# Relatório trabalho I - Meta-heurísticas

**Mateus Pinto da Silva - 3489**[1]

[1]Ciência da Computação – Universidade Federal de Viçosa (UFV-caf) – Florestal – MG – Brasil

(mateus.p.silva[1])@ufv.br

***Resumo.*** *Neste trabalho será apresentado uma implementação da meta-heurística Hill Climb e da Iterated Local Search, e ambos serão testados utilizando duas funções, cada uma testada em dois intervalos diferentes. Optei por utilizar a chance de perturbação de cada variável em 50% para ambos algoritmos. O HC utiliza ruído de até 10% do intervalo passado. O ILS utiliza ruído de até 100% do intervalo passado, e utiliza o HC para refinar sua tentativa, porém com apenas 1% do intervalo passado como ruído. O critério de parada foi de 1000 iterações sem melhoria em ambos algoritmos, embora o HC utilizado pelo ILS pare em 100 iterações sem melhoria. Conclui-se que a implementação foi satisfatória, alcançando bons resultados, e que o ILS aparenta a funcionar melhor que o HC para funções cujo intervalo abrange várias bacias de atração.*

## 1. Como executar

Para executar os algoritmos implementados, é necessário o compilador Rust. Ele pode ser baixado em Rustup.rs (hyperlink). Depois de baixado e instalado, basta acessar a pasta pelo terminal e executar o comando "cargo run --release". Por padrão, será executada 30 vezes o exercício 1a, porém isso pode ser mudado trocando a função executada e os limites no arquivo src/main.rs. Deixei todas as modificações necessárias comentadas no arquivo, a fim de facilitar a correção do professor/monitor.

## 2. Decisões de implementação

Quanto aos quesitos técnicos, optei pela linguagem Rust por ser muito performática, a fim de abusar do número de iterações sem muito custo de tempo. Quanto a decisões científicas (do próprio algoritmo), optei por utilizar 50% de chance de perturbação de cada variável, o que faz com que em 75% das iterações pelo menos uma das variáveis seja perturbada e em 25% as duas sejam. Testei outros valores, porém com esse tive resultados mais satisfatórios. Optei por utilizar o ruído baseado no próprio intervalo da função. Como o objetivo do algoritmo Hill Climb é encontrar a melhor solução de uma bacia de aproximação, utilizei o valor de 10% do intervalo, e como o do Iterated Local Search é achar bacias de aproximação, utilizei 100% do intervalo. Testei outros valores também, porém novamente esses levaram a melhores resultados.

## 3. Discussão dos resultados

| Problema com função objetivo 1 com intervalo a) para as variáveis de decisão | | | | |
|---|---|---|---|---|
| Algoritmo | Mínimo | Máximo | Média | Desvio-padrão |
| HC | -1.913223 | 1.228413 | -0.447133 | 1.567299 |
| ILS | -1.913223 | -1.913223 | -1.913223 | 0 |

| Problema com função objetivo 1 com intervalo b) para as variáveis de decisão | | | | |
|---|---|---|---|---|
| Algoritmo | Mínimo | Máximo | Média | Desvio-padrão |
| HC | -1.913223 | -1.913222 | -1.913223 | 0 |
| ILS | -1.913223 | -1.913223 | -1.913223 | 0 |

| Problema com função objetivo 2 com intervalo c) para as variáveis de decisão | | | | |
|---|---|---|---|---|
| Algoritmo | Mínimo | Máximo | Média | Desvio-padrão |
| HC | -959.400959 | -306.178546 | -653.666388 | 177.850723 |
| ILS | -959.640457 | -959.630904 | -959.638219 | 0.002274 |

| Problema com função objetivo 2 com intervalo d) para as variáveis de decisão | | | | |
|---|---|---|---|---|
| Algoritmo | Mínimo | Máximo | Média | Desvio-padrão |
| HC | -959.640652 | -959.639267 | -959.64019 | 0.000377 |
| ILS | -959.640663 | -959.640662 | -959.640663 | 0 |

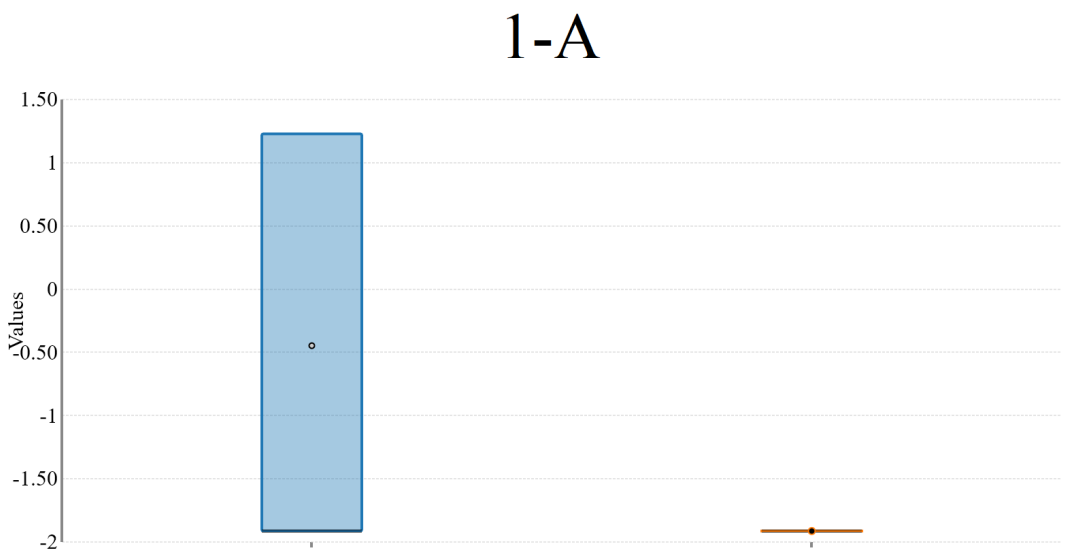


**Figura 1. Azul para o Hill-Climb, vermelho para o ILS**

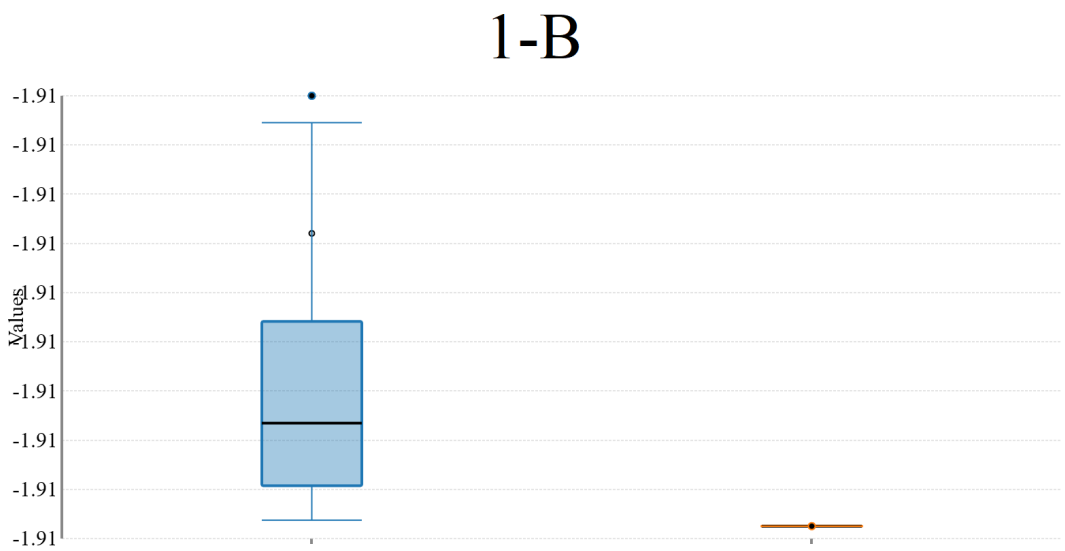


**Figura 2. Azul para o Hill-Climb, vermelho para o ILS**

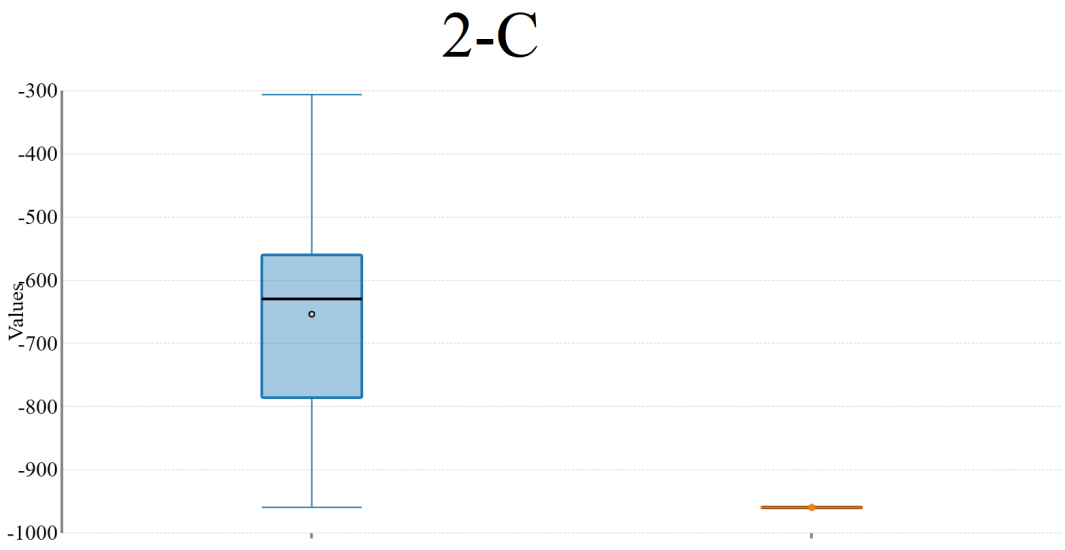


**Figura 3. Azul para o Hill-Climb, vermelho para o ILS**

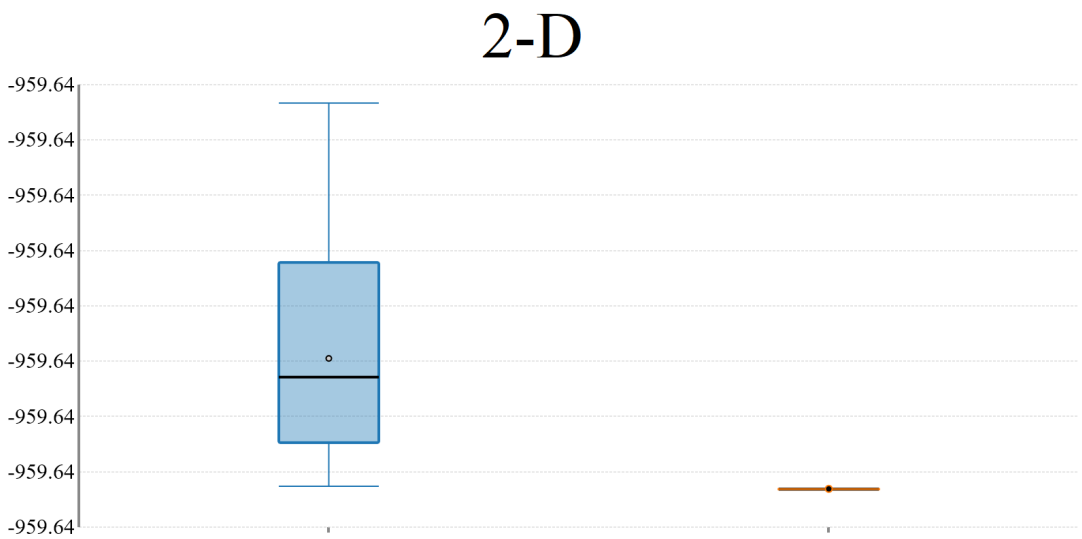


**Figura 4. Azul para o Hill-Climb, vermelho para o ILS**

Através da análise dos gráficos, das tabelas apresentadas neste relatório e dos gráficos das funções mostradas na especificação, é possível perceber que o algoritmo ILS funciona bem melhor do que o HC, visto que a maioria de suas execuções alcançou resultados melhores. Isso acontece pois o algoritmo HC tende a achar mínimos locais, que podem ser globais ou não. Essa diferença, no entanto, é menor nos intervalos B e D, pois eles são menores e contemplam o mínimo global de cada uma de suas respectivas funções. Além disso, o algoritmo HC tende a performar um pouco melhor em funções com poucas (ou apenas uma) bacias de atração.