Instituto Tecnológico de Aeronáutica - ITA

Divisão de Engenharia Eletrônica

Departamento de Eletrônica Aplicada

Laboratório de CES-12

Pedro Alves de Souza Neto (alvesouza.pedro97@gmail.com)

1: Descrição da implementação

Solução

BB:

O algoritmo escolhido foi o Branch and Bound, este algoritmo de utiliza de recursão para se mover em forma similar ao in-order numa arvore. Explicação do que foi feito no código é que ele recebe o valor que sobra da "mochila" e o index do respectivo item, caso o index não possuir item correspondente ele volta para a chamada recursiva anterior e retorna um "false". Caso o "value" for zero ele retorna "true".

Na parte intermediaria da função, ele chama duas recursões uma com o "value" subtraído pelo peso do item correspondente (indicando que o item foi tomado), o outro com o próprio value que só será chamada se a primeira recursão resultar em falso.No final ele seta "true" no output do item caso este seja tomado.

Ilustração correspondente ao percurso das recursões:

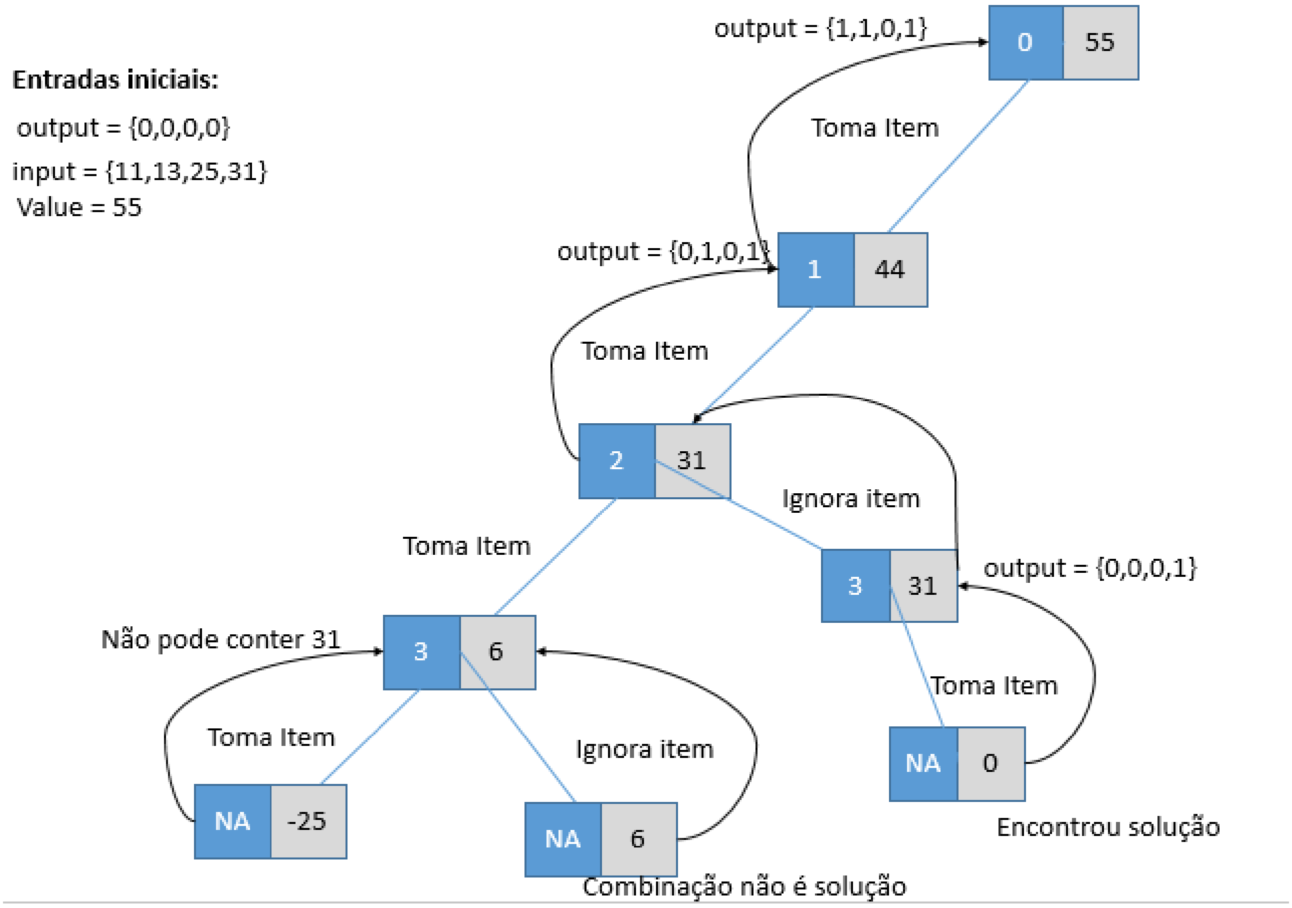

Figura 1.1.Representação do algoritmo.

PD:

O algoritmo do PD recebe o tamanho da mochila representado por “value”, uma lista de item com seus respectivos pesos, e a lista de output que representa a lista que soluciona o problema.

No começo se cria uma tabela com tamanho input.size()X(value+1) , para preencher a tabela com os preços da soma dos itens usados no que no caso deste problema é o peso dos itens.

Para preencher a tabela enquanto o programa está no item “i” ele só poderá usar os valores dos itens [0 1 2...i-1 i], e o de um ponto da tabela tem que ser menor ou igual ao imediatamente em baixo, caso o peso do item for maior que o tamanho da mochila correspondente ao índice, o valor dessa posição será igual ao imediatamente acima, se não existir o valor será zero.

Caso o peso do item seja menor que peso correspondente ao índice, será escolhido o maior valor entre (soma do preço do item e da combinação que ainda cabe na mochila ou o valor logo encima da posição), este padrão continua até completar a tabela ou encontrar uma solução.

Para se determinar quais os itens escolhidos, parte-se do maior tamanho, no último item, caso a posição acima possui um valor igual ao atual, ele irá subir na tabela, reduzindo os índices. Quando a posição acima for menor que a atual, o item correspondente é acionado no output e reduz o peso da mochila correspondente ao peso do item (fazendo ele se mover pela esquerda).

Entradas iniciais:

output = {0,0,0,0}

input = {1,3,5,7}

Value = 10

i\Tamanho	0	1	2	3	4	5	6	7	8	9	10
0(1)	0	1	1	1	1	1	1	1	1	1	1
1(3)	0	1	1	3	4	4	4	4	4	4	4
2(5)	0	1	1	3	4	5	6	6	8	9	9
3(7)	0	1	1	3	4	5	6	7	8	9	10

output = {0,1,0,1}

output = {0,0,0,1}

Figura 2.2.Representação do preenchimento da tabela e do output.

2: Comparação dos resultados

BB

P3:

Devido a sua recursão tomando um caminho similar ao de uma arvore BB possui um tempo exponencial.

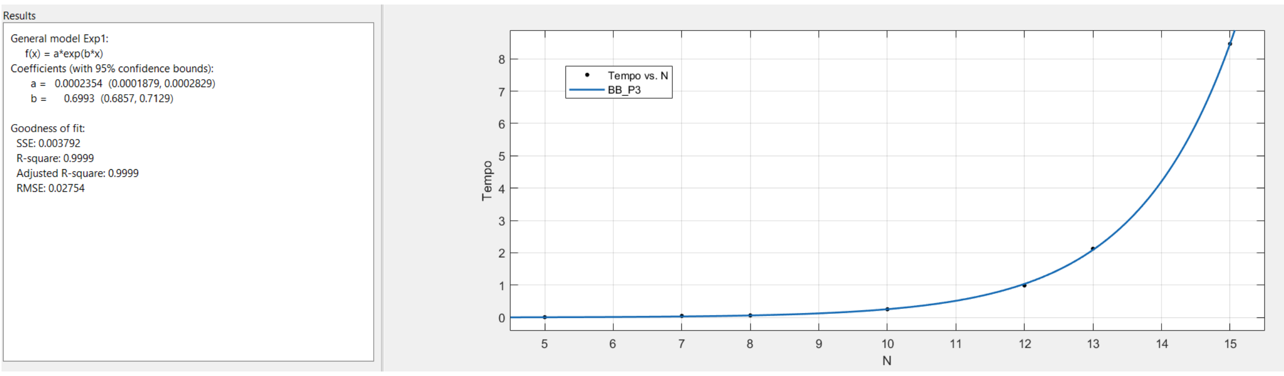

Figura 3.3.Regressão feita pelo MatLab.

P4:

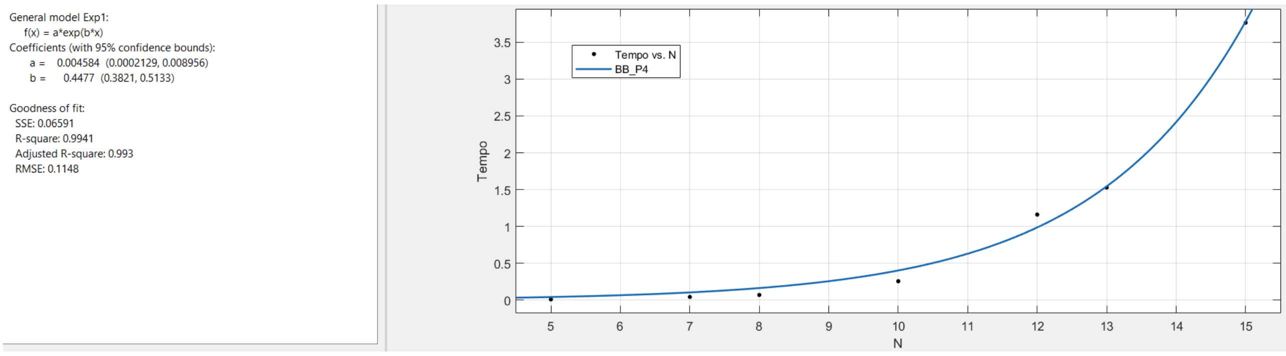

Figura 3.2.Regressão feita pelo MatLab.

P5:

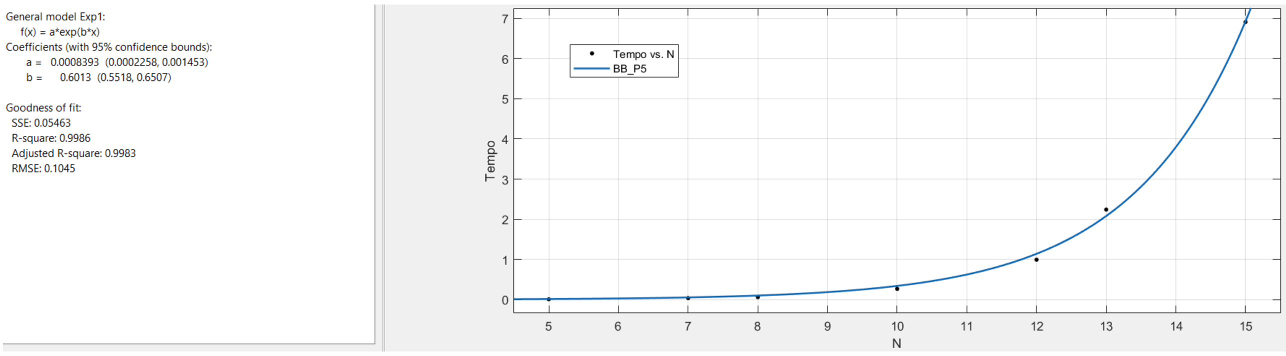

Figura 3.3.Regressão feita pelo MatLab.

AVIS:

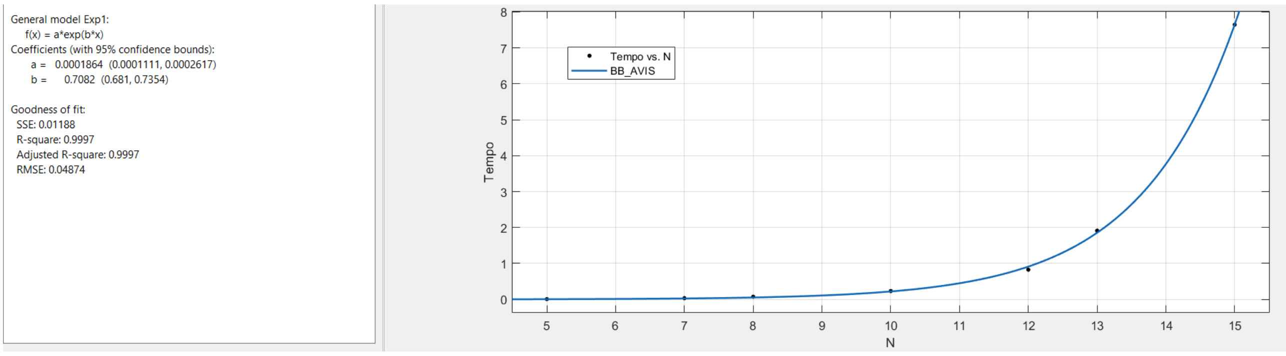

Figura 3.4.Regressão feita pelo MatLab.

EVOD:

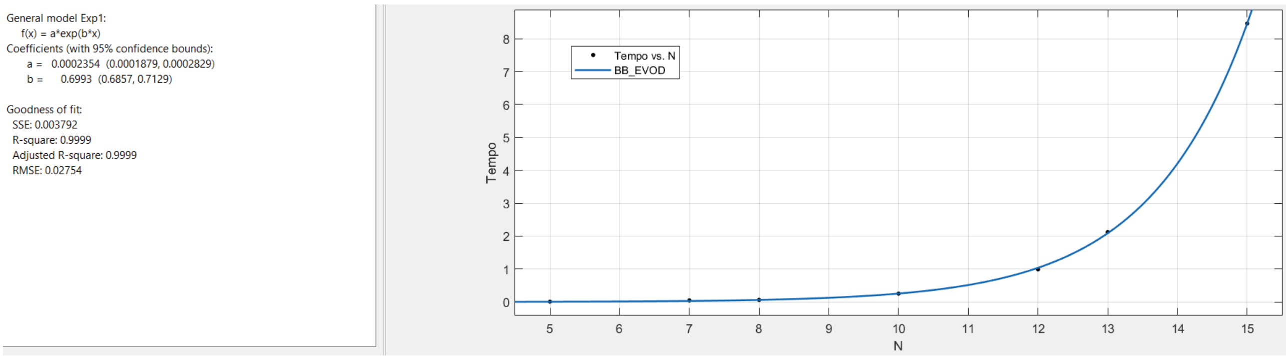

Figura 4.5.Regressão feita pelo MatLab.

PD

P3:

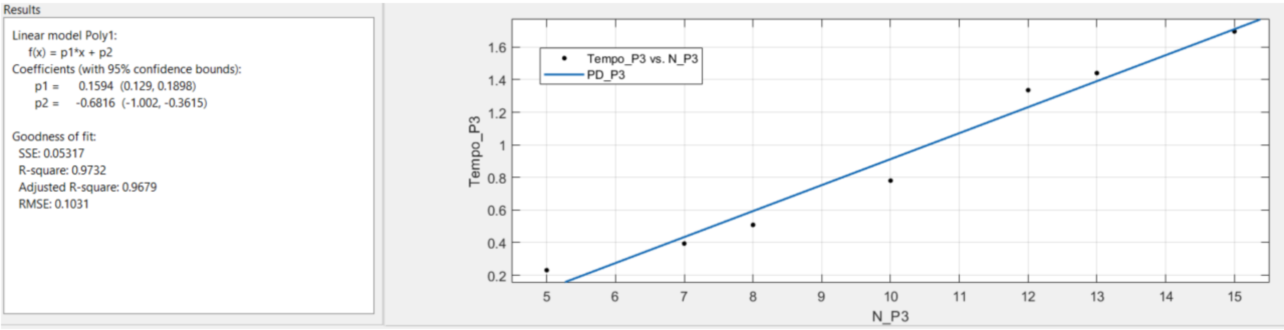

Figura 4.4.Regressão feita pelo MatLab.

P4:

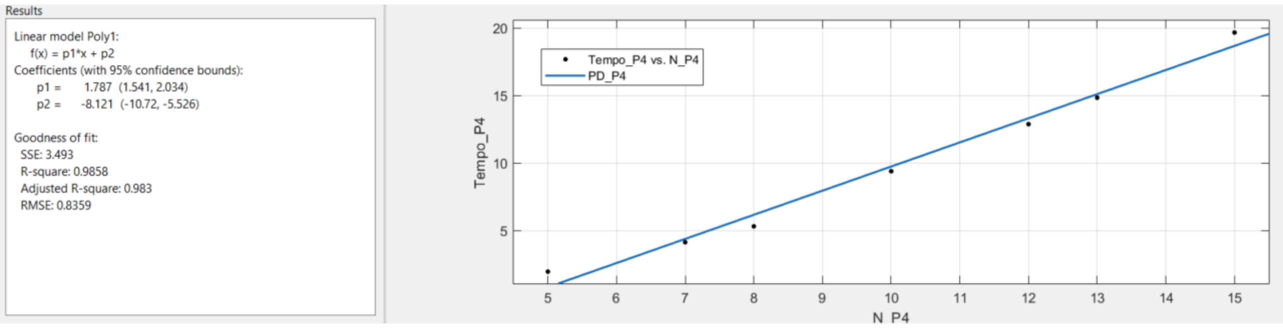

Figura 4.2.Regressão feita pelo MatLab.

P5:

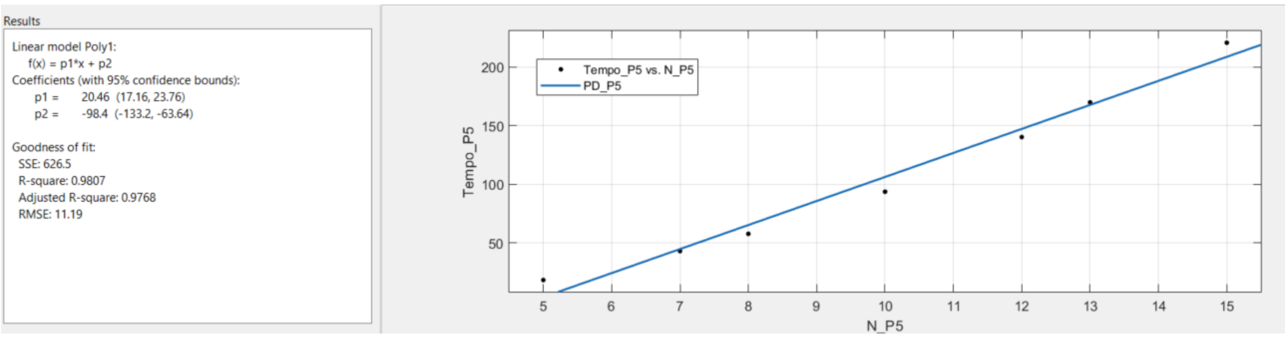

Figura 4.3.Regressão feita pelo MatLab.

AVIS:

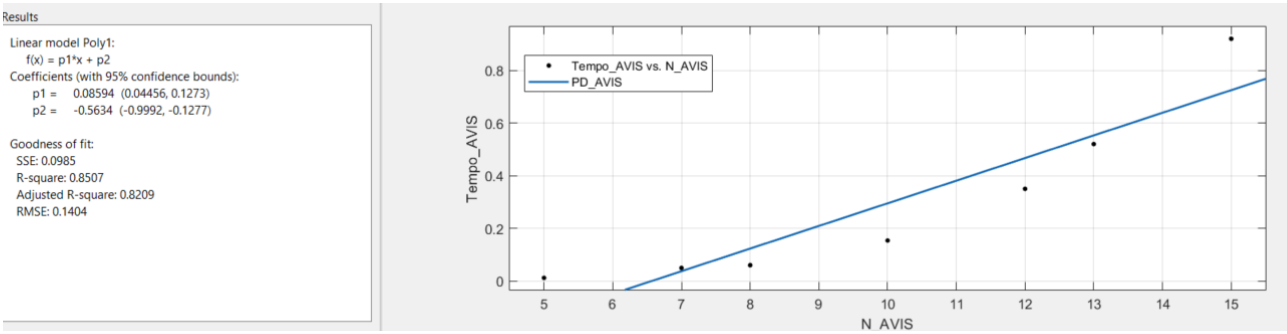

Figura 4.4.Regressão feita pelo MatLab.

EVOD:

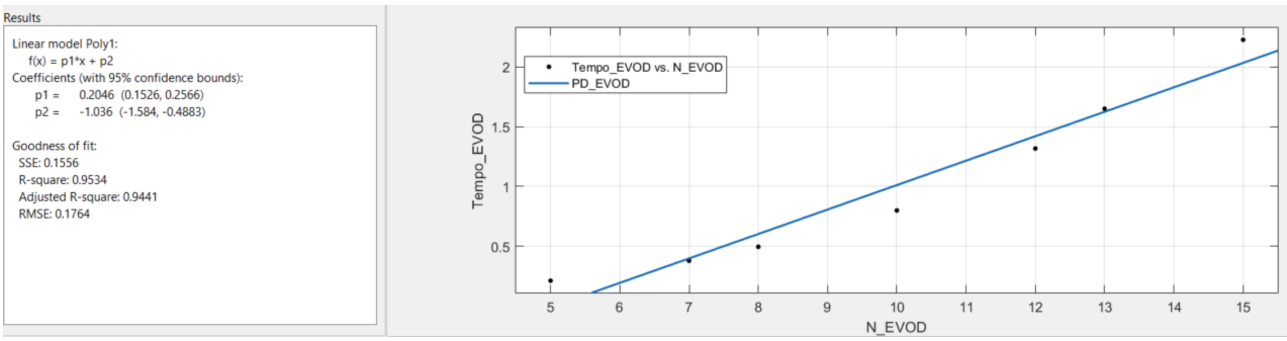

Figura 4.4.Regressão feita pelo MatLab.