Branch & Bound (B&B): a tutorial with theoretical insights and a worked-out example

Alessandro Bombelli[1]

[1]*Air Transport & Operations, Aerospace engineering faculty, Delft University of Technology*

1 Introduction

In this tutorial, the basics behind the Branch & Bound (B&B) solution technique, the most popular solution technique for optimization problems featuring decision variables that are not only continuous, are shown. To do so, we start with a quick recap of the Simplex method and show how embedding the Simplex method into a decision tree forms the basics behind B&B in Sec. 2. Then, in Sec. 3 a fully worked-out example is presented so that the theoretical concepts are displayed in a practical and intuitive way. Finally, in Sec. 4 we highlight some of the aspects of the B&B solution technique that are not presented in this tutorial. In relation to this last point, I would like to point out that this tutorial does not want to replace textbooks or more official sources. Because I witnessed quite a few students struggling with the theory behind B&B (struggles that we, as teachers, are partially responsible for), I decided to work on a tutorial that I hope can ease the learning curve. This tutorial is probably way too verbose sometimes. Some concepts are repeated and stressed, maybe unnecessarily, multiple times. The worked-out example might seem trivial. Notwithstanding, I hope this learning material to be useful for at least some of you. In addition, as mentioned at the end of the tutorial, this is a work-in-progress and any suggestion that could improve the quality of the content is very much appreciated.

2 B&B solution technique

2.1 The Simplex method

In the AE4441-16 course, we analyzed quite extensively the Simplex method, which is a solution method that efficiently finds the optimal solution of a Linear Program (LP), i.e., a model with a (i) linear objective function, (ii) linear constraints, and (iii) continuous decision variables. Requirements (i) and (ii) are necessary to set up the Simplex method, which is a more elaborated version of the solution of a linear system $Ax = b \rightarrow x = A^{-1}b$. The Simplex method is more elaborated than the previous formula for several reasons.
First, when setting up an LP, the constraint matrix will generally not be square. Hence, it cannot be simply inverted. If you recall how the tableau in the Simplex method is defined, for a problem with n decision variables (the original ones, plus all the slack, artificial, and surplus decision variables needed to put the problem in standard form) and m constraints ($n > m$), at every iteration we choose m basic variables and $n - m$ non-basic variables. The non-basic decision variables are set to zero, hence reducing the tableau to an $m \times m$ matrix. Such matrix does not even need to be inverted because the m columns of the tableau associated with the basic variables define an orthonormal matrix. This means that each of these columns has $m - 1$ 0s and a single 1, with all the 1s appearing in different rows. Below, we show three different 3×3 orthonormal matrices (Note: the identity matrix is a special case of orthonormal matrix).

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

Because of this setting, the basic variable associated with the 1 in each row will directly take the value shown on the right-hand side of the tableau.
Second, we should not forget that in each LP we aim at maximizing or minimizing the objective function, and hence the selection of which basic decision variable becomes non-basic and vice versa is subject to the best increment in the objective. Without loss of generality, in the following, we will assume a maximization problem. Hence, in every iteration of the Simplex method, we will make basic the non-basic variable that contributes to the largest increase to the objective function. The current value of the objective function can be always traced on the right-hand side of the first row of the tableau, namely, the objective row.
Once the objective function cannot be improved any further, the tableau of the final iteration will display the optimal value of the objective and which decision variables are non-zero (and their values).
To quickly recap some of the Simplex method concepts addressed above, let us consider the Wyndor Glass Co. problem:

$$\max \quad J = 3x_1 + 5x_2 \tag{2.1}$$

subject to:

$$x_1 \leq 4 \tag{2.2}$$
$$2x_2 \leq 12 \tag{2.3}$$
$$3x_1 + 2x_2 \leq 18 \tag{2.4}$$
$$x_1, x_2 \geq 0 \tag{2.5}$$

which, after adding the slack variables x_3, x_4, and x_5 is transformed into proper form and ready for the Simplex method as follows

$$\max \quad J = 3x_1 + 5x_2 + 0x_3 + 0x_4 + 0x_5 \tag{2.6}$$

subject to:

$$x_1 + x_3 = 4 \tag{2.7}$$
$$2x_2 + x_4 = 12 \tag{2.8}$$
$$3x_1 + 2x_2 + x_5 = 18 \tag{2.9}$$
$$x_1, x_2, x_3, x_4, x_5 \geq 0 \tag{2.10}$$

The initial tableau for the problem is shown in Table 1.

	x_1	x_2	x_3	x_4	x_5	R.H.S.
J	-3	-5	0	0	0	0
	1	0	1	0	0	4
	0	2	0	1	0	12
	3	2	0	0	1	18

Table 1: Wyndor Glass Co. problem, initial tableau.

where x_1 and x_2 are set to be non-basic (hence, $x_1 = x_2 = 0$) while in shaded red the three columns of the orthonormal matrix are shown. In the left-most column, associated with x_3, the single 1 appears in the first constraint row, hence $x_3 = 4$. If you are not convinced, we can always perform element-by-element multiplication in that row: $1 \times 0 + 0 \times 0 + 1 \times x_3 + 0 \times x_4 + 0 \times x_5 = 4 \rightarrow x_3 = 4$. With similar reasoning, we have that $x_4 = 12$ and $x_5 = 18$. The current solution ($Z = 0$) is not optimal because both x_1 and x_2 appear with negative coefficients in the objective row (please re-check the Simplex method theory if this is not clear!), we need more iterations. In the first iteration, x_2 becomes basic and x_4 non-basic, resulting after a bit of manipulation in the tableau shown in Table 2.
We now notice that the objective has increased from $J = 0$ to $J = 30$. x_1 is still non-basic, now together with x_4, while x_2, x_3, and x_5 are basic. (Note: in the tableau, at any iteration basic variables should be associated with a 0 coefficient in the objective row, while non-basic variables generally display a non-zero coefficient).

	x_1	x_2	x_3	x_4	x_5	R.H.S.
J	-3	0	0	$\frac{5}{2}$	0	30
	1	0	1	0	0	4
	0	1	0	$\frac{1}{2}$	0	6
	3	0	0	-1	1	6

Table 2: Wyndor Glass Co. problem, tableau after the first iteration.

The columns of the orthonormal matrix are still highlighted in red. Following the same reasoning as above, we have that $x_3 = 4$ (first constraint row), $x_2 = 6$ (second constraint row), and $x_5 = 6$ (third constraint row). Because the coefficient of x_1 is still negative in the objective row, we can proceed with another iteration where x_1 becomes basic and x_4 non-basic. The resulting final tableau is shown in Table 3.

	x_1	x_2	x_3	x_4	x_5	R.H.S.
J	0	0	0	$\frac{3}{2}$	1	36
	0	0	1	$\frac{1}{2}$	-$\frac{1}{2}$	2
	0	1	0	$\frac{1}{2}$	0	6
	1	0	0	-$\frac{1}{2}$	$\frac{1}{2}$	2

Table 3: Wyndor Glass Co. problem, final tableau.

The current solution, i.e., $J = 36$, is now optimal because there are no negative coefficients in the objective row. As usual, the three columns of the orthonormal matrix are highlighted in red. We have that $x_3 = 2$, $x_2 = 6$, and $x_1 = 2$. We can also double-check that $3 \times x_1 + 5 \times x_2 = 3 \times 2 + 5 \times 6 = 36 = J$.
Notwithstanding the increased complexity with respect to a classic linear system, the Simplex method relies indeed on algebraic to compute the optimal values of the decision variables. This means their values will be, in general, fractional or, using a different perspective, they are not guaranteed to be integer. This is the reason why requirement (iii) is paramount. When working on optimization problems, some decision variables might indeed be continuous. This is the case if we need a decision variable expressing the height of a dam or the overall weight of parcels to be stored in a trailer. On the other hand, if we need to map how many aircraft to acquire, then such a decision variable should be integer. In a similar fashion, we can have decision variables mapping YES (1) or NO (0) decisions, i.e., binary decision variables (which are a special case of integer decision variables). Hence, how can we still use the Simplex method in a way that ensures that integer and binary decision variables are not treated as continuous? This is where the B&B solution technique comes into play.

2.2 Basics of B&B

B&B is an efficient solution technique for optimization problems where some decision variables are not continuous. If we are dealing with a problem with only integer decision variables, such a problem is referred to as Integer Program (IP). If we are only dealing with a problem with binary decision variables, such a problem is referred to as Binary Program (BP). More generally, if we are dealing with a problem that entails a mix of continuous, integer, and binary decision variables, such a problem is referred to as a Mixed Integer Linear Program (MILP).
The underlying principle of B&B relatively simple. When using the Simplex method and optimizing a specific problem, it might be the case that some decision variables take integer values in the final solution, although we still treat them as continuous (recall the Wyndor Glass Co. Product-Mix Problem). Hence, when dealing with a MILP, one thing we can try to do is to relax all non-continuous decision variables to be continuous and to solve the associated relaxed MILP. Maybe we are lucky, and in the final solution, all basic decision variables satisfy their nature. If this is the case, we are done with our optimization problem. Most likely,

this will not be the case and some integer/binary decision variables will have fractional values, hence yielding a solution that has no practical use.
To enforce the correct domain to all fractional decision variables, a decision tree is generated starting from the initial model where all decision variables were continuous. In the rest of this tutorial, we will use the term node to define a specific variation of the original MILP we are trying to solve. We do this because, in the tree structure, every node is indeed a slightly different MILP we want to solve to optimality. In the decision tree, from each parent node, two children nodes are created (two branches are created) that inherit all the properties of the parent node. In addition, each of the two children nodes is given an additional constraint that tries to re-instate the nature of a decision variable that was fractional in the parent node. In a classic decision tree fashion, the two constraints define mutually exclusive sets. If the fractional decision variable (let us define it x_1) is binary, then one of the two children nodes will feature the additional constraint $x_1 = 0$, and the other one will have the additional constraint $x_1 = 1$. This example is shown in Fig. 1, where we assume all decision variables are binary. On the other hand, if the fractional decision variable is integer (let us assume it was defined as $x_1 = \{0, 1, \cdots, 9, 10\}$, hence a decision variable that can take any integer value from 0 to 10), and was returned with the value $x_1 = 3.6$ in our relaxed model, one of the two children nodes will inherit the additional constraint $x_1 \leq 3$ while the other children node will inherit the additional constraint $x_1 \geq 4$. This example is shown in Fig. 2, where we assume all decision variables are integer. Note: both examples do not refer to a complete B&B decision tree, but are used for the sole purpose of providing some insights into the methodology. A fully developed B&B decision tree is shown in Sec. 3.

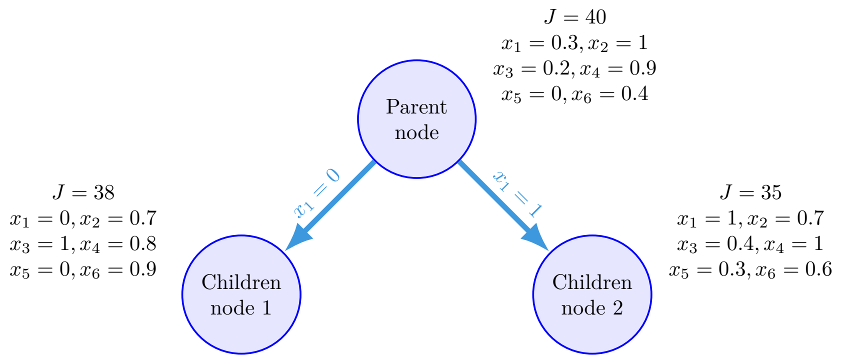


Figure 1: Tree structure for the example discussed in the text with binary-only decision variables. Note: in such model, our goal is to maximize the objective function J.

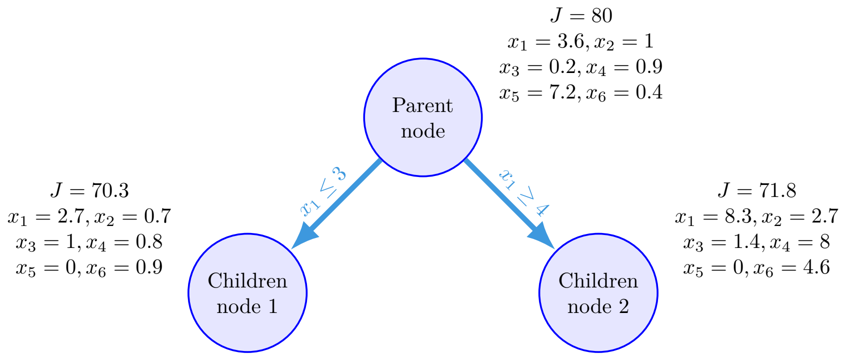


Figure 2: Tree structure for the example discussed in the text with integer-only decision variables. Note: in such model, our goal is to maximize the objective function J.

We can highlight some similarities and differences between the examples shown in Fig. 1 and Fig. 2.
When it comes to similarities, we can notice how in both decision trees the quality of the solution degrades as we move from a parent to a children node (hence, J decreases for a maximization problem and increases for a minimization problem). This is correct, because every children node inherits all the constraints of the parent node, plus an additional constraint (the constraint depicted on the branch connecting the parent with the children). Hence, the children is more constrained than the parent, resulting in an objective that can

only be worse. Another similarity is that, in bot decision trees, no node has been identified that yields a solution satisfying all the integrality constraints. In Fig. 1 and Fig. 2, in both children nodes at least one decision variable is fractional, hence not satisfying the binary (resp. integer) nature of the decision variables. Hence, in both cases we do not have a solution yet that is implementable in practice. Finally, in both cases multiple decision variables were fractional in the parent node (x_1, x_3, x_4, and x_6 in Fig. 1, x_1, x_2, x_3, x_4, and x_6 in Fig. 2). In both situations, we decided to branch on x_1, but this choice was arbitrary. As will be explained in Sec. 4, there are different rules to decide on which decision variable to branch first in case of multiple fractional variables in a node. While different choices might increase or decrease the convergence time of the B&B process, this will not affect the final solution as long as the B&B decision tree is fully explored.

In terms of differences, a major differences between branching on fractional binary decision variables (Fig. 1) and fractional integer decision variables (Fig. 2) is the following. In the first case, in the two children nodes we are enforcing the value of such decision variable, as the two disjoint constraints are $x_1 \leq 0$ and $x_1 \geq 1$. Given than the original binary decision variable (i.e., $x_1 \in \{0, 1\}$) was relaxed to be continuous (i.e., $x_1 \in [0, 1]$), the two constraints become $x_1 = 0$ and $x_1 = 1$. In the second case, given the fractional integer decision variable, we compute the floor and ceiling of the fractional value (in the example, we have resp. $\lfloor 3.6 \rfloor = 3$ and $\lceil 3.6 \rceil = 4$), and impose that the decision variable should be smaller or equal to the floor, or greater or equal to the ceiling (i.e., $x_1 \leq 3$ and $x_1 \geq 4$ resp.). As such, in both children nodes the decision variable could still be fractional (as a matter of fact, it still is in Fig. 2, with $x_1 = 2.7$ in the left-most children node and $x_1 = 8.3$ in the right-most children node). Hence, when branching a fractional integer decision variable the first time, we are generally not assigning a specific value to it in the two children nodes. Conversely, we are reducing the interval where the decision variable is defined. In the example, we went from $x_1 = [0, 10]$ in the parent node (similarly to above, the original decision variable is integer $x_1 = \{0, 1, \cdots, 9, 10\}$] and has been relaxed to be continuous), to $x_1 = [0, 3]$ and $x_1 = [4, 10]$ in the left- and right-most nodes, respectively. If we were to develop the B&B decision tree further, we would probably reach a depth where even for fractional integer decision variables the additional inequality constraint becomes an equality, but this is not generally achieved during the first branching, as it happens with fractional binary decision variables instead.

The examples shown in Fig. 1 and Fig. 2 served as an introduction to anticipate some key concepts needed to formally describe the B&B process, namely the best bound, best incumbent, and root node.

Best bound $\mathbb{BB}$: the best bound is the best objective value J (best means highest for a maximization problem, lowest for a minimization problem) across all unexplored nodes with fractional solutions, i.e., where at least one decision variable is fractional. With the term unexplored, we mean a node that has not been branched further yet. In both examples (Figs. 1 and 2), the parent node has been explored, while both children nodes are unexplored. In the first example, the current best bound is $J = 38$ (left-most children node), because that is the highest value across the two unexplored nodes. In the second example, the best bound is $J = 71.8$ (right-most children node), because that is the highest value across the two unexplored nodes.

Best incumbent $\mathbb{BI}$: the best incumbent is the best objective value J (best means highest for a maximization problem, lowest for a minimization problem) across all nodes with solutions that are non-fractional, i.e., solutions that cab be used in practice. Note that nodes featuring non-fractional solutions are by definition explored. Given that we can branch on fractional decision variables only, nodes where there are no fractional decision variables cannot be branched any further. Note: in both examples, we do not have a best incumbent yet, as there is no node with a solution without any fractional decision variable.

Root node: in a B&B tree, the root node is the first node starting the tree, where all decision variables are relaxed to be continuous. Given it is the least constrained node (in terms of which values the decision variables can take), this node will be characterized by the first and highest (for a maximization problem) or lowest (for a minimization problem) best bound.

We can now combine these three elements to describe how a B&B tree grows. Given a MILP, the first step is to relax all decision variables to be continuous and to solve this relaxed problem. The associated node in the tree is, as defined above, the root node. Because in this special node all decision variables are relaxed to be continuous with no further restrictions, this node defines a full linear relaxation of the original MILP. If the associated solution satisfies all integrality constraints, then the associated J is both a best bound and a best incumbent and the solution is optimal already. If some decision variables are fractional, then the J of the root node defines the first best bound and the B&B decision tree is started branching on a fractional decision variable. As more nodes are added and the decision tree is expanded, the following properties will hold throughout the process:

- the best bound cannot improve as mode nodes are explored. This is because, as we reintroduce constraints on the binary/integer nature of decision variables, we will make the fully relaxed model of the root node "less relaxed" or, in other words, more constrained. If we add constraints to a model, its optimal solution will either stay the same or get worse. As such, the best bound for a maximization problem is actually an upper bound, and will generally decrease over time. On the other hand, the best bound for a minimization problem is actually a lower bound, and will generally increase over time;

- unless the original MILP is infeasible, eventually a first best incumbent will be found. As more nodes are explored, the best incumbent can only stay the same or improve. This is true because, as we explore more nodes, the Simplex method might find in a new node a combination of decision variables that satisfies all binary/integer constraint (although some of these variables are still relaxed to be continuous) while yielding a better J with respect to the current best incumbent. As such, for a maximization problem the best incumbent will generally increase over time, while in a minimization problem the best incumbent will generally decrease over time;

- at any point during the B&B decision tree growth process, we can track the quality of our optimization process with a quantity defined optimality gap $\mathbb{OG}$ that is defined as $\mathbb{OG} = \left|\frac{\mathbb{BB}-\mathbb{BI}}{\mathbb{BI}}\right| \times 100$. The optimality gap is a percentile measure that assesses how far our $\mathbb{BI}$ is with respect to the "theoretical" optimal solution of the MILP we are dealing with, i.e., $\mathbb{BB}$. It should be noted that $\mathbb{BB}$, given the definition we provided above, is a best bound because it defines an objective value of a relaxed version of the MILP, where all original inequality and equality constraints (in matrix form $A_{in}\vec{x} \leq \overrightarrow{b_{in}}$ and $A_{eq}\vec{x} = \overrightarrow{b_{eq}}$), but where at least a binary/integer decision variable is fractional. As such, our B&B decision tree is proven to have converged to the optimal solution when the optimality gap is zero, which implies $\mathbb{BB} = \mathbb{BI}$. If we were to keep track of the evolution of the $\mathbb{BB}$ and $\mathbb{BI}$ values during a B&B process, the following trends are expected. Assuming enough computational time is allocated to that the MILP converges to optimality, for a maximization problem it always holds that $\mathbb{BB} \geq \mathbb{BI}$ (we already anticipated that for a maximization problem $\mathbb{BB}$ is an upper bound), i.e., the best bound and the best incumbent curves will get closer to each other with $\mathbb{BB}$ approaching $\mathbb{BI}$ from above. For a minimization problem, being $\mathbb{BB}$ a lower bound, the best bound and the best incumbent curves will get closer to each other with $\mathbb{BB}$ approaching $\mathbb{BI}$ from below.

  In addition, while optimality is mathematically proven when $\mathbb{BB} = \mathbb{BI}$, an $\mathbb{OG}$ greater than 0% does not mean our current $\mathbb{BI}$ is not the optimal one. It could be the case that $\mathbb{BI}$ cannot be improved already, but that more nodes needs to be explored so that $\mathbb{BB}$ can decrease (maximization problem) or increase (minimization problem) to close the gap.

  : Example

  For sake of clarity, let us consider the following example. We have a MILP we want to maximize and, given the current status of the B&B process, we have $\mathbb{BB} = 200$ and $\mathbb{BI} = 100$, resulting in an $\mathbb{OG} = 100\%$. Does it mean we are 100% off from the optimal solution? Not necessarily. The optimal solution could be indeed slightly lower than 200. It cannot be exactly 200, because otherwise the node currently associated to the $\mathbb{BB} = 200$ would have yielded a non-fractional solution. Going to the other extreme of the spectrum, it could be that our current $\mathbb{BI} = 100$ is already the optimal solution, but the B&B tree needs to be explored much further so that the best bound can decrease. It could also be that the optimal solution falls within the $[100, 200]$ interval, and more nodes need to be explored so that both the best bound decreases and the best incumbent increases.

- as we move down depth-wise and add more nodes to the B&B tree, we can track all the additional constraints that restrict the domain of the binary/integer decision variables (with respect to their initial continuous relaxation) by following all the branches connecting that node back to the root node. All the constraints characterizing all those branches are the additional constraints the node are considering is subject to with respect to the root node. This is an extension of the property that every child node is the same MILP as the parent node, plus the additional constraint deriving from the branch connecting the two nodes. Note that, while nodes that are deeper in the B&B are generally more strict regarding the domain of relaxed binary/integer decision variables (recall Figs. 1 and 2), in every node of the B&B process all inequality and equality constraints $A_{in}\vec{x} \leq \overrightarrow{b_{in}}$ and $A_{eq}\vec{x} = \overrightarrow{b_{eq}}$ should be satisfied no matter what.

⚙: Example

Let us consider Fig. 2 again, and let us assume one inequality constraints of such MILP is $2x_1 + x_4 + x_5 \leq 16$. We can verify that such constraint is verified in the parent node (which, as we now know, is also the root node of such B&B):
$2 \times 3.6 + 0.9 + 7.2 = 15.3 \leq 16$ ✓
Let us now consider children node 1:
$2 \times 2.7 + 0.8 + 0 = 6.2 \leq 16$ ✓
The constraint is also satisfied in such node. We now need to check what happens in children node 2:
$2 \times 8.3 + 8 + 0 = 22.6 \leq 16$ ⚠
The constraint is not satisfied in children node 2! Hence, given that an optimal solution (to the associated relaxations) was found both in children node 1 and children node 2, the constraint $2x_1 + x_4 + x_5 \leq 16$ cannot be part of the original MILP.

We have now reached a stage where we (hopefully) are familiar with the general setting of the B&B, and with terms such as root node, linear relaxation, best bound, and best incumbent. A particularly impatient reader might still be wondering what is the added value of the whole B&B tree structure with respect to full enumeration of all the possible combinations of decision variables, for example, and they would be right! What we have not covered yet, is a very powerful property of the B&B solution method that makes it more efficient than full enumeration, given the latter is not even possible for large problems. Such property is the possibility to fathom nodes, i.e., to avoid exploring nodes still featuring fractional solutions because we already know such exploration would be pointless. We can identify the following conditions when exploring new nodes in the B&B tree:

- a node in the B&B tree is associated with an infeasible MILP model. This means that the additional set of constraints on the values of the relaxed binary/integer decision variables makes it impossible to even find a feasible solution to such MILP. Given that the MILP is infeasible, branching even further will not restore feasibility, hence such node is fathomed;

- a node in the B&B tree is associated with a MILP whose optimal solution is non-fractional (although many binary/integer decision variables are still relaxed to be continuous). In this case, given our current $\mathbb{BI}$, we need to check if the objective value J of the node is better (higher for a maximization, lower for a minimization problem) than $\mathbb{BI}$ or not. In the first case, J becomes the new $\mathbb{BI}$ because it dominates it. In the second case, we found another incumbent solution (i.e., a solution to a relaxed version of the original MILP that satisfies all $A_{in}\vec{x} \leq \overrightarrow{b_{in}}$ and $A_{eq}\vec{x} = \overrightarrow{b_{eq}}$ constraints and where all binary/integer decision variables are non-fractional). In both cases, given that there are no fractional decision variables to branch on, the node cannot be branched any further;

- a node in the B&B tree is associated with a MILP whose optimal solution is fractional. In this case, we need to check if the optimal solution J is better (higher for a maximization, lower for a minimization problem) or worse than $\mathbb{BI}$. In the first case, we should still branch on one of the fractional decision variables. On the other hand, if J is already worse than $\mathbb{BI}$, there is no point in branching even further, as adding more restrictions on the relaxed values on binary/integer decision variables can only make J even worse. This is the most important fathoming condition, as it is the only one preventing the exploration of unnecessary nodes. In every B&B tree, many potential branches of the tree are fathomed because of this property, which substantially speeds up the convergence of the process.

We now have all the main ingredients needed to fully understand the process and convergence of the B&B solution technique.

⚙: Recap

In B&B we use a decision tree where every node is a relaxed version of the original MILP we are trying to solve. While all $A_{in}\vec{x} \leq \overrightarrow{b_{in}}$ and $A_{eq}\vec{x} = \overrightarrow{b_{eq}}$ constraints should be satisfied in every node, binary/integer decision variables are relaxed to deal with an "easier" problem and to leverage the efficiency of the Simplex method, that solves to optimality every node. When we obtain a fractional solution in a node, we can branch on one of the fractional decision variables and "reduce" the relaxation in each of the created children nodes (we basically re-add some of the complexity we were initially neglecting). While generating more and more nodes, we should constantly keep track of the best bound $\mathbb{BB}$ (i.e., the best objective across all unexplored nodes with fractional solutions) and the best incumbent $\mathbb{BI}$ (i.e., the best objective across all nodes with non-fractional solutions). For a maximization problem, it holds that $\mathbb{BB} \geq \mathbb{BI}$, while for a minimization problem it holds that $\mathbb{BB} \leq \mathbb{BI}$. Once $\mathbb{BB} = \mathbb{BI}$, then the original MILP has been solved to optimality.

3 An illustrative example

In this section, we analyze step-by-step a BP (as a recap, an optimization problem with only binary decision variables) and how the BP is solved via B&B. While, in practice, commercial software will efficiently apply B&B for us, this example has been tailored to be so small and simple than B&B can be tracked manually and every node of the process can be explored. Hence, the goal of this section is to show how to apply the insights from Sec. 2 to a real problem. When dealing with large problems, B&B will occur behind the scenes, but the insights gained here should be helpful to keep track of how the algorithm is performing and of the solution quality as more nodes are explored.
In this problem, we assume we go out for dinner with our friends in a pizzeria that allows customers to build their own pizza. we want to build our own pizza, starting with a pizza margherita that costs 6€. Our budget is 13€. We are given the menu with the list of potential toppings and prices as given in Fig. 3.

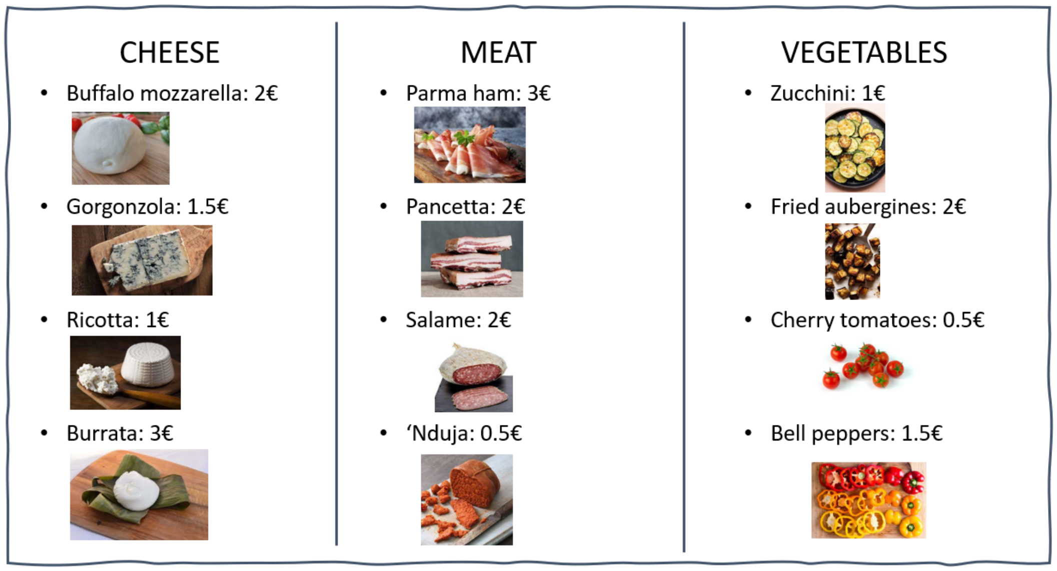


Figure 3: Original topping menu with prices as provided by the pizzeria.

That defines our set of toppings $\mathcal{T}$, with 12 elements ranging from 1=buffalo mozzarella to 12=bell peppers (see blue numbers in Fig. 4).
Adding each topping to our initial margherita increases our satisfaction by a quantity equivalent to the associated red number. Our goal is to create the most satisfactory pizza we can while staying within the budget. To do so, we need to solve a MILP model with 12 binary decision variables $x_i \quad i = 1, \cdots, 12$, where x_i=1 means we are adding the i-th topping to our pizza. Parameters are S_i and P_i (satisfaction and price,

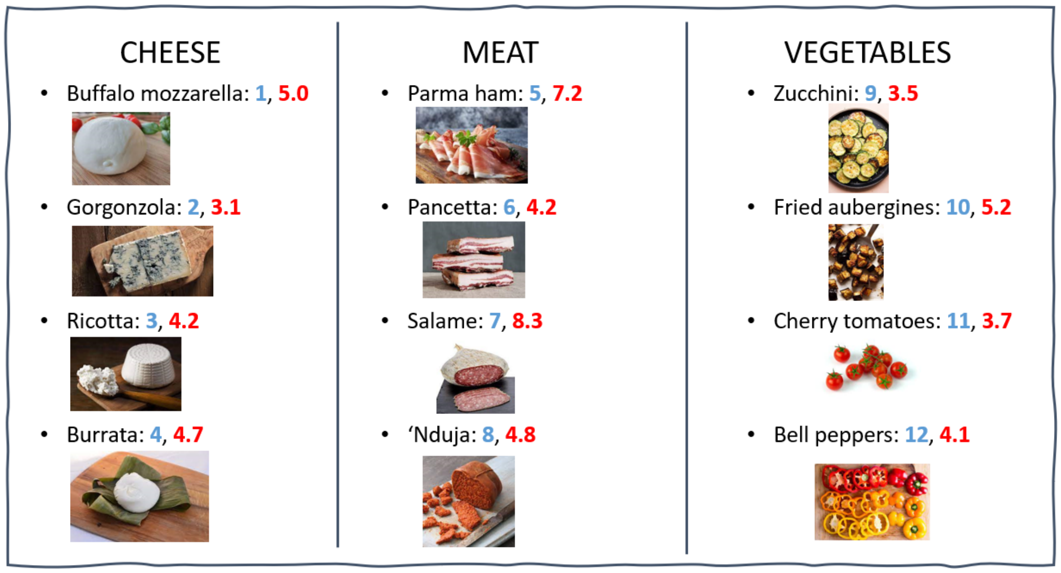


Figure 4: Topping menu with indices of decision variables (x_1 is buffalo mozzarella, x_2 is gorgonzola, $\cdots$, x_{12} is bell peppers) and satisfaction value that the customer (us) assigns to each topping.

resp., of topping i).

In matrix notation, the quantities of interest of our MILP are

$$\vec{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_{12} \end{bmatrix} \in \{0,1\} \qquad \vec{C}^T = \underbrace{[5.0\ 3.1\ \cdots\ 3.7\ 4.1]}_{\text{satisfaction parameters } S_i}$$

$$A_{in} = \underbrace{[2.0\ 1.5\ \cdots\ 0.5\ 1.5]}_{\text{prices } P_i} \qquad \overrightarrow{b_{in}} = [7]$$

$$A_{eq} = [] \qquad \overrightarrow{b_{eq}} = []$$

If we carry out all vectorial products, we end up with the following MILP

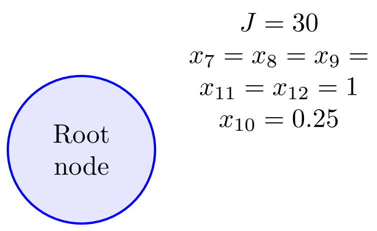
$$\begin{aligned} \max \quad & 5.0x_1 + 3.1x_2 + 4.2x_3 + 4.7x_4 + \\ & 7.2x_5 + 4.2x_6 + 8.3x_7 + 4.8x_8 + \\ & 3.5x_9 + 5.2x_{10} + 3.7x_{11} + 4.1x_{12} \quad (\vec{C}^T\vec{x}) \end{aligned} \tag{3.1}$$

subject to:

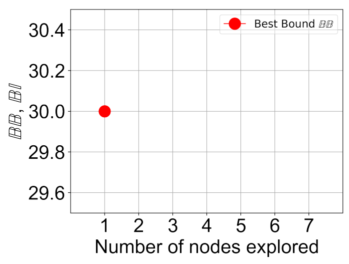
$$\begin{aligned} & 2.0x_1 + 1.5x_2 + 1.0x_3 + 3.0x_4 + 3.0x_5 + 2.0x_6 + \\ & 2.0x_7 + 0.5x_8 + 1.0x_9 + 2.0x_{10} + 0.5x_{11} + 1.5x_{12} \le 7 \quad (A_{in}\vec{x} \le \overrightarrow{b_{in}}) \end{aligned} \tag{3.2}$$

$$x_i \in \{0,1\} \quad \forall i \in \mathcal{T} \quad \text{(decision variables)} \tag{3.3}$$

Given that the problem at hand is quite simple (12 decision variables, one inequality constraint), we can go over the B&B solution method step-by-step. The first step entails computing the solution related to the root node, where all 12 decision variables are related to being continuous, i.e., $x_i \in \{0,1\} \rightarrow x_i \in [0,1]\, \forall i \in \mathcal{T}$. The solution in terms of decision variables (Fig. 5a) and evolution of $\mathbb{BB}$ and $\mathbb{BI}$ (Fig. 5b) is shown in Fig. 5.

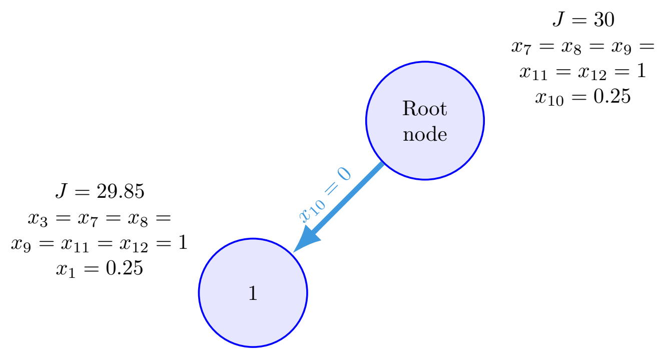


(a) Tree structure

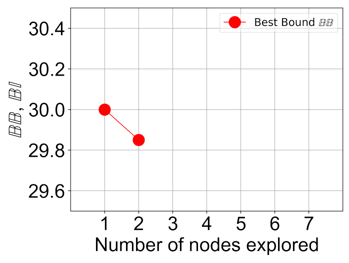


(b) Evolution of $\mathbb{BB}$ and $\mathbb{BI}$.

Figure 5: B&B for the pizza example: root node.



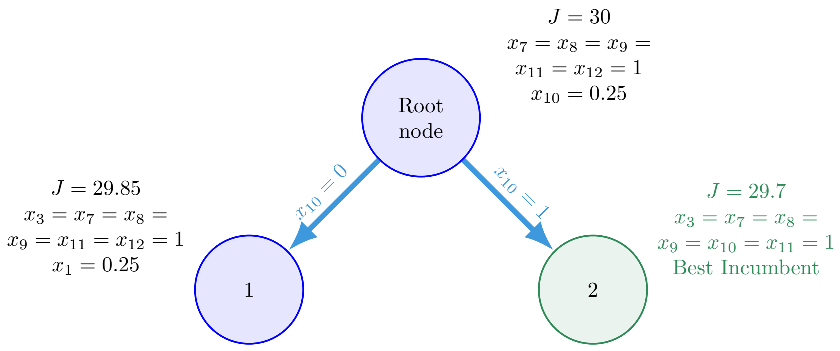
(a) Tree structure



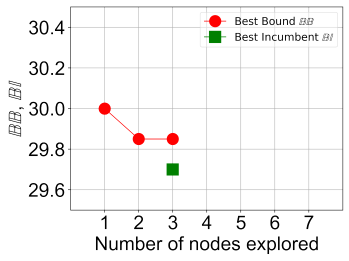
(b) Evolution of $\mathbb{BB}$ and $\mathbb{BI}$.

Figure 6: B&B for the pizza example: two nodes explored.

We can see that the solution is indeed a relaxation, because five decision variables are unitary (x_7, x_8, x_9, x_{11}, and x_{12}), but one is fractional ($x_{10} = 0.25$). This also means the objective value of the root node $J = 30$ is the first $\mathbb{BB}$ while we do not have a $\mathbb{BI}$ yet. Note that, if the solution at the root node had satisfied all integrality constraints already, we would not need any B&B because that solution would be optimal already. On the other hand, we can branch on x_{10} because it is the only fractional decision variable. From the root node, we can generate two children nodes that inherit exactly the MILP properties of the root node, but one of the two will have the additional constraint $x_{10} = 0$, and the other $x_{10} = 1$ instead.
In Fig. 6 we show the case $x_{10} = 0$. We notice that the solution is still fractional, because now $x_1 = 0.25$, and that J dropped from 30 to 29.85. This was expected as adding constraints to a MILP can only worsen its objective. Note that $J = 29.85$ is also the new $\mathbb{BB}$. Let us know consider the case $x_{10} = 1$, still stemming from the root node, in Fig. 7.

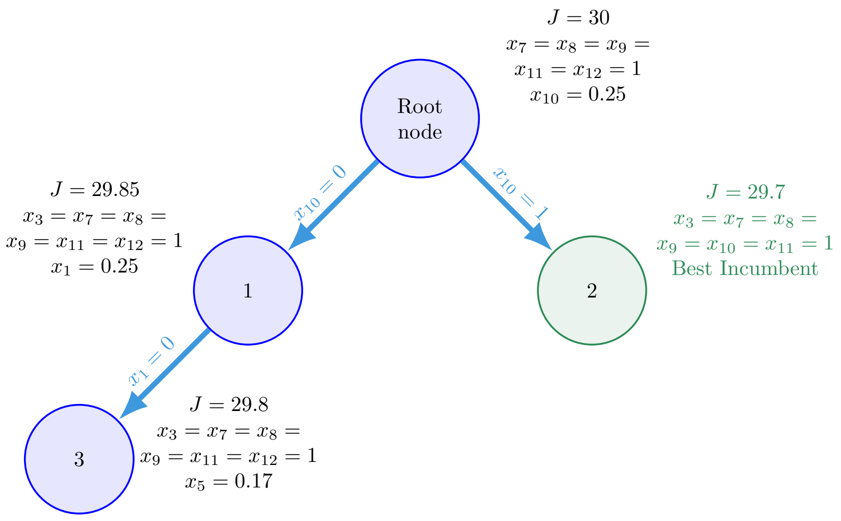


(a) Tree structure

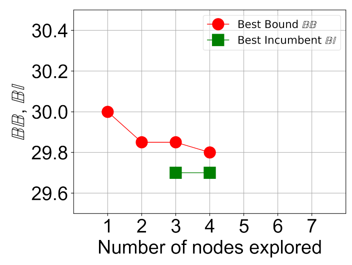


(b) Evolution of $\mathbb{BB}$ and $\mathbb{BI}$.

Figure 7: B&B for the pizza example: three nodes explored.



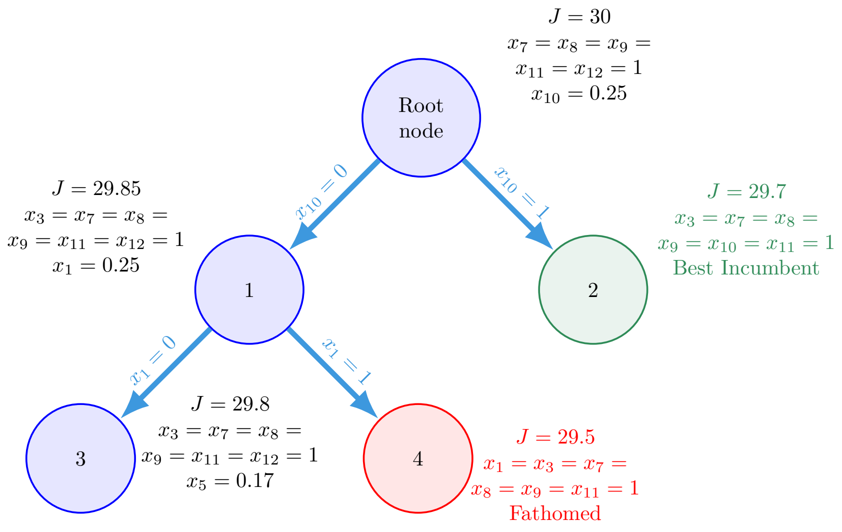
(a) Tree structure



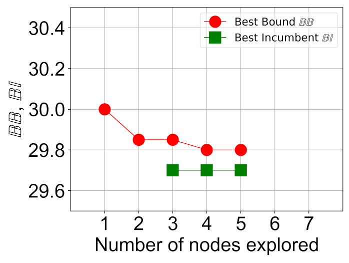
(b) Evolution of $\mathbb{BB}$ and $\mathbb{BI}$.

Figure 8: B&B for the pizza example: four nodes explored.

In this case, a solution that satisfies all integrality constraints is found. Hence, $J = 29.7$ defines our first $\mathbb{BI}$. This also means that we cannot further branch node 2, because there are no fractional decision variables to branch. Conversely, we can still branch from node 1 and create two children MILPs, one where $x_1 = 0$ and the other where $x_1 = 1$. We show the former in Fig. 8. The solution is still fractional, with $\mathbb{BB}$ further decreasing to 29.8. In Fig. 9 we focus on the case $x_1 = 1$ instead.
In node 4, a feasible solution is obtained. Because its objective value is lower than the current $\mathbb{BI}$ ($29.5 < 29.7$), node 4 is dominated by node 2 and hence fathomed. Note that no branches could stem from node 4 anyway, as its solution satisfies integrality constraints. On the other hand, since in node 3 we have a fractional decision variable ($x_5 = 0.17$), we can further branch there. In Figs. 10 and 11 we show, respectively, the cases where $x_5 = 0$ and $x_5 = 1$.
As shown in Fig. 10, node 5 is fathomed because it features a fractional solution ($x_6 = 0.25$) and an objective that is worse than the current $\mathbb{BI}$ ($29.65 < 29.7$). Hence, defining two children nodes, one where $x_6 = 0$ is enforced and the other one where $x_6 = 1$ is enforced, would only lead to an even lower J. Note that $\mathbb{BB}$ is still 29.8 (from node 3), as that is the highest objective value related to a fractional solution that has not been fully branched. In fact, we still need to solve node 6, which is the children of node 3 with the additional constraint $x_5 = 1$, as previously anticipated. Fig. 11 focuses on that node.
Similarly to node 4, node 6 features a solution that satisfies all integrality constraints, but whose J is lower than the current $\mathbb{BI}$ ($28.2 < 28.7$), as shown in Fig. 11a. Hence, the node is fathomed. In addition, no more nodes can be explored, as the four nodes that were not branched yet (nodes 2, 4, 5, and 6) either represent the $\mathbb{BI}$ (node 2), or are fathomed (nodes 4, 5, and 6). As a consequence, the $\mathbb{BB}$ is artificially lowered to 29.7 so that it matches the $\mathbb{BI}$. Since we have reached a condition where $\mathbb{BI} = \mathbb{BB}$, our solution is optimal.

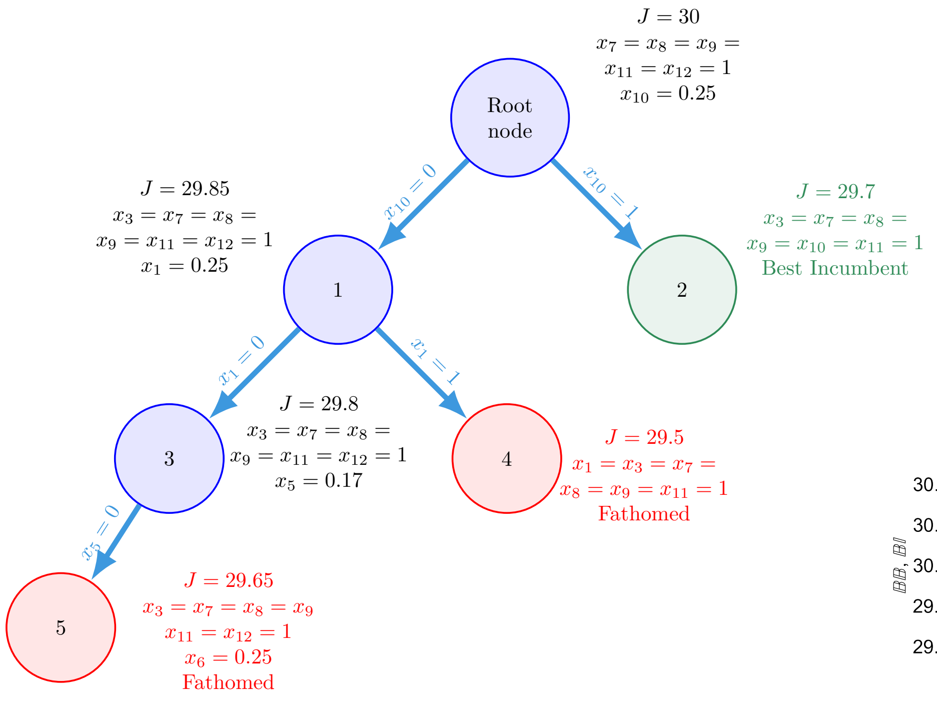


(a) Tree structure

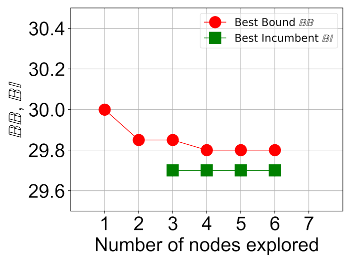


(b) Evolution of $\mathbb{BB}$ and $\mathbb{BI}$.

Figure 9: B&B for the pizza example: five nodes explored.

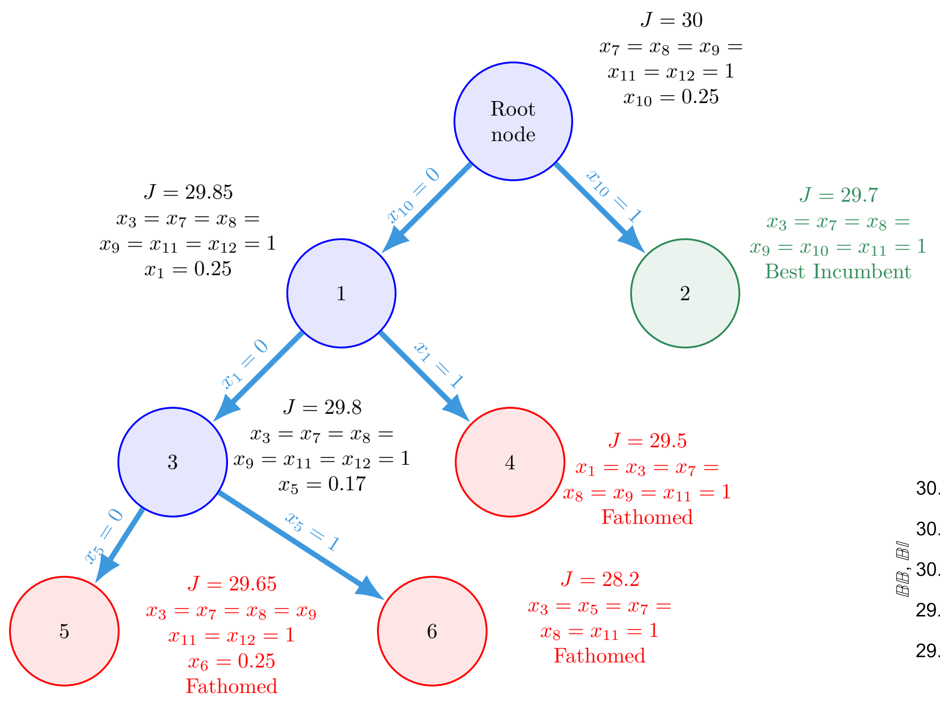


(a) Tree structure

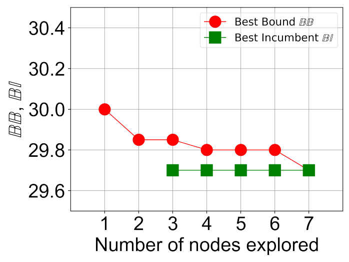


(b) Evolution of $\mathbb{BB}$ and $\mathbb{BI}$.

Figure 10: B&B for the pizza example: six nodes explored.



(a) Tree structure



(b) Evolution of $\mathbb{BB}$ and $\mathbb{BI}$.

Figure 11: B&B for the pizza example: seven nodes explored.

4 What was not covered in this tutorial?

Although this tutorial addresses the B&B solution technique, it was also shown how this technique heavily relies on the Simplex method. While in Sec. 2.1 a recap, together with a well-known problem from the theory, have been shown, it is assumed that readers are familiar with the Simplex method. For example, in the different tableaus it was not shown how to select the entering and exiting basic variables. In a similar fashion, it was not shown how to apply row operations to modify the tableau so that the proper form is achieved. I encourage students to fully understand the Simplex method before even trying to understand B&B.
Addressing now B&B, no mention whatsoever on the computational complexity of such a solution method has been made. While for small problems the B&B solution method finds an optimal solution in a blink of an eye, computational issues might arise quite quickly as the size of the problem grows. While B&B is indeed more efficient than pure enumeration, the size of the full decision tree is so large that such a tree cannot be explored in a limited time, hence yielding large optimality gaps after hours or even days of computation. Sometimes, there are so many nodes and data to be stored, that a B&B could result in memory allocation issues. A way to limit such a problem is to resort to Branch & Cut (B&C), which is basically a smarter version of the B&B that adds constraints (cuts) to the original MILP while exploring the decision tree so that these cuts further eliminate parts of the fractional solution space that would yield no meaningful results. It should be noticed that all these cuts "tighten" the linear relaxation of the original model, to use a bit of jargon, but do not change anyhow the optimal solution to the problem. They are basically ad-hoc constraints that help to further fathom parts of the decision tree. Some well-known cuts are Gomory cuts, half-plane cuts, flow cover cuts, just to name a few.
To summarize, when dealing with large-scale problems, a basic B&B is generally not sufficient, and hence extensions of it (exact methods), meta-heuristics, or combinations of a meta-heuristics with a simpler version of the MILP, i.e., math-heuristics (approximate methods) are needed to get good quality solutions. In practice, for problems that are generally meaningful in real-life situations, what was covered in this tutorial is a good start, but it is definitely not sufficient ☺.

Contacts

This document is a continuous (no pun intended) work-in-progress. As such, comments and feedback on typos, clarifications, or improvements of any kind are very much appreciated. If so, please contact me:

Alessandro Bombelli
a.bombelli@tudelft.nl
Air Transport & Operations section, Control & Operations department
Aerospace engineering faculty, Delft University of Technology