Instituto Tecnológico de Aeronáutica - ITA
Inteligência Artificial para Robótica Móvel - CT213
Aluno: Gustavo Ferraresi Gottschild (ELE-25)
Data: 15 de abril de 2025

Relatório do Laboratório 3 - Otimização com Métodos de Busca Local

1 Breve Explicação em Alto Nível da Implementação

1.1 Definições fornecidas "*a priori*"

Foram definidas, pelo professor, as seguintes constantes para o presente algoritmo:

- **Função custo:** como um dos objetivos da presente atividade é a comparação dos aspectos de três do métodos de busca local ensinados na aula 3, todos os métodos deveriam utilizados para otimizar a mesma função custo. No caso, é exatamente isso que foi feito, com a função de custo denominada `"cost_function()"` e fornecida pelo professor;
- **Número máximo de iterações (`max_iterations`)**: foi de 1000, 1000 e 5000 para, respectivamente, os algoritmos *gradient descent*, *hill climbing* e *simulated annealing*;
- **Ponto inicial no espaço de busca (`theta0`)**: foi a origem $(0; 0)$, para todos os algoritmos;
- **Custo de parada (ε)**: foi de $\varepsilon = 1 \cdot 10^{-10}$ para todos os métodos;

1.2 Critérios de Parada

Todos os algoritmos de otimização baseados em busca local citados no presente relatório foram executados até que seus critérios de parada fossem cumpridos (ambos referentes ao melhor custo obtido até o momento e ao número máximo de iterações permitido).

1.3 Descida do Gradiente

São fornecidos pelo professor tanto a função de custo `"cost_function()"`, quanto o seu gradiente `"gradient_function()"`. Já a implementação propriamente dita do algoritmo *gradient descent* usou o hiperparâmetro "taxa de aprendizagem" $\alpha = 0,1$. Feito isso, fez-se, até que ou o custo do melhor ponto encontrado até o momento fosse menor que ε, ou o número máximo de iterações fosse excedido: foi multiplicado por α e subtraído do ponto `theta` o gradiente do custo calculado nesse mesmo ponto.

1.4 *Hill Climbing*

A ideia básica do método *hill climbing* é a seguinte:

1. São dados um espaço de buscas, um ponto inicial θ_0 em tal espaço e uma função custo definida em toda a sua extensão;
2. A partir de θ_0, encontram-se todos os seus pontos vizinhos. Normalmente, define-se o espaço de vizinhos de um ponto como sendo simétrico (na presente atividade, todos os 8 vizinhos de cada ponto estão a uma distância $\Delta = 2 \cdot 10^{-3}$ dele e equi-espaçados radialmente de 45^o);
3. Checa-se qual dos vizinhos tem o menor valor de função custo;
4. Caso o vizinho de menor custo tenha custo menor que o de θ_0, então a esse ponto é atribuído o valor de seu vizinho;
5. Caso nenhum dos vizinhos tenha custo menor que o custo de θ_0, retorna-se esse ponto como ponto ótimo.

Para a implementação desse algoritmo, foram implementados os subprogramas

- "`neighbors()`", a qual, dado um ponto θ_0, retorna a sua lista de vizinhos;
- "`hill_climbing()`", o qual implementa a execução do método de otimização em si.

1.5 *Simulated Annealing*

Primeiramente, vale ressaltar que, ao contrário do que foi dito em aula, "*annealing*" significa "*recozimento*", não "*têmpera*", os quais são tratamentos térmicos bastante diferentes.

No caso, sua ideia é a de usar uma "temperatura" para possibilitar que se foque mais em *exploration* e que, assim, se evite a convergência prematura para mínimos locais.

O procedimento da otimização *simulated annealing* é:

- São dados um espaço de buscas, um ponto inicial θ_0 pertencente a tal espaço, um *scheduling* de temperatura (isto é: uma definição de como a temperatura deve variar com as iterações, a qual simula a transferência de calor);
- Atualiza-se a temperatura $T[i]$ de acordo com o *scheduling* definido e com a iteração i do algoritmo;
- Caso ocorra de a temperatura ser negativa, retorna-se θ;
- Para o ponto θ, no qual se está efetuando a busca no momento, acha-se um vizinho seu de forma aleatória (reproduzindo a natureza estocástica em escala atômica do fenômeno da difusão). No caso, o espaço amostral do sorteio dos vizinhos é a circunferência de raio $\delta = 2 \cdot 10^{-3}$ com distribuição uniforme de probabilidade;

- Calcula-se, então, a "diferença de energia térmica `delta_E`", a qual é o custo do vizinho menos o custo do ótimo atual θ;

- Caso `delta_E` < 0, atribui-se a θ o valor (posição) desse vizinho. Ademais, diferentemente do método *hill climbing*, caso `delta_E` > 0, simula-se o procedimento da difusão atômica, a qual permite tornar o *simulated annealing* menos *greedy* que o *hill climbing*: sorteia-se uma variável aleatória $R \sim Uniforme([0; 1])$, gerando-se um número aleatório r. Então, θ receberá seu vizinho se, e somente se $r < exp(-\Delta E/T)$. Note que o termo exponencial remete à natureza difusiva das soluções das leis de Fick, que são EDPs semi-lineares parabólicas que determinam a dinâmica da difusão física;

2 Figuras Comprovando Funcionamento do Código

Em todas as execuções, foi guardado o histórico de pontos ótimos atuais, de forma a possibilitar a visualização de como transcorreram as otimizações. Na figura abaixo, uma "estrela de cinco pontas amarela" representa o ponto inicial θ_0 do espaço de buscas e um "X vermelho" representa o resultado da otimização.

Vale notar que, para o resultado da *simulated annealing*, a trajetória é mais ruidosa que as duas demais, algo explicado pela "difusão" presente no algoritmo.

2.1 Descida do Gradiente

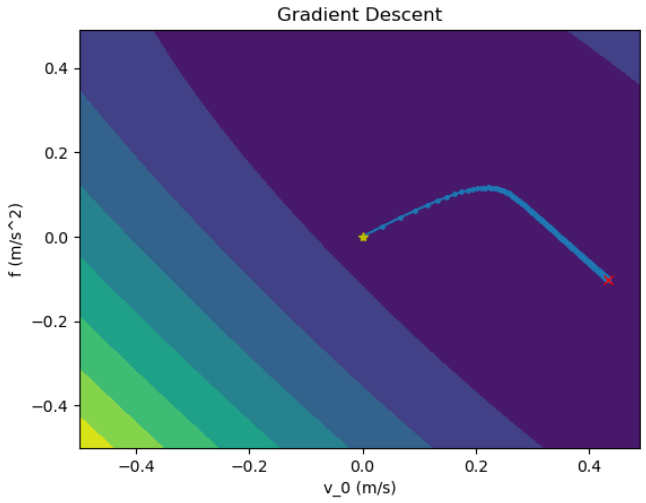

Figura 1: Trajetória da otimização via *gradient descent*.

2.2 *Hill Climbing*

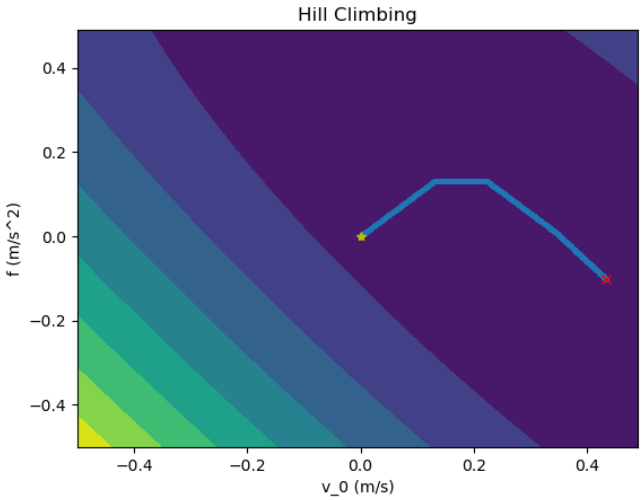

Figura 2: Trajetória da otimização via *hill climbing*.

2.3 *Simulated Annealing*

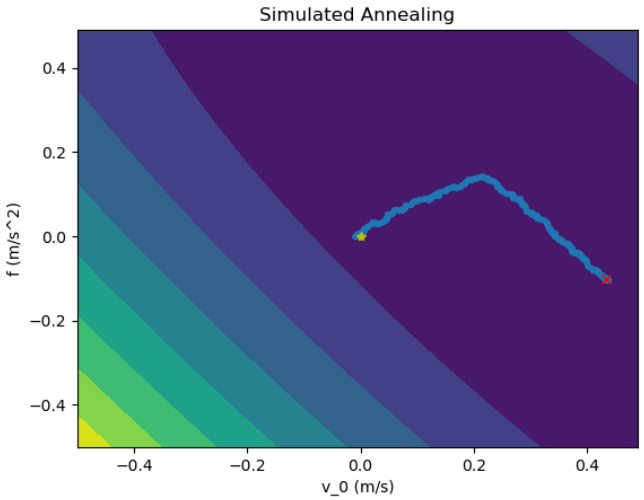

Figura 3: Trajetória da otimização via *simulated annealing*.

3 Comparação entre os métodos

Nas figuras abaixo, têm-se as comparações entre as otimizações, tanto em trajetória no espaço de buscas, quanto em retas obtidas *versus* dados experimentais (vale notar que, como os valores

de v_0 e de f obtidos nas quatro otimizações são muito próximos entre si, na Fig. 5, é muito difícil distinguir as três retas).

No caso das retas ajustadas, foi, para efeito de comparação, usado o método dos mínimos quadrados para ajustar uma quarta reta aos dados.

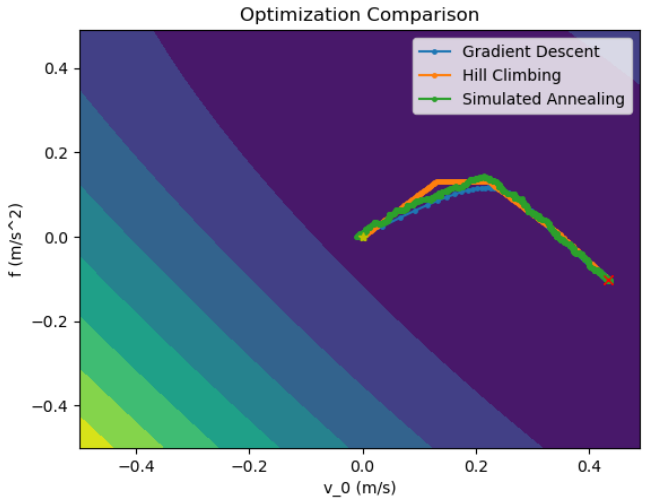

Figura 4: Trajetórias das três otimizações realizadas.

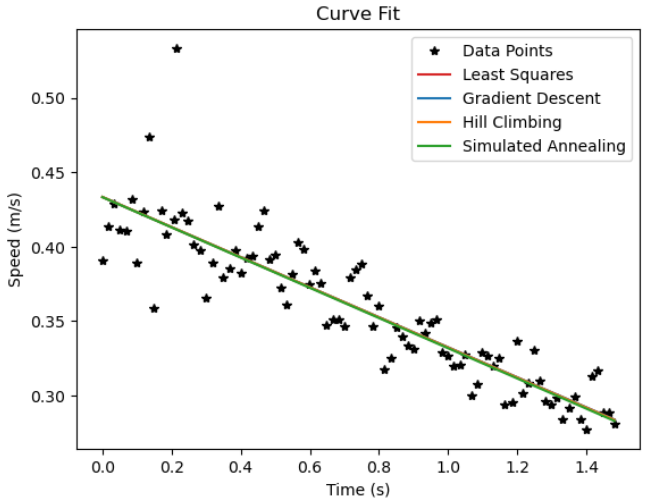

Figura 5: Retas resultado das otimizações comparadas aos pontos experimentais.

Na Tab. 1, tem-se a comparação dos parâmetros da regressão linear obtidos pelos 3 métodos de otimização estudados, além dos parâmetros obtidos via *least squares*. **Nota-se que, para todas as quatro otimizações, os parâmetros são bastante próximos entre si, o que corrobora a corretude das três técnicas de busca local no presente problema.**

Tabela 1: Parâmetros da regressão linear obtidos pelos métodos de otimização.

Método	v_0	f
MMQ	0,433373	-0,101021
Descida do gradiente	0,433371	-0,101018
Hill climbing	0,433411	-0,101196
Simulated annealing	0,433279	-0,101545