UNIVERSIDADE
LUSÓFONA

Inteligência Artificial

Trabalho Prático

Realizado por: Vera Oliveira a21801187

20 de janeiro de 2020

Ano Letivo: 2019/2020

Introdução

Este projeto foi desenvolvido no âmbito da disciplina de Inteligência Artificial, lecionada pelo professor Dr. David Rua.

Este trabalho consiste na realização e implementação de algoritmos heurísticos aprendidos na sala de aulas e realizar a minimização destes com funções propostas.

Para a minimização escolhi as funções Ackley e a função Rastrigin, e dos algoritmos heurísticos foram usados os de Entropia Cruzada e a Evolução Diferencial, em que foi proposto utilizar o x1 e x2. Para além destes dois, também usei o Algoritmo Genético, porém só consegui mostrar a sua utilização para x.

Escolhi os dois primeiros algoritmos mencionados por serem mais rápidos e por a Evolução Diferencial, ser em certa parte um melhoramento do Algoritmo Genético. Também, por ter uma grande mutação, isso iria fazer com que os resultados fossem diferentes.

Problema:

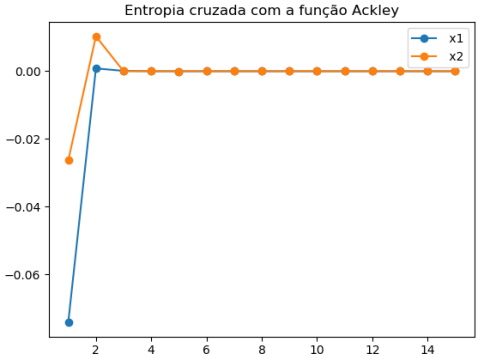
O primeiro problema com que me deparei foi a composição dos algoritmos. Foi aprendido nas aulas a teoria dos algoritmos heurísticos mencionados em cima, porém passar para código parecia e foi muito mais complicado. Após feitos os algoritmos heurísticos, com apenas um x, foi dado início a decomposição das funções para uma aplicação chamada de Desmos, para poder visualizar o resultado pretendido. Com estes valores em mente, comecei a mudar os valores dados inicialmente para que no final o valor fosse perto do valor dado no Desmos.

Entropia Cruzada com a função Ackley:

Função: 10*D + np.exp(1) - 20 * np.exp((-0.2) * np.sqrt((0.5)*(x1**2 + x2**2)))

Parametros de testes:

```
mu1 = 2
mu2 = 2
# 5 é a variancia
sigma1 = np.sqrt(5)
sigma2 = np.sqrt(5)
t = 0
maxits = 50
N = 1000
Ne = 10
```



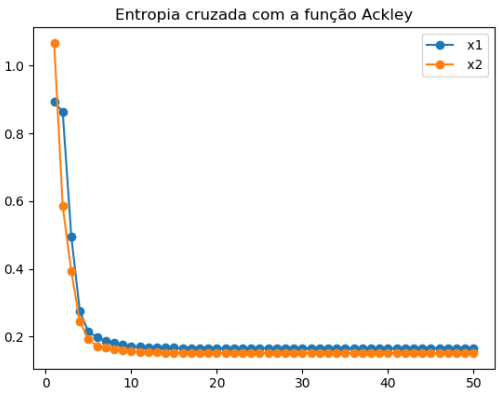
Valor:
```
mu1 = -2.5898774796892383e-16
mu2 = -6.467560092271607e-17
```
(perto do pretendido, mas não otimizado)

N=50

Valor final:

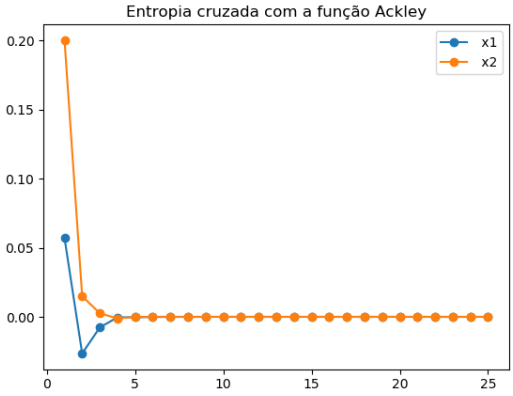
```
mu1 = 0.16577617382022064
mu2 = 0.15138677246968626
```

(perto do pretendido, mas não otimizado)



Versão final:

```
mu1 = 2
mu2 = 2
# 5 é a variancia
sigma1 = np.sqrt(5)
sigma2 = np.sqrt(5)
t = 0
maxits = 50
N = 100
Ne = 10
```

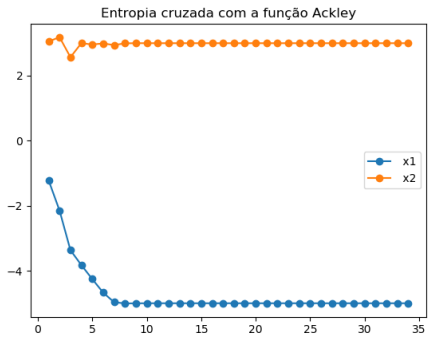


Com estes valores, foi feito o teste 100 vezes para fazer uma avaliação total do desempenho do algoritmo, este foi o resultado:

Algoritmo de Entropia Cruzada com a funcao de Ackley				
	Iteração de paragem	MU1 / X1	MU2 / X2	Tempo
Min	24	-1,933628334E-15	-8,35624255322E-05	5,983500
Média	26	-4,143519483E-16	1,20583529504E-18	6,982300
Max	50	5,163484045E-05	8,07615145000E-09	12,965400

Para salientar, caso seja gerado valores diferentes inicialmente, o gráfico fica diferente.

```
def ackley(x1, x2):
        x1 = x1 + 5
        x2 = x2 - 3
       return 20 + np.exp(1) - 20 * np.exp((-
0.2) * np.sqrt((0.5)*(x1**2 + x2**2)))
```



Valor final:
```
mu1 = -5.0
mu2 = 3.0
```
(como era pretendido)

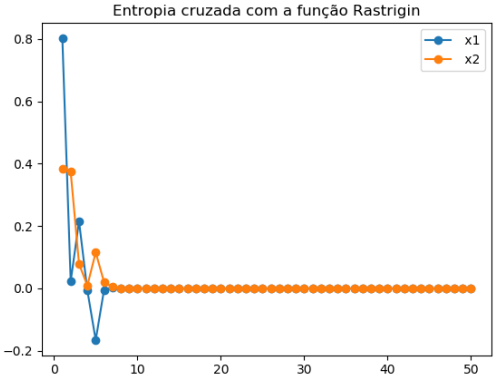
Entropia Cruzada com a função Rastrigin:

Após testes com a entropia cruzada com a função Ackley, foi muito mais fácil realizar o Rastrigin.

Função: 10*D + ((x1**2 - 10 * np.cos(2*np.pi * x1)) + (x2**2 - 10 * np.cos(2*np.pi * x2)))

Parametros de testes:

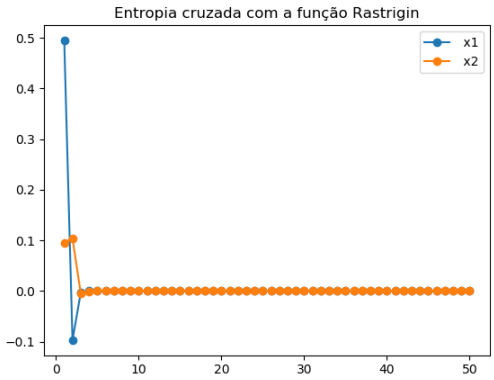
```
    mu1 = 2.26

    mu2 = 2.26
    # 5.12 é a variancia
    sigma1 = np.sqrt(5.12)
    sigma2 = np.sqrt(5.12)
    t = 0
    maxits = 50
    N = 100
    Ne = 10
```



Valor:
```
mu1 = -3.749678707233447e-09
mu2 = -6.656868659489069e-10
```
(perto do pretendido, mas não otimizado)

Versão final:

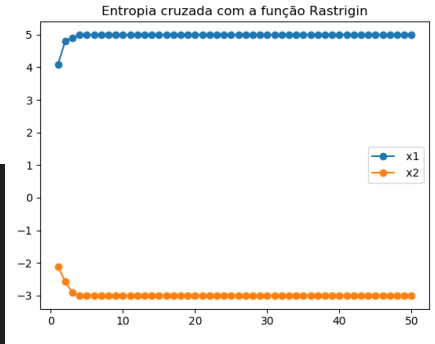
```
    mu1 = 2
    mu2 = 2
    # 5 é a variancia
    sigma1 = np.sqrt(5)
    sigma2 = np.sqrt(5)
    t = 0
    maxits = 50
    N = 100
    Ne = 10
```



Com estes valores, foi feito o teste 100 vezes para fazer uma avaliação total do desempenho do algoritmo, este foi o resultado:

Algoritmo de Entropia Cruzada com a funcao de Ratrigin				
	Iteração de paragem	MU1 / X1	MU2 / X2	Tempo
Min	50	-3,749815533E-09	-2,85794077959E-04	42,935000
Média	50	-3,749687221E-09	7,60809619760E-12	54,853050
Max	50	9,949586343E-01	9,94958637781E-01	70,840500

Para salientar, caso seja gerado valores diferentes inicialmente, o gráfico fica diferente.




```
def rastrigin(x1, x2):
        x1 = x1 - 5
        x2 = x2 + 3
        return 20 + ((x1**2 - 10 * np.cos(2*np.pi * x1)) + (x2**2 - 10 *
np.cos(2*np.pi * x2)))
```

Evolução Diferencial com a função Ackley:

A principal diferença que senti deste algoritmo para o de Entropia Cruzada foi o tempo que era necessário para correr o algoritmo.

A função era idêntica ao do Entropia Cruzada com Ackley, mas neste caso foi passado diretamente no ciclo, como:

```
result = list(de(lambda x1, x2: (20 + np.exp(1) - 20 * np.exp((-0.2)
* np.sqrt((0.5)*(x1**2 + x2**2)))), bounds=[(-5, 5)]))
```

Parâmetros iniciais: mut=0.6, crossp=0.7, popsize=20, its=100

Resultados obtidos:

Time = 1778.2895 ms

x1 = 0.0 with f(x1) = 2.7182818284590446

x2= -0.1270371851541725 with f(x2) = 2.7182818284590446

Versão final:

Parâmetros: mut=0.8, crossp=0.7, popsize=20, its=1000

Resultados finais:

Time = 1768.27 ms

x1=0.0 with f(x1) = 2.7182818284590446

x2=0.14400965795554743 with f(x2) = 2.7182818284590446

Com estes valores, foi feito o teste 100 vezes para fazer uma avaliação total do desempenho do algoritmo, este foi o resultado:

	Algoritmo de Diferencial Evolution com a funcao de Ackley				
	X1	f(x1)	X2	f(x2)	Tempo
Min	-8,881784197E-01	2.7182818284590446	-7,41881725103E-03	2.7182818284590446	1654,574700
Média	0,440892099	2.7182818284590446	-1,02050767968E-01	2.7182818284590446	1752,313800
Max	0,000000000E+00	2.7182818284590446	-1,96682718685E-01	2.7182818284590446	1850,052900

Evolução Diferencial com a função Rastrigin:

Mais uma vez, senti que após ter realizado este algoritmo com outra função, foi muito mais fácil de implementar o do Rastrigin.

A função era idêntica ao do Entropia Cruzada com Rastrigin, mas neste caso foi passado diretamente no ciclo, como:

```
result = list(de(lambda x1, x2: (20 + (x1**2 + x2**2 - 10 * (np.cos(2*np
.pi * x1) + (np.cos(2*np.pi * x2))))), bounds=[(-5.12, 5.12)]))
```

Parâmetros iniciais: mut=0.6, crossp=0.7, popsize=30, its=100

Resultados obtidos:

Time = 2621.9889 ms

x1=2.092983564239148e-10 with f(x1) = 0.0

x2=0.000226658787251921 with f(x2) = 0.0

Versão final:

Parâmetros: mut=0.8, crossp=0.7, popsize=20, its=1000

Resultados finais:

Time = 1771.7736 ms

x1=7.947278390929569e-10 with f(x1) = 0.0

x2=0.021499427177996466 with f(x2) = 0.0

Com estes valores, foi feito o teste 100 vezes para fazer uma avaliação total do desempenho do algoritmo, este foi o resultado:

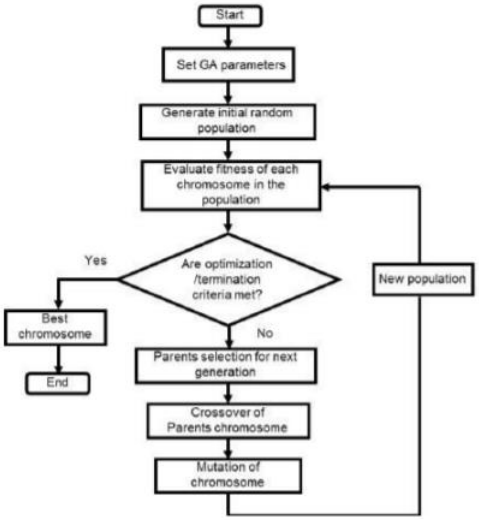
	Algoritmo de Diferencial Evolution com a funcao de Rastrigin				
	X1	f(x1)	X2	f(x2)	Tempo
Min	-9,889129515E-10	0	-2,09582226239054	0	1654,574700
Média	-2,559619183E-10	0,0	-0,48818936861455	0,0	1752,313800
Max	4,769891149E-10	1.9899181141865796	1,11944352516145	1.9899181141865796	1850,052900

P2: Servidores

Eu não consegui implementar esta parte do trabalho, mas no meu ver e juntamente com os valores gerados para o meu projeto P23, esta parte do trabalho consiste em gerar uma lista otimizada de gestor de tarefas consoante a capacidade do servidor, como a RAM e o CPU.

Com os dados fornecidos, era necessário filtrar e ordenar quais os trabalhos que cada servidor consegue fazer e quanto tempo demorava a realizar esse trabalho.

No meu parecer o algoritmo que teria mais eficácia seria o Algoritmo Genético, uma vez sendo um problema discreto, ou pode realizar ou não pode. Com isto fazer todos os passos que este algoritmo tem.



Conclusão

Com este trabalho consegui compreender melhor cada algoritmo e a sua complexidade. Na primeira parte do trabalho demonstrei os dois algoritmos que trabalhei, deixando de parte deste relatório o Algoritmo Genético, pela simples razão de não ter conseguido implementar as duas variáveis de x, mas este estará no repositório do GitLab.

No final deste trabalho sinto que o Entropia Cruzada foi o que melhor me adaptei, e que melhor resolveu o problema, uma vez que a Evolução Diferencial demorava muito mais tempo a fazer a sua execução. Também neste segundo senti que por muito que mudasse os parâmetros os tempos e as soluções não oscilavam muito.