Universidade Federal de Viçosa - Campus Florestal

Trabalho Prático I

Professor: Marcus Henrique Soares Mendes

Disciplina: Meta-heurísticas – CCF-480

Arthur Marciano - 3019

Florestal, MG
2023

ÍNDICE

1. Introdução

O objetivo deste trabalho é a implementação dos algoritmos de busca ILS (Iterated Local Search) e HC (Hill-Climbing) para a minimização de funções. A linguagem de programação escolhida foi o Python pela sua facilidade em realizar análises estatísticas e plotar gráficos.

Foram realizadas 30 execuções de cada algoritmo (ILS e HC) para cada uma das funções apresentadas na documentação do trabalho. Os valores e resultados obtidos serão apresentados ao longo deste trabalho.

2. Desenvolvimento

Implementação do ILS (Iterated Local Search)

Para a implementação do ILS começamos gerando uma solução inicial, onde os valores de x e y são obtidos aleatoriamente dentro do intervalo especificado. Em seguida o algoritmo utiliza o valor da solução inicial para obter o valor da função objetivo, é importante ressaltar que na primeira iteração a melhor função objetivo é a solução com os valores x e y iniciais.

```python
def ils(max_iter, perturbacao, inferior, superior, func_escolhida):
    melhor_solucao = init_solution.solucaoInicial(inferior, superior) #na primeira iteração a melhor solução é a inicial
    melhor_solucaoObj = obj_function.funcaoObjetivo(melhor_solucao, func_escolhida) #na primeira iteração a melhor função objetivo
                                                                                    #é a solução com os valores de x e y iniciais

    for i in range(max_iter):
        #faz uma busca local na solução atual
        solucao_atual, solucaoObjAtual = busca_local.buscaLocal(melhor_solucao, perturbacao, func_escolhida)

        # Se a função objetivo perturbada for melhor que a melhor solução objetivo encontrada até o momento ela atualiza
        if solucaoObjAtual < melhor_solucaoObj:
            melhor_solucao = solucao_atual
            melhor_solucaoObj = solucaoObjAtual


    return melhor_solucao, melhor_solucaoObj
```

Imagem 1: Implementação da função do ILS

A condição de parada do algoritmo ILS é quando o número máximo de iterações é atingido. Enquanto a condição não for satisfeita, o algoritmo entra em um loop, onde ele faz uma busca local utilizando a melhor solução encontrada até o momento.

A função de busca local é baseada no Hill-Climbing e recebe uma solução inicial e o intervalo de perturbação, e realiza a perturbação da solução atual iterativamente até encontrar uma solução vizinha que tenha uma função objetivo melhor. O critério de parada é quando não é mais possível melhorar a função objetivo.

```python
def buscaLocal(solucao, perturbacao, func_escolhida):
    x, y = solucao
    solucao_atual = x, y
    melhor_funcaoObj = obj_function.funcaoObjetivo(solucao_atual, func_escolhida)

    while True:
        #perturba a solução atual (melhor_solução)
        solucao_perturbada = perturbação.perturba(solucao_atual, perturbacao)
        funcaoObj_perturbada = obj_function.funcaoObjetivo(solucao_perturbada, func_escolhida)

        #verifica se a solução perturbada é melhor que a solução atual
        if funcaoObj_perturbada < melhor_funcaoObj:
            solucao_atual = solucao_perturbada
            melhor_funcaoObj = funcaoObj_perturbada
        else:
            break

    return solucao_atual, melhor_funcaoObj
```

Imagem 2: Implementação da função de busca local

A perturbação é feita na função 'perturba', ela recebe a solução atual e o intervalo de perturbação previamente definido como parâmetros de entrada. Nesta função também utilizamos valores aleatórios entre gerados entre o intervalo de perturbação, de modo a ter uma maior exploração do espaço de busca.

```python
def perturba(solucao, perturbacao):
    x, y = solucao
    #perturba a solução adicionando um valor aleatório à x e y
    perturba_x = x + random.uniform(-perturbacao, perturbacao)
    perturba_y = y + random.uniform(-perturbacao, perturbacao)

    return perturba_x, perturba_y
```

Imagem 3: Implementação da perturbação

Implementação do HC (Hill-Climbing)

A implementação do Hill-Climbing é um pouco mais simples que a do ILS, para esta parte foi criada uma função responsável por realizar os passos do algoritmo HC. Primeiro é feita a inicialização, gerando uma solução candidata aleatória utilizando como base a equação apresentada no material da disciplina:

- $vi = (lsi - lii) * ri + lii$,
    - $ri \rightarrow$é um número aleatório gerado por uma distribuição uniforme relativo à i-ésima variável de decisão.
    - $lsi \rightarrow$é o limite superior
    - $lii \rightarrow$limite inferior

Com base nesta equação foram gerados os valores aleatórios iniciais para x e y e atribuídos ao valor da solução candidata como mostra a figura abaixo.

```python
def hillClimbing(max_iter, inferior, superior, tam_passo, func_escolhida):

    #inicialização
    #gera uma solução candidata aleatória
    #vi = (lsi - lii) * ri + lii
    x = (superior - inferior) * random.uniform(0,1) + inferior
    y = (superior - inferior) * random.uniform(0,1) + inferior
    solucao = (x,y)
    #calcula o valor da solução objetivo inicial
    melhor_solucaoObj = obj_function.funcaoObjetivo(solucao, func_escolhida) #na primeira iteração a melhor função objetivo
                                                                              #é a solução com os valores de x e y iniciais
```

Imagem 4: Inicialização do Hill Climbing

Em seguida é feita a modificação, nesta etapa as soluções candidatas sofrem uma perturbação (dar um passo). Em seguida é feita a verificação da função objeto para ver se o candidato em questão é melhor, então só aceita a mudança da solução atual caso a solução do candidato seja melhor que a solução atual.

```python
    #modificação
    for i in range(max_iter):
        #dar um passo
        x_candidato = solucao[0] + random.uniform(0, 1) * tam_passo
        y_candidato = solucao[1] + random.uniform(0, 1) * tam_passo
        candidato = (x_candidato, y_candidato)

        #avaliar ponto candidato
        valor_candidato = obj_function.funcaoObjetivo(candidato, func_escolhida)

        #verifica se o candidato é melhor
        if valor_candidato <= melhor_solucaoObj:
            #armazena a nova solução
            solucao, melhor_solucaoObj = candidato, valor_candidato


    return solucao, melhor_solucaoObj
```

Imagem 5: Etapa de modificação Hill-Climbing

Configurações escolhidas para realização do algoritmo:

- **ILS (Iterated Local Search)**
  - Tamanho da perturbação: 0.5
  - Critério de parada: número máximo de iterações
  - Número máximo de iterações: 500

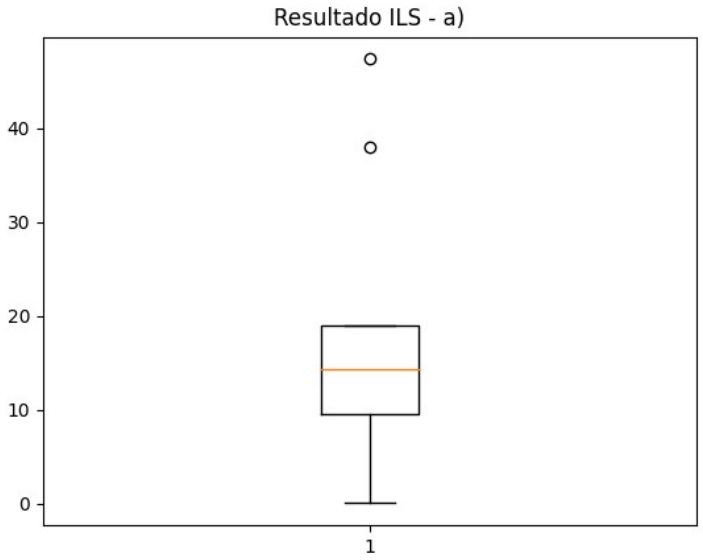
- HC (Hill-Climbing)
  - Critério de parada: número máximo de iterações
  - Número máximo de iterações: 500
  - Tamanho da perturbação: 0.5

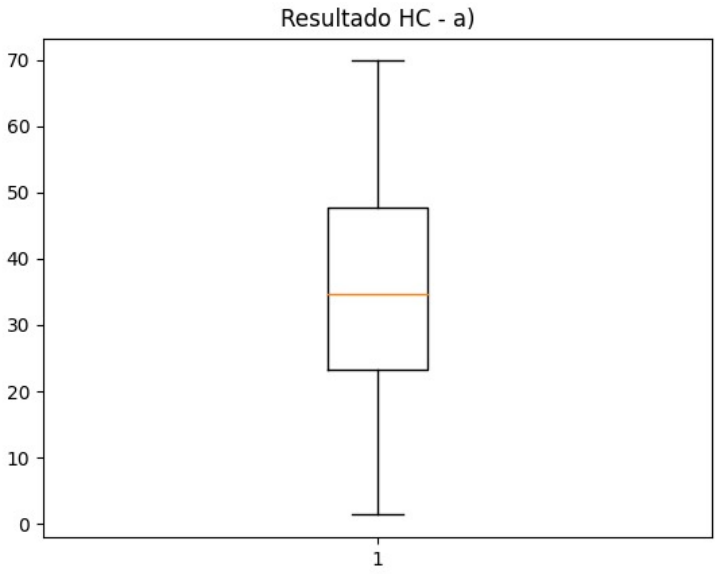
3. Resultados

a) $f(x, y) = x^2 + y^2 + 25(sen^2(x) + sen^2(y))$ com $-5 \leq x, y \leq 5$

Algoritmo	Mínimo	Máximo	Média	Desvio-padrão
ILS	0.014171631383 9151	47.42125530477 704	14.88183266757 3835	9.692109071674 848
Hill-Climbing	1.386703352743 6026	69.82304139732 884	35.13449746203 954	15.09851541743 6595

Boxplots:

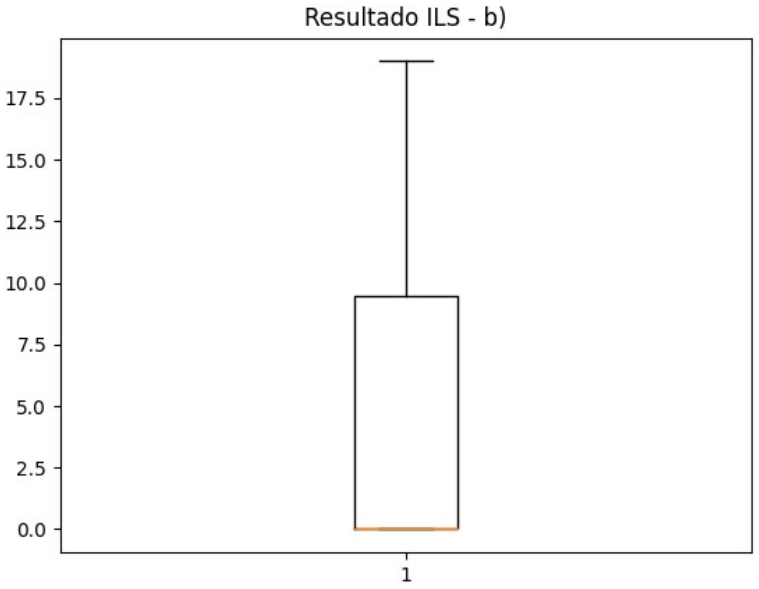


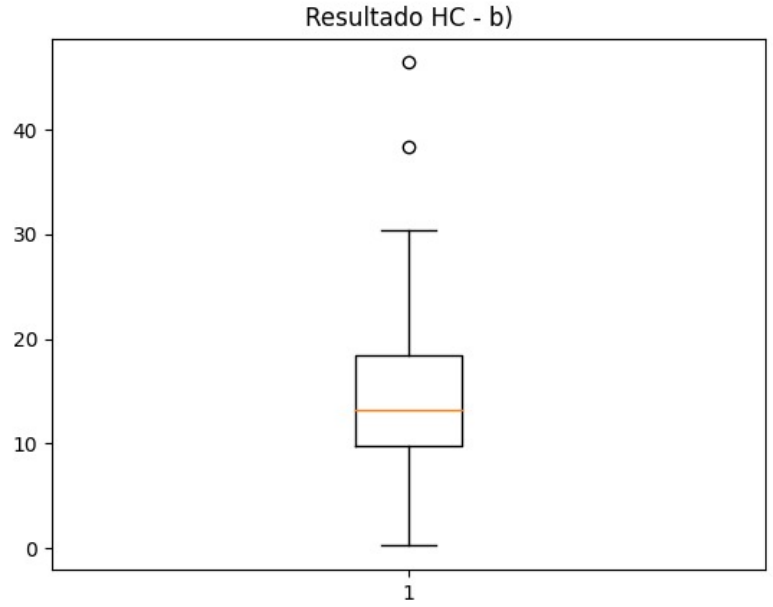


b) $f(x,y) = x^2 + y^2 + 25(sen^2(x) + sen^2(y))$ com $-2 \leq x, y \leq 2$

Algoritmo	Mínimo	Máximo	Média	Desvio-padrão
ILS	0.000726208541 1915967	19.00347656621 8207	3.813617089593 0523	5.253925346231 0285
Hill-Climbing	0.222828402509 17443	46.42521540054 565	15.48479730591 917	10.39183564491 6615

Boxplots:





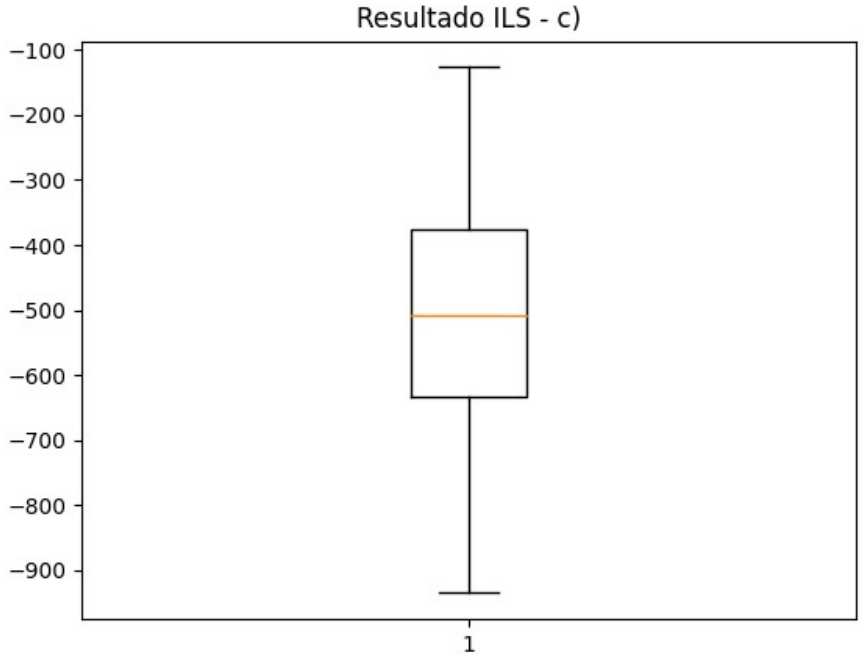
Valores das variáveis de decisão obtidos para a melhor solução encontrada no exercício b):

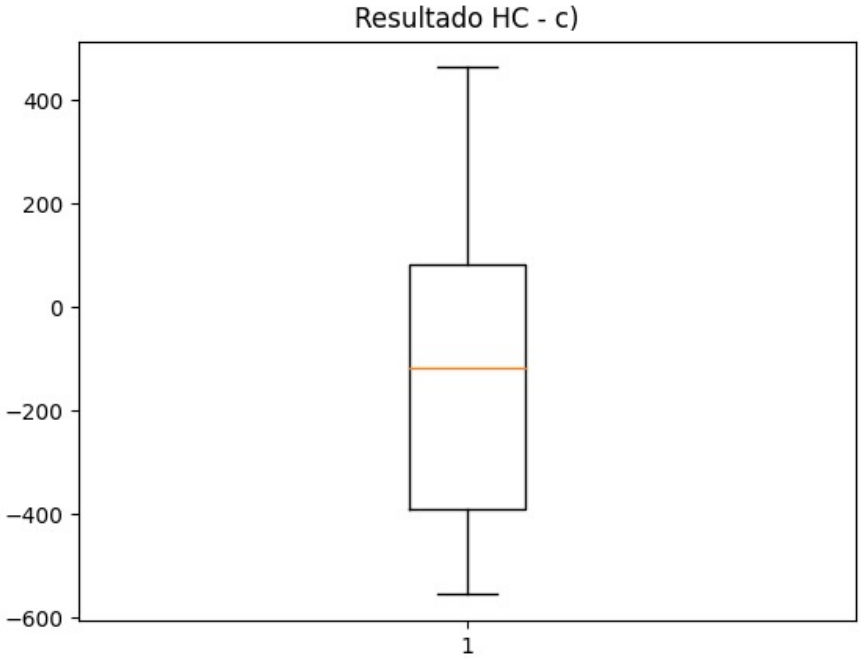
- **ILS**
  - x = -0.0027397850127156564
  - y = 0.004519383541045197
- **HC**
  - x = -0.007553622253245934
  - y = 0.0923937082700308

c) $f(x,y) = -(y+47)sin\left(\sqrt{|y+0.5y+47|}\right) - xsin\left(\sqrt{|x-(y+47)|}\right)$ com $-512 \leq x, y \leq 512$

Algoritmo	Mínimo	Máximo	Média	Desvio-padrão
ILS	-935.3376958425606	-126.42380783427839	-521.7946981591147	204.66551121165904
Hill-Climbing	-555.4342742680554	462.75319684374401	-108.3610554452565	291.63493789036005

Boxplots:





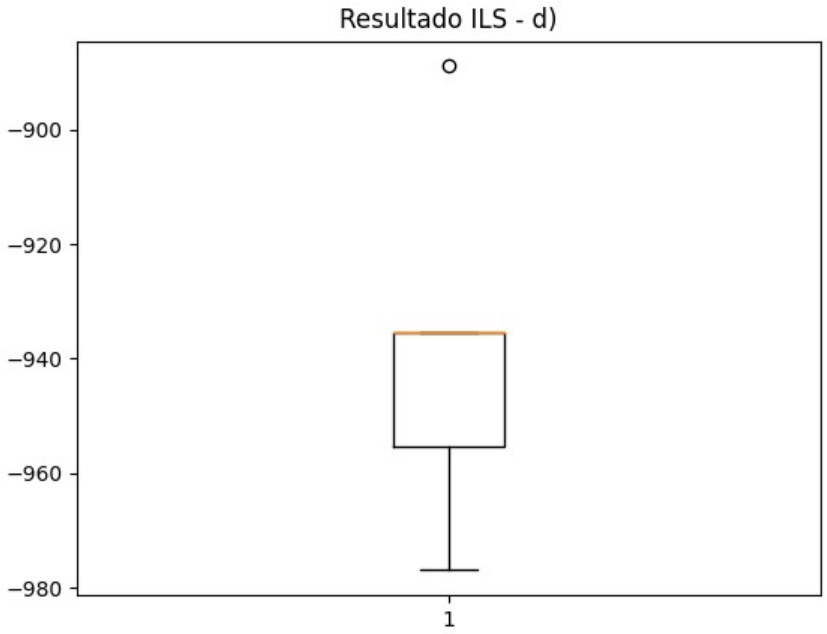
Valores das variáveis de decisão obtidos para a melhor solução encontrada no exercício c):

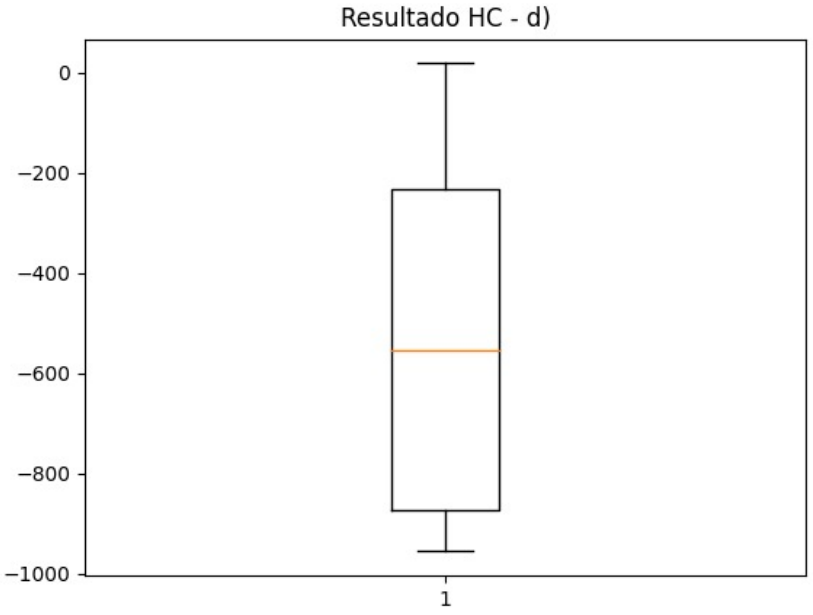
- **ILS**
  - x = 439.4438059465013
  - y = 453.9481050623504
- **HC**
  - x = -410.95352939343337
  - y = 97.84948523796903

d) $f(x,y) = -(y+47)sin\left(\sqrt{|y+0.5y+47|}\right) - xsin\left(\sqrt{|x-(y+47)|}\right)$ com $400 \leq x, y \leq 51\ 2$

Algoritmo	Mínimo	Máximo	Média	Desvio-padrão
ILS	-976.9108541831184	-888.9490246612246	-946.7224068658172	18.72362708797684
Hill-Climbing	-955.8015619746895	17.82571092628291	-523.7313946175859	339.0236272028435

Boxplots:





Valores das variáveis de decisão obtidos para a melhor solução encontrada no exercício c):

- **ILS**
  - x = 522.1368339788971
  - y = 413.3014336266153
- **HC**
  - x = 479.96959306133357
  - y = 430.4619914797496

4. Conclusão

Com a realização deste trabalho é possível ter uma visão mais clara a respeito do funcionamento de alguns dos algoritmos de busca estudados previamente em sala.

Analisando os resultados obtidos, e levando em conta que os dois tiveram o mesmo critério de parada, pode-se tirar de conclusão que o algoritmo ILS (Iterated Local Search) leva uma certa vantagem de desempenho em relação ao HC (Hill-Climbing), já que o mesmo atingiu valores de mínimos melhores que o HC.

Por fim, acredito que o trabalho foi muito útil, servindo de grande aprendizado e ajudando na fixação dos conceitos vistos na matéria.

-