620-362
Applied Operations Research

Branch & Bound

Department of Mathematics and Statistics
The University of Melbourne

Some contents of this presentation are adapted from year 2005 course notes for 620-362 Applied Operations Research, Department of Mathematics and Statistics, The University of Melbourne (compiled by Prof Natashia Boland and Dr Renata Sotirov)

Branch and Bound (B&B)

…method compute the optimal solution to IP, MIP and COP by *enumerating the points* in a subproblem's feasible region.

Recall:

- combinatorial optimisation problem (COP) is any optimisation problem that has a finite number of feasible solutions.

- integer programming problem (IP) is an optimisation problem in which unknown variables are all required to be integers.

- mixed integer programming problem (MIP) is an optimisation problem in which only some of the variables are required to be integers.

"Divide and Conquer"

- B&B is a divide and conquer approach

On branching…

- suppose S is the feasible region for some MILP and we wish to solve:

$$\min_{x \in S} c^T x$$

- let $S = S_1 \cup \ldots \cup S_k$, then

$$\min_{x \in S} c^T x = \min_{1 \le i \le k} \left(\min_{x \in S_i} c^T x \right)$$

  i.e. we can optimise over each subset separately.

- dividing the original problem into subproblems is called **branching**.
- taken to the extreme, this scheme is equivalent to complete enumeration.
- the complete enumeration is impossible for most problems as soon as the number of variables in an integer program exceeds 20 or 30 (!)

"Divide and Conquer"

Example: Enumeration tree for $S \subseteq \{0,1\}^3$

1. Divide S into

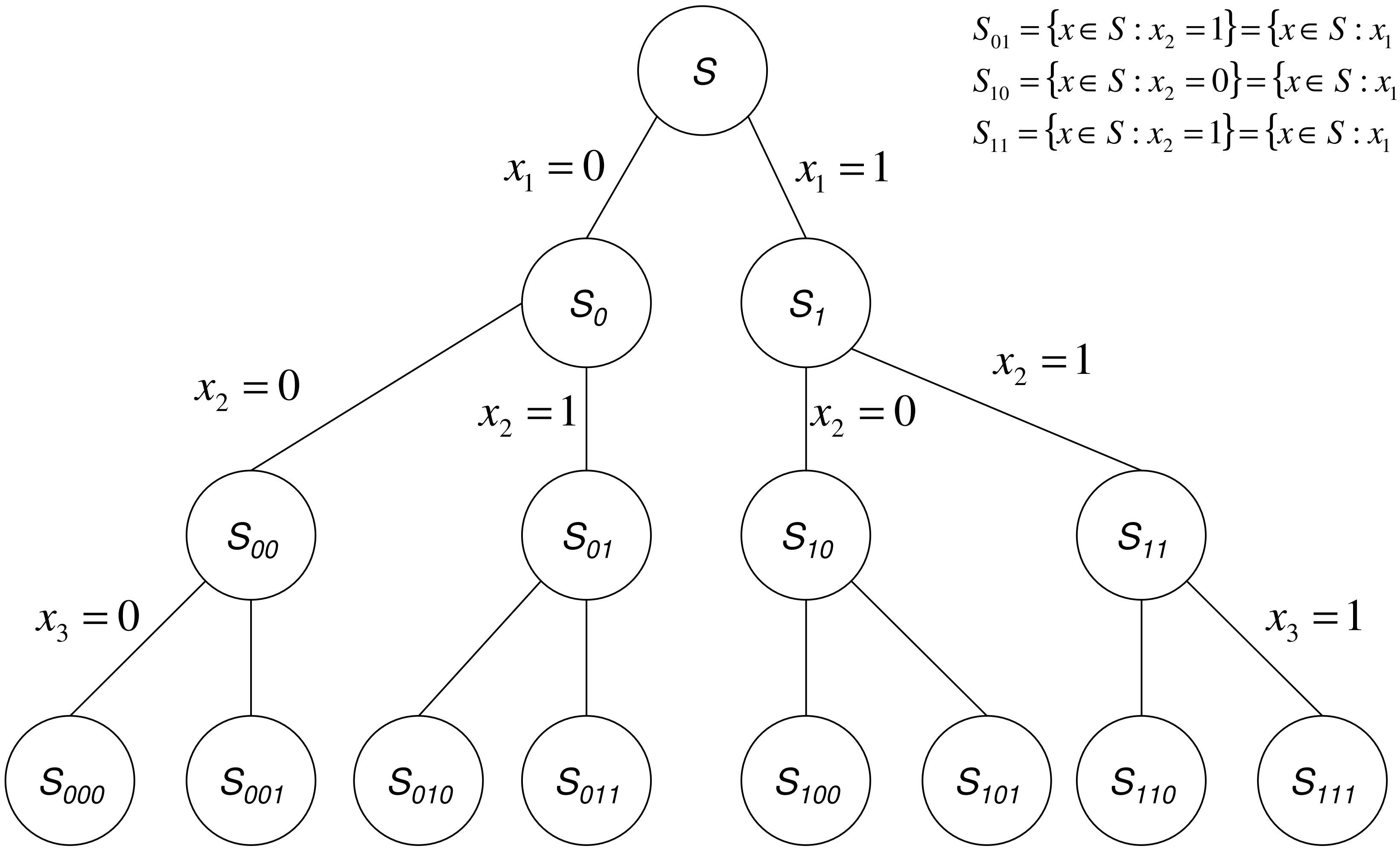
$$S_0 = \{x \in S : x_1 = 0\} \text{and } S_1 = \{x \in S : x_1 = 1\}$$

2. Then

$$S_{00} = \{x \in S : x_2 = 0\} = \{x \in S : x_1 = x_2 = 0\}$$
$$S_{01} = \{x \in S : x_2 = 1\} = \{x \in S : x_1 = 0, x_2 = 1\}$$
$$S_{10} = \{x \in S : x_2 = 0\} = \{x \in S : x_1 = 1, x_2 = 0\}$$
$$S_{11} = \{x \in S : x_2 = 1\} = \{x \in S : x_1 = x_2 = 1\}$$

and so on…

“Divide and Conquer”



$$S_{00} = \{x \in S : x_2 = 0\} = \{x \in S : x_1 = x_2 = 0\}$$
$$S_{01} = \{x \in S : x_2 = 1\} = \{x \in S : x_1 = 0, x_2 = 1\}$$
$$S_{10} = \{x \in S : x_2 = 0\} = \{x \in S : x_1 = 1, x_2 = 0\}$$
$$S_{11} = \{x \in S : x_2 = 1\} = \{x \in S : x_1 = x_2 = 1\}$$

Terminology

- if we picture the subproblems graphically, then we form a **search tree**
- each subproblem is linked to its **parent** and eventually to its **children**
- eliminating a problem from further consideration is called **pruning** or **fathoming**
- the act of bounding and then branching is called **processing**
- a subproblem that has not yet been considered is called a **candidate** for processing
- the set of candidates for processing is called the **candidate list**
- going back on the path from a node to its root is called **backtracking**

On bounding…

successors of a node
(children, grandchildren, etc.)
have more constrained feasible regions

$\Downarrow$

relaxations at successors nodes have higher (minimizing)/lower (maximizing) values

$\Downarrow$

searching the successors of a node cannot yield an integer feasible solution with value better than the relaxation value at the node

$\Downarrow$

if the relaxation value at a node is worse than the value of some integer solution we have already found, then we can stop exploring that node

$\Downarrow$

fathoming the node

LP-based Branch-and-Bound

Binary IP:

$$\begin{aligned} &\min cx \\ &\text{s.t. } Ax \geq b \\ &\quad x \in \{0,1\}^n \end{aligned}$$

LP relaxation:

$$\begin{aligned} &\min cx \\ &\text{s.t. } Ax \geq b \\ &\quad 0 \leq x \leq 1 \end{aligned}$$

Branching: LP solution x^* has $x^*_i \in (0,1)$

$x_i = 0$ $\qquad$ $x_i = 1$

LP-based Branch-and-Bound

IP:

$$\begin{aligned} &\min\ cx \\ &\text{s.t. } Ax \geq b \\ &\qquad x \in \mathbb{Z}_+^n \end{aligned}$$

LP relaxation:

$$\begin{aligned} &\min\ cx \\ &\text{s.t. } Ax \geq b \\ &\qquad x \geq 0 \end{aligned}$$

Branching: LP solution x^* has $x^*_i \in (a, a+1)$, $a \in \mathbb{Z}$

$$x_i \leq a \qquad\qquad x_i \geq a+1$$

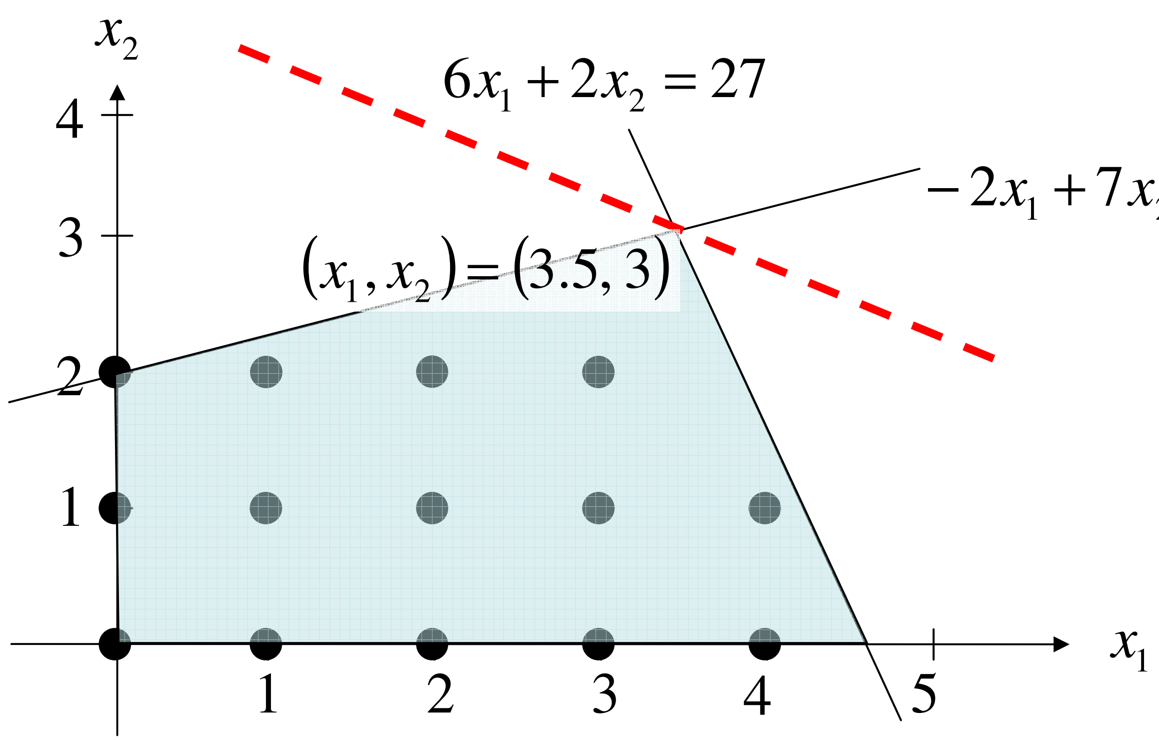
LP-based Branch-and-Bound

Example 1

$$\max\ z = x_1 + 2x_2$$
$$\text{s.t.} -2x_1 + 7x_2 \le 14$$
$$6x_1 + 2x_2 \le 27$$
$$x_1, x_2 \text{ integer}$$

Best objective = $NONE$

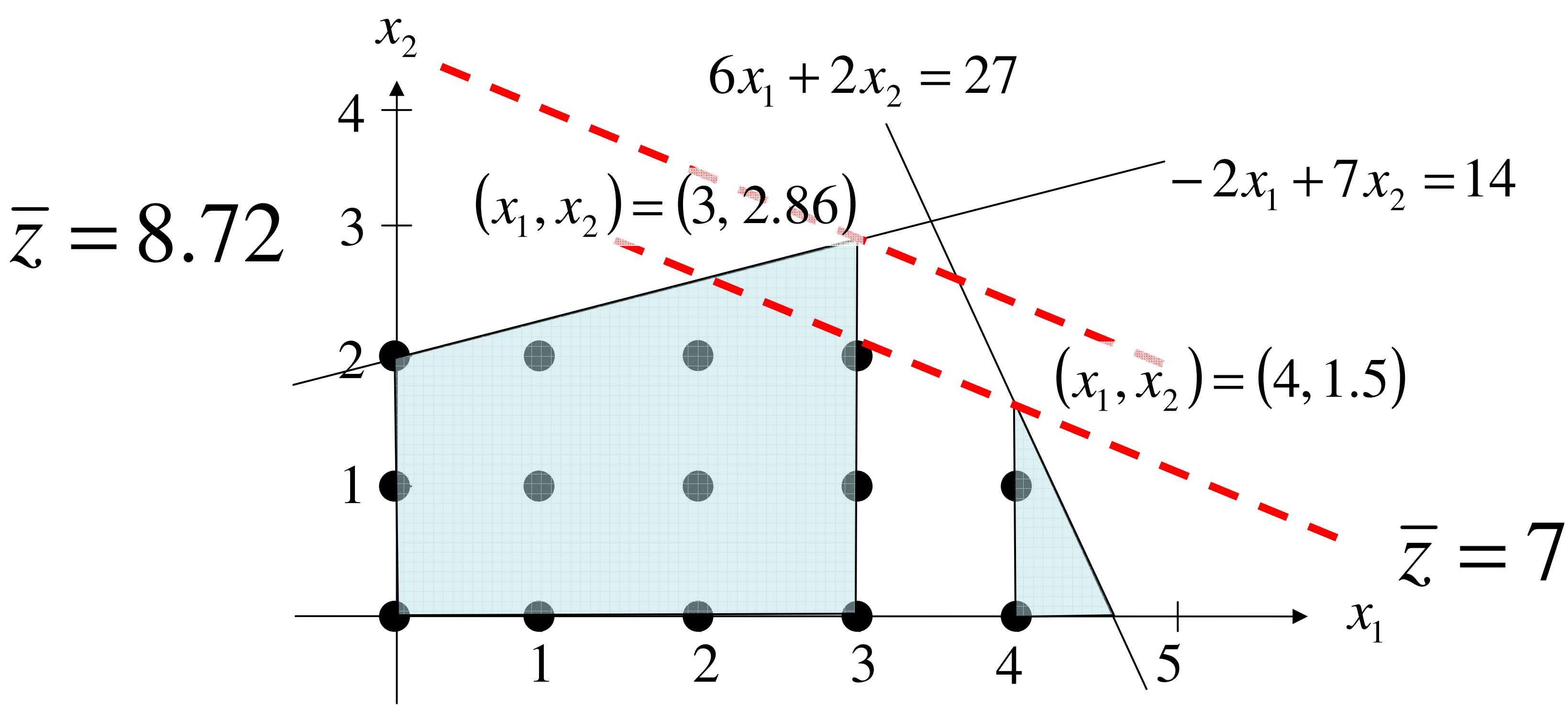
$\bar{z} = 9.5$



LP-based Branch-and-Bound

$$\max\ z = x_1 + 2x_2$$
$$\text{s.t.} -2x_1 + 7x_2 \le 14$$
$$6x_1 + 2x_2 \le 27$$
$$x_1, x_2 \text{ integer}$$

Best objective = *NONE*

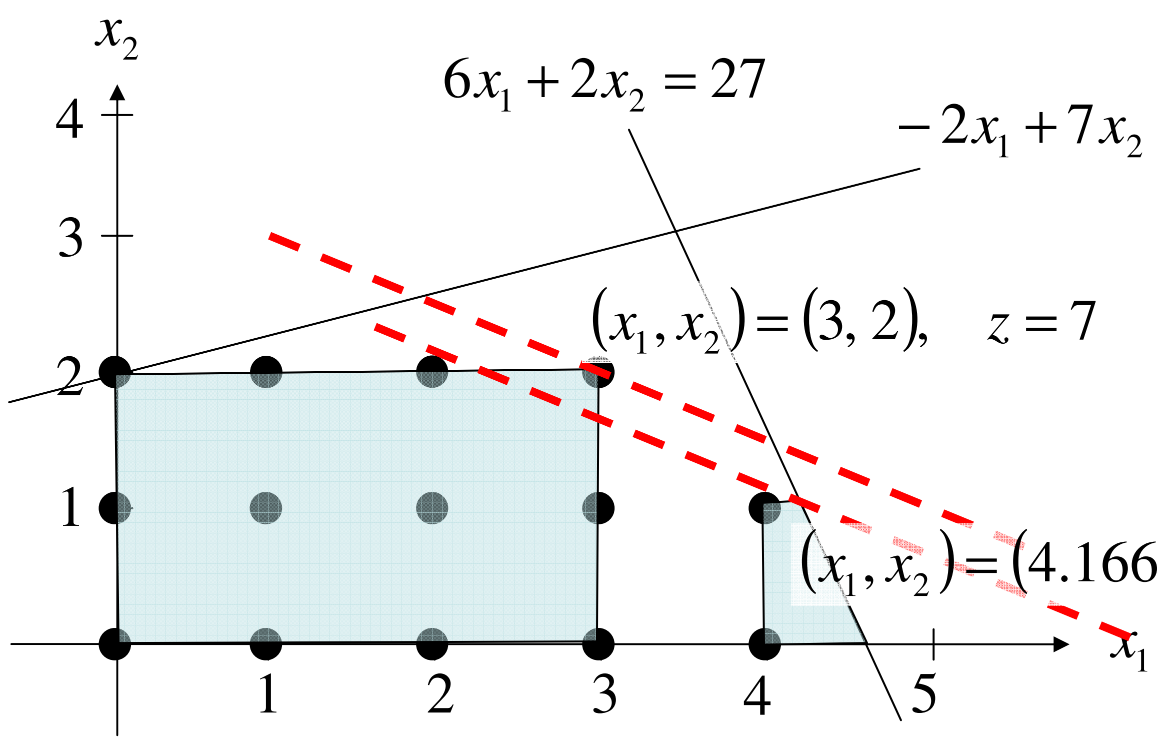


LP-based Branch-and-Bound

$$\max\ z = x_1 + 2x_2$$
$$\text{s.t.} -2x_1 + 7x_2 \le 14$$
$$6x_1 + 2x_2 \le 27$$
$$x_1, x_2 \text{ integer}$$

Best objective = 7

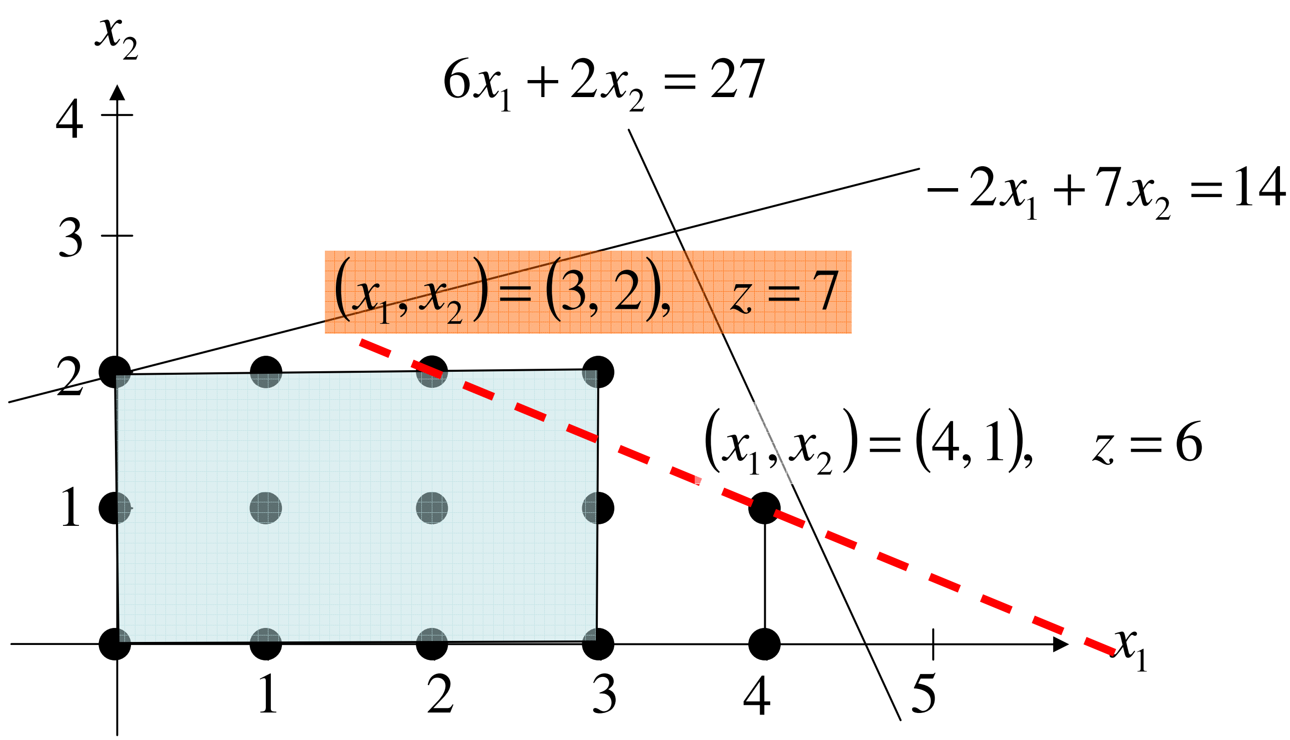
$\bar{z} = 8.72$



LP-based Branch-and-Bound

$$\max\ z = x_1 + 2x_2$$
$$\text{s.t.} -2x_1 + 7x_2 \le 14$$
$$6x_1 + 2x_2 \le 27$$
$$x_1, x_2 \text{ integer}$$

Best objective $= 7$



How to choose a node?

Depth first search
(also known as last in, first out - LIFO):

Rule: if the current node is not pruned, the next node considered is one of its two children

- note that it is always easy to resolve the LP relaxation when simple constraint is added and the *optimal basis available*
- experience indicate that feasible solutions are more likely to be found *deep in the tree* than at nodes near the root
- good for feasibility
- efficiency *is bad* for deep trees (can always place a bound on the search depth)

How to choose a node?

Breadth first search:

Rule: all of the nodes at given level are considered before any nodes at the next lower level

– this node selection is *not practical* for solving general IP using LP relaxations, but it has interesting properties that are used in *heuristics*.

How to choose a node?

Best bound search:

Rule: choose the node with the best bound

- *continuous improvement* of global bound (upper bound if maximising, lower bound if minimising)

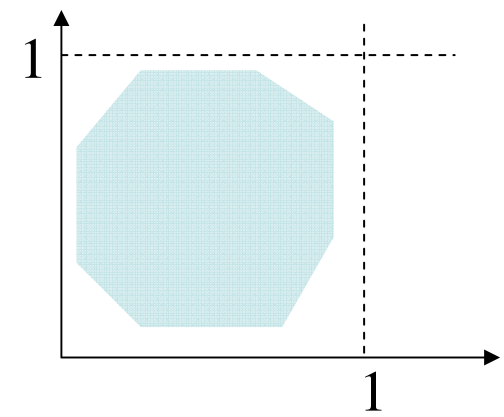
Good strategy in B&B tree:

Depth-first until an initial integer feasible solution, then switch to best-bound search

Infeasibility

It is possible for an IP to be infeasible whilst its LP relaxation is feasible and has a solution (!)

If IP infeasible, fathoming is not possible.

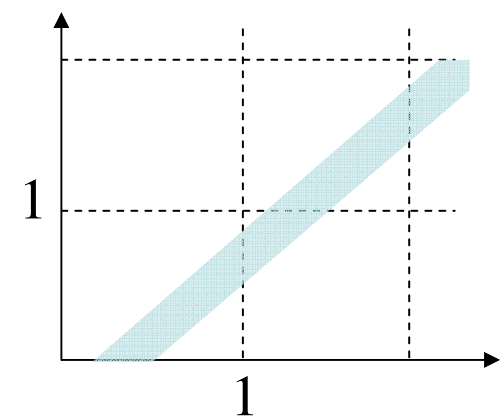


Case 1: LP feasible region **bounded**.

- B&B will generate the entire tree whose leaves are all infeasible LPs

Case 2: LP feasible region **unbounded**.

- B&B may produce an infinite number of nodes and will never detect the fact that the original IP is infeasible



Branch-and-Bound Algorithm

$$
\begin{array}{lll}
(IP) & \max & z = cx \\
 & s.t. & Ax \le b \\
 & & x \ \text{integer}
\end{array}
$$

Notation:

IP^i: integer program at node i of the branch and bound tree

LP^i: LP relaxation of IP^i

L: set of unexplored nodes

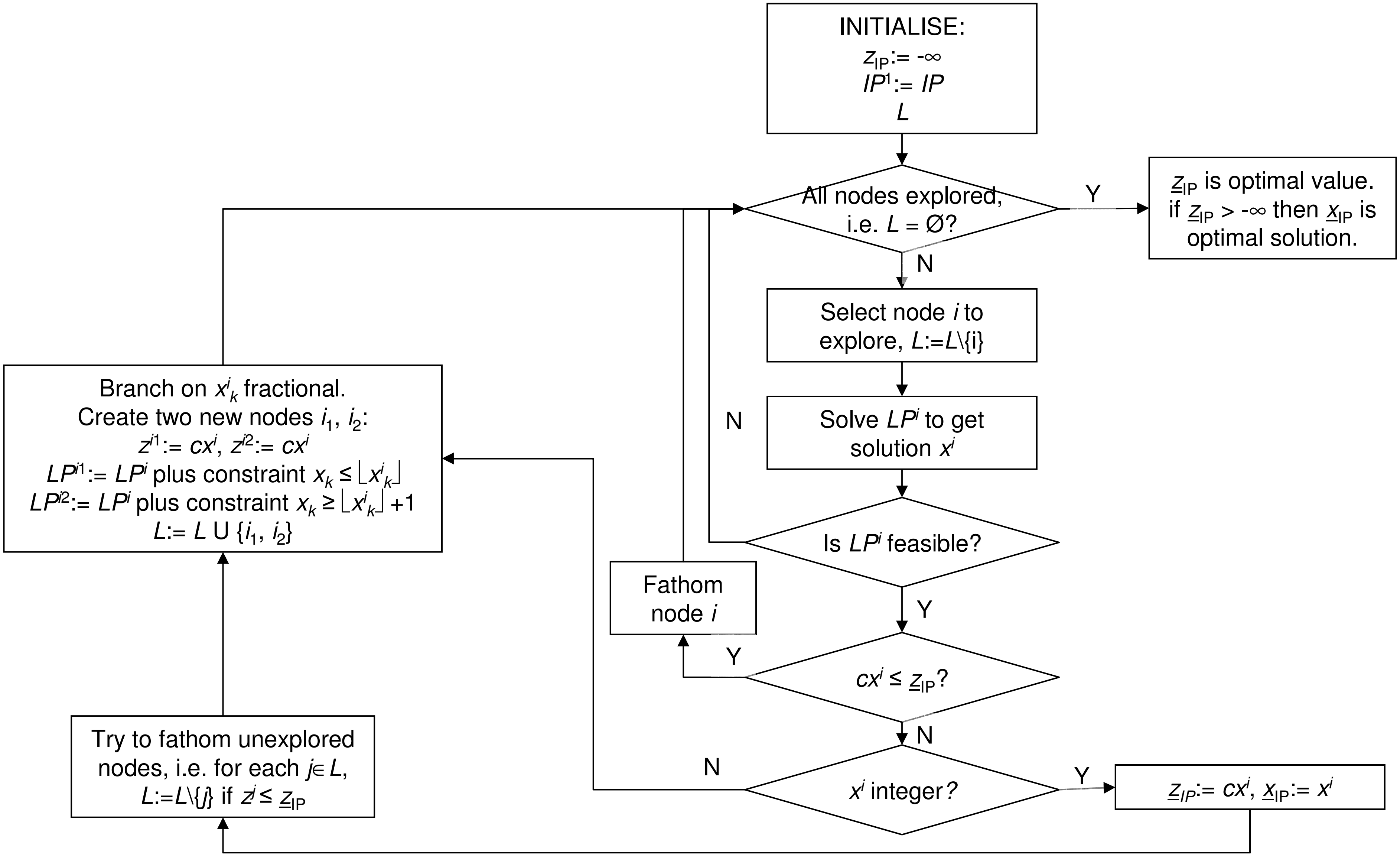
x^i: optimal solution to LP^i

z^i: value of upper bound on node i

$\underline{z}_{IP}$: best lower bound (best solution)

$\underline{x}_{IP}$: integer feasible solution associated with best lower bound

Branch-and-Bound Algorithm (Max. prob.)



Branch-and-Bound Algorithm

The basic philosophy of the B&B method is:

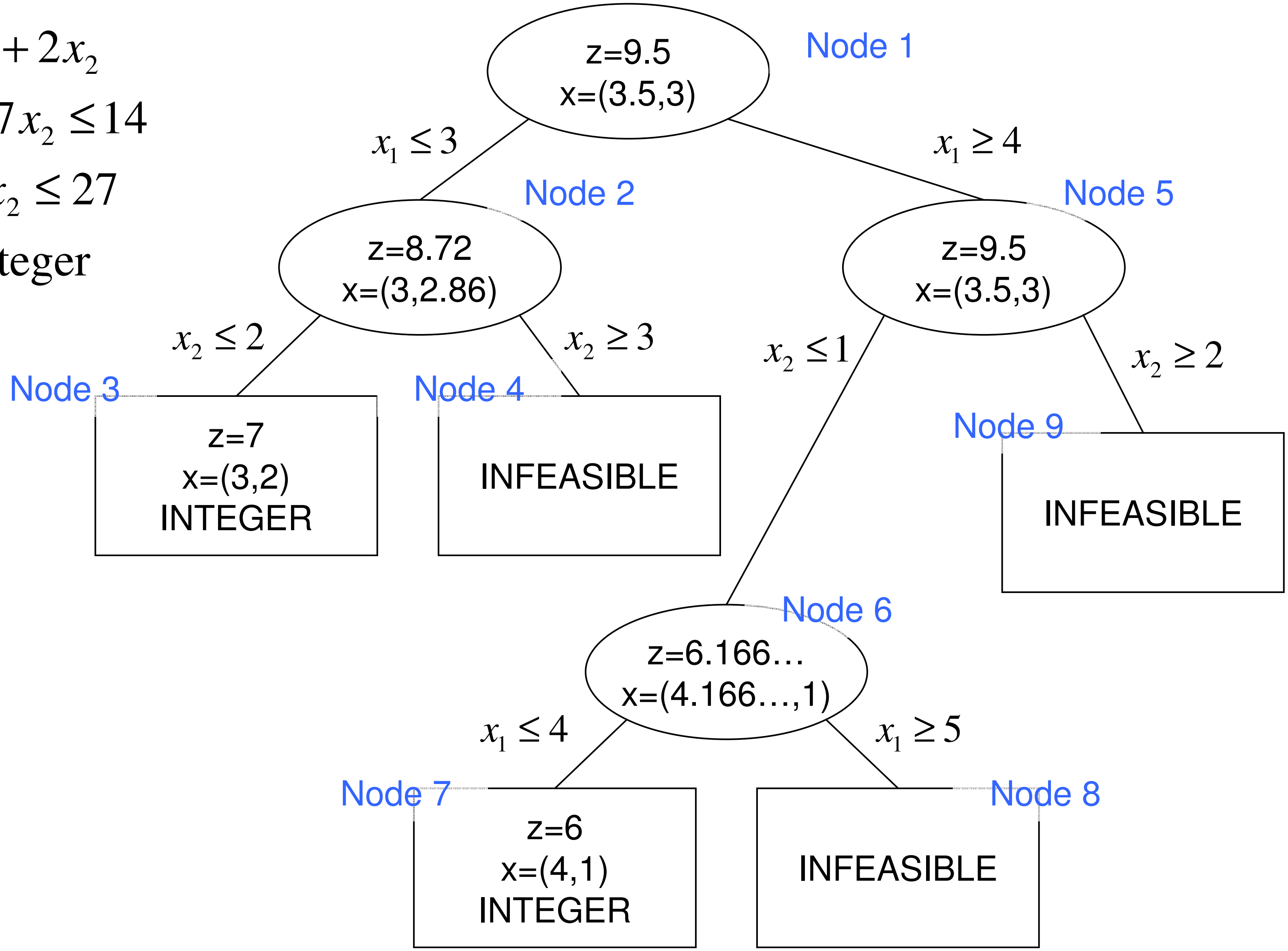
- to solve and resolve the linear programming relaxations as rapidly as possible
- to branch intelligently

B&B solvers also apply:

- preprocessing (e.g. root,in B&B tree)
- primal heuristic (e.g. diving heuristics, local branching)
- branching schemes (which variable to branch on first e.g. dichotomy, priority – user specified)
- different ways of solving the LP relaxation (e.g. primal simplex, dual simplex, subgradient method)
- alternative bounds to the LP relaxation bound (e.g. Lagrangian, heuristics)
- constraint generation (Gomory, Lift-and-project)
- special implementation (branch-and-price)

Example 1: B&B Tree

$$\max\ z = x_1 + 2x_2$$
$$\text{s.t.} -2x_1 + 7x_2 \le 14$$
$$6x_1 + 2x_2 \le 27$$
$$x_1, x_2 \text{ integer}$$



Further reading…

Winston Chapter 9

(9.3, 9.4, 9.5)