UNIVERSIDADE DE TRÁS-OS-MONTES E ALTO DOURO

Módulo 1
Hill Climbing e *Simulated Annealing*

Licenciatura em Engenharia Informática

Inteligência Artificial

Paulo Oliveira

Eduardo Pires

Autor

- Marcelo Queirós Pinto – Al60102

Vila Real, 2016

Índice

Índice de Figuras

1. Introdução

No âmbito da Unidade Curricular Inteligência Artificial, foi proposto um trabalho sobre os algoritmos *Hill Climbing* e *Simulated Annealing*, o qual envolveu uma parte prática na qual se desenvolveu código, com a orientação do professor Paulo Oliveira durante as aulas, e uma parte teórica, presente neste relatório.

O estudo destes algoritmos, *Hill Climbing* e *Simulated Annealing*, é relevante para a compreensão dos algoritmos de otimização mais elaborados e eficientes, e sendo assim, pretendo com este relatório adquirir conhecimentos e desenvolver competências relativamente aos algoritmos em estudo mas também facilitar o desenvolvimento futuro de conhecimentos acerca de outros algoritmos de otimização.

Para a realização deste trabalho foi-nos fornecido um protocolo etc O presente relatório encontra-se dividido pontos etc

O desenvolvimento deste relatório é feito a partir de princípios hermenêuticos, baseando-se portanto na revisão bibliográfica e código efetuado, além da minha interpretação do que foi pesquisado e dos conhecimentos que adquiri aquando da elaboração deste trabalho. Para a realização deste trabalho foi utilizado o *software Matlab*.

2. Enquadramento Teórico

Após concluída a pesquisa de informação pedida no Ponto 1 do protocolo, fiz uma síntese da informação recolhida, a qual é pedida no Ponto 2 do protocolo e apresento de seguida, dividida nos pontos que considero relevantes. Na última etapa deste relatório encontram-se as referências bibliográficas que utilizei.

2.1. Algoritmos de Otimização

Um algoritmo de otimização é um algoritmo que tem como objetivo encontrar o máximo ou o mínimo de uma função, dependendo do problema, num determinado espaço de pesquisa, podendo as variáveis terem restrições.

Os algoritmos de otimização podem ser determinísticos ou Estocásticos/Aleatórios. Um algoritmo de otimização determinístico é um algoritmo que não utiliza métodos aleatórios para o cálculo de variáveis, e portanto é possível prever todos os seus passos conhecendo o seu ponto de partida, porque tem sempre a mesma resposta para o mesmo ponto inicial. Um algoritmo Estocástico ou Aleatório é um algoritmo que possui variáveis que de alguma forma são calculadas com base em métodos aleatórios, podendo simular processos reais.

Existem vários algoritmos de otimização, entre eles: algoritmo de *Ford-Fulkerson*, algoritmos genéticos, algoritmos de estimação de distribuição, *Branch and bound*, Decomposição de *Dantzig-Wolfe*, Divisão e Conquista, algoritmos gulosos, método do gradiente conjugado, *Minimax*, otimização por enxame de partículas, pesquisa tabu, algoritmos com programação dinâmica, algoritmos com programação inteira, algoritmo *simplex*, *Hill Climbing* e *Simulated Annealing*, entre outros. Os dois últimos são estudados de seguida.

2.1.1. *Hill Climbing*

O algoritmo *Hill-Climbing* é um algoritmo estocástico (definição de algoritmo estocástico atrás) que tem como base começar num ponto aleatório do espaço de pesquisa e ir pesquisando no seu raio de vizinhança melhores soluções, melhorando a solução ao longo das iterações até estagnar. A estagnação pode acontecer na solução ideal ou o algoritmo pode ter sido enganado por um máximo ou mínimo local ou ter estagnado numa planície, ambas as situações por não ter conseguido encontrar uma melhor solução no raio de vizinhança e portanto nestes casos não consegue encontrar o máximo ou mínimo global (dependendo se se trata de um problema de maximização ou minimização). Assim, este algoritmo define-se como um algoritmo com

aperfeiçoamento iterativo porque a cada iteração deverá acontecer uma melhoria da solução ou caso não aconteça o novo ponto é descartado, nunca piorando a solução.

2.1.2.Variantes do algoritmo *Hill Climbing*

Existem várias variações do algoritmo Hill Climbing, entre as quais:

- ***Basic Hill Climbing:*** O algoritmo básico do *Hill Climbing* faz uma pesquisa intensiva pela vizinhança, melhorando a sua solução até encontrar uma solução inferior à solução anterior;
- ***First Choice*:** Este algoritmo é normalmente utilizado quando existe um número elevado de sucessores. Começa por gerar sucessores aleatoriamente, até que um desses pontos gerados tenha melhor valor que o ponto atual. Se este for encontrado, o ponto atual é substituído.
- ***Best First Choice:*** Este algoritmo define que será escolhida a primeira solução de teste na vizinhança que origine uma melhoria, portanto apenas no seu pior caso a vizinhança será toda analisada;
- ***Stochastic Hill Climbing:*** Este algoritmo não examina toda a vizinhança, em vez disso escolhe aleatoriamente um vizinho e decide, mediante o melhoramento da vizinhança;
- ***Random Best First Choice:*** Neste algoritmo um vizinho é escolhido de forma aleatória e será aceite caso o seu valor seja melhor que o da solução atual, caso contrário será escolhido outro vizinho. O método dá-se por terminado quando não é encontrado um vizinho mais adequado num número fixo de iterações ou quando se depara com um máximo/mínimo local;
- ***Multiple Restart:*** Este algoritmo faz reinicializações do método *Hill Climbing* quando cumprido um determinado critério de estagnação, gerando portanto um conjunto de soluções finais possivelmente diferentes. A possibilidade de este método ser enganado por máximos ou mínimos locais é muito menor que utilizando o *Hill Climbing* habitual, uma vez que este método gera mais soluções finais diferenciadas e provavelmente de sítios diferentes do espaço de pesquisa, sendo muito mais provável encontrar a solução ideal.

2.2.3.Aplicações do algoritmo *Hill Climbing*

Apesar de existirem algoritmos mais eficientes e vantajosos para a maioria dos problemas, O algoritmo *Hill Climbing* tem imensas aplicações.

Após pesquisa na internet, encontrei alguns problemas que utilizaram o algoritmo *Hill Climbing* para chegar à solução, o primeiro problema que encontrei diz respeito à conceção de produtos químicos para o custo mínimo por unidade. O segundo problema lida com um método para obter o melhor perfil de temperatura numa reação química.

Além destes problemas particulares, outros problemas mais vulgares podem ser solucionados de maneira mais ou menos eficiente a partir do algoritmo *Hill Climbing*, entre eles: problema do caixeiro viajante, o problema oito rainhas e projeto de circuito.

2.1.3.*Simulated Annealing*

O algoritmo *Simulated Annealing* foi originalmente inspirado a partir de um processo termodinâmico que requer o aquecimento de um metal até atingir uma determinada temperatura seguido do arrefecimento lento para evitar ruturas. Este método envolve o aquecimento e arrefecimento de um material para alterar as suas propriedades físicas devido às mudanças na sua estrutura interna. Como o metal arrefece, a sua nova estrutura torna-se fixa, e consequentemente, faz com que o metal obtenha estas novas propriedades.

Assim, este algoritmo utiliza uma variável temperatura, a qual é utilizada para o cálculo da probabilidade de aceitar novos valores (falarei disto em detalhe na etapa Resultados Obtidos, analisando e discutindo a importância deste parâmetro, assim como outros), e portanto, este é um algoritmo estocástico (definição de algoritmo estocástico atrás) de otimização que tem como finalidade a aproximação da solução ótima global de uma dada função, para um dado problema. É um algoritmo que utiliza meta heurísticas, o que significa, no caso do algoritmo em questão, que aceita uma melhoria ou pioria mediante uma probabilidade, variável em cada implementação de forma a ser adaptável ao problema em questão. É um ótimo algoritmo quando o espaço de pesquisa é discreto, tendo uma taxa de sucesso muito boa, mas também é um algoritmo muito bom para espaços de pesquisa contínuos, para os quais será estudado neste relatório. Além disso, funciona com taxa de sucesso bastante melhor que o algoritmo *Hill Climbing* quando se pretende encontrar um máximo/mínimo global, fazendo-o num tempo bastante aceitável, sendo preferível muitas

vezes quando comparado com alternativas como o gradiente descendente, algoritmo não estudado neste relatório.

2.1.4.Aplicações do algoritmo *Simulated Annealing*

Após pesquisa na internet, encontrei alguns problemas que utilizaram o algoritmo *Simulated Anneling* para chegar à solução: Sistemas de Distribuição de Água - usado para agendar as bombas de água em um sistema. Para avaliar o desempenho de cada configuração e para verificar as restrições necessárias, é utilizado um sistema de simulação; Simulações físicas; Problema da mochila e Caixeiro Viajante.

3. Resultados Obtidos

Após explicados os algoritmos, apresento os resultados obtidos para os pontos 3, 4, 5 e 6 do protocolo, que envolviam respetivamente *Hill Climbing*, *Restart Hill Climbing* e *Simulated Annealing* para os últimos 2 pontos.

Nos pontos 3, 4 e 5 a função utilizada é:

$$f_1(x) = 4(\sin(5\,\pi\,x + 0.5)^6 \exp(\log_2((x-0.8)^2))) \quad 0 \le x \le 1.6$$

No ponto 6 é utilizada a função:

$$f_2(x,y) = 0.5 + \frac{\sin(sqrt((x^2+y^2)^2 - 0.5))}{1 + 0.001(x^2+y^2)^2}, \quad -2.048 \le x, y \le +2.048$$

Após vários testes feitos com diferentes parâmetros para cada algoritmo, escolhi os parâmetros com que consegui obter melhores resultados com eficácia em termos de tempo. De seguida apresento os resultados.

3.1. *Hill Climbing*

Para o algoritmo *Hill Climbing* utilizei raio=1.6/100 e iterações=100, de seguida apresentam-se os diferentes resultados que são obtidos:

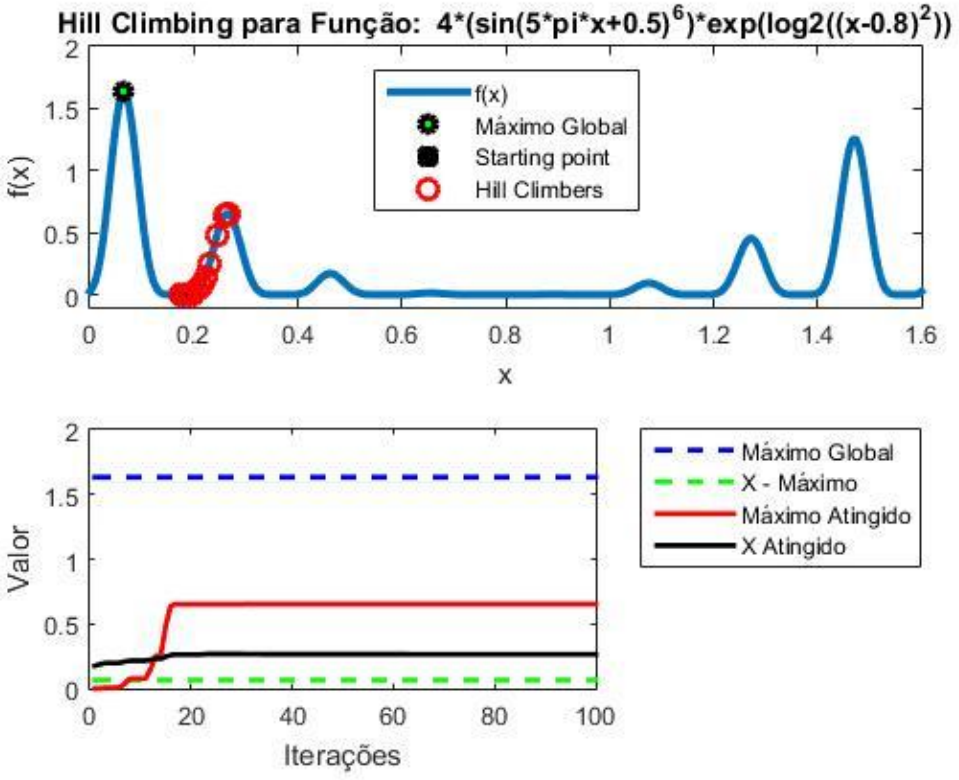

Figura 1: Hill Climbing falhou para um máximo local

Na figura acima verifica-se que o algoritmo Hill Climbing não conseguiu encontrar o máximo global e subiu para um máximo global, situação muito frequente neste algoritmo, revelando-se um forte problema quando se pretende chegar à melhor solução, neste caso, o máximo global. O algoritmo fica preso num máximo local porque o raio de vizinhança não permite encontrar um valor superior ao atual e portanto, como o algoritmo não aceita atualizações inferiores ao valor atual não muda o valor e ficará preso durante o resto das iterações. Também não seria bom o aumento do raio de vizinhança porque o algoritmo perderia precisão, não conseguiria chegar a um máximo facilmente porque não conseguiria subir de forma adequada uma colina, saltando logo para outra, isto, na maioria das vezes.

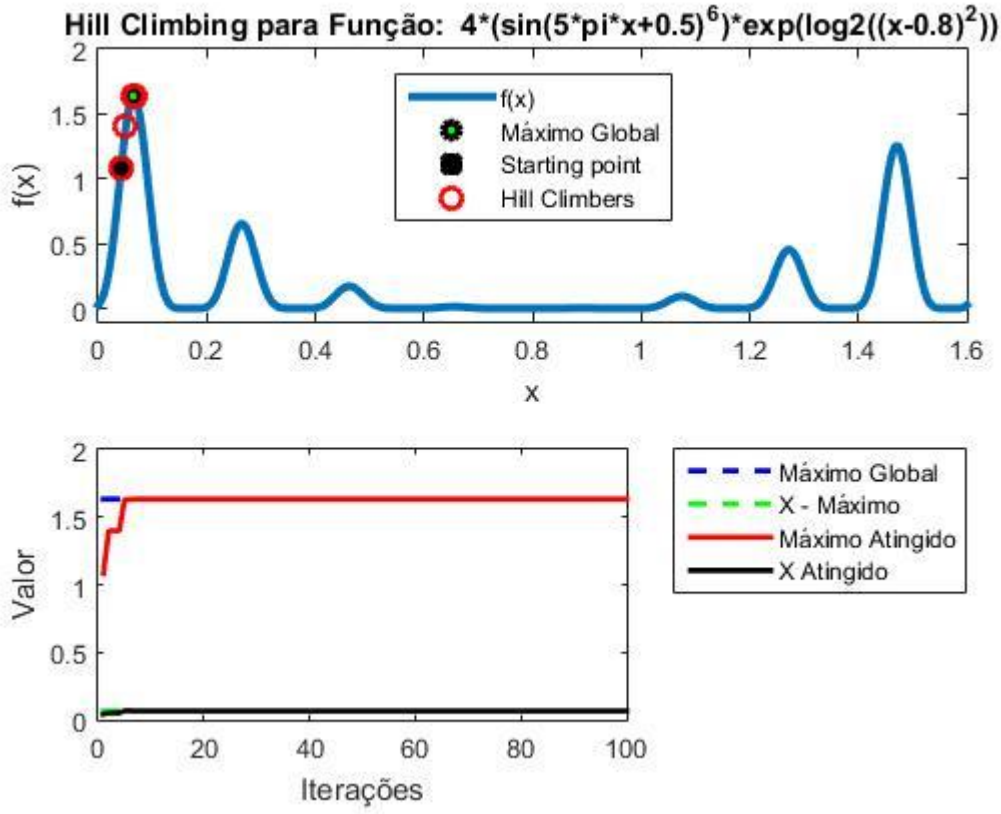

Figura 2: Hill Climbing encontrou o máximo global

Na figura acima verifica-se que o algoritmo conseguiu chegar à melhor solução, alcançando o máximo global. Neste caso, o ponto inicialmente gerado aleatoriamente calhou na colina que contém o máximo global e portanto facilmente subiu, atingindo o máximo global em poucas iterações. Pelos testes que realizei, esta situação, com o algoritmo em causa, acontece uma em cada oito vezes, revelando uma probabilidade de sucesso de 12,5%.

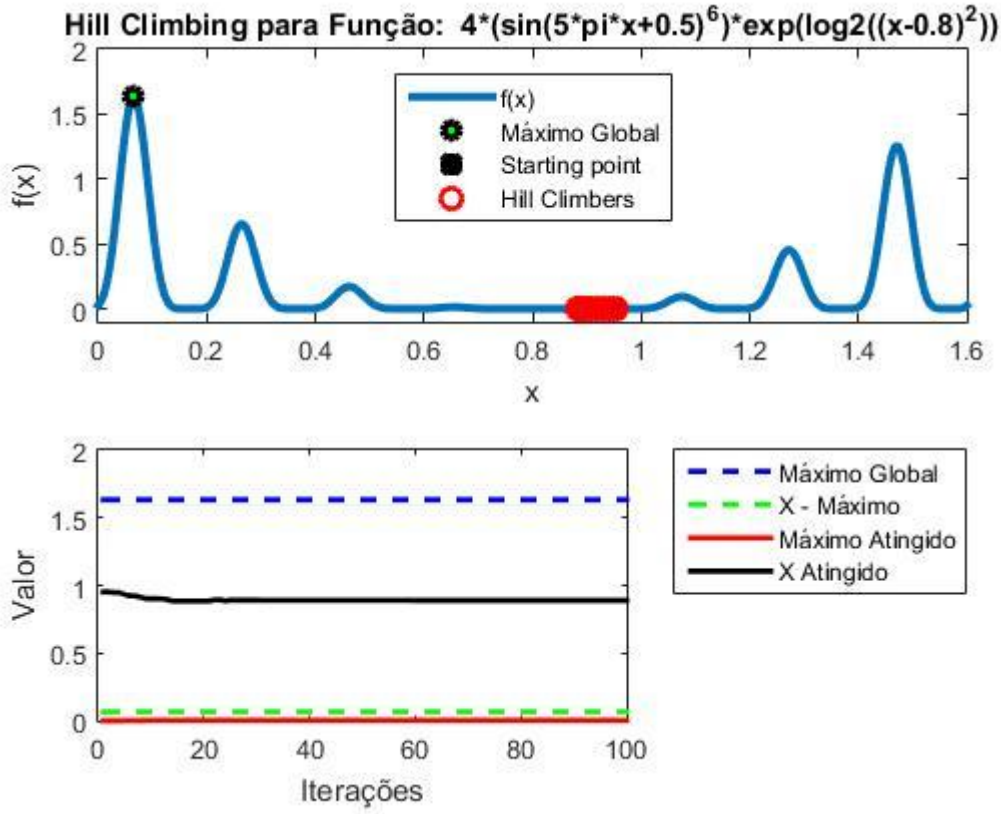

Figura 3: Hill Climbing estagnou numa planície

Uma situação também frequente nesta função, com o algoritmo em causa é a verificada na figura acima, uma estagnação numa planície. Esta situação acontece porque o algoritmo apenas altera o seu valor atual para valores melhores e como o seu raio de vizinhança contém valores equivalentes ao atual, o algoritmo não vai obter movimento ou apenas movimentos repetitivos. Aumentar o raio de vizinhança também não seria solução, já explicado acima.

3.2.*Multiple Restart Hill Climbing*

Para o algoritmo *Multiple Restart Hill Climbing* utilizei raio=1.6/100, iterações=100 e critério de estagnação para saltar quando a média dos últimos 5 valores for igual ao valor atual, de seguida apresentam-se os diferentes resultados que são obtidos:

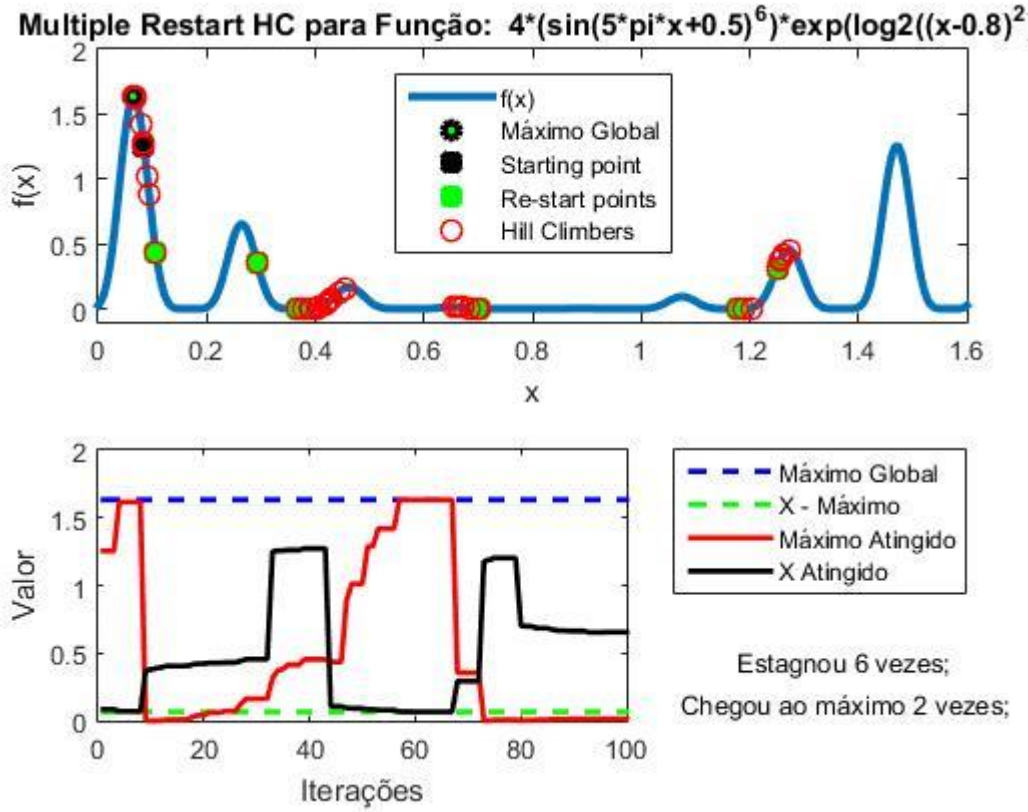

Figura 4: Multiple Restart Hill Climbing encontrou o máximo global

Na figura acima verifica-se que o *Multiple Restart Hill Climbing* conseguiu chegar à solução, alcançando o máximo global. No entanto, podemos verificar que deparou-se com os mesmos problemas do *Basic Hill Climbing* referidos anteriormente, a pesquisa ficou estagnada várias vezes, tanto em planícies como em máximos locais. Estas limitações são combatidas pelo método utilizando reinicialização aleatória de um novo ponto quando existe estagnação, que neste caso é após 5 pontos sucessivos com o mesmo valor. Esta variante ajuda a tornar o método Hill Climbing mais eficiente, uma vez que a probabilidade de encontrar o máximo global é maior, sendo assim uma boa opção para combater o problema dos máximos locais.

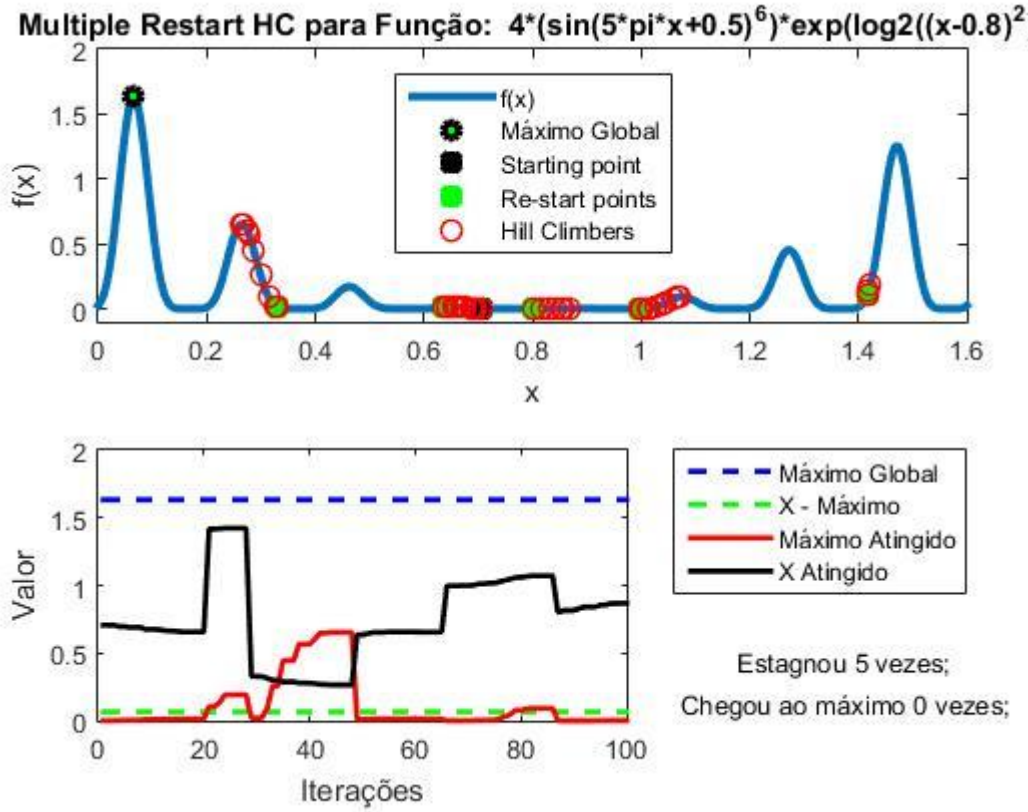

Figura 5: Multiple Restart Hill Climbing falhou em encontrar o máximo global

No entanto, o método também falha algumas vezes, como verificado na figura acima, neste caso o máximo global não foi encontrado. Na figura, no canto inferior direito aparece o número de vezes que o a pesquisa estagnou e o número de vezes que conseguiu chegar ao máximo global, que neste caso foram 0. Em nenhuma das vezes que um ponto foi gerado aleatoriamente, esse ponto calhou perto da colina com o máximo global e portanto a pesquisa não o conseguiu detetar em 100 iterações. Provavelmente, com mais algumas iterações, o máximo global seria encontrado.

3.3.*Simulated Annealing* -Maximização

Para o algoritmo *Simulated Annealing* utilizei raio=1.6/10, iterações=100 com 3 repetições por cada iteração, temperatura inicial=10 e decaimento da temperatura igual a 0.93, de seguida apresentam-se os diferentes resultados que são obtidos:

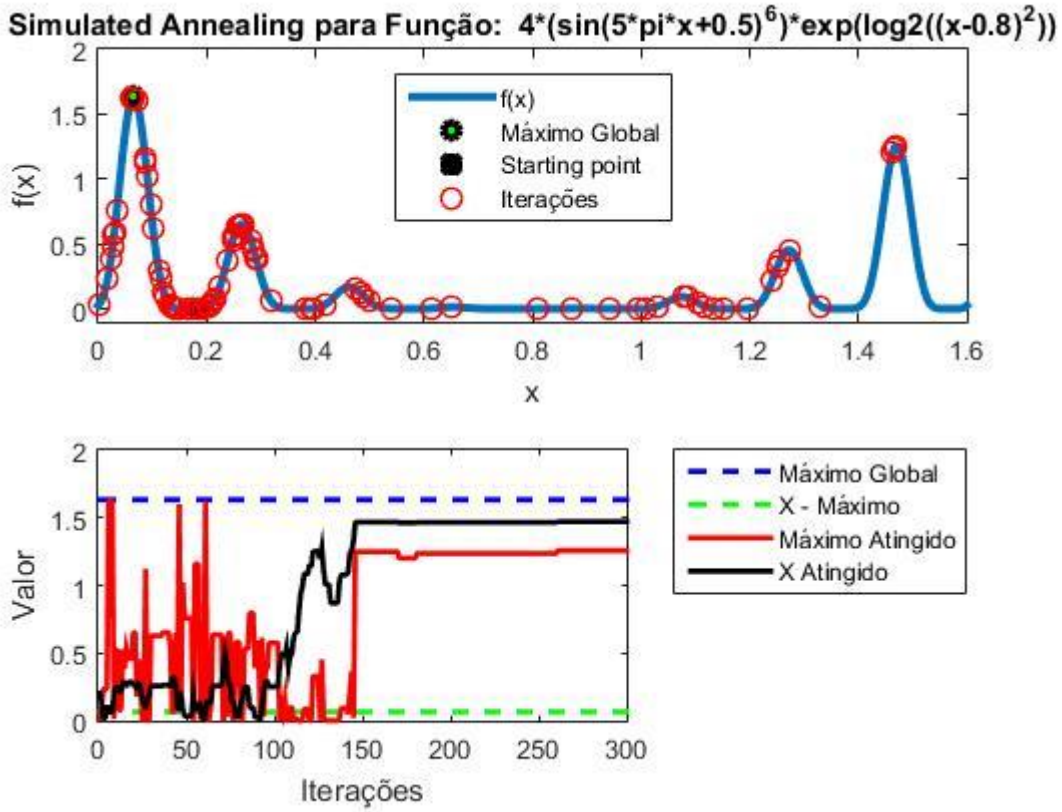

Figura 6: Simulated Annealing encontrou o máximo global

Na figura acima é verificado o sucesso da pesquisa do *Simulated Annealing*, este método, como já explicado, não aceita apenas soluções que envolvem uma melhoria em relação ao ponto atual e portanto funciona melhor que o método *Hill Climbing*, isto porque, por vezes, exemplificando este caso, é necessário descer de uma colina para poder subir outra, e essa poderá conter um máximo maior que o anterior. Neste método não existem estagnações porque estaticamente como existe uma probabilidade de aceitar soluções não boas, ele acabará por sair da zona onde possivelmente estagnaria.

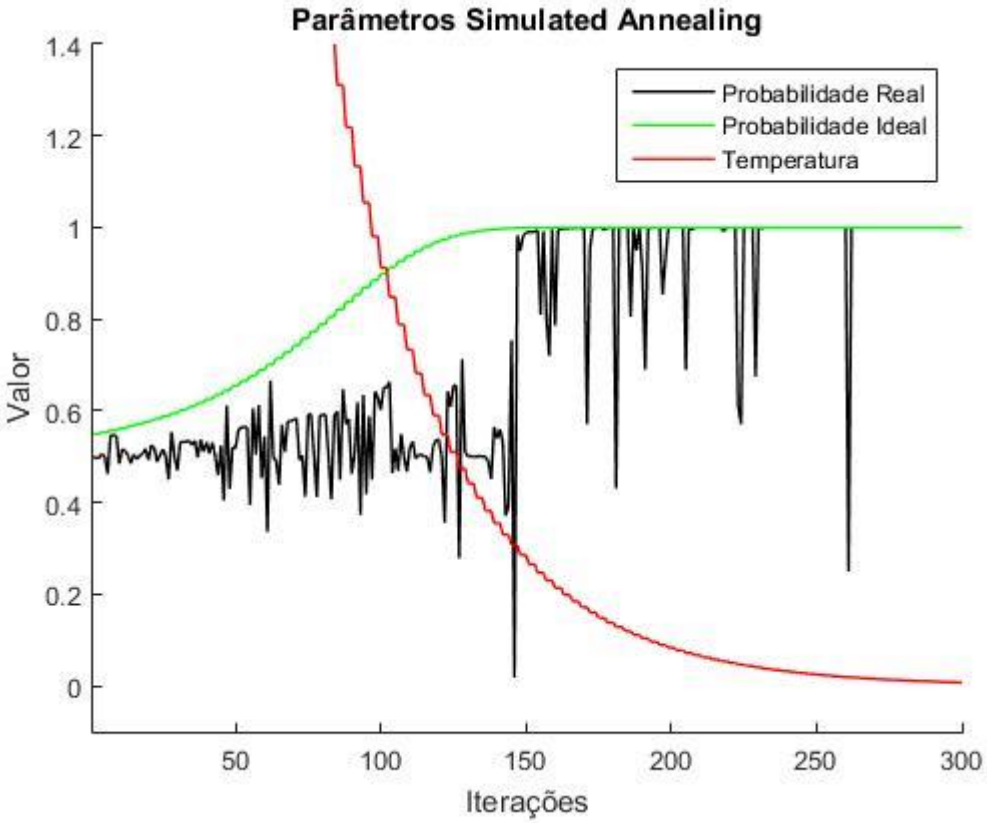

Figura 7: Parametros vs Iterações correspondente à figura anterior

Na figura acima é apresentado o gráfico que envolve os parâmetros Probabilidade Real, Probabilidade Ideal e Temperatura com o número de iterações. A temperatura decaí ao longo das iterações com um fator de 0.93, enquanto que a probabilidade real se aproxima da probabilidade ideal, falhando alguns pontos pelo facto de o delta não ser constante, ou seja os pontos obtidos não estarão sempre à mesma distancia do anterior e portanto influenciará o delta e consequentemente o formula da probabilidade.

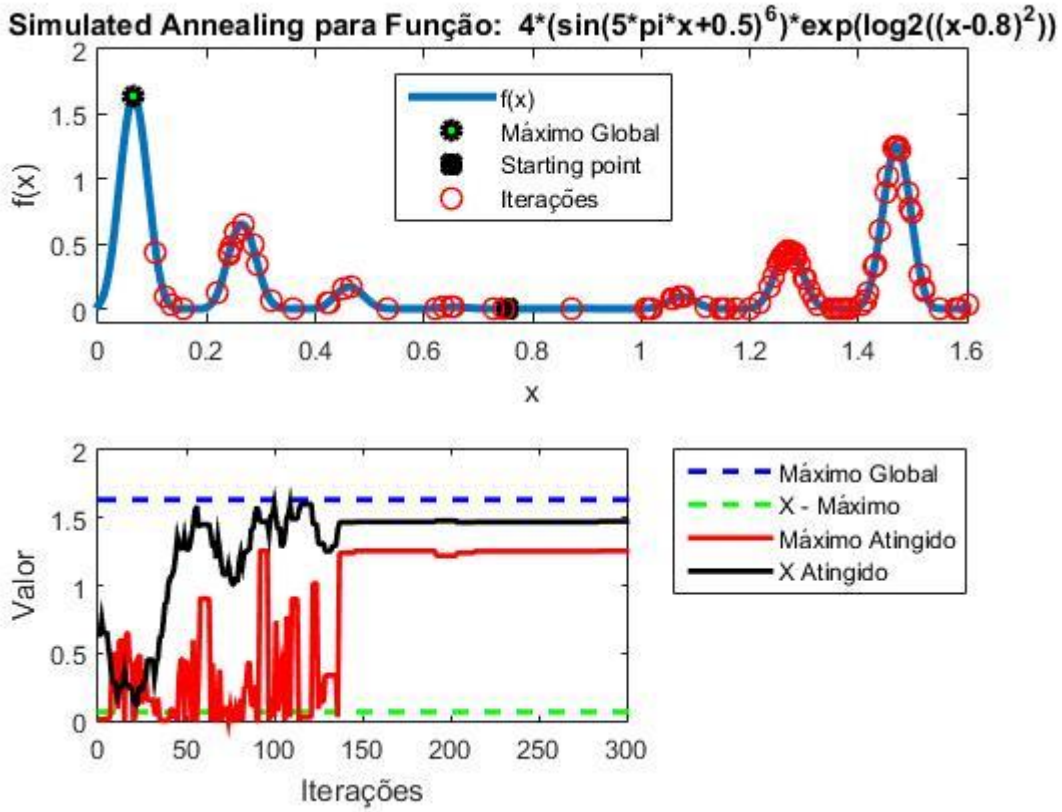

Figura 8: Simulated Annealing falhou em encontrar o máximo global

Na figura acima verifica-se um caso de falhanço do algoritmo Simulated Annealing. No entanto, é de destacar que não ficou preso em máximos locais e ao analisarmos as iterações, podemos observar que o algoritmo conseguiu ultrapassar algumas das limitações do algoritmo *Hill Climbing*, nomeadamente, ficar preso em máximos locais e planícies. Como podemos observar no gráfico das iterações, este método vai "saltando" de um lado para o outro, sendo influenciado por uma probabilidade, ou seja, se o custo de mover-se para a solução vizinha for aceitável, existe uma probabilidade de passar para a nova solução. Enquanto a variável temperatura obter um valor alto, a probabilidade de tomar decisões menos aceitáveis é elevada. No entanto, à medida que a temperatura vai reduzindo, esta probabilidade é reduzida, fazendo com que este método se torne mais determinístico.

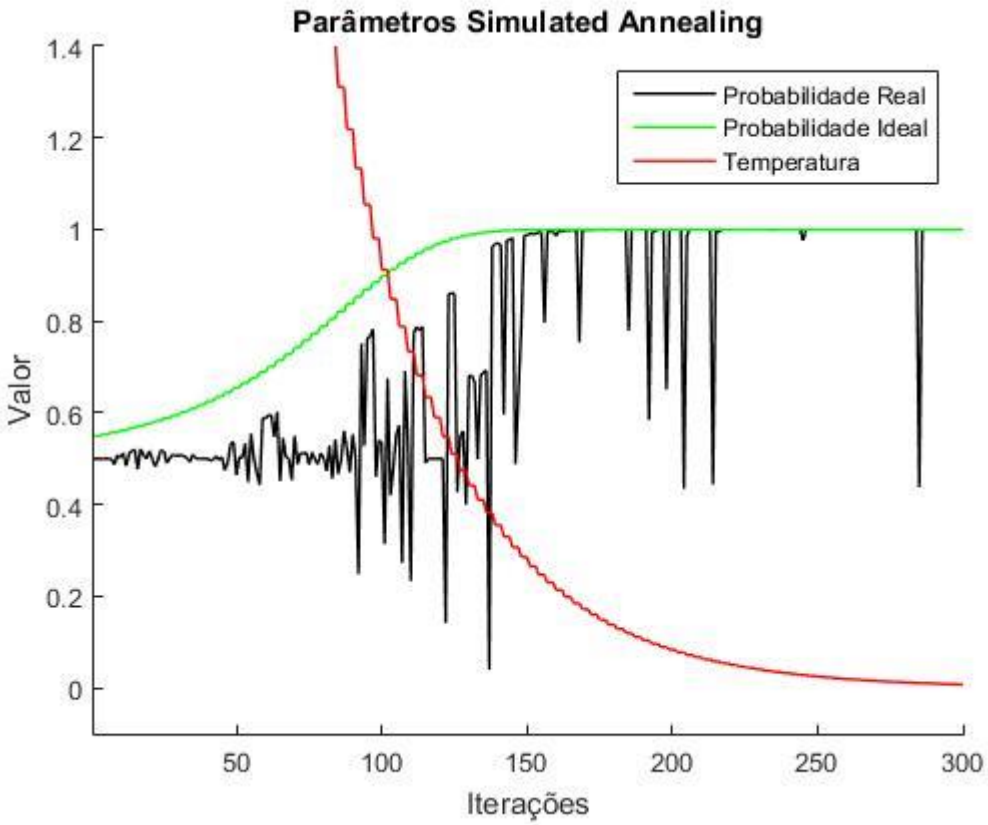

Figura 9: Parâmetros vs Iterações correspondente à figura anterior

Na figura acima é apresentado outro gráfico parâmetros vs iterações, que é semelhante ao anterior gráfico do mesmo género e portanto é explicado da mesma forma.

3.4.*Simulated Annealing* em função 3D - Minimização

Para o algoritmo *Simulated Annealing em função 3D* utilizei raio=0.1, iterações=100 com 10 repetições por cada iteração, temperatura inicial=50 e decaimento da temperatura igual a 0.93. De salientar que os ideais do Simulated Anneling acima explicados para o problema de maximização para a função 2D são os mesmos para este problema. De seguida apresentam-se os diferentes resultados que são obtidos:

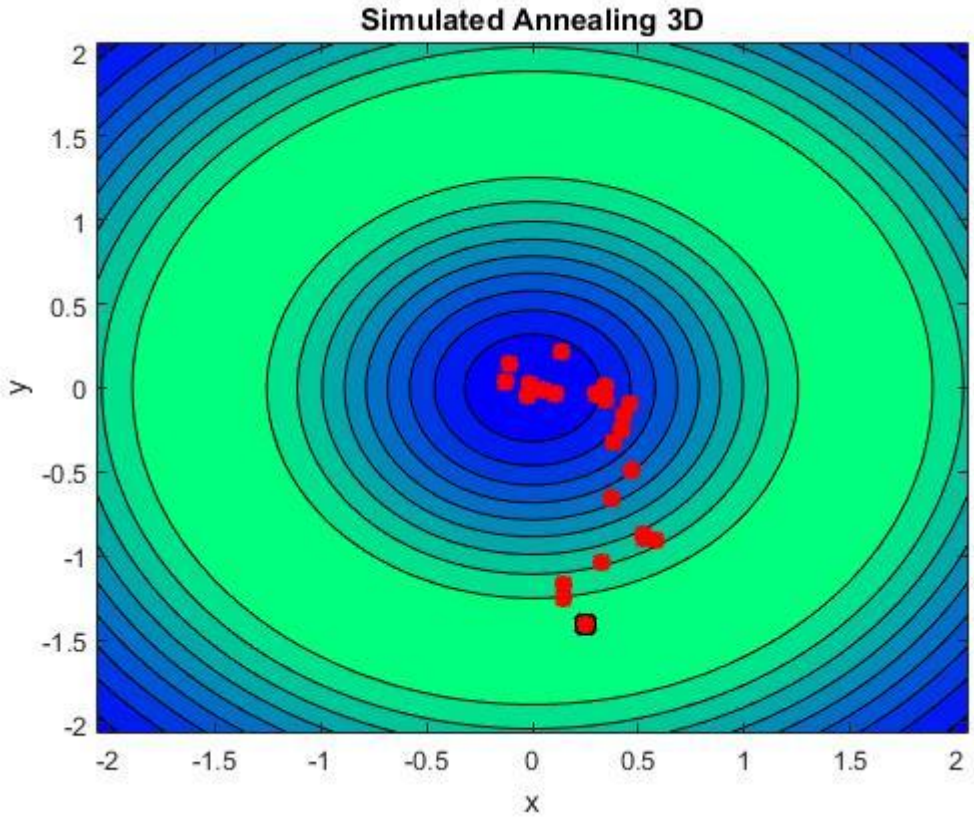

Figura 10: Simulated Annealing encontrou o máximo global

Na figura acima é apresentado o mesmo algoritmo, *Simulated Annealing*, mas para uma função tridimensional. O algoritmo é processado exatamente da mesma forma, envolvendo apenas detalhes por se ter que lidar com outra dimensão. Neste caso verifica-se que houve sucesso, o primeiro ponto gerado aleatoriamente calhou mais perto do mínimo global do que de um local e por isso atingiu a solução ideal.

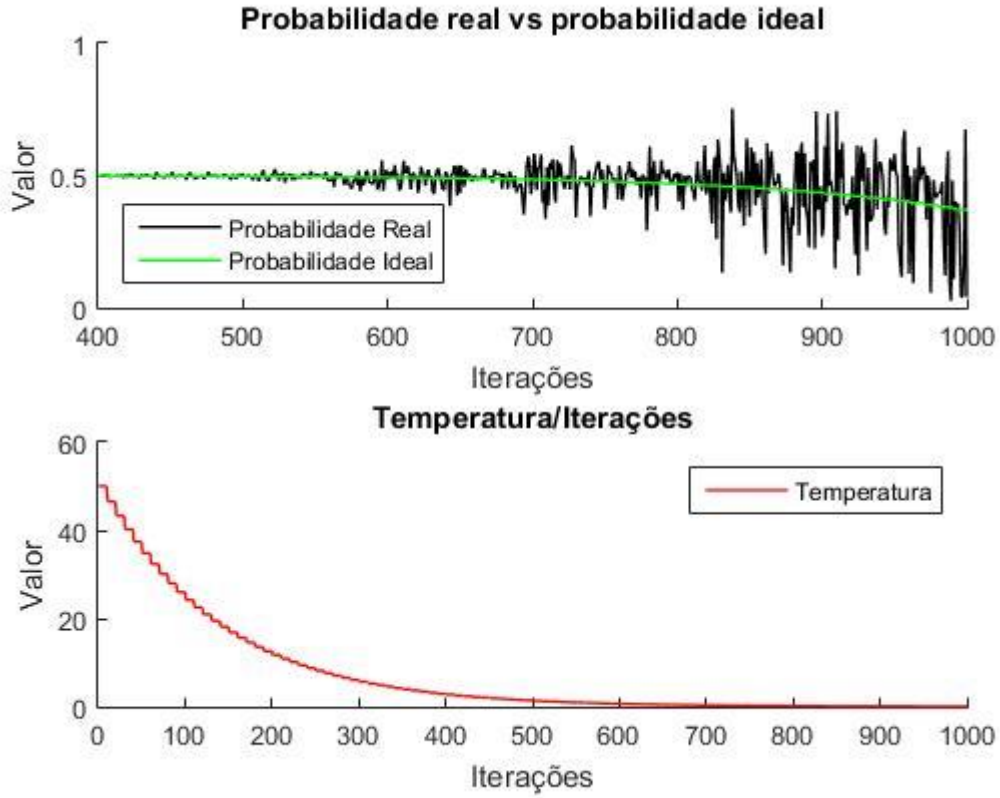

Figura 11: Parâmetros vs Iterações correspondente à figura anterior

Na figura acima é apresentado os gráficos que envolvem os parâmetros Probabilidade Real, Probabilidade Ideal e Temperatura com o número de iterações. A temperatura decaí ao longo das iterações com um fator de 0.93, enquanto que a probabilidade real se aproxima da probabilidade ideal, falhando alguns pontos pelo facto de o delta não ser constante, ou seja os pontos obtidos não estarão sempre à mesma distancia do anterior e portanto influenciará o delta e consequentemente o formula da probabilidade.

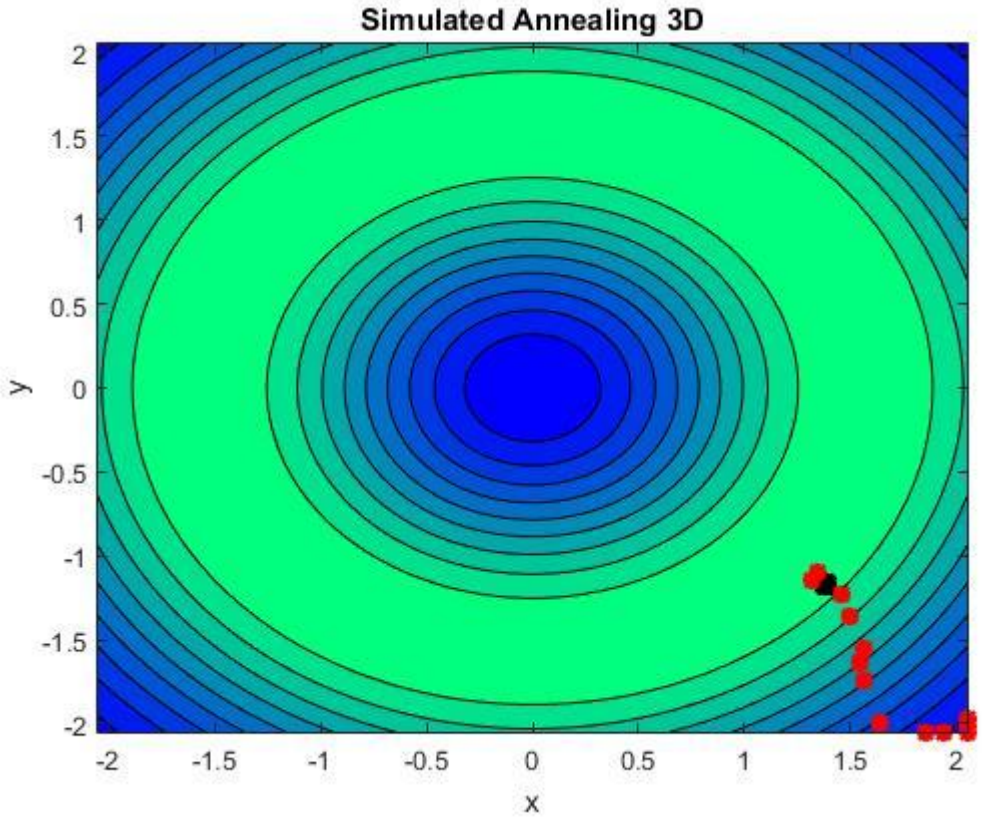

Figura 12: Simulated annealing falhou em encontrar o máximo global

Verifica-se acima um caso de insucesso com o algoritmo *Simulated Annealing*, o primeiro ponto foi gerado perto de um mínimo local, e o algoritmo não conseguiu aceitar outras soluções mais desfavoráveis que poderiam originar a obtenção de uma solução mais favorável e caiu na solução errada, dando como solução um mínimo local.

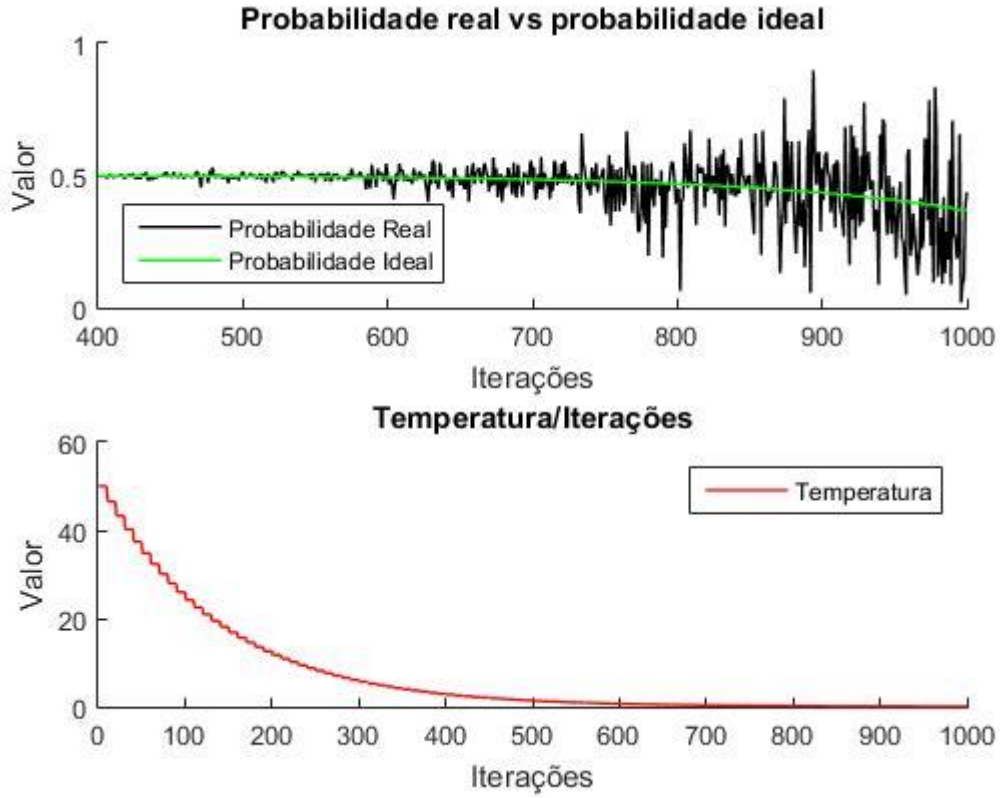

Figura 13: Parâmetros vs Iterações correspondente à figura anterior

Na figura acima é apresentado os gráficos parâmetros vs iterações, semelhantes aos anteriores gráficos e são explicados da mesma forma.

4. Conclusão

As conclusões a que chego é que adquiri conhecimentos e desenvolvi de competências relativas aos algoritmos *Hill Cimbing* e *Simulated Annealing*, assim como algumas variantes, além de outras competências em outros assuntos e algoritmos relacionados, originadas pela pesquisa que fiz. Também consegui melhorar a minha performance relativamente à utilização do programa Matlab.

Relativamente aos algoritmos estudados, conclui-o que o algoritmo Hill Climbing é facilmente enganado caindo numa armadilha de máximo/mínimo local ou estagnação, portanto não deve ser utilizado para resolver problemas em que o espaço de pesquisa é muito instável, contendo vários máximos e mínimos e quando se pretende chegar à solução ideal.

O algoritmo *Simulated Annealing* é um bom método com boa taxa de sucesso, tendo vantagens como não ficar preso em máximos locais, ou seja, por vezes utiliza a probabilidade e a quantidade certa de aleatoriedade ao invés da lógica para evitar soluções sub-ótimas.

Entre os dois algoritmos, o Simulated Annealing revela-se mais eficaz mas também se encontra longe de ser perfeito, falhando também algumas vezes.

5. Bibliografia

Evaristo, Métodos de Otimização. Disponível em: <http://www2.peq.coppe.ufrj.br/Pessoal/Professores/Evaristo/CO897_2014/M%E9todos%20N%E3o%20Determin%EDsticos/aula1.pdf>. Acesso em 15 de Outubro de 2015.

Cohen, W., Greiner, R., Schuurmans, D. (1994). Probabilistic Hill-Climbing. Computational Learning Theory and Natural Learning Systems, Vol. II. Edited by Hanson, S., Petsche, T., Rivest R., Kearns M. MITCogNet, Boston:1994.

Russell, Stuart J.; Norvig, Peter (2003), Artificial Intelligence: A Modern Approach (2nd ed.), Upper Saddle River, New Jersey: Prentice Hall, pp. 111–114, ISBN 0-13-790395-2

Breve, F. (2007). Computação Evolutiva – Parte I. Disponível em: http://www.fabriciobreve.com/material/compavancada/CA-Aula2-ComputacaoEvolutiva.pdf. Acesso em 17 de Outubro de 2016

Porto Editora. Disponível em: <https://www.portaleducacao.com.br/educacao/artigos/33378/o-que-e-um-algoritmo-de-otimizacao> . Acesso em 17 de Outubro de 2016

V.Vassilev, A.Prahova: "The Use of Simulated Annealing in the Control of Flexible Manufacturing Systems", International Journal INFORMATION THEORIES & APPLICATIONS, VOLUME 6/1999

6. Código anexado

```
%Primeiro trabalho Inteligência Artificial
%Ponto 3 - Hill Climbing Normal
%Ponto 3 pede:
%       - 100 iterações por teste
%       - Raio de pesquisa 0.01

clc %limpar qualquer input ou output do minitor
clear %limpar variáveis
close all %apaga figuras anteriores

limits = [0,1.6] %limites da pesquisa
range= limits(2) - limits(1) %espaço de pesquisa

figure(1)%cria uma nova figura
set(0, 'defaultlinelinewidth', 3); %definir propriedades do gráfico
subplot(2,1,1); %divide a figura em graficos
ezplot('4*(sin(5*pi*x+0.5)^6)*exp(log2((x-0.8)^2))', [0,1.6]); %desenha a
função
axis([0,1.6,-0.1,2]); %delimitar os eixos
title('Hill Climbing para Função:  4*(sin(5*pi*x+0.5)^6)*exp(log2((x-
0.8)^2))')
xlabel('x');
ylabel('f(x)');
hold on %congelar gráficos existentes para os novos plots não apagarem os
anteriores

x_radius=range/100; %raio de pesquisa

x_current=limits(1) + rand*range; %primeiro valor de x
                                  %rand devolve um valor entre 0 e 1
                                  %logo x_current = 0 + [0,1]*1.6
                                  %logo x_current = [0,1.6] <- pretendido

%chama função objetivo
F_current=4*(sin(5*pi*x_current+0.5)^6)*exp(log2((x_current-0.8)^2));

%a solução aproximada é: [0.066,1.6332]
%plot da solução
plot(0.066,1.6332, 'ok', 'markersize',6,'markerfacecolor', 'g');

%plot primeiro ponto
plot(x_current, F_current, '*k', 'markersize', 8);

%solução inicial
xplot(1)=x_current;
Fplot(1)=F_current;

It=100;

for j=2:It,
    x_new= (x_current-x_radius)+rand*2*x_radius   %calculo do novo X
    %subtrai raio e adiciona de 0 a 2*raio (dependendo do valor do rand)
    %logo x_new varia de [x_current-raio, x_current+raio]
```

```
    %limitar procura no espaço de pesquisa
    if x_new < limits(1)
        x_new = limits(1);
    end
    if x_new > limits(2)
        x_new = limits(2);
    end

    %calculo da imagem de x
    F_new=4*(sin(5*pi*x_new+0.5)^6)*exp(log2((x_new-0.8)^2));

    %ver se a nova imagem é melhor que a antigo
    if(F_new > F_current)
        x_current = x_new;
        F_current = F_new;
    end

    %guardar valores em vetores para depois fazer plot
    xplot(j)=x_current;
    Fplot(j)=F_current;
end

%plot dos hill climbers
plot(xplot,Fplot,'or', 'Linewidth', 2, 'markersize', 8);

legend('f(x)', 'Máximo Global', 'Starting point','Hill Climbers', 'Location',
'North');
hold off

it = 1:1:It;

%x e f(x) do máximo global
F_ideal = 1.6332*ones(1,It);
x_ideal = 0.066*ones(1,It);

subplot(2,1,2);
axis([1,It,0,2]);%delimitação dos eixos
plot(it, F_ideal, 'b--', 'linewidth', 2);
hold on
plot(it, x_ideal,'g--','linewidth', 2)
plot(it, Fplot, 'r-', 'linewidth', 2);
plot(it, xplot,'k-', 'linewidth', 2);
legend('Máximo Global', 'X - Máximo', 'Máximo Atingido', 'X Atingido',
'Location', 'NorthEastOutside');
xlabel('Iterações');
ylabel('Valor');
hold off
```

```
%Primeiro trabalho Inteligência Artificial
%Ponto 4 - Multiple Re-start Hill Climbing
%Ponto 4 pede:
%       - considerar estagnação quando a solução
%       corrente não se altera em 5 iterações

clc %limpar qualquer input ou output do minitor
clear %limpar variáveis
close all %apaga figuras anteriores

limits = [0,1.6] %limites da pesquisa
range= limits(2) - limits(1) %espaço de pesquisa

figure(1)%cria uma nova figura
set(0, 'defaultlinelinewidth', 3); %definir propriedades do gráfico
subplot(2,1,1); %divide a figura em graficos
ezplot('4*(sin(5*pi*x+0.5)^6)*exp(log2((x-0.8)^2))', [0,1.6]); %desenha a
função
axis([0,1.6,-0.1,2]); %delimitar os eixos
title('Multiple Restart HC para Função:  4*(sin(5*pi*x+0.5)^6)*exp(log2((x-
0.8)^2))')
xlabel('x');
ylabel('f(x)');
hold on %congelar gráficos existentes para os novos plots não apagarem os
anteriores

x_radius=range/100; %raio de pesquisa

x_current=limits(1) + rand*range; %primeiro valor de x
                                  %rand devolve um valor entre 0 e 1
                                  %logo x_current = 0 + [0,1]*1.6
                                  %logo x_current = [0,1.6] <- pretendido

%chama função objetivo
F_current=4*(sin(5*pi*x_current+0.5)^6)*exp(log2((x_current-0.8)^2));

%a solução aproximada é: [0.066,1.6332]
%plot da solução
plot(0.066,1.6332, 'ok', 'markersize',6,'markerfacecolor', 'g');

%plot primeiro ponto
plot(x_current, F_current, '*k', 'markersize', 8);

%solução inicial
xplot(1)=x_current;
Fplot(1)=F_current;

xn(1:5)=0; %vetor para depois guardar os ultimos 5
FlagStag=0; %aumenta quando os ultimos 5 são iguais
nstag=1; %quando a FlagStag for igual a nstag é considerada estagnação
jumps=0; %para contar numero de saltos
nmax=0; %para contar quantas vezes atinge o máximo

It=100;
for j=2:It,
    x_new = (x_current-x_radius)+rand*2*x_radius;   %calculo do novo X
    %subtrai raio e adiciona de 0 a 2*raio (dependendo do valor do rand)
    %logo x_new varia de [x_current-raio, x_current+raio]
```

```
    if(FlagStag==nstag)%se o contador de mudanças em 5 gor 0 salta
        x_new = limits(1) + rand*range; %calculo de novo x
        jumps=jumps+1; %incrementa saltos
        disp('Jump');  %para fazer display de Jump
    end
    %limitar procura no espaço de pesquisa
    if x_new < limits(1)
        x_new = limits(1);
    end
    if x_new > limits(2)
        x_new = limits(2);
    end

    %calculo da imagem de x
    F_new=4*(sin(5*pi*x_new+0.5)^6)*exp(log2((x_new-0.8)^2));

    if((nstag>FlagStag) && (F_new > F_current))  %ver se a nova imagem é
melhor que a antigo
        x_current = x_new;
        F_current = F_new;
    elseif(FlagStag == nstag)   %caso seja um novo x tem de mudar o x_current
também
        x_current = x_new;
        F_current = F_new;
        %plot da nova solução inicial
        %plot(x_current,F_current, '*k');
        new_start_x(jumps)=x_new;  %para saber para onde saltou
        new_start_F(jumps)=F_new;  %para saber para que imagens saltou
        FlagStag=0;
    end

    %guardar valores em vetores para depois fazer plot
    xplot(j)=x_current;
    Fplot(j)=F_current;

    %calculo da media
    xn(5)=xn(4);
    xn(4)=xn(3);
    xn(3)=xn(2);
    xn(2)=xn(1);
    xn(1)=x_current;
    avg=(xn(5)+xn(4)+xn(3)+xn(2)+xn(1))/5;

    if (avg == x_current)%se a media for igual ao valor todos os 5 valores
devem ser iguais
        FlagStag=FlagStag+1;
        xn(1:5)=0; %reset ao vetor
        if(Fplot(j)>1.6)%a solução é aproximadamente acima de 1.6, logo acima
de 1.6 considero que chegou ao máximo
            nmax=nmax+1; %pretendo saber quantas vezes atingiu o máximo
        end
        disp('Pesquisa estagnada');
    end
end
if(jumps>0)%fazer plot dos reinicios se houve saltos
    plot(new_start_x, new_start_F, '*g', 'markersize', 8);
end
```

```
%plot dos hill climbers
plot(xplot,Fplot,'or', 'Linewidth', 1, 'markersize', 8);

legend('f(x)', 'Máximo Global', 'Starting point','Re-start points','Hill
Climbers', 'Location', 'North');
hold off

it = 1:1:It;

%x e f(x) do máximo global
F_ideal = 1.6332*ones(1,It);
x_ideal = 0.066*ones(1,It);

subplot(2,1,2);
axis([1,It,0,2]);%delimitação dos eixos
plot(it, F_ideal, 'b--', 'linewidth', 2);
hold on
plot(it, x_ideal,'g--','linewidth', 2)
plot(it, Fplot, 'r-', 'linewidth', 2);
plot(it, xplot,'k-', 'linewidth', 2);
legend('Máximo Global', 'X - Máximo', 'Máximo Atingido', 'X Atingido',
'Location', 'NorthEastOutside');
xlabel('Iterações');
ylabel('Valor');

axes('Position',[0 0 1 1],'Visible', 'off'); %grafico invisivel para escrever
apenas
string = int2str(jumps);  %passar para string
string = strcat('Estagnou',{' '}, string, ' vezes;'); %juntar strings
text(0.7,0.2,string); %escrever

string = int2str(nmax);%passar para string
string = strcat('Chegou ao máximo',{' '}, string, ' vezes;'); %juntar strings
text(0.65,0.15,string); %escrever
hold off
```

```
%Primeiro trabalho Inteligência Artificial
%Ponto 5 - Simulated Annealing

clc %limpar qualquer input ou output do minitor
clear %limpar variáveis
close all %apaga figuras anteriores

limits = [0,1.6] %limites da pesquisa
range= limits(2) - limits(1) %espaço de pesquisa

figure(1) %cria uma nova figura
set(0, 'defaultlinelinewidth', 3);
subplot(2,1,1); %divide a figura em graficos
ezplot('4*(sin(5*pi*x+0.5)^6)*exp(log2((x-0.8)^2))', [0,1.6])
axis([0,1.6,-0.1,2]); %delimitar os eixos
title('Simulated Annealing para Função:  4*(sin(5*pi*x+0.5)^6)*exp(log2((x-
0.8)^2))')
xlabel('x');
ylabel('f(x)');
hold on %congelar gráficos existentes para os novos plots não apagarem os
anteriores

x_radius=range/10; %raio de pesquisa

x_current=limits(1) + rand*range; %primeiro valor de x
                                  %rand devolve um valor entre 0 e 1
                                  %logo x_current = 0 + [0,1]*1.6
                                  %logo x_current = [0,1.6] <- pretendido

%chama função objetivo
F_current=4*(sin(5*pi*x_current+0.5)^6)*exp(log2((x_current-0.8)^2));

%a solução aproximada é: [0.066,1.6332]
%plot da solução
plot(0.066,1.6332, 'ok', 'markersize',6,'markerfacecolor', 'g');

%plot primeiro ponto
plot(x_current, F_current, '*k', 'markersize', 8);

%solução inicial
xplot(1)=x_current;
Fplot(1)=F_current;

T=10; %temperatura inicial
Alpha=0.93; %para diminuir a temperatura
It=100;
Rep_It=3;
nmrIt=1;
DeltaTeste=-2; %delta ideal
for n=1:It,
    for i=1:Rep_It,
        x_new = (x_current-x_radius)+rand*2*x_radius;   %calculo do novo X
        %subtrai raio e adiciona de 0 a 2*raio (dependendo do valor do rand)
        %logo x_new varia de [x_current-raio, x_current+raio]

        %limitar procura no espaço de pesquisa
        if x_new < limits(1)
            x_new = limits(1);
```

```
        end
        if x_new > limits(2)
            x_new = limits(2);
        end

        %calculo da imagem de x
        F_new=4*(sin(5*pi*x_new+0.5)^6)*exp(log2((x_new-0.8)^2));

        %calculo do delta
        Delta=F_new-F_current;
        %calculo da probabilidade
        P=1/(1+exp(Delta/T)); %formula da probabilidade
        P_Real(nmrIt)=P; %para fazer plot da probabilidade
        % Se a nova solução for melhor que a atual
        if(Delta>0)
            x_current=x_new;
            F_current=F_new;
        elseif rand(1)> P
                x_current=x_new;
                F_current=F_new;
        end
        %Guardar nos vetores as coordenadas, respetivamente, do melhor valor
        xplot(nmrIt)=x_current; % melhor em cada nivel (1 por)
        Fplot(nmrIt)=F_current;
        P_Ideal(nmrIt)=1/(1+exp(DeltaTeste/T));
        Vect_T(nmrIt)=T;
        nmrIt=nmrIt+1;
    end
    T=Alpha*T;
end

%plot das iterações
plot(xplot,Fplot,'or', 'Linewidth', 1, 'markersize', 8);

legend('f(x)', 'Máximo Global','Starting point', 'Iterações', 'Location',
'North');
hold off

nmrIt=nmrIt-1;
it = 1:1:nmrIt;

%x e f(x) do máximo global
F_ideal = 1.6332*ones(1,nmrIt);
x_ideal = 0.066*ones(1,nmrIt);

subplot(2,1,2)
axis([1,nmrIt,0,3]); %delimitação dos eixos
plot(it, F_ideal, 'b--', 'linewidth', 2);
hold on
plot(it, x_ideal,'g--','linewidth', 2)
plot(it, Fplot, 'r-', 'linewidth', 2);
plot(it, xplot,'k-', 'linewidth', 2);
legend('Máximo Global', 'X - Máximo', 'Máximo Atingido', 'X Atingido',
'Location', 'NorthEastOutside');
xlabel('Iterações');
ylabel('Valor');

figure(2)
```

```
hold on
plot(it, P_Real, 'k-', 'linewidth', 1);
plot(it, P_Ideal, 'g-', 'linewidth', 1);
plot(it, Vect_T, 'r-', 'linewidth', 1);
legend('Probabilidade Real', 'Probabilidade Ideal','Temperatura', 'Location',
'NorthEast');
axis([1, nmrIt,-0.1,1.4]);
title('Parâmetros Simulated Annealing')
xlabel('Iterações');
ylabel('Valor');
hold off
```

```
%Primeiro trabalho Inteligência Artificial
%Ponto 6 - Simulated Annealing função 3D

clc %limpar qualquer input ou output do minitor
clear %limpar variáveis
close all %apaga figuras anteriores

limits = [-2.048, 2.048;-2.048, 2.048] %limites da pesquisa
range= limits(2) - limits(1) %espaço de pesquisa

figure(1) %cria uma nova figura
colormap winter
ezcontourf('0.5+((sin(sqrt(x^2+y^2)))^2-0.5)/((1+0.001*(x^2+y^2))^2)',[-
2.048,2.048],[-2.048,2.048]);
title('Simulated Annealing 3D');
hold on

Dim=2;
%primeira solução;
jj=1;
while (jj <= Dim) %varia de 1 a 2
    %Shubbert
    VarLBounds(jj)=-2.048;
    VarUBounds(jj)=+2.048;
    x(jj)=VarLBounds(jj)+rand(1)*(VarUBounds(jj)-VarLBounds(jj));
    jj=jj+1;
end
%plot primeira solução
plot(x(1),x(2), '*k', 'markersize', 8);

%chama função objetivo
F_current=0.5+((sin(sqrt(x(1)^2+x(2)^2)))^2-
0.5)/((1+0.001*(x(1)^2+x(2)^2))^2);

MinBest=Inf;
F_verybest=Inf;
x_verybest=x; %Melhor x
%MinBest----Melhor F
if(MinBest>F_current)
    MinBest=F_current;
end

T=50; %Temperatura Inicial = 50
Alpha=0.93; %para diminuir a temperatura
DeltaTeste=0.02;
It=100;
Rep_It=10;
nmrIt=1;
raio=0.1;
for n=1:It,
    for i=1:Rep_It,
        x_new=x+raio*randn(1,2);

        if x_new(1) < VarLBounds(1)
            x_new(1) = VarLBounds(1);
        end
        if x_new(2) < VarLBounds(2)
            x_new(2) = VarLBounds(2);
```

```
        end
        if x_new(1) > VarUBounds(1)
            x_new(1) = VarUBounds(1);
        end
        if x_new(2) > VarUBounds(2)
            x_new(2) = VarUBounds(2);
        end

        F_new=0.5+((sin(sqrt(x_new(1)^2+x_new(2)^2)))^2-
0.5)/((1+0.001*(x_new(1)^2+x_new(2)^2))^2);
        Delta=F_new-F_current;
        P=1/(1+exp(Delta/T)); %formula da probabilidade
        P_Real(nmrIt)=P;
        P_Ideal(nmrIt)=1/(1+exp(DeltaTeste/T));
        Vect_T(nmrIt)=T;
        % Se a nova solução for melhor que a atual
        if(Delta<=0)
            x=x_new;
            F_current=F_new;
        elseif rand < P
                x=x_new;
                F_current=F_new;
                %pause(0.1);
        end

        if(F_current < F_verybest)
            F_verybest = F_current;
            x_verybest=x;
        end

        if(F_current < F_verybest)
            F_verybest = F_current;
            x_verybest = x;
        end

        plot(x_verybest(1), x_verybest(2), 'r*',
'Markersize',6,'LineWidth',2);
        nmrIt=nmrIt+1;
    end
    T=Alpha*T;
    disp(['x= ' num2str(x) ' fx= ' num2str(F_current) ' Temp= ' num2str(T)])
end

nmrIt=nmrIt-1;
it = 1:1:nmrIt;

figure(2)
subplot(2,1,1);
hold on
plot(it, P_Real, 'k-', 'linewidth', 1);
plot(it, P_Ideal, 'g-', 'linewidth', 1);
legend('Probabilidade Real', 'Probabilidade Ideal','Location', 'SouthWest');
title('Probabilidade real vs probabilidade ideal')
axis([400, nmrIt,0,1]);
xlabel('Iterações');
ylabel('Valor');
hold off
subplot(2,1,2);
hold on
```

```
plot(it, Vect_T, 'r-', 'linewidth', 1);
legend('Temperatura', 'Location', 'NorthEast');
title('Temperatura/Iterações')
xlabel('Iterações');
ylabel('Valor');
hold off

%figure(3)
%surf(peaks)
```
