Branch and Bound Algorithm

A Introduction

The Branch and Bound (B&B) algorithm is a systematic approach used to solve optimization problems, particularly in cases where an exhaustive search is impractical due to the problem's size or complexity. It is widely employed for solving combinatorial optimization problems, including integer programming (IP) and mixed-integer programming (MIP) problems.

Basic Principles:

- **Exploration of Solution Space**: The algorithm systematically explores the solution space by dividing it into smaller subspaces (branches).
- **Bounding**: It uses bounds (upper and lower bounds) to prune the search space efficiently, discarding branches that cannot possibly contain an optimal solution.

Steps in the Branch and Bound Algorithm:

1. **Initialization**: Begin with an initial relaxation of the problem (e.g., solving the LP relaxation for an IP problem).
2. **Branching**: If the relaxation does not yield an integer solution, select a variable to branch on:
    - **Binary Variables**: Create two subproblems by fixing the variable to 0 in one subproblem and 1 in another.
    - **Integer Variables**: Depending on the variable's domain, explore different discrete values or fractional parts.
3. **Bounding**: After branching, solve each subproblem (often using LP relaxation):
    - Compute bounds (lower and upper bounds) for each node in the search tree based on the solutions found so far.
    - Update the best-known solution if an integer feasible solution is found that improves upon the current best.
4. **Pruning**: Use bounds to prune parts of the search tree that cannot contain an optimal solution:
    - **Optimality Cuts**: If the lower bound of a node exceeds the current best solution, prune that branch.
    - **Feasibility Cuts**: If a feasible solution is found in a node, use it to eliminate parts of the search space where better solutions cannot exist.
5. **Termination**: The algorithm terminates when all branches of the search tree have been explored or pruned, and no further improvements to the best solution can be made.

Advantages of Branch and Bound:

- **Optimality**: Guarantees finding the globally optimal solution (if feasible) for problems where all constraints and the objective function are linear or convex.
- **Versatility**: Applicable to a wide range of optimization problems, including discrete, combinatorial, and mixed-integer problems.
- **Efficiency**: Efficiently narrows down the search space using bounds and pruning techniques, making it feasible to solve large-scale problems.

Limitations and Considerations:

- **Computational Complexity**: The algorithm's performance heavily depends on the problem's structure and the quality of the bounds used.
- **Heuristics and Enhancements**: Practical implementations often incorporate heuristic methods, preprocessing steps, and specialized techniques (like cutting planes in IP/MIP) to improve efficiency and effectiveness.
- **Solver Dependency**: Different implementations and solvers may yield different results and efficiencies due to variations in algorithmic choices and heuristics.

B The Branch-and-Bound Algorithm

Consider the following IP problem:

$$\begin{cases} Max & 10x_1 & + & 15x_2 & = & z \\ s.t. & 8x_1 & + & 4x_2 & \leq & 40 \\ & 15x_1 & + & 30x_2 & \leq & 200 \\ & x_1 \geq 0,\ x_2 \geq 0 \\ & x_1 \text{ and } x_2 \text{ integer} \end{cases}$$

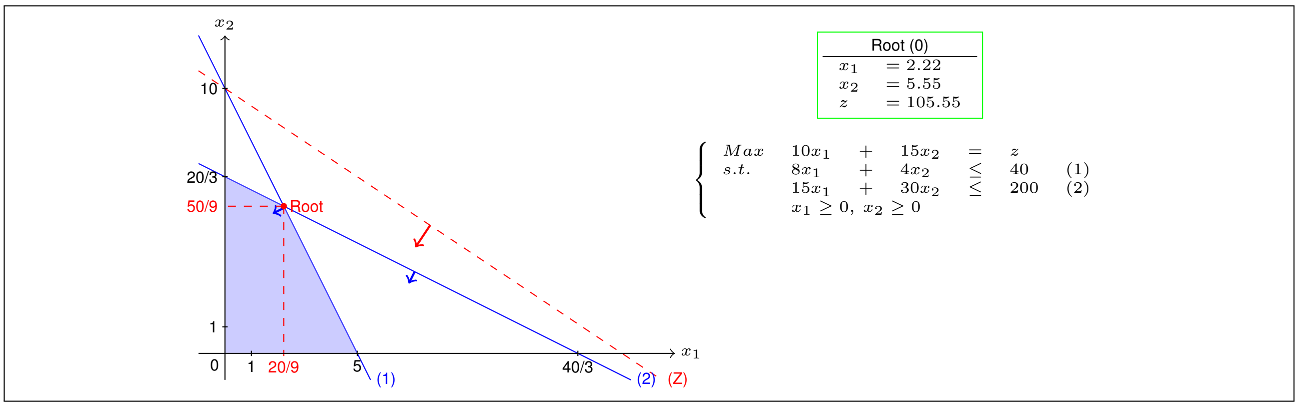
Step1: Build the root node from the linear relaxation

Since we cannot solve directly the problem, we can try to solve its linear relaxation which is the equivalent problem without the integrality constraint:

Integer Model **Linear Relaxation**

$$\begin{cases} Max & 10x_1 & + & 15x_2 & = & z \\ s.t. & 8x_1 & + & 4x_2 & \leq & 40 \\ & 15x_1 & + & 30x_2 & \leq & 200 \\ & x_1 \geq 0,\ x_2 \geq 0 \\ & x_1 \text{ and } x_2 \text{ integer} \end{cases} \Rightarrow \begin{cases} Max & 10x_1 & + & 15x_2 & = & z \\ s.t. & 8x_1 & + & 4x_2 & \leq & 40 \\ & 15x_1 & + & 30x_2 & \leq & 200 \\ & x_1 \geq 0,\ x_2 \geq 0 \end{cases}$$

The linear relaxation can be solved with the simplex method. However, for this problem, we will use the graphical method because we have only two variables.



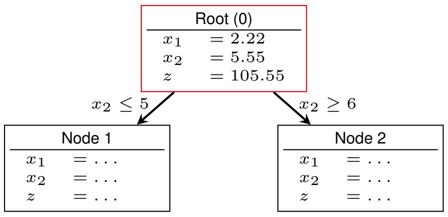
Root (0)
$x_1 = 2.22$
$x_2 = 5.55$
$z = 105.55$

$$\begin{cases} Max & 10x_1 & + & 15x_2 & = & z & \\ s.t. & 8x_1 & + & 4x_2 & \leq & 40 & (1) \\ & 15x_1 & + & 30x_2 & \leq & 200 & (2) \\ & x_1 \geq 0,\ x_2 \geq 0 \end{cases}$$

Step 2: We branch on a Fractional Variable

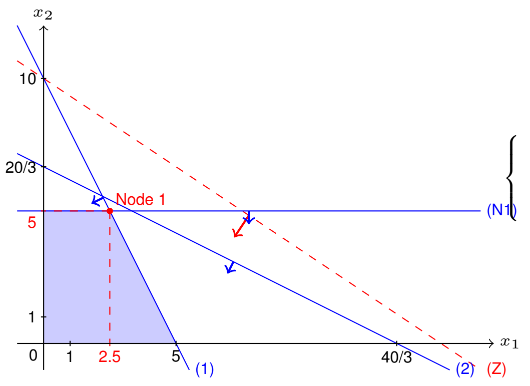
If there is no fractional variable, then we do not branch. Otherwise, we just mark the node as an integer and select another pending node with a fractional variable. If there are no pending nodes with a fractional variable, then the optimal solution is the best integer solution based on the objective.

Note that, sometimes we get an integer solution at the root, which means we found an optimal solution and no further exploration is required. Here we decide to branch on variable x_2, but because x_1 is also fractional, we can also choose to branch on x_1. Both decisions are correct. We create two new nodes, where, in Node 1, we will impose $x_2 \leq 5$ and $x_2 \geq 6$ since $2 < x_2 = 5.55 < 6$ for Node 1 and Node 2, respectively. An exploration tree is then used to track the search process:



We have to get the solution of the two newly created nodes before going to the next step.

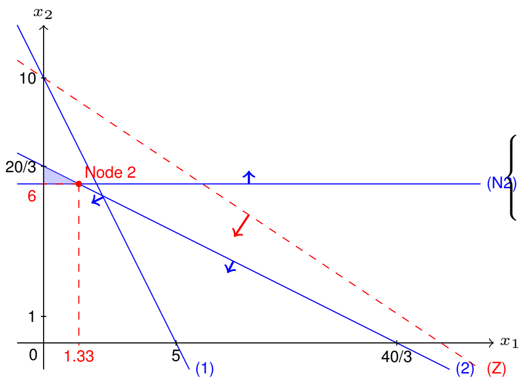
Node 1:



Node 1	
x_1	$= 2.5$
x_2	$= 5$
z	$= 100$

$$\begin{cases} \begin{array}{llclcll} Max & 10x_1 & + & 15x_2 & = & z & \\ s.t. & 8x_1 & + & 4x_2 & \leq & 40 & (1) \\ & 15x_1 & + & 30x_2 & \leq & 200 & (2) \\ & & & x_2 & \leq & 5 & (N1) \\ & \multicolumn{6}{l}{x_1 \geq 0,\ x_2 \geq 0} \end{array} \end{cases}$$

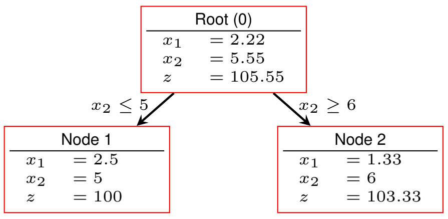
Node 2:



Node 2	
x_1	$= 1.33$
x_2	$= 6$
z	$= 103.33$

$$\begin{cases} \begin{array}{llclcll} Max & 10x_1 & + & 15x_2 & = & z & \\ s.t. & 8x_1 & + & 4x_2 & \leq & 40 & (1) \\ & 15x_1 & + & 30x_2 & \leq & 200 & (2) \\ & & & x_2 & \geq & 6 & (N2) \\ & \multicolumn{6}{l}{x_1 \geq 0,\ x_2 \geq 0} \end{array} \end{cases}$$

Now, we update our exploration tree:



We repeat from Step 2

We have two pending Nodes, Node 1 and Node 2, with fractional variables. To make a choice we will follow this simple rule:

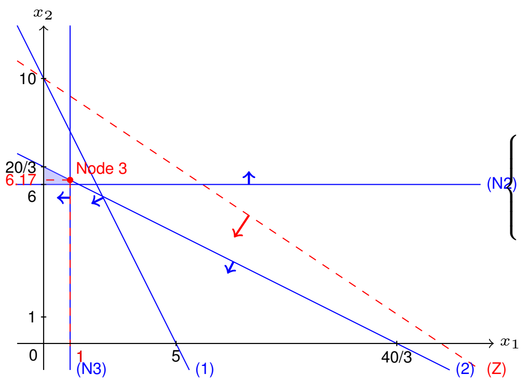
- If we are maximizing, we select the pending node with the highest value of z;
- If we are minimizing, we select the pending node with the lowest value of z;

This assures that we explore the feasible region with the best potential to find an optimal solution or, at least, quite close to an optimal one. Here, we will pursue our search prioritizing Node 2.
Again, we create two new nodes from Node 2. Since x_1 is the only fractional variable in Node 2, we have no choice but to branch on it, considering $1 < x_1 = 1.33 < 2$

- Node 3: by imposing $x_1 \leq 1$;
- Node 4: by imposing $x_1 \geq 2$.

We first solve both nodes and then we update the exploration tree.

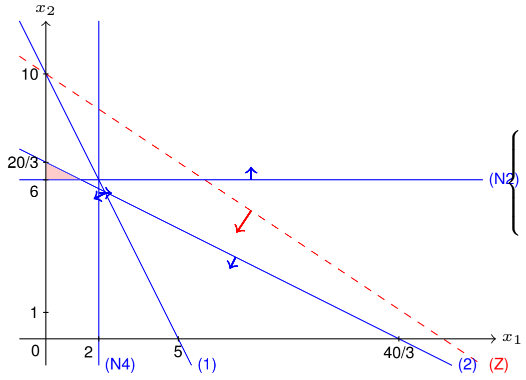
Node 3:



Node 3	
x_1	$= 1$
x_2	$= 6.17$
z	$= 102.55$

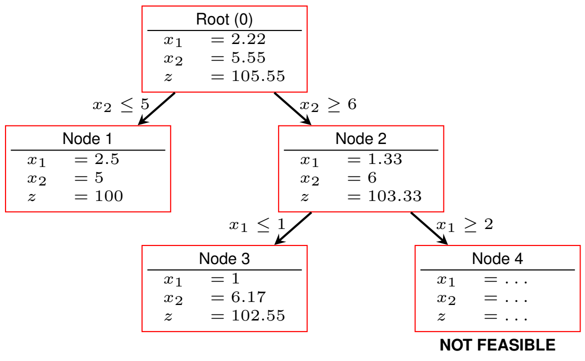
$$\begin{cases} Max & 10x_1 + 15x_2 = z \\ s.t. & 8x_1 + 4x_2 \leq 40 \quad (1) \\ & 15x_1 + 30x_2 \leq 200 \quad (2) \\ & x_2 \geq 6 \quad (N2) \\ & x_1 \leq 1 \quad (N3) \\ & x_1 \geq 0,\ x_2 \geq 0 \end{cases}$$

Node 4: Infeasible



Node 4	
x_1	$= \ldots$
x_2	$= \ldots$
z	$= \ldots$

$$\left\{\begin{array}{lccccll} Max & 10x_1 & + & 15x_2 & = & z & \\ s.t. & 8x_1 & + & 4x_2 & \leq & 40 & (1) \\ & 15x_1 & + & 30x_2 & \leq & 200 & (2) \\ & & & x_2 & \geq & 6 & (N2) \\ & x_1 & & & \geq & 2 & (N4) \\ & \multicolumn{3}{l}{x_1 \geq 0,\ x_2 \geq 0} & & & \end{array}\right.$$



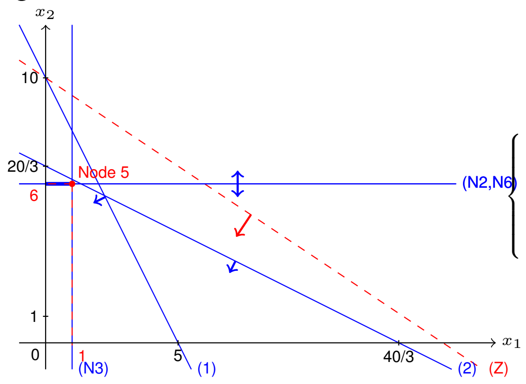
We repeat from Step 2

We have two pending nodes, Node 1 and Node 4, with fractional variables. Node 4 is discarded because it is infeasible. We will branch from Node 3, on $x_2 = 6.17$, because it has the highest value of z and generates:

- Node 5: by imposing $x_2 \leq 6$;
- Node 6: by imposing $x_2 \geq 7$.

We solve both nodes and update the exploration tree.

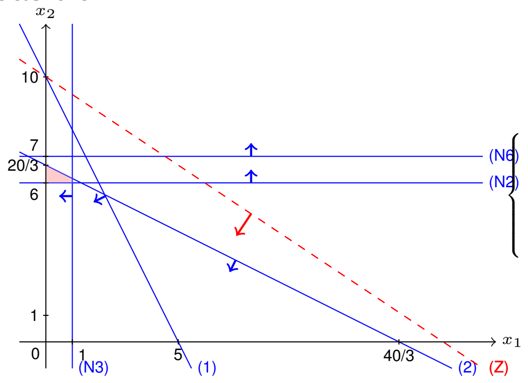
Node 5: Integer Solution



Node 5	
x_1	$= 1$
x_2	$= 6$
z	$= 100$

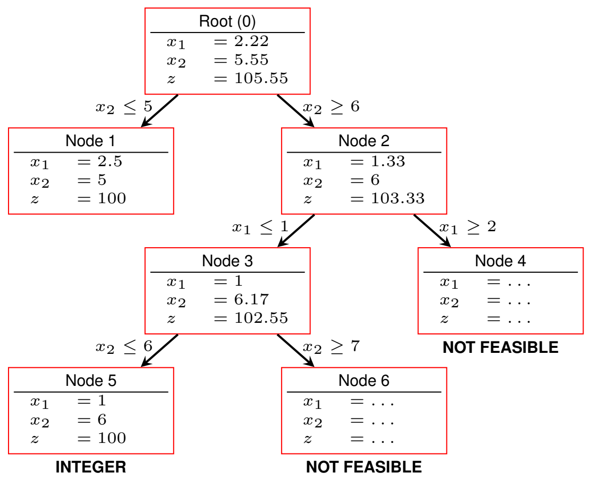
$$\left\{\begin{array}{lccccll} Max & 10x_1 & + & 15x_2 & = & z & \\ s.t. & 8x_1 & + & 4x_2 & \leq & 40 & (1) \\ & 15x_1 & + & 30x_2 & \leq & 200 & (2) \\ & & & x_2 & \geq & 6 & (N2) \\ & x_1 & & & \leq & 1 & (N3) \\ & & & x_2 & \leq & 6 & (N5) \\ & \multicolumn{3}{l}{x_1 \geq 0,\ x_2 \geq 0} & & & \end{array}\right.$$

Node 6: Infeasible



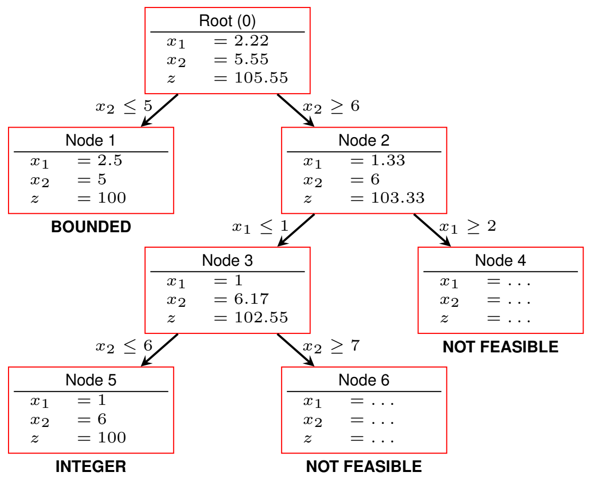
Node 6	
x_1	$= \ldots$
x_2	$= \ldots$
z	$= \ldots$

$$\left\{\begin{array}{lccccll} Max & 10x_1 & + & 15x_2 & = & z & \\ s.t. & 8x_1 & + & 4x_2 & \le & 40 & (1) \\ & 15x_1 & + & 30x_2 & \le & 200 & (2) \\ & & & x_2 & \ge & 6 & (N2) \\ & x_1 & & & \le & 1 & (N3) \\ & & & x_2 & \ge & 7 & (N6) \\ & \multicolumn{4}{l}{x_1 \ge 0,\ x_2 \ge 0} & & \end{array}\right.$$



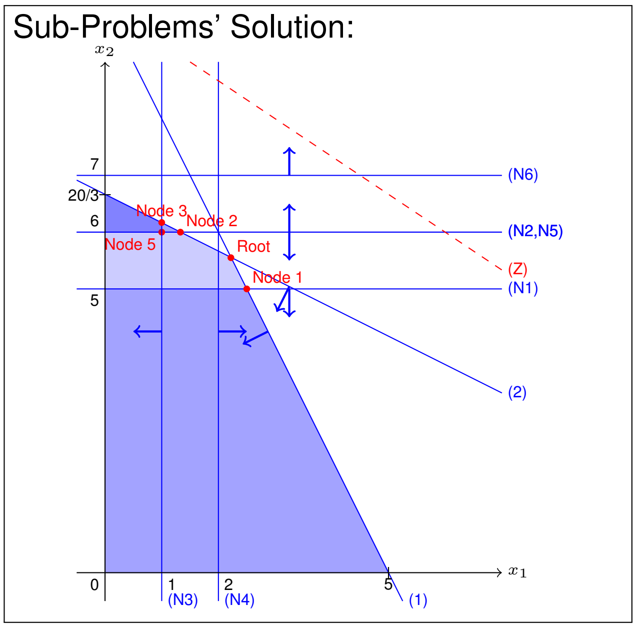
We repeat from Step 2

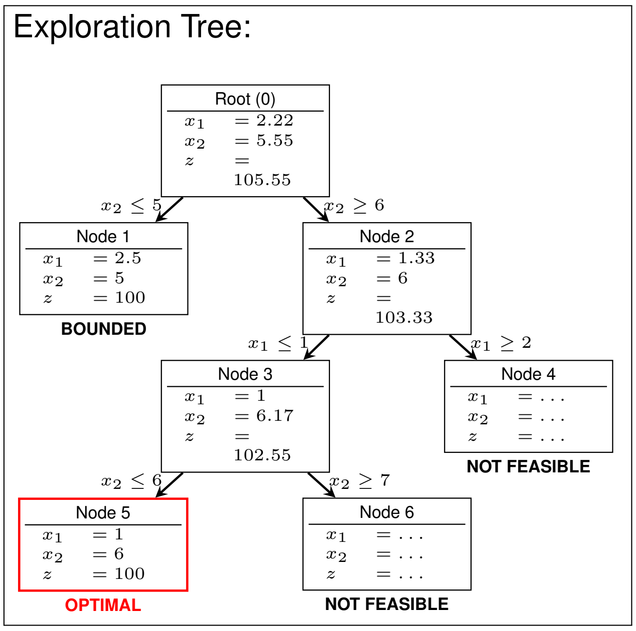
We still have one pending node, Node 1 (Node 5 has no fractional variables). If we branch on Node 1, we may find other integer solutions. The value of $z = 100$ at Node 1 tells us that the best we can expect from this exploration is to find an integer solution with the same value of z, not higher. However, from the previous steps, we already found an integer solution with a value of $z \ge 100$ at Node 5. Therefore, we can save the effort and avoid exploring Node 1 because it will not bring a better solution than the one already found. In this case, we say that Node 1 is **Bounded** by Node 5.



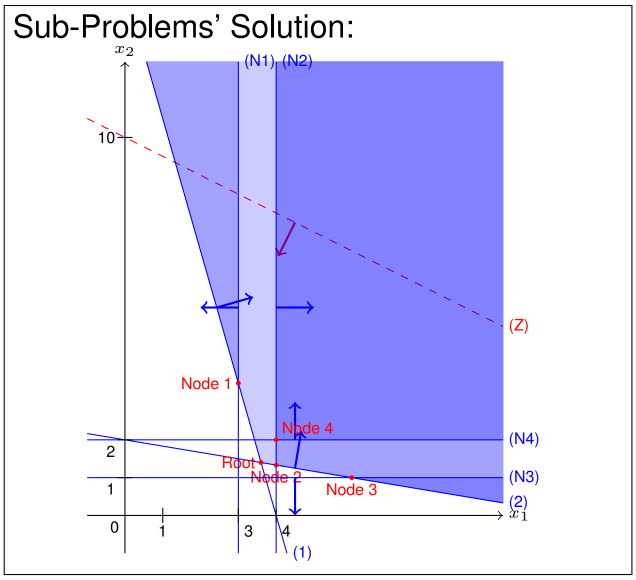
We conclude

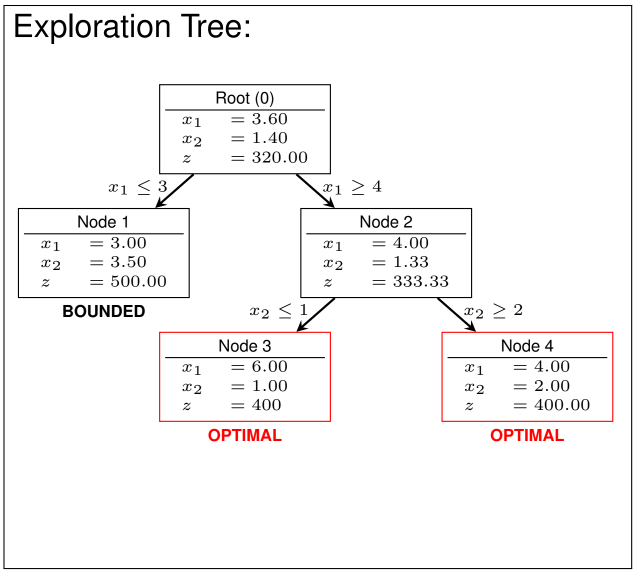
There are no more pending nodes, then the algorithm stops and we finish our exploration. The best integer solution found during the process is obtained at Node5, therefore we can conclude it is the optimal solution of the IP: $x_1 = 1$, $x_2 = 6$, and $z = 100$.





C Solution of the Dorian Problem


Sub-Problems' Solution:
x_2
x_1
(N1)
(N2)
(1)
(2)
(N3)
(N4)
(Z)
Node 1
Node 4
Root
Node 2
Node 3


Exploration Tree:
Root (0)
$x_1 = 3.60$
$x_2 = 1.40$
$z = 320.00$
$x_1 \leq 3$
$x_1 \geq 4$
Node 1
$x_1 = 3.00$
$x_2 = 3.50$
$z = 500.00$
BOUNDED
Node 2
$x_1 = 4.00$
$x_2 = 1.33$
$z = 333.33$
$x_2 \leq 1$
$x_2 \geq 2$
Node 3
$x_1 = 6.00$
$x_2 = 1.00$
$z = 400$
OPTIMAL
Node 4
$x_1 = 4.00$
$x_2 = 2.00$
$z = 400.00$
OPTIMAL