la cual se decide utilizar 500 iteraciones antes de detener el algoritmo. Sin embargo, con $N = 8$ la mayoría de las veces logra encontrar una solución válida, y se pudo encontrar soluciones para $N = 16$, e incluso para $N = 32$.

Link al repositorio con la resolución de la tarea: https://github.com/wiwillym/CC5114_Tarea2