Branch-and-bound

Aula baseada em diversas fontes:

"Integer programming" de L. Wolsey, 1998.

"Integer and combinatorial optimization" de Nemhauser e Wolsey, 1988

Slides do curso "Optimization methods in management sciente" – J. Orlin et al., MIT Opencourseware

Slides do curso "Systems Optimization" – John Vande Vate, MIT Opencourseware

Como resolver problemas combinatórios

- Exemplo 1:
  - três variáveis:
    - x_1
    - x_2
    - x_3
- Cada uma delas pode assumir valores 0 ou 1.

- Possibilidade: enumeração explícita (listar todas as soluções possíveis e verificar qual a melhor).

Exemplo 1

- Através de ramificações, exploramos (enumeramos) todas as possíveis soluções.

  Ex.: problema com três variáveis binárias

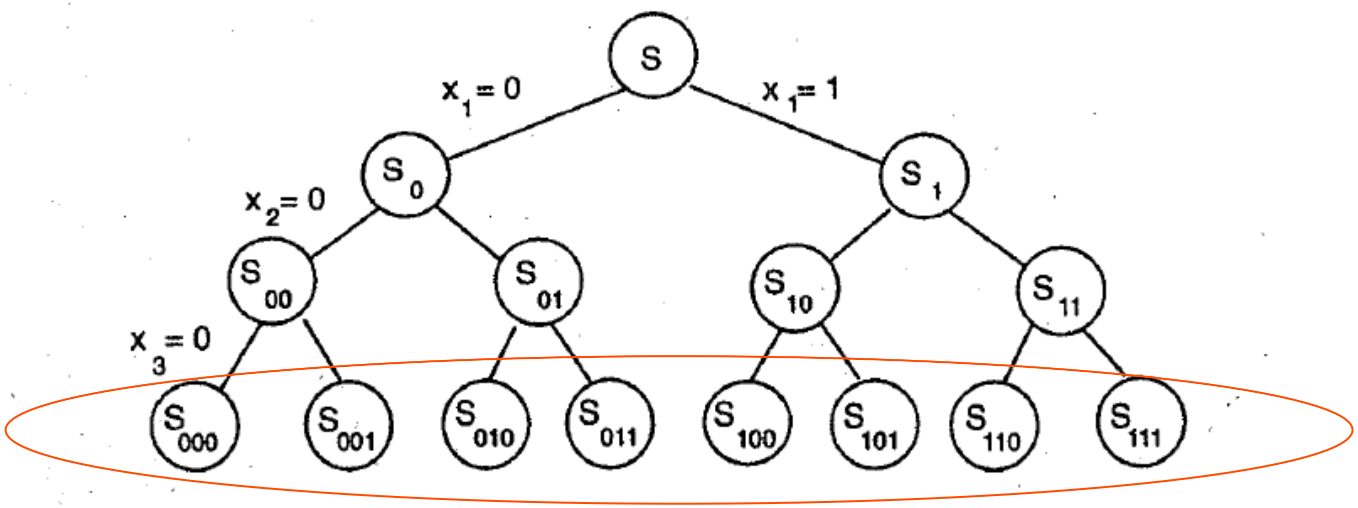


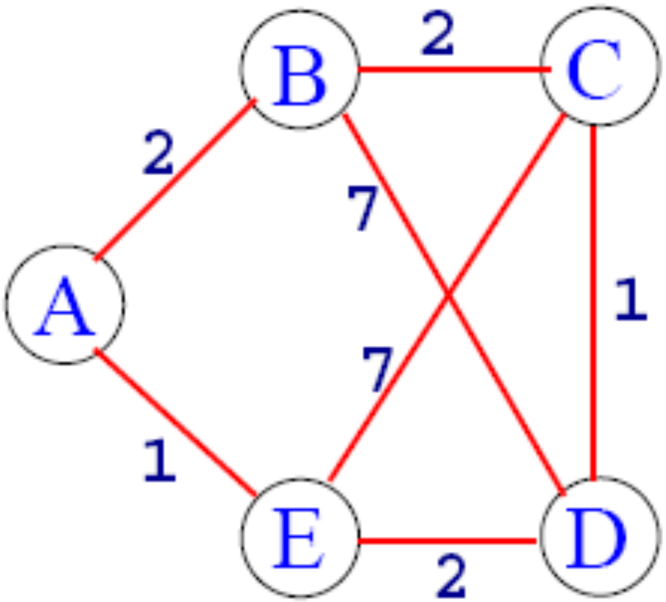
Fig. 7.1 Binary enumeration tree

problemas que precisam ser resolvidos

Wolsey, L. A., Integer Programming.Wiley-Interscience, 1998

Exemplo 2

- Um problema do caixeiro viajante com 5 cidades

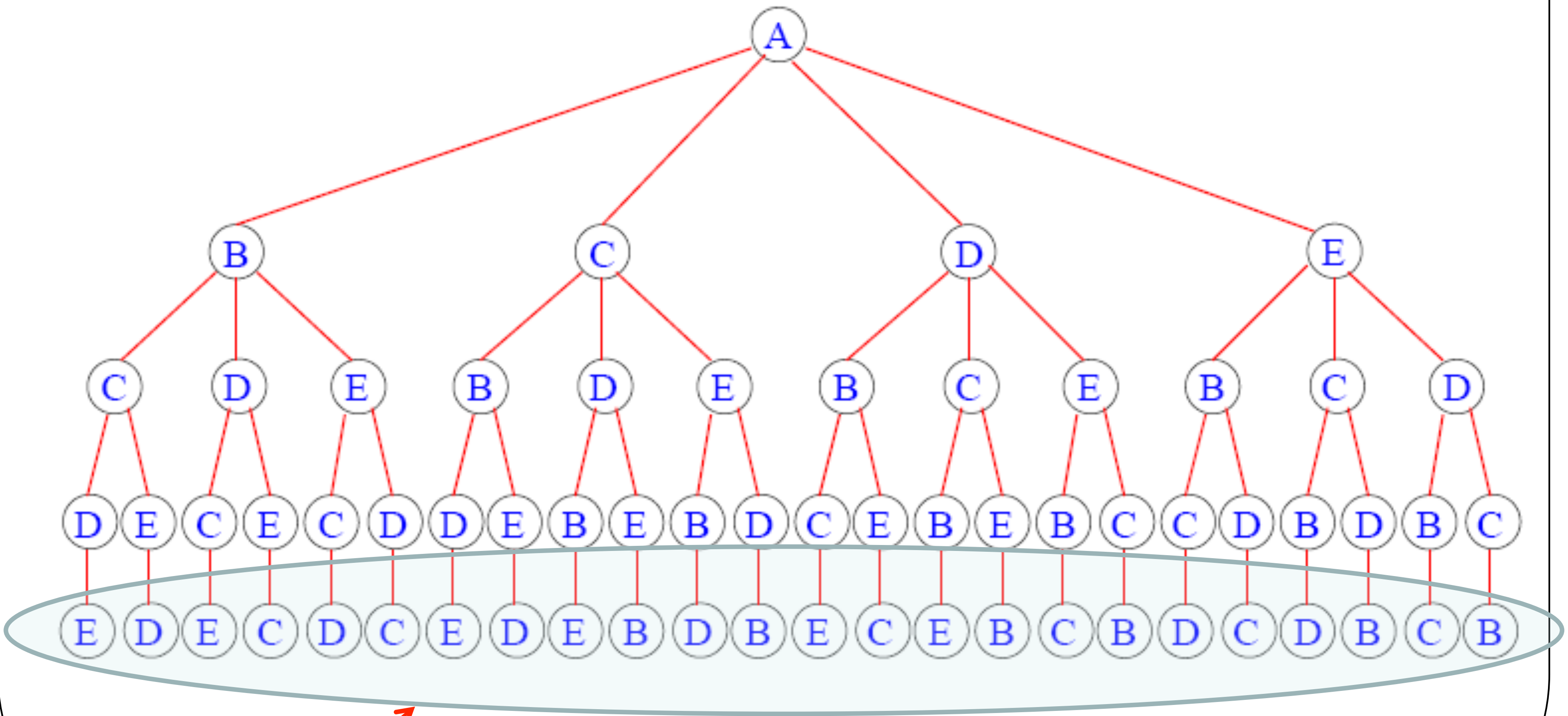


Estagiário PAE: Guilherme Albuquerque

Exemplo 2

- Para o problema do caixeiro viajante:
  - os circuitos são representados por caminhos de *n* vértices;
  - os nós da árvore representam os vértices;
  - as ramificações de um nó para outro representam as arestas.

Exemplo 2

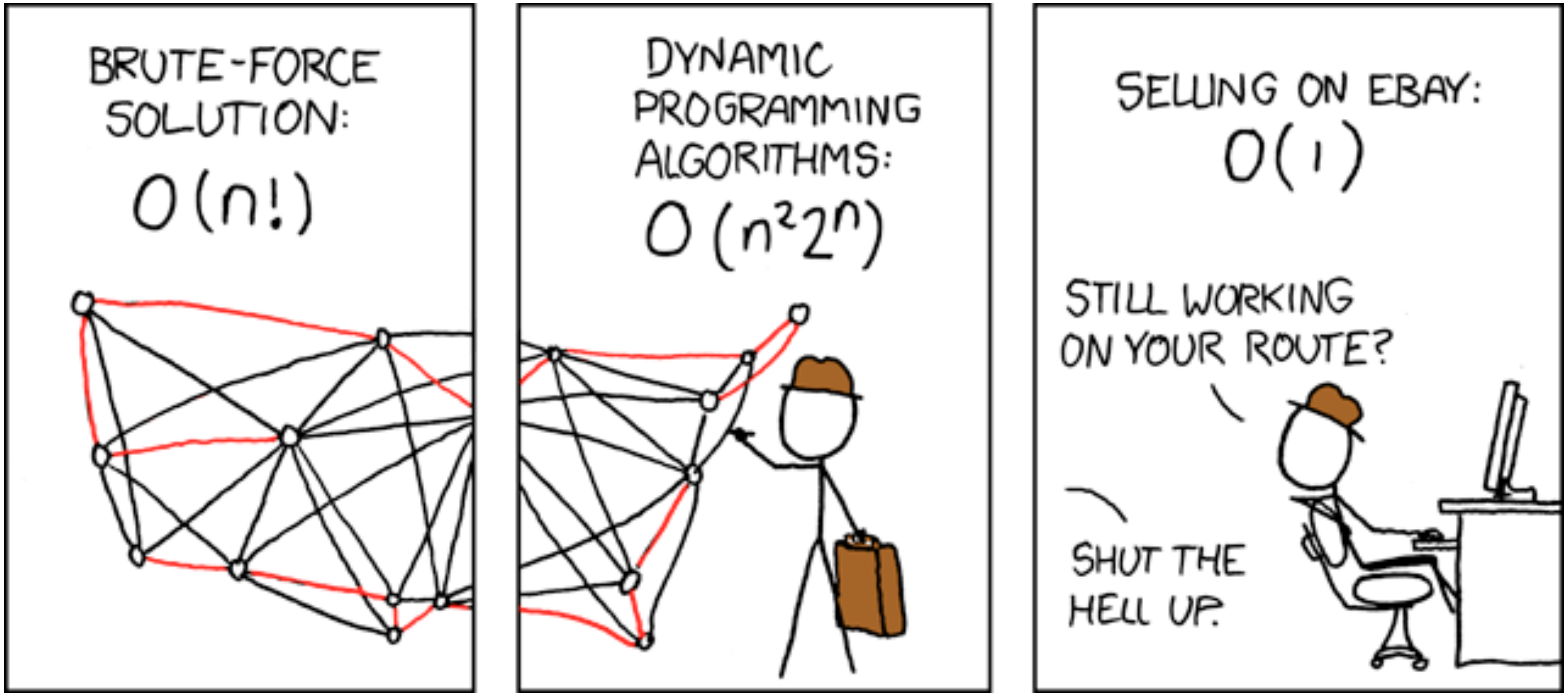


É preciso avaliar todas estas soluções

Estagiário PAE: Guilherme Albuquerque

Enumeração Explícita

- Problema:
  - Enumeração completa é impraticável quando o número de variáveis em um problema é grande.
    - Número de soluções para o ex1: 2^3 (caso geral 2^n)
    - Número de soluções para o ex2: (5-1)! (caso geral (n-1)!
- Proposta:
  - Fazer mais do que dividir indefinidamente!
    - Não só *branch*… mas também *bound*!



http://imgs.xkcd.com/comics/travelling_salesman_problem.png

Crescimento exponencial

On complete enumeration

- Suppose that we could evaluate 1 trillion solutions per second, and instantaneously eliminate 99.9999999% of all solutions as not worth considering
- Let n = number of binary variables
- Solutions times
  - n = 70, 1 second
  - n = 80, 17 minutes
  - n = 90 11.6 days
  - n = 100 31 years
  - n = 110 31,000 years

19

Slides do curso “Optimization methods in management sciente” – J. Orlin et al., MIT

Enumeração Implícita

- **Objetivo** - reduzir o espaço de busca;
  - subconjuntos de soluções são implicitamente considerados e descartados

- **Meio** – utilizar informação da solução do nó corrente para ramificá-lo (ou não ramificá-lo);
  - testar as soluções segundo critérios de eliminação.

Ideia básica

- Enumeração *implícita*.

ECONOMETRICA

VOLUME 28 July, 1960 NUMBER 3

AN AUTOMATIC METHOD OF SOLVING DISCRETE PROGRAMMING PROBLEMS

BY A. H. LAND AND A. G. DOIG

In the classical linear programming problem the behaviour of continuous, nonnegative variables subject to a system of linear inequalities is investigated. One possible generalization of this problem is to relax the continuity condition on the variables. This paper presents a simple numerical algorithm for the solution of programming problems in which some or all of the variables can take only discrete values. The algorithm requires no special techniques beyond those used in ordinary linear programming, and lends itself to automatic computing. Its use is illustrated on two numerical examples.

Land and Doig, Econometrica, 28, 497-520

Professor Ailsa Land

Emeritus Professor of Operational Research

Recent publications

- PAUSE: a computationally tractable combinatorial auction
  Land, Ailsa, Powell, Susan and Steinberg, Richard (2006) PAUSE: a computationally tractable combinatorial auction. In: Cramton, Peter C., Shoham, Yoav and Steinberg, Richard, (eds.) Combinatorial Auctions. MIT Press, Cambridge, MA, USA, pp. 139-157. ISBN 9780262033428
- Minimising a linear function subject to linear, quadratic and discrete constraints
  Land, Ailsa and Powell, Susan (1994) Minimising a linear function subject to linear, quadratic and discrete constraints. Operational Research working papers, LSEOR 94.12. Department of Operational Research, London School of Economics and Political Science, London, UK.
- Computer program for Data Envelopment Analysis
  Land, Ailsa and Powell, Susan (1992) Computer program for Data Envelopment Analysis. Operational Research working papers, LSEOR 92.3. Department of Operational Research, London School of Economics and Political Science, London, UK.

Complete list of publications

Share:

http://www.lse.ac.uk/management/people/aland.aspx

Tutor:

Dr. Alison Harcourt (nee Doig)

Alison is an esteemed member of our OR community and has research interests in linear programming and a background in integer programming: she co-invented the branch-and-bound method in the 60s. She currently tutors in the department with such mathematical prowess that she takes on a broad range of subjects.

www.zoominfo.com/p/Alison-Harcourt/1448488990

THE UNIVERSITY OF MELBOURNE

ALUMNI & FRIENDS

DEPARTMENT OF MATHEMATICS & STATISTICS

THE ALUMNI NEWSLETTER OF THE DEPARTMENT OF MATHEMATICS & STATISTICS, BRINGING PAST STUDENTS TOGETHER.

MESSAGE FROM THE HEAD

Professor Peter Taylor

As I come to the end of six years as Head of Department, it is a good time to reflect on the position we are in.

In general, we can be very proud of the way the Department is travelling. Our undergraduate student numbers have increased dramatically since the introduction of the Melbourne Model, and the feedback that we receive from surveys and other sources indicate that we do a very good job in looking after our students.

One inevitable consequence of the discontinuation of double degrees is that there will be fewer students completing majors in Mathematics and Statistics. We plan to respond to this by making the two-year research-transition MSc a degree of choice, not only for students completing undergraduate Mathematics and Statistics majors at the University of Melbourne, but also for those completing majors at other universities, particularly elsewhere in Australia and New Zealand.

The MSc started in 2009, but in 2011 it will be the first time that the entering cohort will largely consist of new generation undergraduate students, and the structure has been revised to take account of this. There is an emphasis on coherence and on a degree of breadth across the mathematical sciences, which we did not previously have in our Honours course.

The results for the 2011 Australian Research Council Discovery grants have just been announced. The Department has chief investigators on seven successful grant applications and is involved in two more hosted at other universities.

Having helped prepare the university's ERA submission in the Mathematical Sciences and having had the opportunity to compare with parts of the submissions of other universities, I am in a good position to judge how the Department stacks up against other departments in the country. I am confident that the University of Melbourne will be ranked highly in the mathematical sciences by the ERA panel in 2011.

Finally, I would like to congratulate Barry Hughes and Omar Foda who have been promoted to Professor. When I took over as Head in January 2005, there were six full professors in the Department. We now have fifteen full professors. Staff at other levels have also been promoted in recent years. This reflects the fact that individual staff are doing many things very well, something that augurs well for the Department's future.

Professor Aleks Owczarek will take over as Head of Department in January 2011. I wish him all the best in this rewarding position.

60 SECONDS WITH ALISON HARCOURT

Most students in the Department know Alison Harcourt as the lovely tutor who uses her exceptional skills to help them with a broad range of subjects. What many people don't know is that fifty years ago the same Alison (then Doig) co-invented the so called branch-and-bound method. The method is used for finding optimal solutions of various optimization problems and is renowned worldwide for its large contributions to the discipline of Mathematics, mainly in the area of Operations Research.

In 60 seconds we give you a snapshot of this remarkable woman.


1965

The Sun, Thursday, Oct. 21,

omen well

MISS ALISON DOIG, of Melbourne University's Department of Statistics, prepares information for a computer.

How did you come to work at the University?

My first trip to the University was by bus along Studley Park Road. The fare was threepence, which was collected by the conductor. I was going to my first public examination, Intermediate (Year 10) Algebra. Because of the war, the Exhibition Buildings had been taken over by the military, so the exam was held in the First Year Lab in the new Chemistry Building. The exam presented no problems, so I had plenty of time to take in my surroundings, and I knew from that instant where I wanted to work and what I wanted to do.

My original majors were Mathematics and Physics. Fortunately, after a mishap with Physics, I needed to take another subject and I opted for Theory of Statistics I. Instant success, as I had found the subject which has remained my central interest ever since. It was at about that time that the Statistics Department achieved autonomy. We were still based in the south-west corner of the (Old) Arts Building.

What do you like most about your work as a tutor?

Meeting new generations of students. It is pleasing when a student sees the point of a particular aspect of statistics, but for some students the subject is subsidiary to their main interest and they are unprepared to give much thought to its study. Of course, it is our job to arouse that interest, but sometimes it's a long hard slog.

What is something about yourself that most of your colleagues wouldn't know?

My mother's father was born a mere eighteen years after the battle of Waterloo. She was the youngest in her family, and I am the youngest of her four children. So our generations stretch over many years.

Name the top three things you couldn't live without

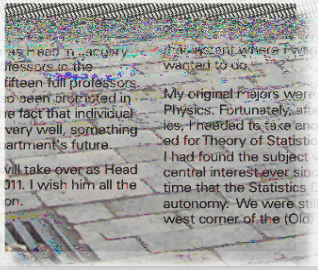
Firstly, the Australian bush and all its wonderful accessories like lyrebirds and echidnas. Second, the music of Bach, Beethoven, Mozart and many others. Third, the belief that St Kilda will win at least one more AFL premiership.

What are you reading at the moment?

I've only just finished reading exam scripts, which is a great relief. I should be reading the latest book from our book club but instead I want to tackle "Such is Life" by Tom Collins (Joseph Furphy); I'm not sure if I have the stamina to finish it. There are three hundred pages plus another one hundred and fifty of annotations!

What do you do to relax?

I've always been pretty good at deep thinking (that is, doing nothing), but other favourite ways of relaxing are Scottish country dancing and rogaining.



60 SECONDS WITH ALISON HARCOURT

Most students in the Department know Alison Harcourt as the lovely tutor who uses her exceptional skills to help them with a broad range of subjects. What many people don't know is that fifty years ago the same Alison (then Doig) co-invented the so called branch-and-bound method. The method is used for finding optimal solutions of various optimization problems and is renowned worldwide for its large contributions to the discipline of Mathematics, mainly in the area of Operations Research.

In 60 seconds we give you a snapshot of this remarkable woman.

1965

The Sun, Thursday, Oct. 21,

omen well

puter all the appropriate information, it can help you decide which drug is more important for a patient."

Miss Doig said that doctors in their own way had used statistics by observation. Later calculating machines had helped.

But only in recent years had it been possible, with computers, to handle a mass of information.

And Miss Doig added: "A computer NEVER makes mistakes.

"A girl working a calculating machine at a desk can question the practicability of a step. But she might make a mistake in handling the information.

"A computer obeys implicitly."

MISS ALISON DOIG, of Melbourne University's Department of Statistics, prepares information for a computer.

Alison Harcourt

Faculty of Science / News / Alison Harcourt named 2019 Victorian Senior Australian of the Year

25 Oct 2018

Alison Harcourt named 2019 Victorian Senior Australian of the Year

Ms Harcourt was honoured for her seminal work in integer linear programming and other contributions to the mathematical and statistical sciences.

Also announced this week was the news that Ms Harcourt will be awarded a honorary Doctor of Science by the University of Melbourne. She was recently profiled in a popular segment on 7.30 on ABC TV.

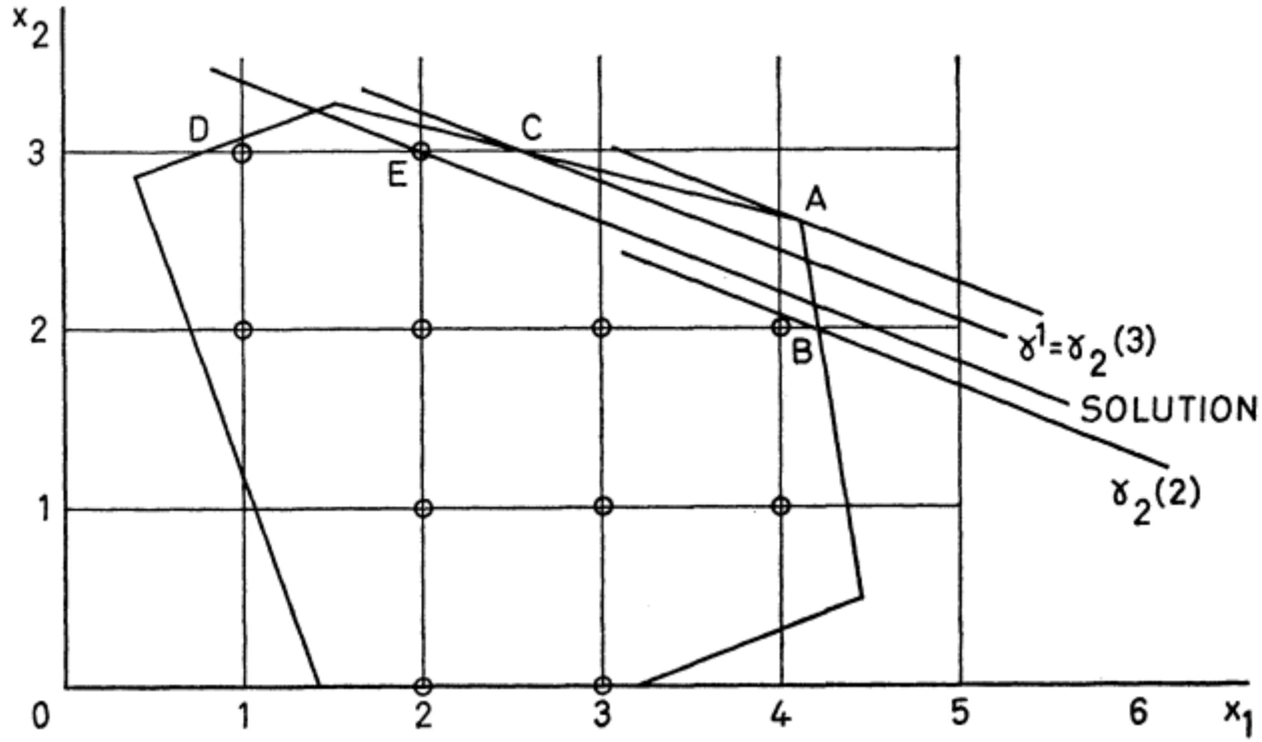


FIGURE 2

Land and Doig, Econometrica, 28, 497-520

Conceitos básicos

Ideia básica – B&B

- Dividir para conquistar!
- Dividir: *branch*
- Conquistar: *bounds*
  - Limitantes:
    - Dual: Relaxação
    - Primal: Solução Factível.

Relaxação

- Ideia básica de vários métodos (incluindo branch-and-bound). Duas condições:

  – O espaço factível da relaxação contém o espaço factível do problema original

  – O valor objetivo da relaxação é *melhor* que o do problema original para cada solução factível possível.

Limitante dual

- Ao se resolver um problema de minimização (maximização), uma relaxação nos fornece um limitante inferior (superior).

Limitante primal

- Ao se resolver um problema de minimização (maximização), uma solução factível nos fornece um limitante superior (inferior).

Partição

- Suppose $\mathcal{S}$ is the feasible set for an MILP and we wish to solve $\max_{x \in \mathcal{S}} c^\top x$.
- Consider a partition of $\mathcal{S}$ into subsets $\mathcal{S}_1, \ldots \mathcal{S}_k$. Then

$$\max_{x \in \mathcal{S}} c^\top x = \max_{\{1 \le i \le k\}} \{\max_{x \in \mathcal{S}_i} c^\top x\}$$

.

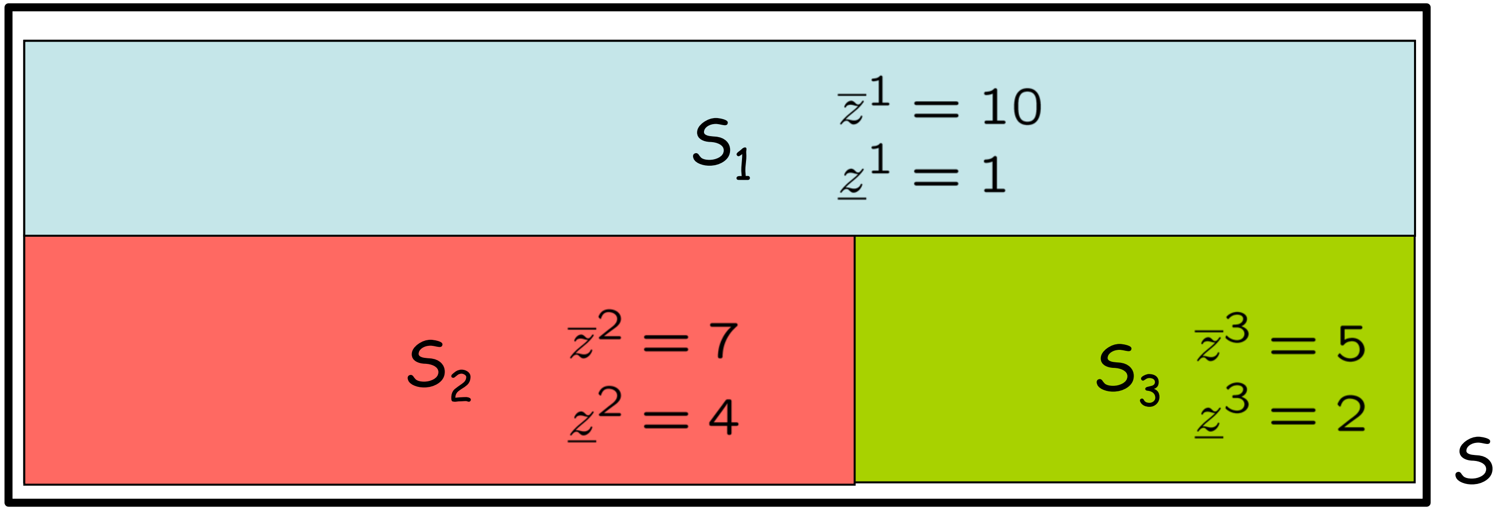
- In other words, we can optimize over each subset separately.
- Idea: If we can't solve the original problem directly, we might be able to solve the smaller *subproblems* recursively.
- Dividing the original problem into subproblems is called *branching*.
- Taken to the extreme, this scheme is equivalent to complete enumeration.

Prof. Ted Ralphs

http://coral.ie.lehigh.edu/~ted/files/ie418/lectures/Lecture5.pdf

Composição de limitantes

Proposition 7.2 *Let $S = S_1 \cup \ldots \cup S_K$ be a decomposition of S into smaller sets, and let $z^k = \max\{cx : x \in S_k\}$ for $k = 1, \ldots, K$, $\overline{z}^k$ be an upper bound on z^k and $\underline{z}^k$ be a lower bound on z^k. Then $\overline{z} = \max_k \overline{z}^k$ is an upper bound on z and $\underline{z} = \max_k \underline{z}^k$ is a lower bound on z.*



$$\overline{z} = 10$$

$$\underline{z} = 4$$

e se fosse *min* ?

Wolsey, L. A., Integer Programming.Wiley-Interscience, 1998

Branch and bound
(básico)

Reorganizando as Ideias...

- Se o poliedro $\bar{P} = \{Ax \leq b, x \in \mathbb{R}^n_+\}$ é fechado, então o número de soluções inteiras factíveis é finito.

- Uma forma de obter uma solução ótima para o problema PI é utilizar um processo exaustivo, a *enumeração completa*.

- Uma forma de reduzir o espaço de busca consiste em utilizar informações do problema PL para executar um procedimento chamado *enumeração implícita*.

- Dividir e Conquistar!

- Retomando o exemplo da enumeração explícita.
- Ideia: não resolver todos os nós-folha, mas eliminar ramos da árvore (com o auxílio de limitantes).

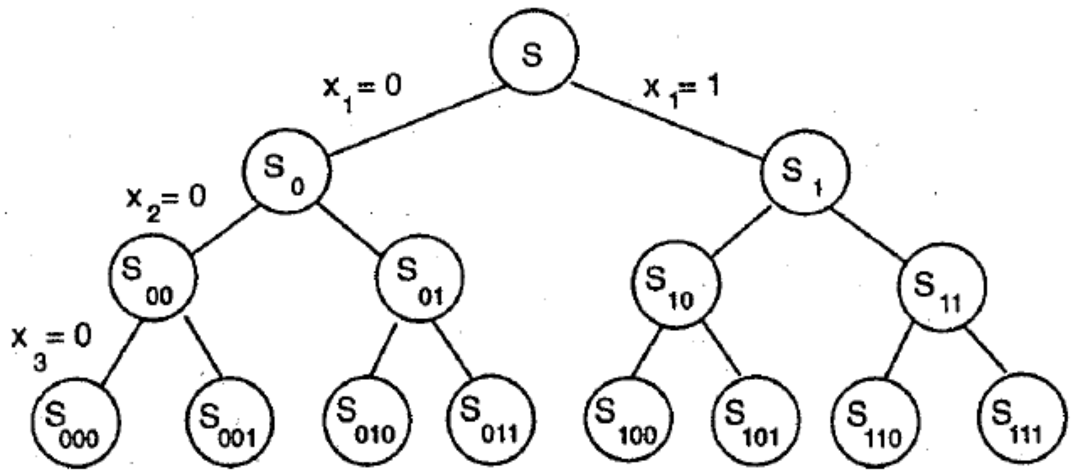


Fig. 7.1 Binary enumeration tree

Wolsey, L. A., Integer Programming.Wiley-Interscience, 1998

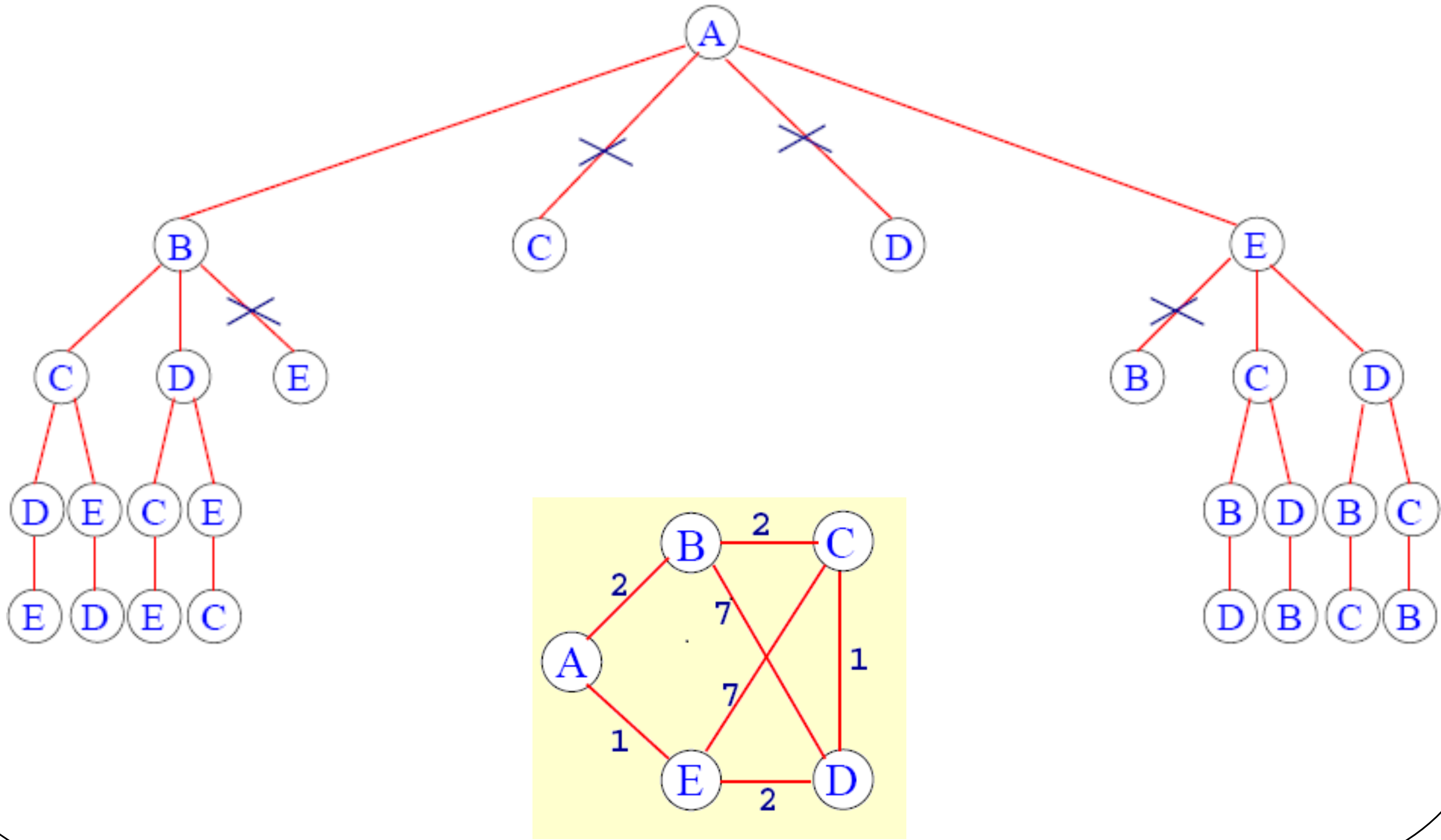
Enumeração Implícita

- Testes para a eliminação de nós:
  - **Eliminação por infactibilidade**
  - **Eliminação por qualidade**
  - **Eliminação por otimalidade**

Enumeração Implícita

- Testes para a eliminação de nós:
  - **Eliminação por infactibilidade:**
    - Se um nó apresenta infactibilidade, seus filhos também apresentarão!
    - O nó deve ser podado por infactibilidade.
  - A árvore de enumeração do Exemplo 2 ficaria...

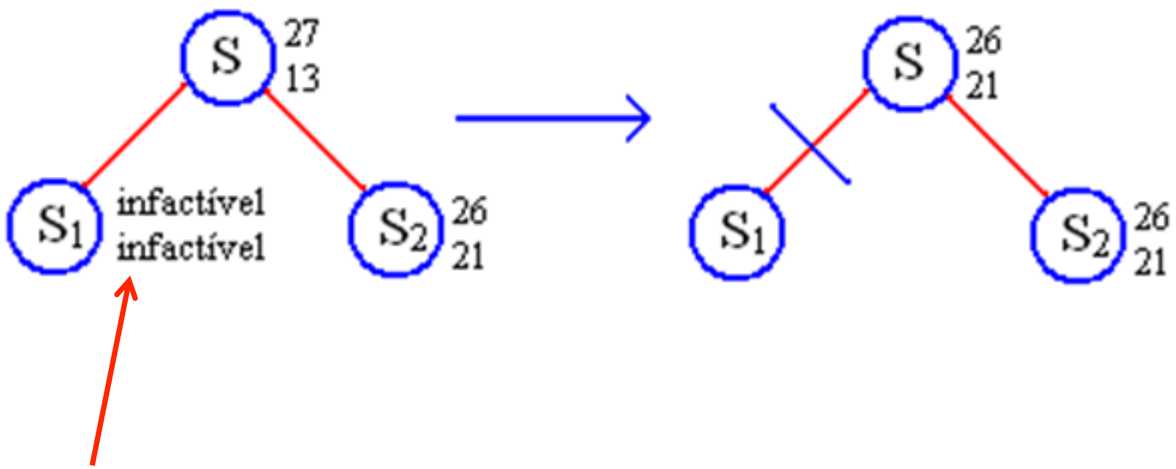
Enumeração Implícita



Enumeração Implícita

- Dado um problema inteiro de maximização:

  - S1 está sondado por infactibilidade:

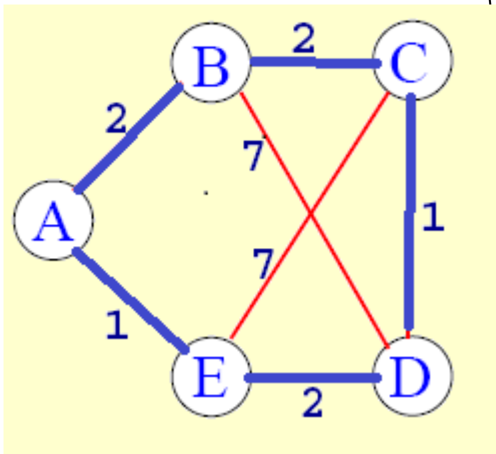


Enumeração Implícita

- Testes para eliminação de nós:

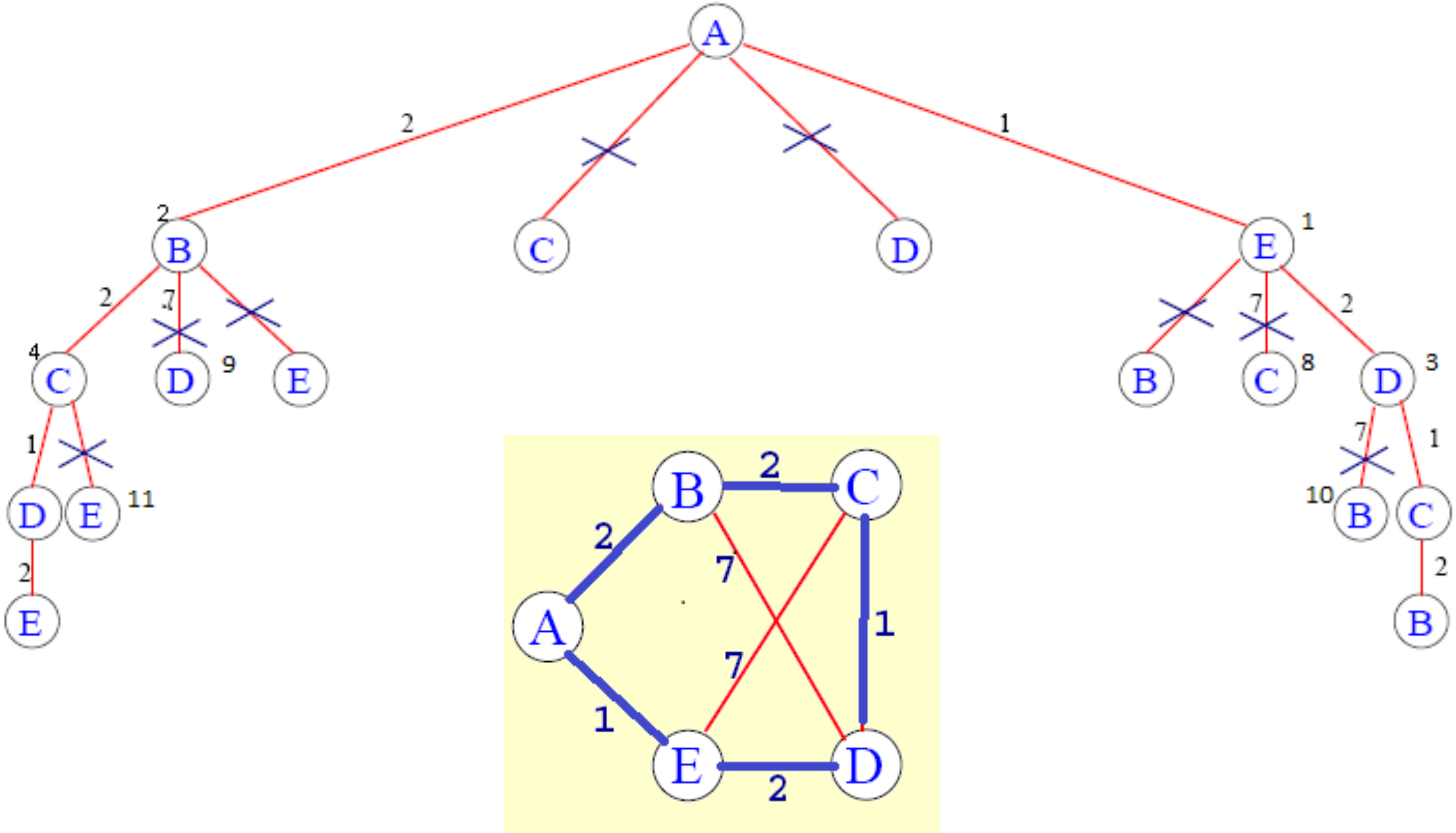
  - **Eliminação por qualidade:**
    - Pode-se comparar o custo das soluções dos sub-problemas;
    - Se a solução de um dado nó for pior que uma solução do problema original obtida previamente, o seu ramo não precisa ser estendido: ele é descartado (podado) por qualidade.

Enumeração Implícita



- **Testes para eliminação de nós:**
  - Vamos incluir eliminação por qualidade no Exemplo 2.
    - Suponha que uma solução foi obtida por um algoritmo guloso, que escolhe o caminho A, B, C, D e E, com valor de solução 8.
    - Se a soma de um percurso parcial for maior ou igual a 8, podamos o ramo.

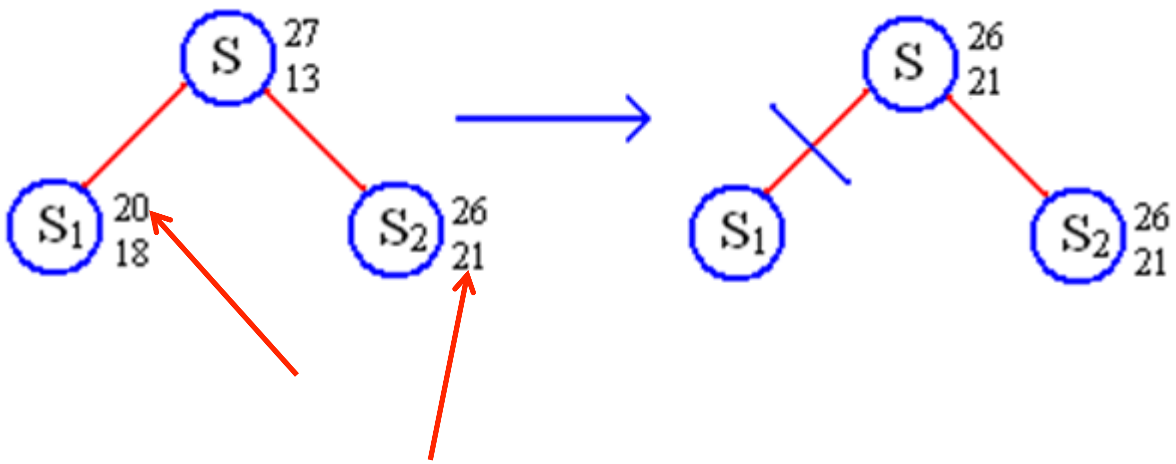
Enumeração Implícita



Estagiário PAE: Guilherme Albuquerque

Enumeração Implícita

- Dado um problema inteiro de maximização:
  - S1 está sondado por qualidade:



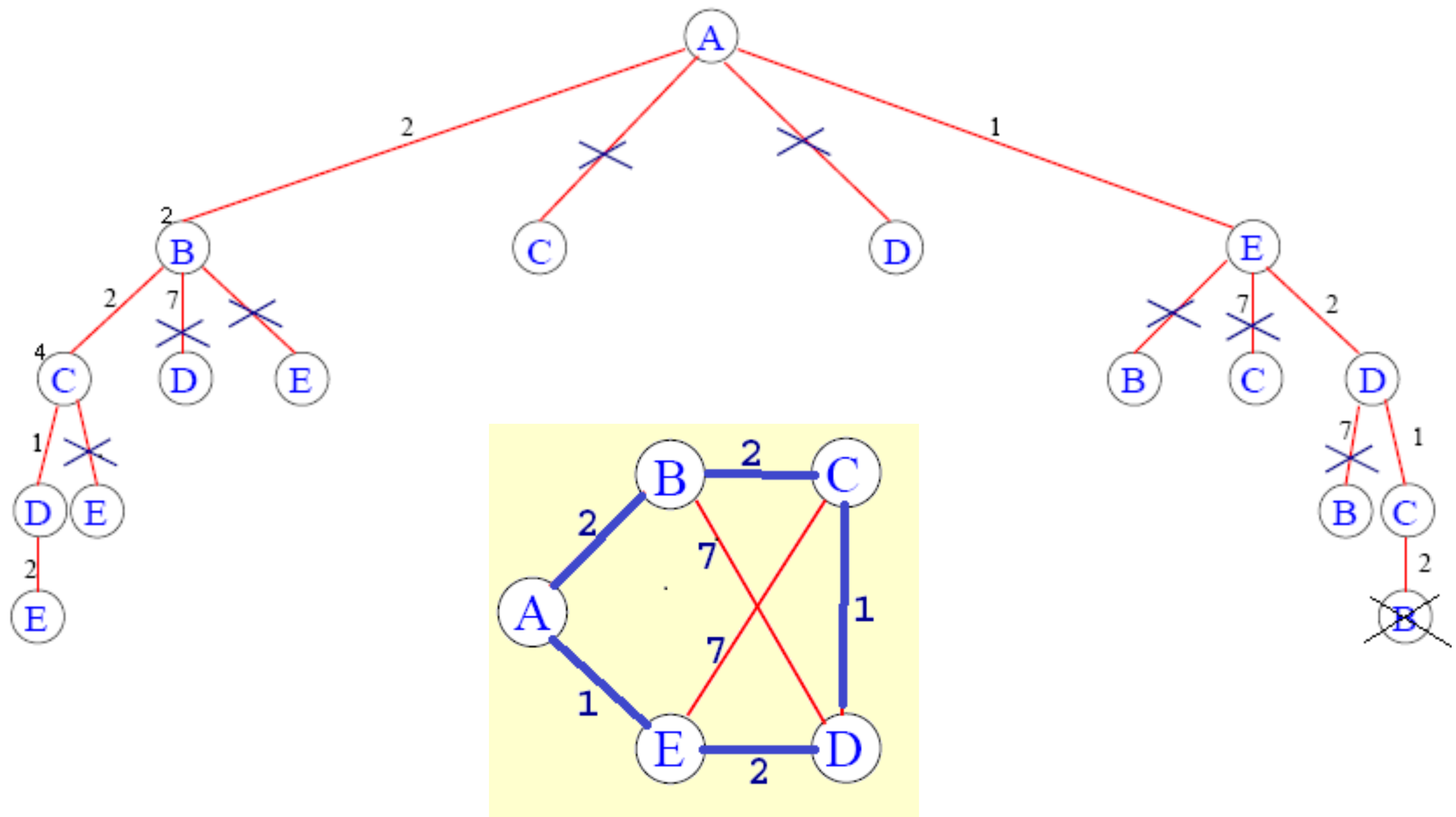
Enumeração Implícita

- Testes para eliminação de nós:

  – **Eliminação por otimalidade:**

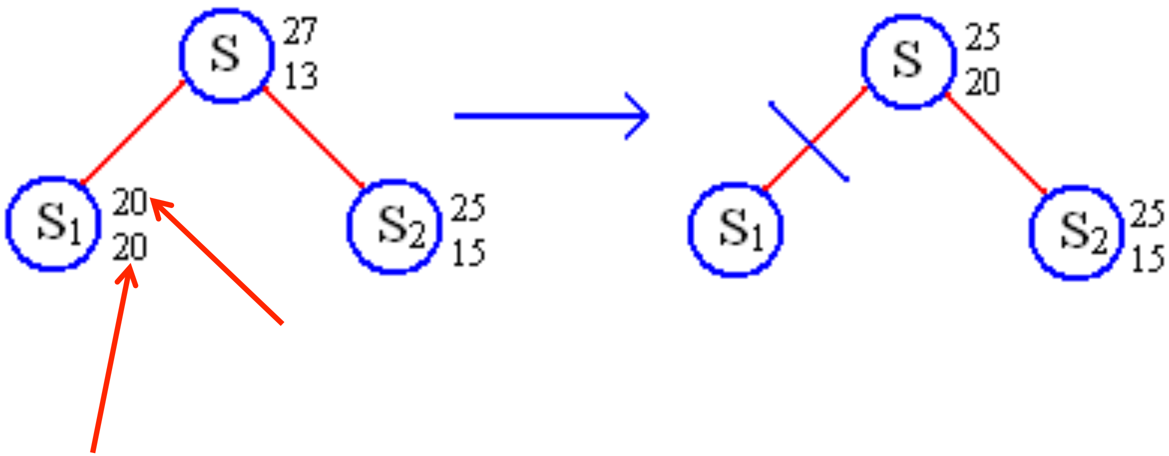
    - Análogo à eliminação por qualidade.
    - Ocorre quando a solução de um sub-problema (pai) é também solução do problema original (é uma solução factível);
    - Já que nenhuma solução filha supera a solução do pai, se em dado ramo da árvore uma solução do problema original foi encontrada, o nó é podado por otimalidade.

Enumeração Implícita



Enumeração Implícita

- Dado um problema inteiro de maximização:
  - S1 está sondado por otimalidade:



Poda (resumo)

- poda por otimalidade:

  $z^t = \max\{ cx : x \in S_t\}$ resolvido.

- poda por limitante:

  $$\overline{z}^t \leq \underline{z}$$

- poda por infactibilidade:

  $S_t = \emptyset$

Wolsey, L. A., Integer Programming.Wiley-Interscience, 1998

Eliminação de nós

Como eliminar nós da árvore *Branch-and-Bound*?

- Se a região factível do PL_i for vazia, então o nó correspondente ao problema inteiro PI_i é eliminado por infactibilidade, pois a região factível do PI está contida na região factível do PL.
- Se o valor ótimo $\bar{z}_i$ do problema PL_i é menor que o valor da melhor solução encontrada até o momento z^*, então PI_i é eliminado por qualidade.
- Se a solução ótima de PL_i é inteira, então PI_i é eliminado por otimalidade.

* Max $c^T x$

Comentários

- A relaxação usada não necessariamente é a linear.

- Frequentemente usa-se a relaxação linear devido, sobretudo, à eficiência computacional dos métodos simplex e de pontos interiores.

Exemplo 1

$$
\begin{aligned}
(P) \quad z = \max\, & 5x_1 - x_2 \\
& 7x_1 - 5x_2 \leq 13 \\
& 3x_1 + 2x_2 \leq 17 \\
& x \in Z_+^2
\end{aligned}
$$

Arenales et al., 2007.

Exemplo 1

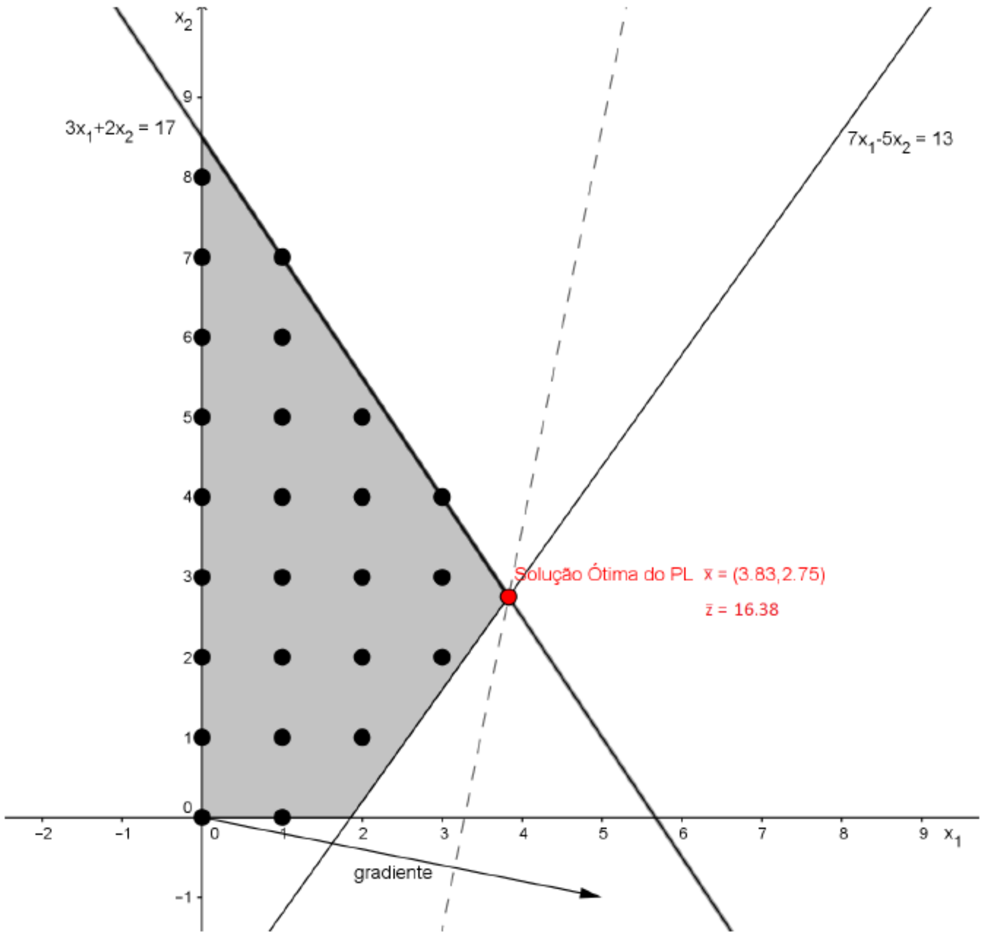


Figura: Representação gráfica das soluções de PI e PL.

Arenales et al., 2007.

Exemplo 1: partição do problema PI

- $\lfloor x \rfloor = max\{y \in \mathbb{Z} : \ y \leq x\}$, o maior inteiro contido em x;
- $\lceil x \rceil = min\{y \in \mathbb{Z} : \ y \geq x\}$, o menor inteiro que contém em x;
- Como tanto x_1 quanto x_2 possuem valores contínuos na solução do PL, podemos escolher qualquer uma das duas para particionar o problema original.

$$
\begin{array}{rlr}
 & & PI_1 \\
z_1 = & max & 5x_1 - x_2 \\
 & s.a. & 7x_1 - 5x_2 \leq 13 \\
 & & 3x_1 + 2x_2 \leq 17 \\
 & & x_1 \leq 3 \\
 & & x_1 \in \mathbb{Z}_+ \\
 & & x_2 \in \mathbb{Z}_+
\end{array}
\qquad
\begin{array}{rlr}
 & & PI_2 \\
z_2 = & max & 5x_1 - x_2 \\
 & s.a. & 7x_1 - 5x_2 \leq 13 \\
 & & 3x_1 + 2x_2 \leq 17 \\
 & & x_1 \geq 4 \\
 & & x_1 \in \mathbb{Z}_+ \\
 & & x_2 \in \mathbb{Z}_+
\end{array}
$$

Essa partição pode ser representada por meio de uma árvore *Branch-and-Bound* em que cada nó corresponde a um problema. Os nós descendentes do nó PI, são chamados de nós filhos (PI_1 e PI_2), e diz-se que a variável x_1 foi ramificada.

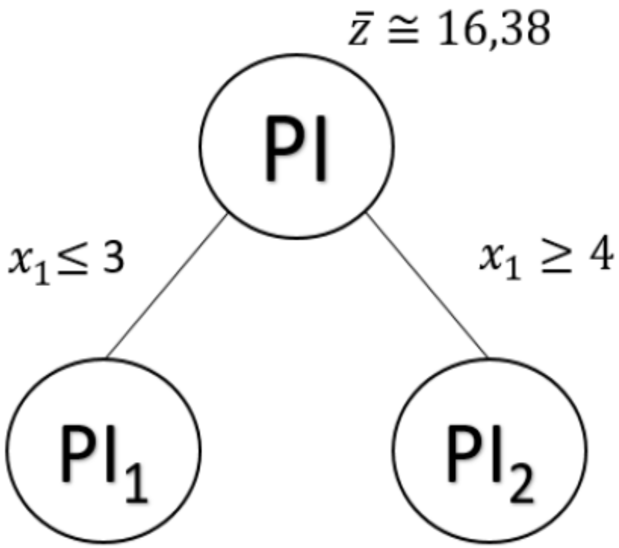


Figura: Partição do problema PI.

Exemplo 1

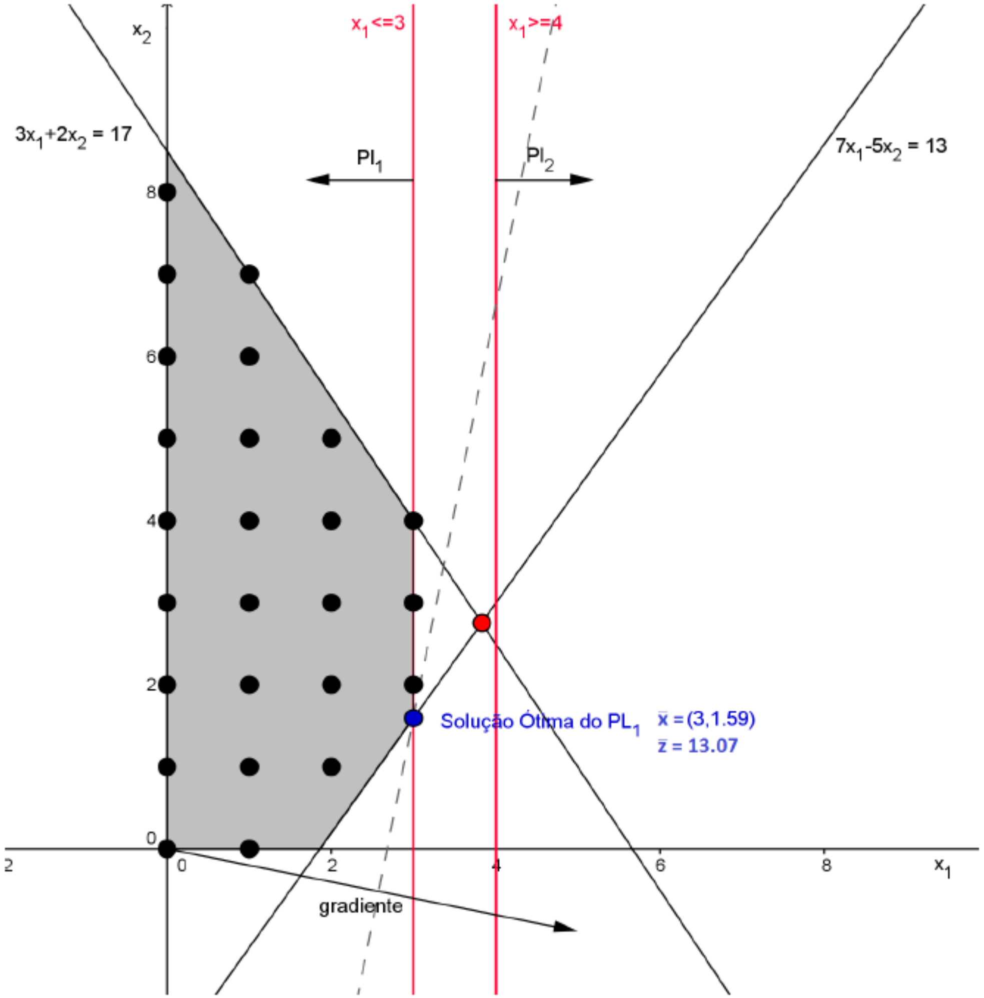


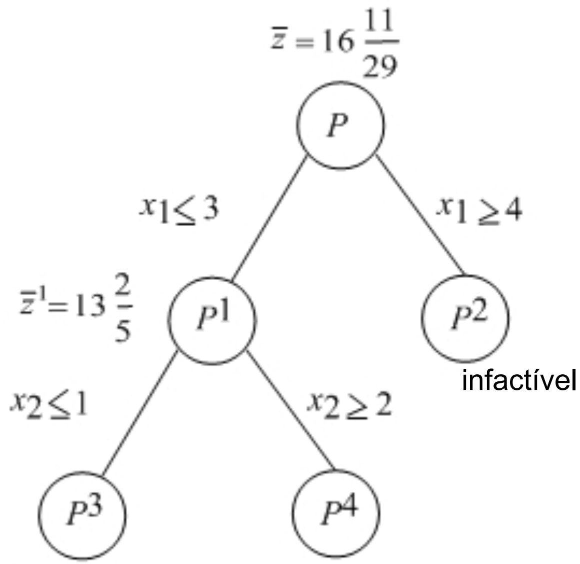
Figura: Representação gráfica das soluções de PI_1 e PI_2.

Arenales et al., 2007.

Exemplo 1

- Podemos dividir (P_1) da seguinte forma:

$$P_3 = P \cap \{x : x_2 \leq \lfloor \bar{x}_2 \rfloor = 1\}$$
$$P_4 = P \cap \{x : x_2 \geq \lceil \bar{x}_2 \rceil = 2\}$$



Arenales et al., 2007.

Exemplo 1

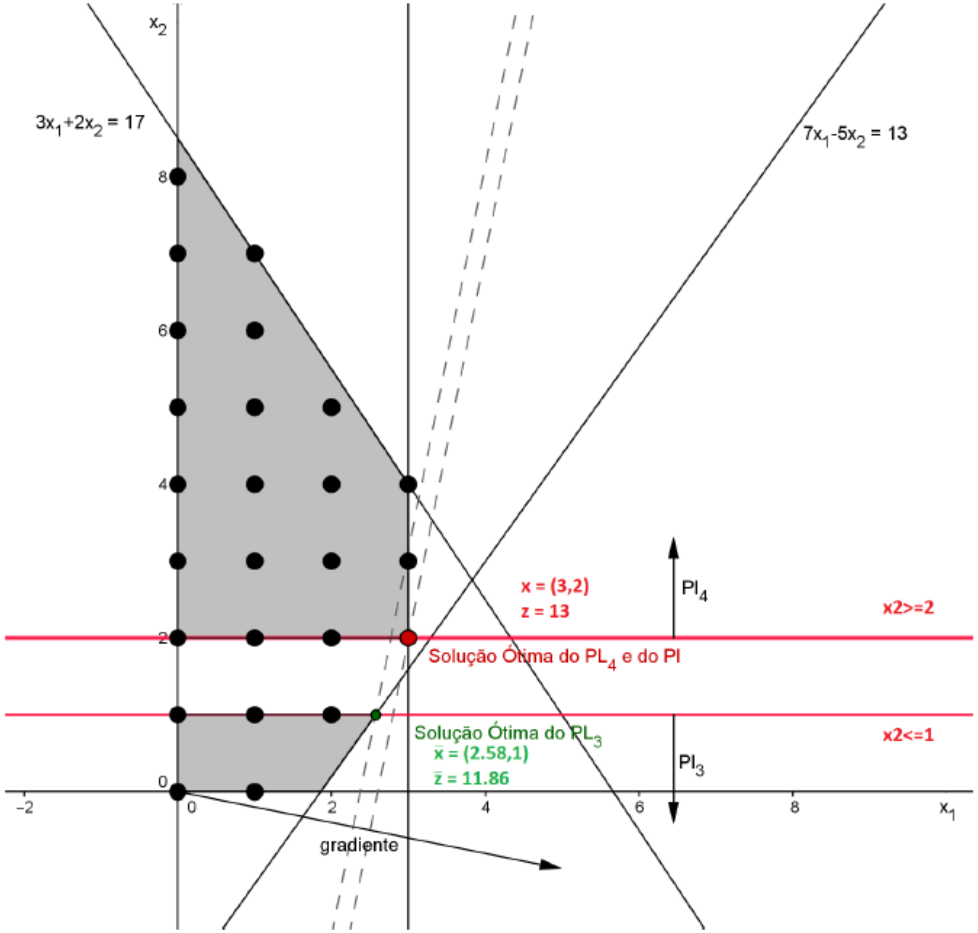
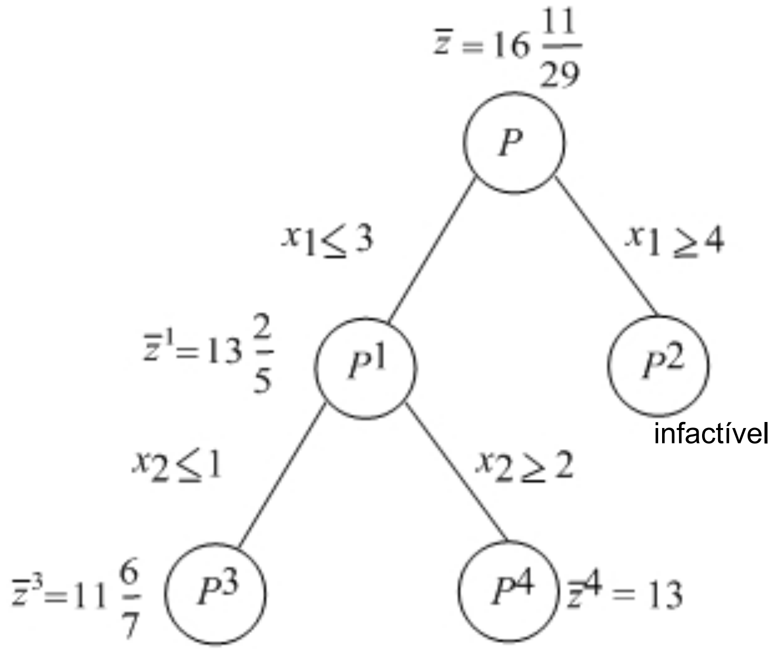


Figura: Representação gráfica das soluções de PI_3 e PI_4.

Arenales et al., 2007.

Exemplo 1

- A solução de (P_4) é a solução ótima:



Arenales et al., 2007.

Exemplo 2

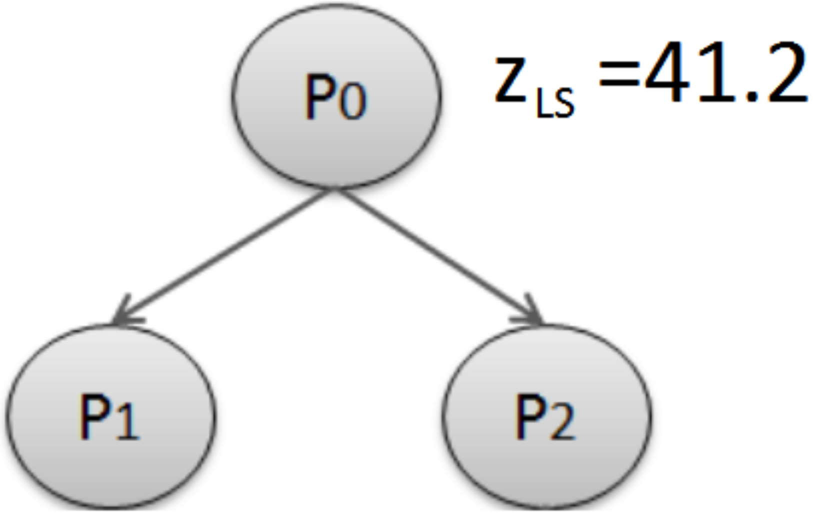
$$\text{Max } 15\,x_1 + 11{,}2\,x_2 + 17{,}5\,x_3 + 4\,x_4 + 3\,x_5 + 4{,}8\,x_6$$

$$\text{s.a. } 5\,x_1 + 4\,x_2 + 7\,x_3 + 2\,x_4 + 3\,x_5 + 6\,x_6 \le 15$$

x_i binária

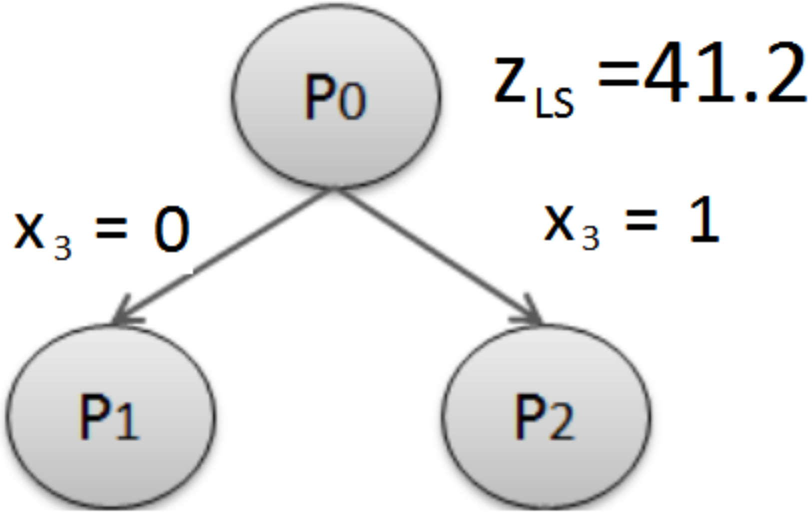
Item	Valor	Peso	Valor/Peso
1	15,0	5	3,0
2	11,2	4	2,8
3	17,5	7	2,5
4	4,0	2	2,0
5	3,0	3	1,0
6	4,8	6	0,8

Exemplo 2



Sol. Relaxada: x1 = x2 = 1 e x3 = 6/7
x4 = x5 = x6 = 0

Exemplo 2



Sol. Relaxada: x1 = x2 = 1 e x3 = 6/7
x4 = x5 = x6 = 0

Slides do curso “Optimization methods in management sciente” – J. Orlin et al., MIT

Exemplo 2

- Dificuldade:

  A árvore gerada se aproxima da árvore da enumeração explícita.

Algoritmo *Branch-and-Bound*

Considere

- z_i: valor ótimo do problema PI_i;
- $\bar{z}_i$: valor ótimo do problema PL_i;
- x^*: melhor solução inteira encontrada até o momento;
- z^*: valor da função objetivo em x^*;

O nó 0 da árvore *Branch-and-Bound* corresponde ao problema original PI. Um nó não eliminado e que ainda não foi ramificado é chamado de nó ativo. Os nós ativos são armazenados em uma lista L.

Algoritmo

Passo 0 (Inicialização): Faça $\bar{z} = \infty$, $z^* = -\infty$, $x^* = \varnothing$, $L = \{PI\}$;

Passo 1 (Seleção de nó): Selecione o nó ativo i, associado ao problema inteiro PI_i, da lista de nós ativos. Se a lista L estiver vazia, vá para o **Passo 6**;

Passo 2 (Teste de eliminação 1): Se a região factível de PL_i for vazia, elimine PI_i da lista e volte ao **Passo 1**;

Passo 3 (Teste de eliminação 2): Se o valor de $\bar{z}_i$ da solução ótima de PL_i é tal que $\bar{z}_i \leq z^*$, elimine PI_i da lista e volte ao **Passo 1**;

Passo 4 (Teste de eliminação 3): Se a solução ótima $\bar{x}_i$ de PL_i é inteira com valor $\bar{z}_i$, e se $\bar{z}_i > z^*$, atualize x^* e z^*. Elimine Pi_i da lista e todos os nós ativos $i's$ tais que $\bar{z}_i \leq z^*$ e volte ao **Passo 1**;

Passo 5 (Ramificação): Selecione uma variável da solução ótima $\bar{x}_i$ de PL_i com valor não inteiro e divida PI_i em dois subproblemas. Adicione estes problemas a lista L e volte ao **Passo 1**;

Passo 6 (Fim): Se $z^* = -\infty$, não existe solução factível; caso contrário, a solução incumbente x^* é uma solução ótima.