Técnicas de Concepção de Algoritmos: **Branch and Bound**

R. Rossetti, A.P. Rocha, João Pascoal Faria

FEUP, MIEIC, CAL, 2013/2014

Branch and Bound

- O que é?
  - BB ou B&B é uma técnica de concepção de algoritmos genérica para encontrar soluções óptimas em vários problemas de optimização
  - Aplicada especialmente em problemas de optimização discreta e combinatorial
  - Aplica-se geralmente quando uma técnica gananciosa (*greedy*) ou programação dinâmica não são de todo apropriadas
  - É geralmente mais lento que as anteriores; pode levar a complexidades temporais exponenciais nos piores casos
  - Se bem aplicada, pode gerar soluções razoavelmente rápidas nos casos médios

Branch and Bound

- Princípio
  - Consiste na enumeração de todas as soluções candidatas
  - Um alargado número de candidatos "não promissores" é descartado, estimando-se limites superiores e inferiores da quantidade a ser optimizada
  - Baseia-se numa pesquisa em largura (*breadth-first search*), mas nem todos os vértices filhos são expandidos
    - Há um critério bem definido para dizer que filhos expandir e quando
    - Outro critério determina quando o algoritmo encontrou a solução óptima
  - Método proposto por A. H. **Land** and A. G. **Doig** (1960) para programação discreta

    A. H. Land and A. G. Doig (1960). "An automatic method of solving discrete programming problems". *Econometrica* **28** (3): pp. 497-520.

Aplicações

- A abordagem é utilizada para resolver alguns problemas conhecidos, ditos *NP-hard*, como:
  - Programação Inteira
  - Programação não-linear
  - Problema do Caixeiro Viajante (*Travelling Salesman)* (TSP)
  - Problema de afectação quadrática (QAP)
  - *The Maximum Satisfiability Problem* (MAX-SAT)
  - Pesquisa do Vizinho mais Próximo (NNS)
  - O problema de corte de stock
  - False Noise Analysis

B&B: Algoritmo Geral

- Considerações
  - Cada solução é assumida poder ser expressa como um array X[1:n], como por exemplo em técnicas de retrocesso
  - Um "preditor", ou função de custo aproximado CC, deve ser definida
- Definições importantes
  - Um "live node" é um vértice que não foi expandido
  - Um "dead node" é um vértice que já foi expandido
  - Um "expanded node" (ou "e-node") é um "live node" com melhor CC

B&B: Algoritmo Geral

```
Procedure B&B()
begin
   E: nodepointer;
   E := new(node); -- this is the root node which
                   -- is the dummy start node
   H: heap;        -- A heap for all the live nodes
      -- H is a min-heap for minimization problems,
      -- and a max-heap for maximization problems.
   while (true) do
      if (E is a final leaf) then
      -- E is an optimal solution
      print out the path from E to the root;
      return;
      endif

      Expand(E);
      if (H is empty) then
         report that there is no solution;
         return;
      endif
      E := delete-top(H);
   endwhile
end
```

B&B: Algoritmo Geral

```
Procedure Expand(E)

begin

   - Generate all the children of E;
   - Compute the approximate cost value CC of each child;
   - Insert each child into the heap H;

end
```


Funções de Custo Aproximado

- Como escolher?
  - **Definição** da Função de Custo *C*: para cada vértice ***x*** na árvore de solução, a função de custo *C*(***x***) é o custo da melhor solução que vai até o vértice ***x***.
  - **Teorema**:
    No caso de problemas de minimização, se *CC*(***x***) <= *C*(***x***) para todo vértice ***x***, e se *CC*(***x***) = *C*(***x***) para todo vértice folha ***x***, então o primeiro vértice a ser expandido (melhor *CC*), que é um vértice folha, corresponde à solução óptima do problema!
  - *Prove!*

Funções de Custo Aproximado

- Critérios para escolha da função de custo aproximado CC
  - Para problemas de minimização:
    CC(***x***) <= *C*(***x***) para todo "*live node*" x
    CC(***x***) = *C*(***x***) para todo vértice folha
  - Para problemas de maximização:
    CC(***x***) >= *C*(***x***) para todo "*live node*" x
    CC(***x***) = *C*(***x***) para todo vértice folha
  - *CC* é chamada uma subestimação de *C* (no caso de minimização), e uma superestimação de *C* (no caso de maximização)
  - Quanto mais próximo que *CC* estiver de *C*, mais rápido será capaz o algoritmo de encontrar a solução óptima!

Exemplo de Aplicação: B&B

- **Problema**
  - Uma companhia está a montar uma equipa para levar a cabo uma série de operações
  - Há **quatro membros** na equipa: A, B, C e D
  - Há **quatro operações** a realizar: 1, 2, 3, e 4
  - Cada membro da equipa pode realizar exactamente uma operação, e Todas as quatro operações devem ser executadas com sucesso, para que para que o projecto todo seja considerado concluído satisfatoriamente.
  - Entretanto, a probabilidade de um membro particular da equipa ser bem sucedido na realização de uma operação particular varia, como indicado na tabela abaixo:

		Operation			
		1	2	3	4
Team member	A	0.9	0.8	0.9	0.85
	B	0.7	0.6	0.8	0.7
	C	0.85	0.7	0.85	0.8
	D	0.75	0.7	0.75	0.7

Exemplo de Aplicação: B&B

- **Problema (cont…)**
  - Se aos membros da equipa, ABCD, fossem atribuídas as tarefas 1234, nesta ordem, então a probabilidade do projecto ser concluído na sua totalidade com sucesso seria:

    (0.9)(0.6)(0.85)(0.7) = 0.3213 ~ 32%

  - **Pressuposto**: Se houver formas possíveis de organizar a equipa de modo a se obter uma taxa de sucesso para o projecto como um todo que exceda aos **45%**, então o gestor irá aprovar o projecto!
  - **Questão**: para esta dada equipa, o gestor irá aprovar o projecto?

		Operation			
		1	2	3	4
Team member	A	0.9	0.8	0.9	0.85
	B	0.7	0.6	0.8	0.7
	C	0.85	0.7	0.85	0.8
	D	0.75	0.7	0.75	0.7

Exemplo de Aplicação: B&B

- **Formulação:**
  - Vértices na árvore: uma atribuição pessoa-tarefa, total ou parcial
  - Política de selecção de vértices: melhor valor global da **função de custo aproximado** (*bounding function*)
  - Política de selecção de variáveis: escolher a próxima operação na ordem natural, ou seja 1 a 4
  - Função de Custo (**bounding function**): para operações não atribuídas, escolher o melhor membro da equipa que esteja disponível, ou seja, o que possui maior probabilidade de sucesso, ainda que a pessoa seja escolhida mais do que uma vez
  - Função Objectivo (critério de paragem): quando um estado-solução candidato tiver o valor resultante da função de custo maior do que o valor do estado-candiato corrente
  - Estado-solução candidato: quando a função de custo retorna uma solução em que cada operação é atribuída a uma pessoa diferente

Team member	Operation 1	2	3	4
A	0.9	0.8	0.9	0.85
B	0.7	0.6	0.8	0.7
C	0.85	0.7	0.85	0.8
D	0.75	0.7	0.75	0.7

Solução candidata: ?

No início, nenhuma decisão foi tomada; Calcula-se o valor máximo do potencial de sucesso para o projecto:

Se A for escolhido para realizar as 4 operações, então:

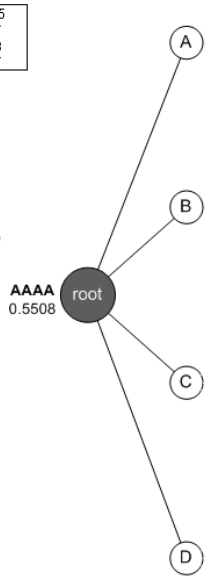
AAAA = (0.9)(0.8)(0.9)(0.85) = 0.5508

Team member	Operation 1	2	3	4
A	0.9	0.8	0.9	0.85
B	0.7	0.6	0.8	0.7
C	0.85	0.7	0.85	0.8
D	0.75	0.7	0.75	0.7

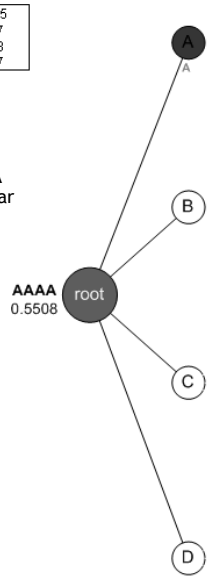
Solução candidata: ?

Para a **Operação 1**, expande-se o vértice raiz (*branching process*).

Cada nó filho corresponde à atribuição da primeira tarefa a um membro específico da equipa…



Team member	Operation 1	2	3	4
A	0.9	0.8	0.9	0.85
B	0.7	0.6	0.8	0.7
C	0.85	0.7	0.85	0.8
D	0.75	0.7	0.75	0.7

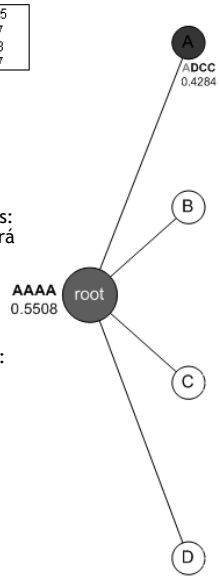


Solução candidata: ?

Neste nó, **A**, decidiu-se que a pessoa A irá realizar a operação 1. Iremos indicar isso colorindo A em verde

A não pode ser seleccionada para realizar qualquer outra tarefa, porque já foi escolhido

Team member	Operation 1	2	3	4
A	0.9	0.8	0.9	0.85
B	0.7	0.6	0.8	0.7
C	0.85	0.7	0.85	0.8
D	0.75	0.7	0.75	0.7



Solução candidata: ?

As outras pessoas serão atribuídas aos outros trabalhos. Isso será feito com o mesmo critério utilizado para os outros: a pessoa com melhor probabilidade será escolhida, ainda que se repita; se houver impasse, escolhe-se arbitrariamente

O resultado da função de custo para este estado **ADCC**, dá-nos um valor de:

ADCC = (0.9)(0.7)(0.85)(0.8) = 0.4284

Team member	Operation 1	2	3	4
A	0.9	0.8	0.9	0.85
B	0.7	0.6	0.8	0.7
C	0.85	0.7	0.85	0.8
D	0.75	0.7	0.75	0.7

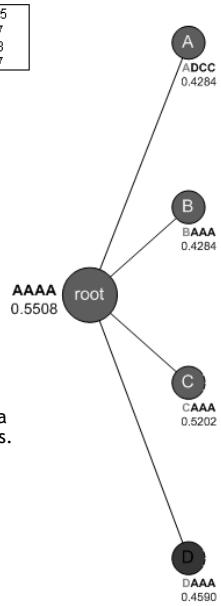
Solução candidata: ?

Passa-se para o próximo nó, em que a decisão é que B realize a operação 1; neste caso, B não poderá realizar qualquer outra operação; as restantes pessoas serão atribuídas às outras operações;

Neste caso, A é a melhor pessoa para realizar 2, 3, e 4, resultando em:

BAAA = (0.7)(0.8)(0.9)(0.85) = 0.4284

Realiza-se o mesmo procedimento para os outros nós filhos a serem explorados.



FEUP Universidade do Porto Faculdade de Engenharia

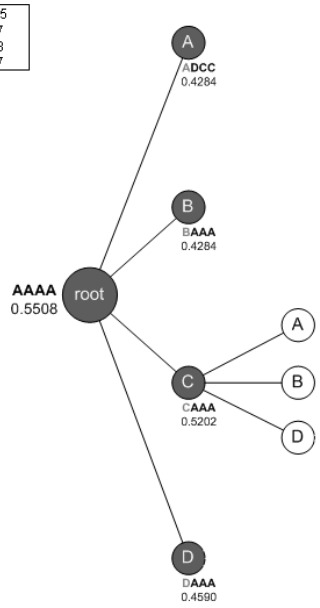
Branch and Bound - CAL, 2013/14 <#>

Team member	Operation 1	2	3	4
A	0.9	0.8	0.9	0.85
B	0.7	0.6	0.8	0.7
C	0.85	0.7	0.85	0.8
D	0.75	0.7	0.75	0.7

Solução candidata: ?

Agora iremos expandir (branch) os nós da árvore.

O nó que escolhemos expandir é o nó que é mais promissor, ou seja, aquele que potencialmente irá levar-nos à melhor solução. Neste caso é o nó **C**, com valor **0.5202**



FEUP Universidade do Porto Faculdade de Engenharia

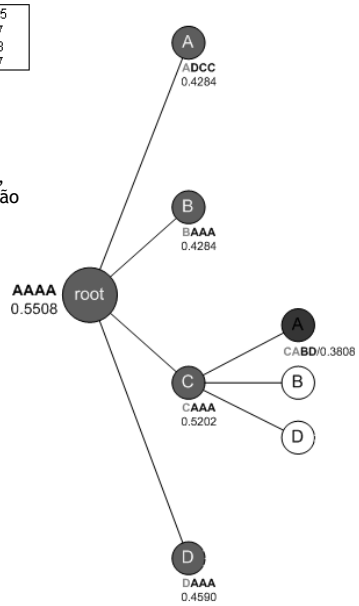
Branch and Bound - CAL, 2013/14 <#>

Team member	Operation 1	2	3	4
A	0.9	0.8	0.9	0.85
B	0.7	0.6	0.8	0.7
C	0.85	0.7	0.85	0.8
D	0.75	0.7	0.75	0.7

Solução candidata: ?

Ao avaliarmos o no filho, a partir de C, verifica-se que este é um estado solução candidato, pois todos os elementos atribuídos às tarefas 1 a 4 são diferentes, nomeadamente **CABD**!

Neste estado, a probabilidade de sucesso do projecto é de ~**38%**



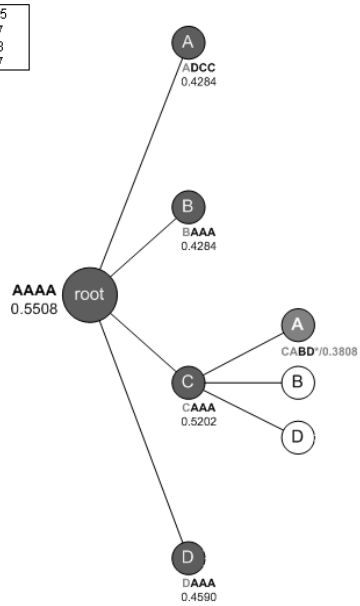
FEUP Universidade do Porto Faculdade de Engenharia

Branch and Bound - CAL, 2013/14 ‹#›

Team member	Operation 1	2	3	4
A	0.9	0.8	0.9	0.85
B	0.7	0.6	0.8	0.7
C	0.85	0.7	0.85	0.8
D	0.75	0.7	0.75	0.7

Solução candidata: **CABD (0.3808)**

Neste caso, marca-se a solução CABD como a solução candidata corrente; qualquer outra solução cujo resultado da função de custo seja melhor, irá substitui-la



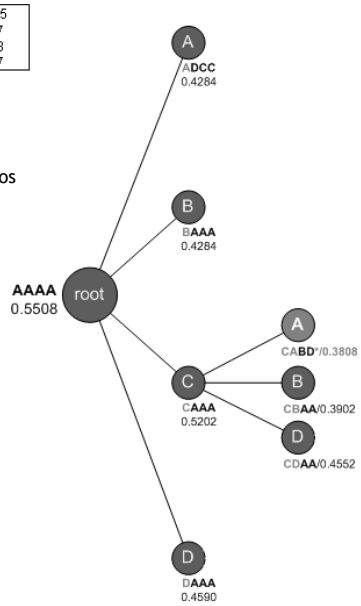
FEUP Universidade do Porto Faculdade de Engenharia

Branch and Bound - CAL, 2013/14 ‹#›

Team member	Operation 1	2	3	4
A	0.9	0.8	0.9	0.85
B	0.7	0.6	0.8	0.7
C	0.85	0.7	0.85	0.8
D	0.75	0.7	0.75	0.7

Solução candidata: **CABD (0.3808)**

Prossegue-se com a avaliação dos outros estados…



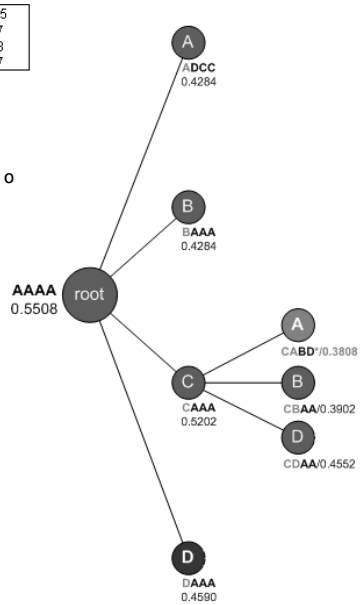
FEUP Universidade do Porto Faculdade de Engenharia

Branch and Bound - CAL, 2013/14 ‹#›

Team member	Operation 1	2	3	4
A	0.9	0.8	0.9	0.85
B	0.7	0.6	0.8	0.7
C	0.85	0.7	0.85	0.8
D	0.75	0.7	0.75	0.7

Solução candidata: **CABD (0.3808)**

… e escolhe-se para expandir (*branch*) o estado mais promissor até o momento encontrado; neste caso, será o vértice D, no nível 1 da árvore!



FEUP Universidade do Porto Faculdade de Engenharia

Branch and Bound - CAL, 2013/14 ‹#›

Team member	Operation 1	2	3	4
A	0.9	0.8	0.9	0.85
B	0.7	0.6	0.8	0.7
C	0.85	0.7	0.85	0.8
D	0.75	0.7	0.75	0.7

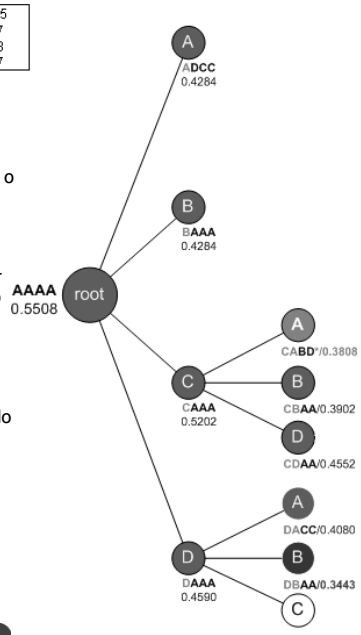
Solução candidata: **CABD (0.3808)**

… e escolhe-se para expandir (*branch*) o estado mais promissor até o momento encontrado; neste caso, será o vértice D, no nível 1 da árvore!

Importante! Ao expandir o nó e avaliar os seus filhos, verifica-se que o estado DBAA tem uma probabilidade máxima esperada de

0.3443 < 0.3808

Que é menor do que o valor encontrado para a solução corrente candidata! Desta forma, não interessa expandir mais a árvore a partir deste estado!



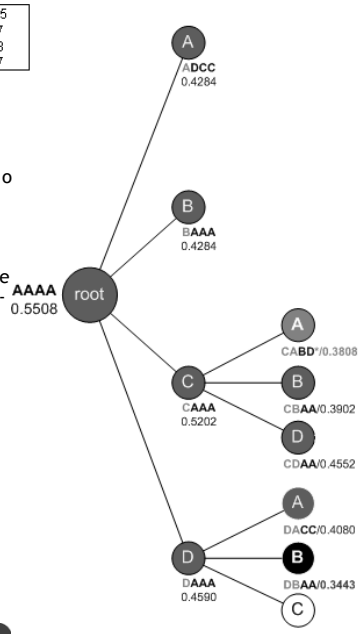
Team member	Operation 1	2	3	4
A	0.9	0.8	0.9	0.85
B	0.7	0.6	0.8	0.7
C	0.85	0.7	0.85	0.8
D	0.75	0.7	0.75	0.7

Solução candidata: **CABD (0.3808)**

Considerando o valor encontrado para o DBAA…

0.3443 < 0.3808

Marca-se este estado como um "vértice morto," não sendo necessário expandi-lo mais

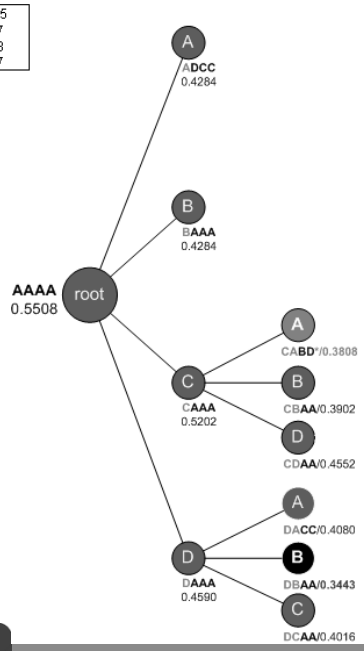


FEUP Universidade do Porto Faculdade de Engenharia
Branch and Bound - CAL, 2013/14 ‹#›

Team member	Operation 1	2	3	4
A	0.9	0.8	0.9	0.85
B	0.7	0.6	0.8	0.7
C	0.85	0.7	0.85	0.8
D	0.75	0.7	0.75	0.7

Solução candidata: **CABD (0.3808)**

Completa-se a expansão do nó D e a avaliação dos estados resultantes a partir daquele nó…



Team member	Operation 1	2	3	4
A	0.9	0.8	0.9	0.85
B	0.7	0.6	0.8	0.7
C	0.85	0.7	0.85	0.8
D	0.75	0.7	0.75	0.7

Solução candidata: **CABD (0.3808)**

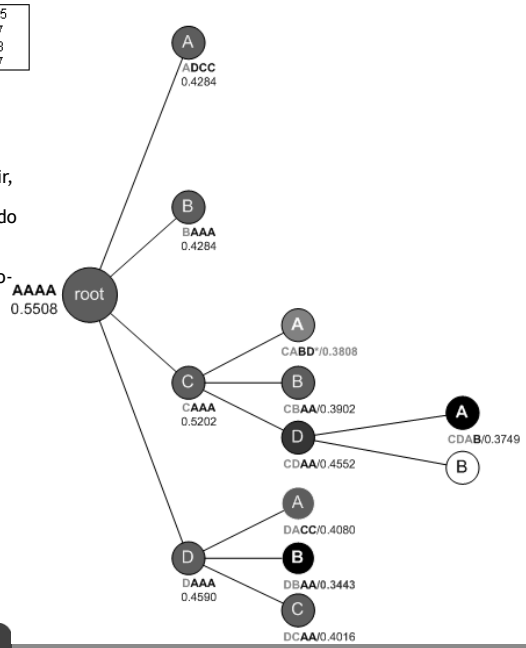
Escolhe-se o próximo estado a expandir, ou seja, aquele que é mais promissor em termos de potencial de conclusão do projecto. Neste caso, o estado **CDAA**

Ao expandi-lo, verifica-se que o estado-filho **CDAB** é um estado-solução candidato!

Porém, como o valor do seu potencial de conclusão é

0.3719 < 0.3808

… a solução **CDAB** é descartada, e marcada como um "vértice morto"



FEUP Universidade do Porto Faculdade de Engenharia
Branch and Bound - CAL, 2013/14 ‹#›

		Operation			
		1	2	3	4
Team member	A	0.9	0.8	0.9	0.85
	B	0.7	0.6	0.8	0.7
	C	0.85	0.7	0.85	0.8
	D	0.75	0.7	0.75	0.7

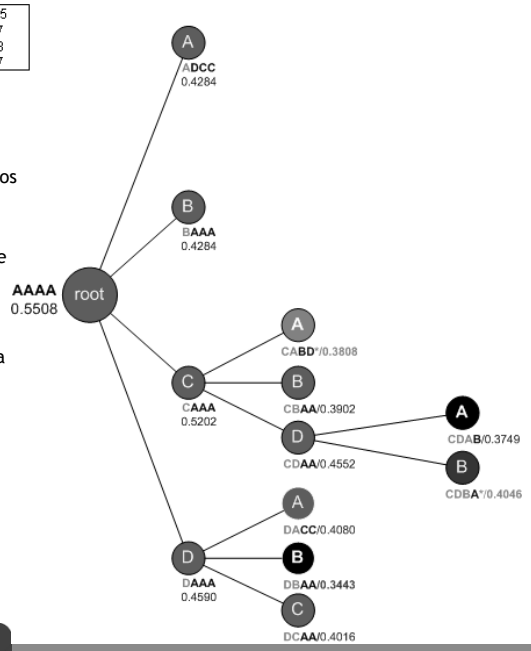
Solução candidata: **CABD (0.3808)**

Conclui-se a avaliação dos estados filhos e verifica-se que o estado CDBA é um estado-solução candidato!

Neste caso, o valor do seu potencial de conclusão é

0.4046 > 0.3808

… e portanto passará a ser considerada a solução candidata!

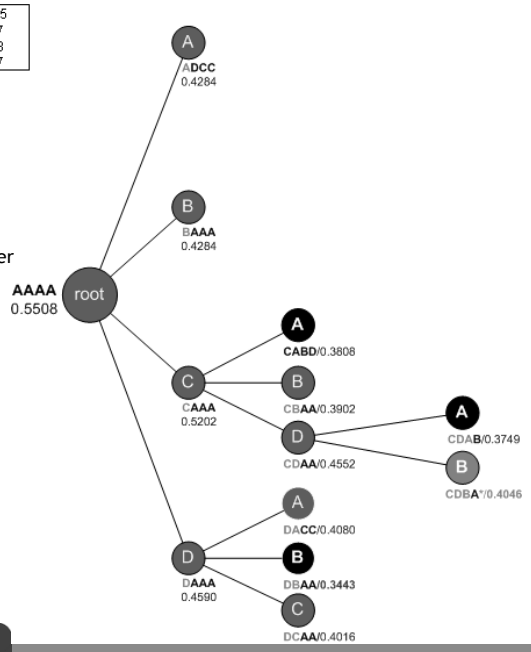


		Operation			
		1	2	3	4
Team member	A	0.9	0.8	0.9	0.85
	B	0.7	0.6	0.8	0.7
	C	0.85	0.7	0.85	0.8
	D	0.75	0.7	0.75	0.7

Solução candidata: ~~**CABD (0.3808)**~~
Solução candidata: **CDBA (0.4046)**

Marca-se a nova solução candidata encontrada: **CDBA**!

A solução candidata anterior passa a ser um "vértice morto"



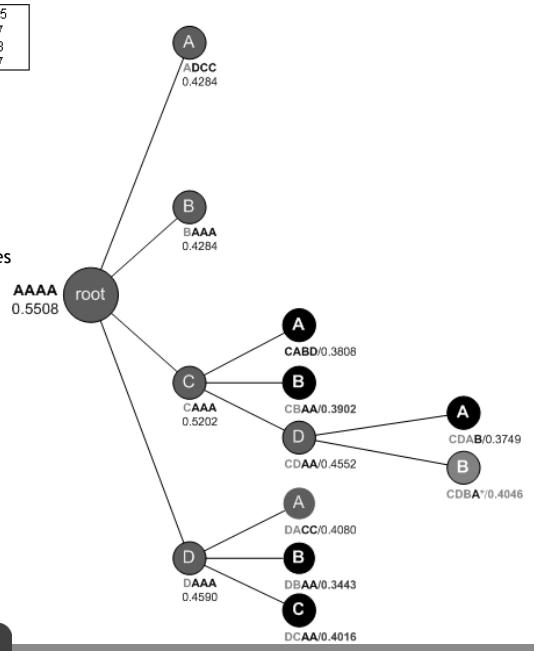
Team member	Operation 1	2	3	4
A	0.9	0.8	0.9	0.85
B	0.7	0.6	0.8	0.7
C	0.85	0.7	0.85	0.8
D	0.75	0.7	0.75	0.7

Solução candidata: ~~**CABD (0.3808)**~~
Solução candidata: **CDBA (0.4046)**

Com o novo valor do potencial de conclusão para a solução candidata corrente, é possível avaliar quão promissores continuam a ser os vértices à espera de serem processados.

Neste caso, todos aqueles cujo valor é menor do que o novo valor corrente, passam a ser considerados não promissores e, portanto, "vértices mortos"

É o que acontece com os estados **CBAA**(0.3902) e **DCAA**(0.4016)

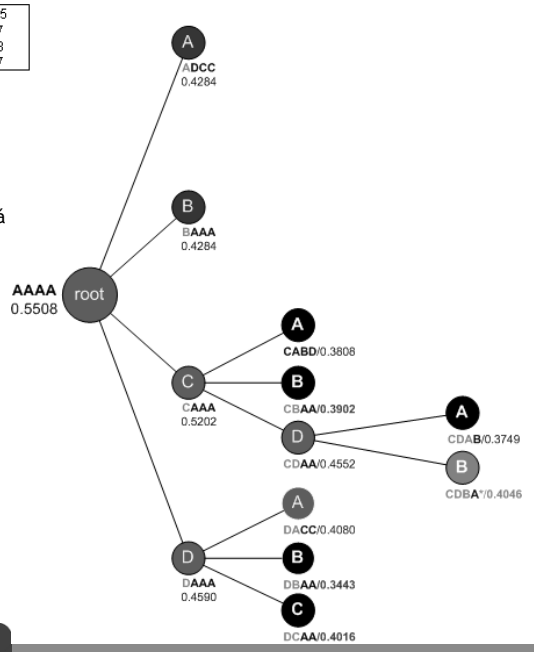


Team member	Operation 1	2	3	4
A	0.9	0.8	0.9	0.85
B	0.7	0.6	0.8	0.7
C	0.85	0.7	0.85	0.8
D	0.75	0.7	0.75	0.7

Solução candidata: ~~**CABD (0.3808)**~~
Solução candidata: **CDBA (0.4046)**

Prossegue-se com a escolha do novo estado a ser expandido. Neste caso, há duas opções com a mesma propensão, pelo que se escolhe arbitrariamente!

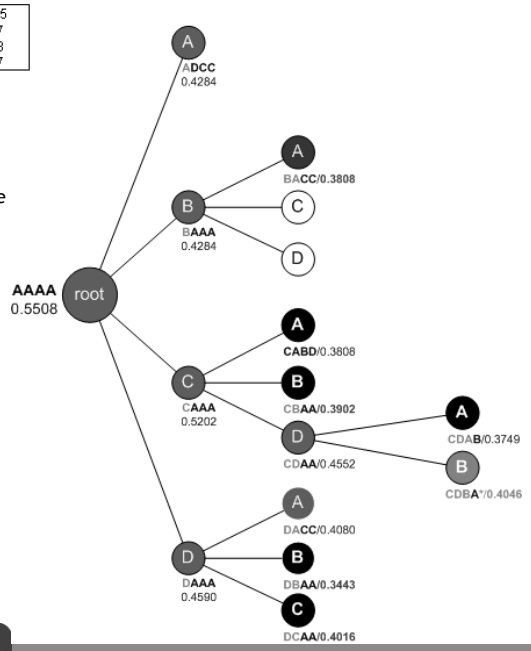
Ambos ADCC e BAAA têm o mesmo potencial: **0.4284**!



Team member	Operation 1	2	3	4
A	0.9	0.8	0.9	0.85
B	0.7	0.6	0.8	0.7
C	0.85	0.7	0.85	0.8
D	0.75	0.7	0.75	0.7

Solução candidata: ~~**CABD (0.3808)**~~
Solução candidata: **CDBA (0.4046)**

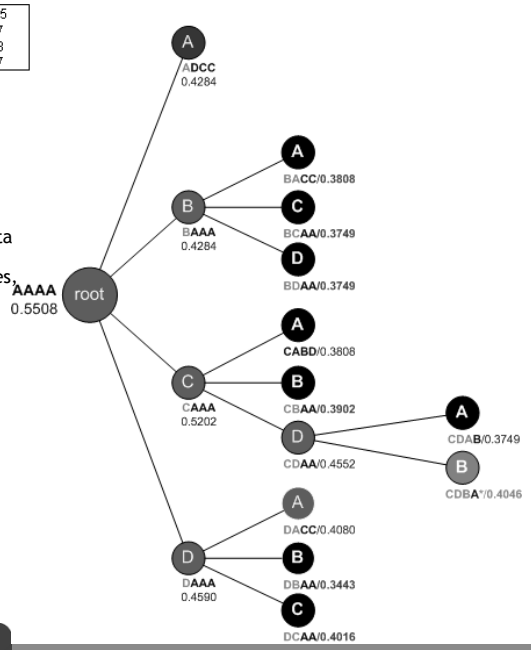
Opta-se por expandir o estado **BAAA**, e compara-se o potencial dos filhos em relação ao valor da solução candidata corrente!



Team member	Operation 1	2	3	4
A	0.9	0.8	0.9	0.85
B	0.7	0.6	0.8	0.7
C	0.85	0.7	0.85	0.8
D	0.75	0.7	0.75	0.7

Solução candidata: ~~**CABD (0.3808)**~~
Solução candidata: **CDBA (0.4046)**

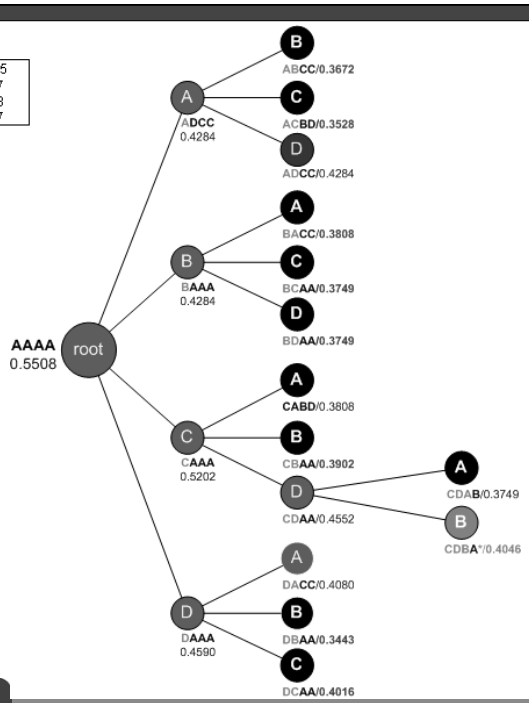
Como todos os valares gerados pela função de custo para os filhos são inferiores ao valor da solução candidata corrente, os estados **BACC**, **BCAA** e **BDAA** são considerados não promissores, e portanto marcados como "vértices mortos"



Team member	Operation 1	2	3	4
A	0.9	0.8	0.9	0.85
B	0.7	0.6	0.8	0.7
C	0.85	0.7	0.85	0.8
D	0.75	0.7	0.75	0.7

Solução candidata: ~~**CABD (0.3808)**~~
Solução candidata: **CDBA (0.4046)**

Continua-se a expandir os próximos estados promissores, neste caso o estado **ADCC**, e a marcar os filhos não promissores como "vértices mortos"



FEUP Universidade do Porto Faculdade de Engenharia
Branch and Bound - CAL, 2013/14 <#>

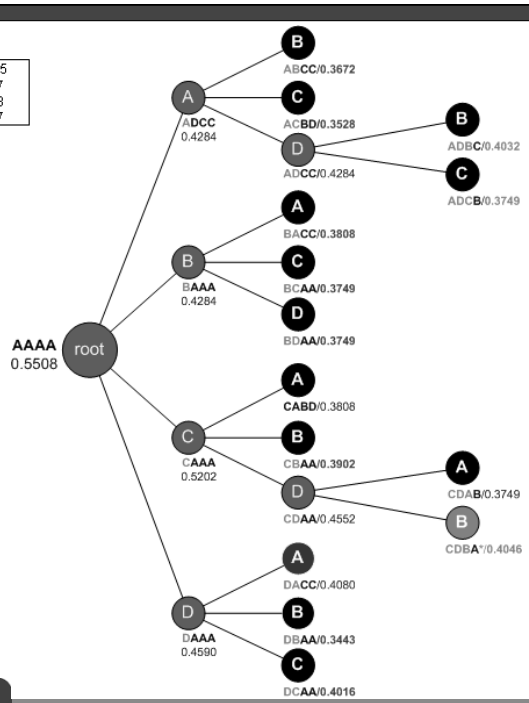
Team member	Operation 1	2	3	4
A	0.9	0.8	0.9	0.85
B	0.7	0.6	0.8	0.7
C	0.85	0.7	0.85	0.8
D	0.75	0.7	0.75	0.7

Solução candidata: ~~**CABD (0.3808)**~~
Solução candidata: **CDBA (0.4046)**

A partir do estado ADCC, chega-se aos estados filhos **ADBC** e **ADCB**, ambos estados-solução do problema.

Entretanto, estes estados são descartados, pois representam uma pior solução do que a solução candidata corrente!

Note que o estado **DACC** é o último estado à espera de ser expandido!



FEUP Universidade do Porto Faculdade de Engenharia
Branch and Bound - CAL, 2013/14 <#>

Team member	Operation 1	2	3	4
A	0.9	0.8	0.9	0.85
B	0.7	0.6	0.8	0.7
C	0.85	0.7	0.85	0.8
D	0.75	0.7	0.75	0.7

Solução candidata: ~~**CABD (0.3808)**~~
Solução candidata: **CDBA (0.4046)**

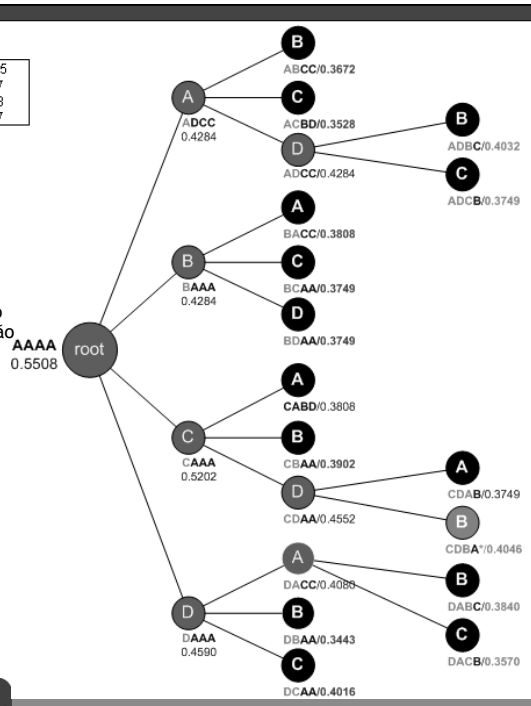
Procede-se com a expansão do último estado, neste caso o estado **DACC**.

Ambos os estados filhos são solução do problema, DABC e DACB. Entretanto são descartados por serem inferior à solução candidata corrente!

O projecto em questão pode ser realizado com uma taxa de sucesso esperada máxima igual a

0.4046 ~40,5%

Para esta equipa, como o valor encontrado é inferior ao desejado, de 45%, o Projecto não será aprovado!



FEUP Universidade do Porto Faculdade de Engenharia
Branch and Bound - CAL, 2013/14 <#>

Algns cuidados a ter em B&B

- Para utilizar B&B com relativo sucesso, é necessário:
  - Ter uma boa função de custo (bouding function): deve ser uma função optimística, mas tão viável quanto possível
  - Reconhecer um estado-solução candidato cedo: é importante para iniciar o processo de poda da árvore, diminuindo o número de estados a serem expandidos
  - Encontrar formas de identificar estados cujos descendentes não são promissores: Este procedimento também evita que vértices sejam expandidos desnecessariamente
  - Ordenar restrições e variáveis, de forma a testar primeiro as mais restritivas: a ideia geral é favorecer que os vértices da árvore sejam considerados não promissores o mais rapidamente possível. O mais próximo da raiz que um vértice é podado, mais a árvore poderá ser simplificada.

FEUP Universidade do Porto Faculdade de Engenharia
Branch and Bound - CAL, 2013/14 <#>

Referências e mais informação

- "Introduction to Algorithms", Second Edition, Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, The MIT Press, 2001
- "The Algorithm Design Manual", Steven S. Skiena, Springer-Verlag, 1998
- A. H. Land and A. G. Doig (1960). "An automatic method of solving discrete programming problems". Econometrica 28 (3): pp. 497-520
- John W. Chinneck (2010) "Practical Optimization: a Gentle Introduction", Carleton University, CA.