U N I V E R S I D A D

D E G U A D A L A J A R A

CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERÍA

DEPARTAMENTO DE CIENCIAS COMPUTACIONALES

Seminario de Solución de Problemas de Inteligencia Artificial I

Producto integrador de la unidad 4. Optimización por algoritmos genéticos

Profesor: Alma Yolanda Alanís García

Alumno: Anthony Esteven Sandoval Márquez

Código: 215660767

Carrera: Ingeniería en Computación

Sección: D01

Fecha: 15/09/2022

Introducción

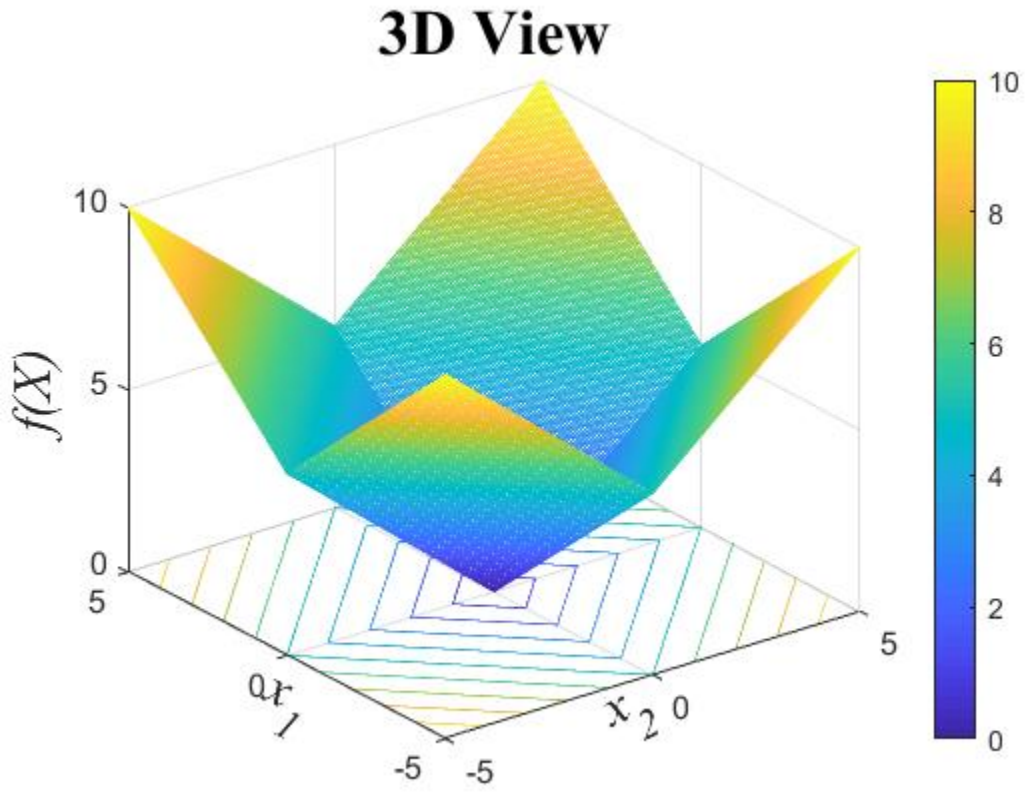
Un algoritmo genético (o AG para abreviar) es una técnica de programación inspirada en la reproducción de los seres vivos y que imita a la evolución biológica como estrategia para resolver problemas de optimización. Hay una gran cantidad de problemas que pueden ser resueltos mediante los algoritmos genéticos. Algunos de estos problemas son: Optimización de rutas, optimización de tareas, gestión automatizada, aprendizaje de comportamiento de robots, sistemas de sector financiero, encontrar errores de programación, etc.

1. Absolute Function

Ecuación:

$$f(x) = \sum_{i=1}^{n} |x_i|$$

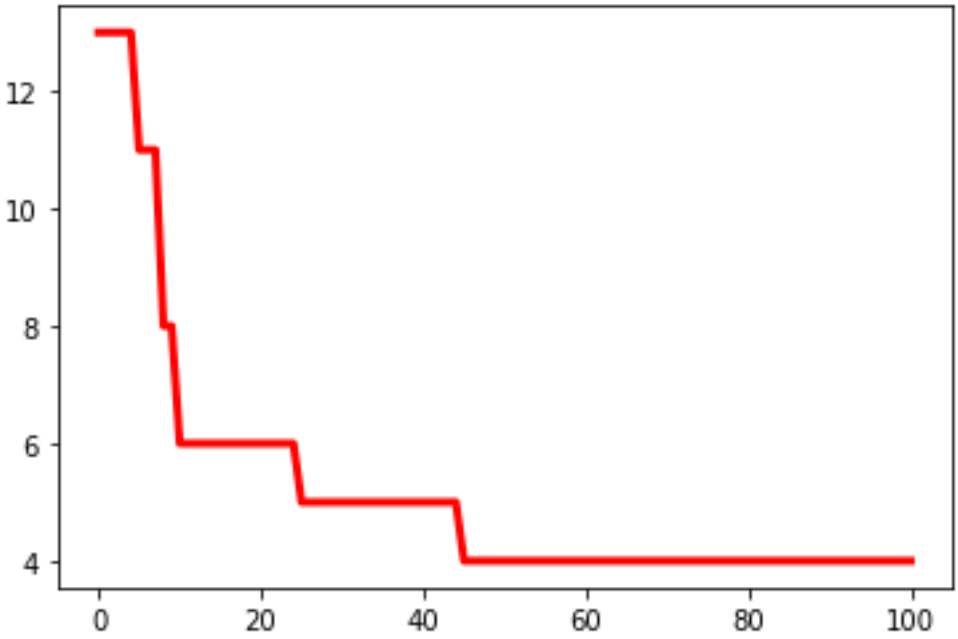
Grafica 3D



Mejores soluciones encontradas: Se realizo el algoritmo con 4 valores dentro del rango [10, -10]

```
MENOR COSTO:
4.2734066396951675
[ 0.15695363 -3.2562563   0.6947668   5.16186   ]
```

Gráfica, evolución del desempeño en 100 iteraciones



Código fuente (Python)

```python
import numpy as np
from numpy import random
import math
import matplotlib.pyplot as plt

class cIndividuos:
    rangos = []
    costo = math.inf


individuos = 4 #Numero de individuos en la poblacion
cromosoma = 4 #Numero de genes en un cromosoma
puntoCruzMenor = 1
puntoCruzMayor = 3

generaciones = 100 #Numero de veces que se repetira el algoritmo
```

```
fitnessValues = np.zeros(individuos) #Matriz para guardar los valores fitness de
la poblacion

varmin = [-10, -10, -10, -10]
varmax = [ 10,  10,  10, 10]
minCost = cIndividuos()


def calculateFunct(x):
    #Absolute function
    return sum(abs(x))

    ########## IMPRESION DE LOS DATOS ##########
def imprimirPoblacion(poblacion):
    for individuo in range(individuos):
        for gen in range(cromosoma):
            print("{:.2f}".format(poblacion[individuo][gen]), end = " ")
        print()
    print('------------------------------------------------------')

def imprimirFitness(fitnessValues):
    for individuo in range(individuos):
        print("{:.2f}".format(fitnessValues[individuo]), end = " ")
    print('------------------------------------------------------')


    ########## Metodo para encontrar al individuo mas fuerte ##########
def mayorCosto():
    mayor = 0
    indiceMayor = 0
    for ind in range(individuos):
        if(fitnessValues[ind]> mayor):
            mayor = fitnessValues[ind]
            indiceMayor = ind
    return indiceMayor

    ########## Metodo para encontrar al individuo mas debil ##########
def menorCosto():
    menor = math.inf
    indiceMenor = 0
    for ind in range(individuos):
        if(fitnessValues[ind] < debil):
            menor = fitnessValues[ind]
            indiceMenor = ind
    return indiceMenor

    ########## Metodo para calcular el fitness de la poblacion ##########
def fitness(minCost):
    for ind in range(individuos):
        fitnessValues[ind] = calculateFunct(poblacion[ind])
        if(fitnessValues[ind] < minCost.costo):
            minCost.costo = fitnessValues[ind]
            minCost.rangos = poblacion[ind]
            #print('Menor costo: ', minCost.costo, ' en ', minCost.rangos)
```

```
    ########## Metodo para intercambiar genes entre dos individuos, necesario
para el metodo cruzamiento() ##########
def intercambio(individuo1, individuo2, punto):
    aux1 = -1
    aux2 = -1
    #Aqui se hace un intercambio de valores entre individuos a partir del indice
indicado por la variable punto
    for gen in range(punto, cromosoma):
        aux1 = poblacion[individuo1][gen]
        aux2 = poblacion[individuo2][gen]
        poblacion[individuo1][gen] = aux2
        poblacion[individuo2][gen] = aux1

    ########## Metodo para cruzar dos individuos ##########
def cruzamiento():
    puntoCruz = random.randint(puntoCruzMenor, puntoCruzMayor)

    for individuo in range(individuos):
        if(individuo % 2 == 0): #Esta condicion es para que ingnore al segundo
individuo
            intercambio(individuo, individuo+1, puntoCruz)

    ########## Metodo para mutar individuos ##########
def mutacion():
    #En la mutacion solo se cambia un bit aleatorio de cada individuo, se tiene
que ingnorar a algun individuo
    #En mi caso el individuo ignorado es el mas debil que fue cambiado por el
fuerte
    for individuo in range(individuos):
        if(individuo != debil):
            mutado = random.randint(0, cromosoma)
            poblacion[individuo][mutado] = np.random.uniform(varmin[mutado],
varmax[mutado], 1)

    ########## Metodo para crear un individuo ##########
def crearIndividuo():
    ind = np.random.uniform(varmin, varmax, cromosoma)
    return ind

    ########## Metodo para crear una poblacion, requiere del metodo
crearIndividuo() ##########
def crearPoblacion():
    poblacion = np.zeros((individuos, cromosoma), dtype=('float32'))
    for individuo in range(individuos):
        poblacion[individuo] = crearIndividuo()
    return poblacion


minCost.costo = math.inf
minCost.rangos = fitnessValues
#Variables para graficar
x = np.zeros((generaciones+1), dtype=('float32'))
y = np.zeros((generaciones+1), dtype=('int32'))

poblacion = crearPoblacion()
```

```
imprimirPoblacion(poblacion)

fitness(minCost)
imprimirFitness(fitnessValues)

for i in range(generaciones+1):
    #Seleccion
    debil = mayorCosto() #Se guarda el individuo mas debil de la poblacion
    poblacion[mayorCosto()] = poblacion[menorCosto()] #Se intercambia el
individuo mas debil con el mas fuerte de la poblacion
    """
    print('SELECCION: ELIMINACION DEL INDIVIDUO MAS DEBIL')
    imprimirPoblacion(poblacion)
    """

    cruzamiento()
    """
    print('CRUZAMIENTO')
    imprimirPoblacion(poblacion)
    """

    #Mutacion
    mutacion()
    """
    print('MUTACION')
    imprimirPoblacion(poblacion)
    """

    #Se calcula de nuevo fitness para los individuos modificados
    fitness(minCost)
    """print('CALCULO DEL FITNESS CON INDIVIDUOS MODIFICADOS')
    imprimirPoblacion(poblacion)
    imprimirFitness(fitnessValues)
    """

    x[i] = i
    y[i] = minCost.costo


imprimirPoblacion(poblacion)

print()
print('MENOR COSTO:')
print(minCost.costo)
print(minCost.rangos)

#Grafica de las generaciones y mejores individuos
plt.plot(x, y, color='red', linewidth=3)
plt.show()
```

Conclusiones

Con la utilización de este tipo de algoritmos me sorprendí de la forma en que estos resuelven los problemas, en un principio no sabia como simular todo lo relacionado con la vida de organismos, pero entiendo todo mejor y ya me di cuenta como se hace. La forma de programar estos algoritmos depende de cada programador, se puede realizar de cualquier forma mientras se sigan los principios del funcionamiento de un algoritmo genético. Resolver problemas con esta técnica es realmente eficaz, a simple vista parece fuerza bruta, pero esto se hace de una forma más optimizada, existe bastante variación entre cada iteración y siempre se conservan los mejores resultados.

Bibliografía.

Conogasi. (2018). Algoritmos genéticos. 2022, Septiembre 15, Conogasi.org Sitio web: https://conogasi.org/articulos/algoritmos-geneticos/