

Programación genética

Alejandro Mendoza Puig

Introducción

Programación genética

- Rama de la computación evolutiva que funciona similar a los algoritmos genéticos (**cruza, mutación, selección y reemplazo**)
- Se realizan los mismos operadores que en AG pero a una función la cual esta buscando optimizar su comportamiento. La población pueden ser vistos como programas de computadora

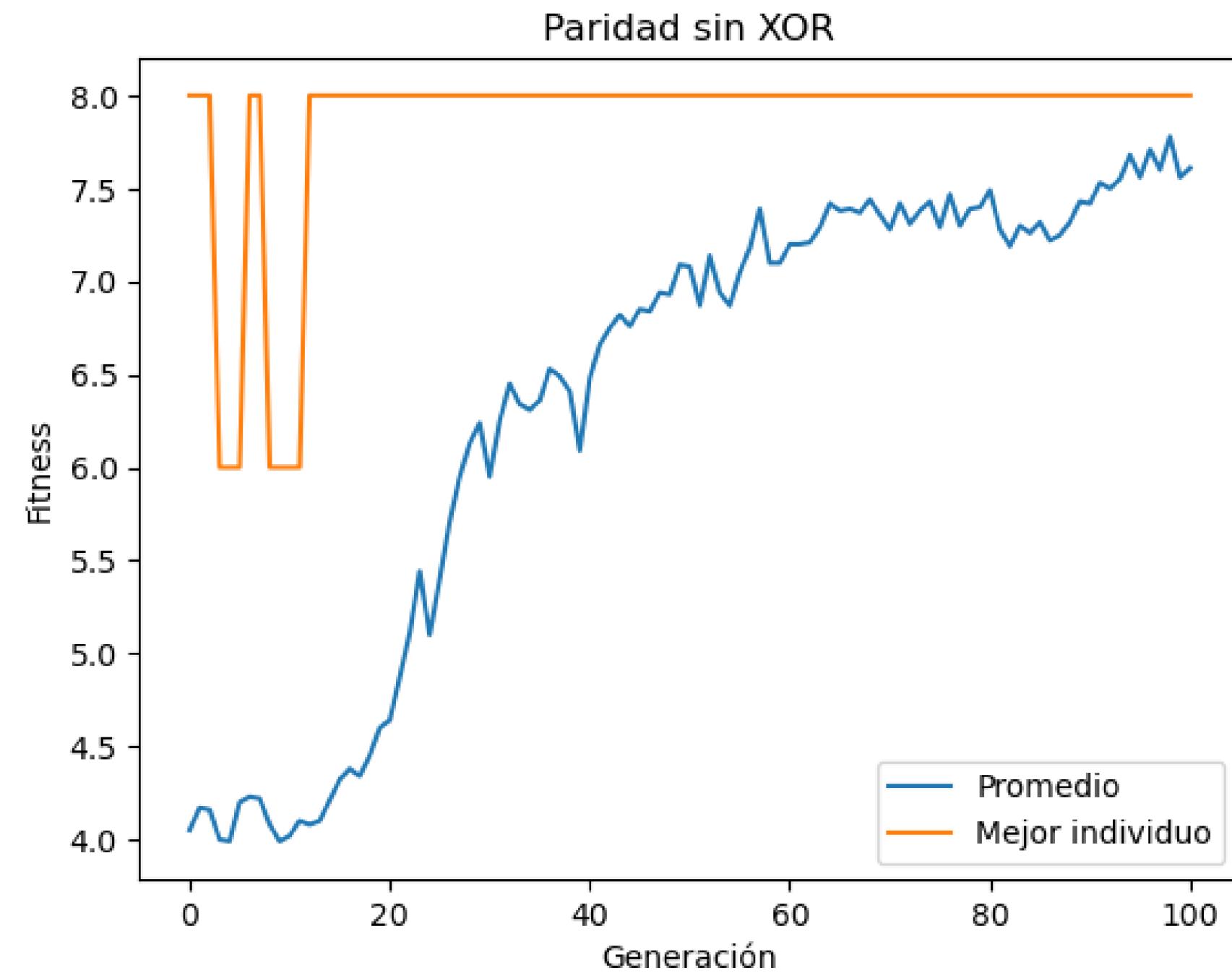
Problemas

- En esta tarea trabajamos sobre dos problemas, el primero es el de paridad, en el cual dados 3 bits se tiene que generar una respuesta binaria, 1 cuando el número de unos en los 3 bits son pares y 0 de lo contrario.
- El segundo problema es una regresión simbólica en la cual contamos con 21 valores de X junto con el resultado de $f(x)$.
- Los resultados mostrados son utilizando la paquetería **DEAP**

Resultado Paridad sin XOR

- 100% accuracy
- Mejor individuo

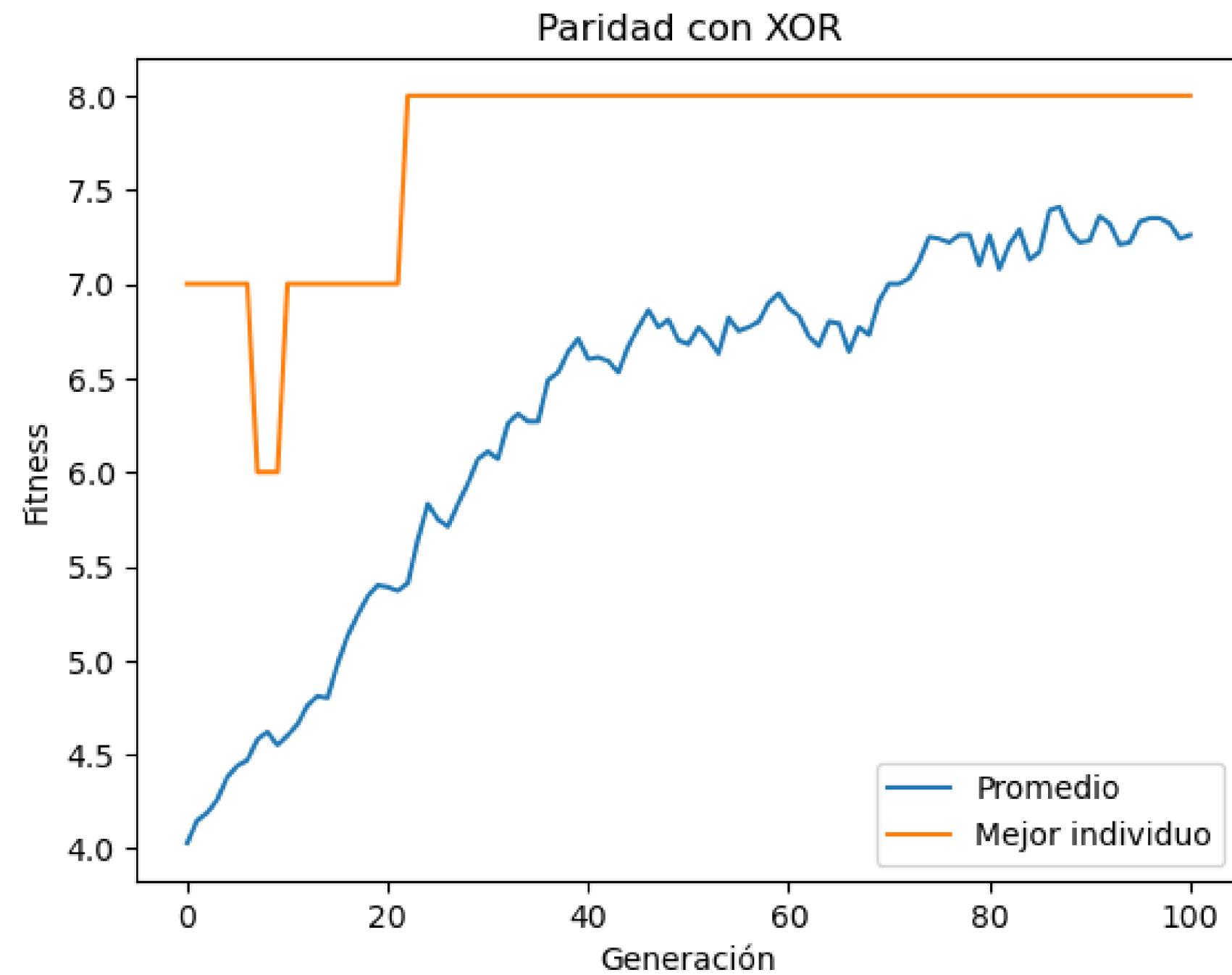
`and_(or_(and_(var0, or_(var1,
not_(var0))), var2), or_(or_(var1,
and_(0, and_(var0, var1))), var0))`



Resultado Paridad con XOR

- 100% accuracy
- Mejor individuo

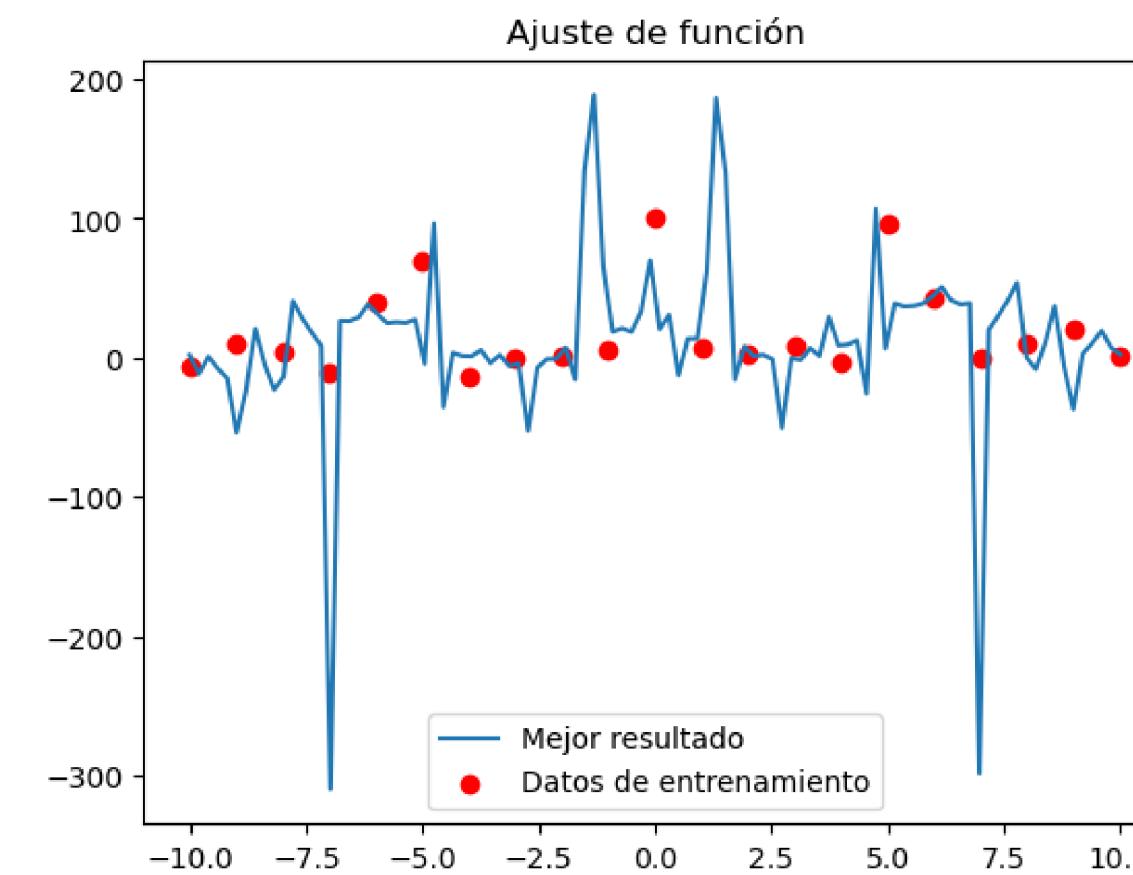
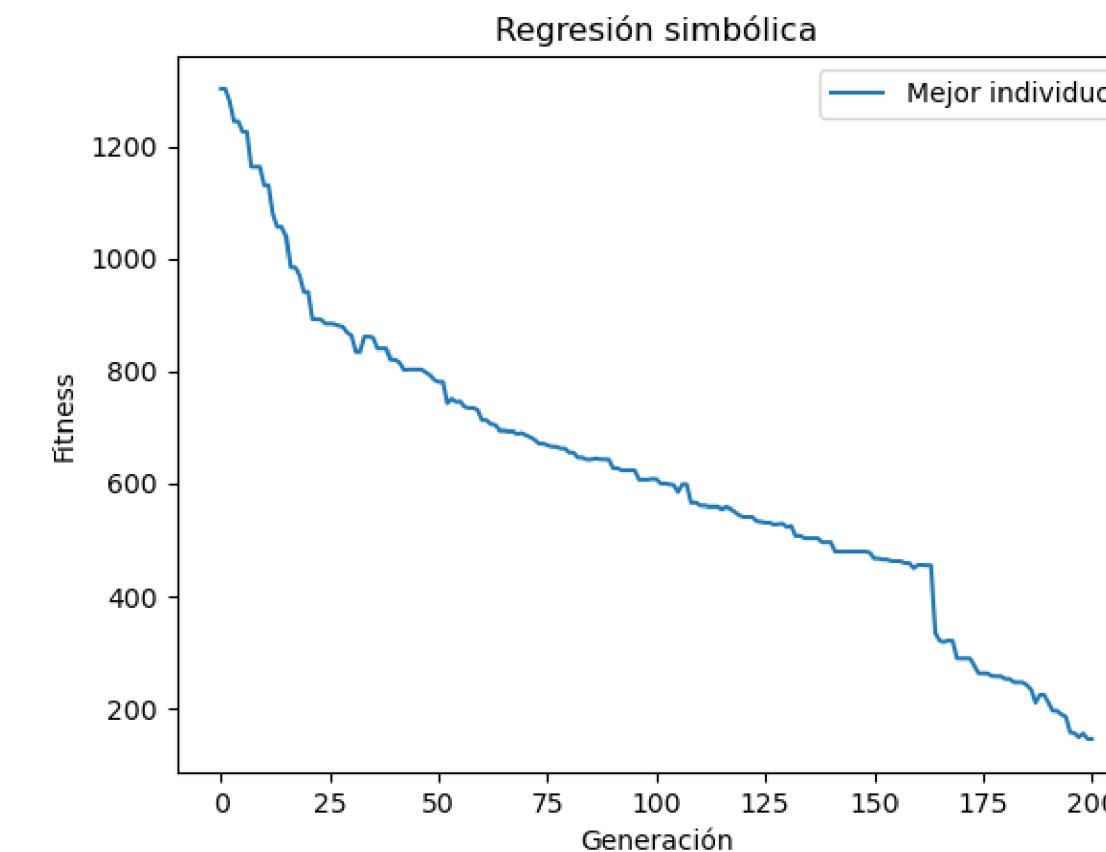
```
xor(and_(and_(var2, var2),  
not_(and_(var1, 0))),  
and_(not_(xor(var0, var1)),  
not_(and_(var1, 0))))
```



Resultado Regresión simbólica

- Fitness: 144
- Mejor individuo

```
sub(sub(sub(add(sub(-1, add(-1, neg(sub(add(cos(x), add(cos(x), 1)),  
add(cos(sub(sub(protectedDiv(-1, protectedDiv(0, cos(-1))), x), x)), -1))))),  
sub(sub(protectedDiv(-1, protectedDiv(add(cos(sub(0, cos(-1))),  
add(sub(sub(protectedDiv(add(cos(-1), add(x, x)), protectedDiv(1, cos(protectedDiv(sub(-1, x),  
neg(1)))), x), sub(0, x)), -1)), x), add(cos(-1), sub(neg(sub(add(cos(neg(x)), add(sub(sub(x, x),  
sub(-1, add(cos(x), add(x, 1)))), 1)), add(cos(1), neg(-1)))), add(add(cos(sub(0, x)),  
add(sub(protectedDiv(1, cos(protectedDiv(sub(-1, 1), neg(x)))), add(-1, add(protectedDiv(sub(-1,  
cos(x)), cos(x)), 1))), 0)), add(cos(-1), add(protectedDiv(sub(sub(protectedDiv(-1, protectedDiv(0,  
cos(1))), x), sub(-1, add(cos(x), add(x, 1))), cos(protectedDiv(sub(-1, 1), neg(x)))), 1)))),  
add(sub(-1, add(cos(cos(protectedDiv(sub(-1, 1), protectedDiv(sub(-1, cos(x)), neg(x)))),  
add(cos(sub(cos(x), x)), add(cos(sub(0, x)), add(sub(sub(protectedDiv(1,  
cos(protectedDiv(sub(-1, 1), protectedDiv(sub(-1, cos(x)), neg(x)))), x), sub(-1, add(add(x, add(-1,  
1)), add(x, cos(sub(-1, 1)))), -1)))), sub(neg(sub(add(cos(cos(-1)), add(sub(protectedDiv(add(x,  
1), neg(x)), sub(-1, add(add(x, add(protectedDiv(1, cos(x)), cos(protectedDiv(sub(-1, x), neg(1)))),  
add(x, 1)))), -1)), add(cos(add(protectedDiv(1, cos(protectedDiv(sub(-1, 1), sub(-1, add(cos(-1),  
cos(x)))), 1))), -1)), add(sub(x, -1), neg(sub(add(protectedDiv(1, cos(protectedDiv(sub(-1,  
cos(cos(x)), protectedDiv(sub(-1, cos(x)), neg(x)))), add(sub(sub(protectedDiv(-1, x), x),  
sub(add(-1, add(protectedDiv(sub(-1, cos(x)), cos(1)), 1)), add(protectedDiv(protectedDiv(1,  
cos(protectedDiv(sub(-1, 1), neg(x)))), cos(x)), sub(protectedDiv(1, cos(protectedDiv(sub(-1, 1),  
protectedDiv(sub(-1, cos(x)), neg(x)))), x)))), 1)), add(cos(-1), neg(x)))),  
mul(sub(add(protectedDiv(sub(protectedDiv(cos(x), cos(protectedDiv(sub(-1, 1), neg(x)))),  
cos(x)), neg(x)), 1), neg(sub(x, add(cos(-1), neg(-1)))), 0), 0))
```



Conclusiones

Paridad: tanto usando como no usando XOR, se llegaron a expresiones la cuales tenían 8/8 correctos.

Regresión: se puede notar que se logra crear una función que es una función sobreajustada (Esto se puede notar ya que para pasar por todos los puntos la función presenta fluctuaciones muy grandes), aunque esto también es de esperarse ya que el set de datos fue generado con números aleatorios por lo cual si dejamos correr las poblaciones por muchas generaciones van a seguir haciendo todo para minimizar la función de evaluación.

