

|  |  |  |
| --- | --- | --- |
|  | 2023/2024 |  |
|  | |  |
| Trabalho Prático 2    **Corpo Docente:**  **José Oliveira**  **Eduardo Pires**  Engenharia Informática  *Inteligência Artificial* | | |
|  |  |  |
|  |  |  |
|  |  |  |

al70759 Cláudia Oliveira da Silva  
al71134 Pedro Miguel Monteiro Barroso

# Resumo

Este relatório propõe-se a realizar uma análise aprofundada de três métodos essenciais de investigação em Inteligência Artificial, centrados na otimização de problemas complexos. Os algoritmos escolhidos - Subida da Colina, Simulated Annealing e Algoritmo Genético - desempenham papéis cruciais na procura de soluções eficientes em vários contextos.

O relatório abordará a implementação específica de cada algoritmo para este cenário, começando pela Subida da Colina, seguida pelo Simulated Annealing e, por fim, o Algoritmo Genético. O uso de parâmetros adequados, como temperatura, fator de decaimento, e probabilidade de cruzamento e mutação, será crucial para o desempenho eficaz de cada algoritmo.

Para além das implementações, serão apresentados resultados detalhados, acompanhados por gráficos que ilustram o desempenho dos algoritmos ao longo de iterações.0

Índice

[Resumo 1](#_Toc153185983)

[1. Introdução 4](#_Toc153185984)

[1.1 Objetivos do Projeto: 4](#_Toc153185985)

[2. Conclusão 5](#_Toc153185986)

[3. Bibliografia 5](#_Toc153185987)

# Introdução

## Objetivos do Projeto:

Adquirir conhecimentos e desenvolver competências relativas aos algoritmos estudados.

# Algoritmos

## Hill Climb(Anexo A)

## Simulated Annealing(Anexo B)

## Genetic Algorithm(Anexo C)

# Conclusão

# Bibliografia

# Anexo A

function [x\_best, f\_best, best\_hc] = HillClimb(f, x0,lower,upper, max\_iters)

figure

hold on

title("HC run")

x=0:10^-5:1;

fx = f(x);

plot(x,fx,'b');

x\_best = x0;

f\_best = f(x\_best);

best\_hc = [];

plot(x\_best,f\_best,'\*b');

for i = 1:max\_iters

r = (rand() - 0.5)\*10^-2;

x\_new = x\_best + r;

while x\_new < lower || x\_new > upper

r = (rand() - 0.5)\*10^-2;

x\_new = x\_best + r;

end

f\_new = f(x\_new);

if f\_new > f\_best

x\_best = x\_new;

f\_best = f\_new;

plot(x\_best,f\_best,'\*r');

end

best\_hc = [best\_hc, f\_best];

end

end

# Anexo B

function [x\_best, f\_best,best\_sa,temp\_sa] = SimulatedAnnealing(f, x0, max\_iters,lower,upper, initial\_temperature, cooling\_rate)

figure

hold on

title("SA run")

x=0:10^-5:1;

fx = f(x);

plot(x,fx,'b');

it = 0;

x\_best = x0;

f\_best = f(x\_best);

best\_sa = f\_best;

temp\_sa = initial\_temperature;

temperature = initial\_temperature;

plot(x\_best,f\_best,'\*r');

% Adiciona vetores para armazenar valores de temperatura e iterações

temp\_values = [];

iter\_values = [];

% prob\_values = [];

%ciclo repetido 5 vezes

while it < max\_iters

r = (rand() - 0.5)/20;

x\_new = x\_best + r;

while x\_new < lower || x\_new > upper

r = (rand() - 0.5)/20;

x\_new = x\_best + r;

end

f\_new = f(x\_new);

delta\_f = f\_new - f\_best;

probability = exp(-abs(delta\_f )/ temperature);

if delta\_f > 0

x\_best = x\_new;

f\_best = f\_new;

else

if rand() < probability

x\_best = x\_new;

f\_best = f\_new;

end

% prob\_values =[prob\_values,probability]

end

temperature = temperature \* cooling\_rate;

plot(x\_best,f\_best,'\*r');

it = it + 1 ;

best\_sa = [best\_sa,f\_best];

temp\_sa = [temp\_sa, temperature];

% Armazena valores de temperatura e iterações

temp\_values = [temp\_values, temperature];

iter\_values = [iter\_values, it];

end

% Gráfico da temperatura em função das iterações

figure

plot(iter\_values, temp\_values, 'g');

title('Temperature vs Iterations');

xlabel('Iterations');

ylabel('Temperature');

%figure

%plot(iter\_values,prob\_values);

%figure

%plot(iter\_values,x\_best);

end

# Anexo C