Facultad de Ciencias
Cómputo Evolutivo - Práctica 6
Palmerin Morales David Gabriel

1. **Escribe una función que genere números pseudo-aleatorios de la distribución normal estándar N(0,1) a partir de números uniformemente distribuidos. Indica el método usado.**

   Se usa el método Box-Muller, el cual mapea números uniformemente aleatorios a una distribución normal con desviación de 1.

   http://www.lmpt.univ-tours.fr/~nicolis/Licence_NEW/08-09/boxmuller.pdf

   **Implementación:** Archivo Utils.py en función box_muller_transform
   **Ejecución:** python3 Practica06.py 1

2. **Implementa el algoritmo (1+1)-ES. Prueba tu algoritmo sobre la función conocida como Sphere. Utilizar un parámetro sigma = 1 y un punto de inicio x = (-99,...,-99)**

   a) **Ejecuta tu algoritmo para d = 10 y para d = 100 ¿Qué tan cerca del óptimo converge y qué tan rápido?**

   En el caso de 10 dimensiones se encontró un óptimo cercano a 0.2, el cual se tardó 10e6 iteraciones en obtenerlo.

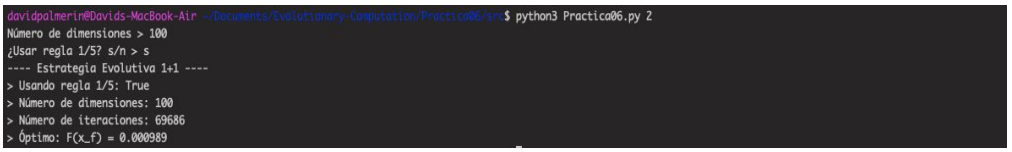

   Por otro lado, para 100 dimensiones se obtuvo un óptimo cercano a el cual tardó 10e6 iteraciones.

**b) Implementa la regla del ⅕ y vuelve a ejecutar tu algoritmo. ¿Qué diferencias observas respecto a la ejecución anterior?**

Claramente la regla ayuda a la convergencia del algoritmo pues termina muchísimo más rápido a comparación de la desviación fija con 1.

Gracias a estos cambios podemos dirigir la búsqueda de una mejor forma y obtener el resultado muchísimo más rápido. La diferencia la podemos ver en el número de iteraciones usadas para obtener los óptimos.


*Ejecución para 10 dimensiones*

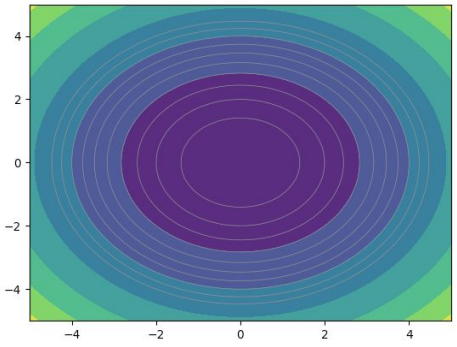



*Ejecución para 100 dimensiones*

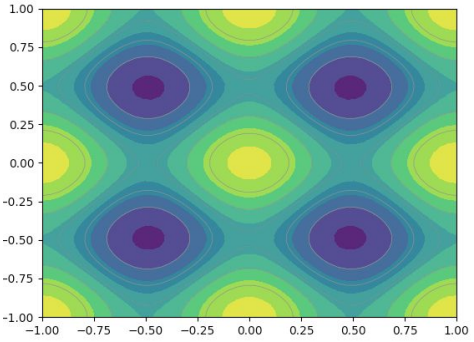
**Implementación:** Archivo ES1vs1.py
**Ejecución:** python3 Practica06.py 2

3. **Implementa el algoritmo (μ +λ)-ES usando una mutación no correlacionada. Para adaptar el tamaño de la mutación usa la recomendación dada en clase t = 1/√2. Prueba tu algoritmo sobre la función Sphere y Ackley. Buscar los parámetros μ, λ hasta alcanzar convergencia al óptimo global con un error por debajo de 0.001. Puedes usar la recomendación de parámetros λ/μ = 7**

   a) **Ejecuta tu algoritmo en *d = 2* y grafica los contornos de nivel de la función en los niveles de f(x): 2,4,6,8,10,12,14,16,18 y 20**




*Función Sphere con d = 2*




*Función de Ackley con d = 2*

   b) **¿Qué tan bien funciona para *d = 10* respecto a la (1+1)-ES?**

      **Comparando el número de iteraciones claramente es muchísimo mejor la estrategia (μ +λ)-ES pues únicamente toma 80 iteraciones para obtener un óptimo con precisión 0.001. Incluso obtiene una solución con mejor fitness.**

```
davidpalmerin@Davids-MacBook-Air ~/Documents/Evolutionary-Computation/Practica06/src$ python3 Practica06.py 3
Seleccionar función:
1. Sphere
2. Ackley
 > 1
Número de dimensiones > 10

Estrategia Evolutiva mu-lambda
        Buscando óptimo...
> Mu = 14 : Lambda = 100
> Número de dimensiones: 10
> Número de iteraciones: 80
> x_f = [-0.003020216478204316, 0.007482720114704975, 0.003215648735696826, 0.008694879881744043, -0.002736812214806863, 0.013509085591156133, 0.008939571445580333, -0.006204007889429547, 0.015813896506
725402, 0.013250343307347381, 0.001321207335540055]
> F(x_f) = 0.000885
```

**Implementación:** Archivo ES_mu_lambda.py
**Ejecución:** python3 Practica06.py 3