

## Unit Commitment 2 – Solution Method

### 1.0 Introduction

In the previous notes, we observed that our problem is a linear mixed integer programming problem. Before we discuss our chosen method of solving this problem, it is useful to see where our problem lies in the general area of optimization.

Linear mixed integer programs are a particular type of problem that falls under the more general heading of integer programs (IPs). IPs may be *pure*, in which case all the decision variables are integer (PIP), or they may be *mixed*, in which case some decision variables are integer and some are continuous valued (MIP).

It is also possible to have problems where the integer variables may take on one of just two values: 0 and 1. Such problems occur frequently when the decisions to be made are of the “yes or no” type. When all integer variables are this way, the problem is considered to be a binary integer program (BIP).

### 2.0 IPs are hard

The material in this section is adapted from [1, ch 13.3].

There are two obvious approaches that come to mind for solving IPs. One is to check every possible solution. We call this exhaustive enumeration. Another is to solve the problem as a linear program without integrality requirements, and then round the values we get to the nearest integer. We call this LP-relaxation with rounding. Let's look at these two approaches.

## 2.1 Exhaustive enumeration

Consider a BIP problem with 3 variables:  $x_1$ ,  $x_2$ , and  $x_3$ , each of which can be 1 or 0. The possible solutions are

$$(0,0,0), (0,0,1), (0,1,0), (0,1,1), \\ (1,0,0), (1,0,1), (1,1,0), (1,1,1).$$

Thus, there are  $2^3=8$  solutions. It would not be too hard to check them all. But consider more typical problems with 30 variables, or even 300.

$$2^{30}=1.0737 \times 10^9=1,073,700,000 \text{ (over a billion possible solutions)}$$

$$2^{300}=2.037 \times 10^{90}$$

Recall our UC formulation in the previous notes. There were three integer variables for every generator:

- $z_{it}$  is 1 if generator  $i$  is dispatched during  $t$ , 0 otherwise,
- $y_{it}$  is 1 if generator  $i$  starts at beginning of period  $t$ , 0 otherwise,
- $x_{it}$  is 1 if generator  $i$  shuts at beginning of period  $t$ , 0 otherwise,

PJM, for example, has over 1100 generators. For PJM's unit commitment problem, the number of integer decision variables is over 3300. We see there are over  $2^{3300}$  possible solutions! It is for this reason that the UC problem is said to suffer from the "curse of dimensionality."

The conclusion is that exhaustive enumeration is simply not feasible, even for the fastest computers.

## 2.2 LP relaxation with rounding

Here, we relax the requirement that the decision variables be integers. Then our problem becomes a standard LP, and we can solve it efficiently using the simplex method. When we examine the resulting solution, some or all of the variables are likely to be non-integer. These variables are then just rounded to the nearest integer, and we call what results our solution.

This approach may in fact work with reasonable accuracy (it may get close to the optimum) for some problems, especially if the values of the variables are large so that rounding creates relatively little error. (This would not be the case, however, for BIPs, as in the case of the UC, where integer variables are either 1 or 0.)

But there are two pitfalls in this approach.

*Pitfall #1:* The solution obtained by rounding the optimal LP solution is not necessarily feasible. To illustrate, consider a problem having the below two constraints.

$$-x_1 + x_2 \leq 3.5$$

$$x_1 + x_2 \leq 16.5$$

Assume the LP-relaxation found a solution that has  $x_1=6.5$  and  $x_2=10$ , so that

$$-6.5 + 10 = 3.5 \leq 3.5$$

$$6.5 + 10 = 16.5 \leq 16.5$$

If we round  $x_1$  to 7, then

$$-7 + 10 = 3 \leq 3.5$$

$$7 + 10 = 17 \text{ not } \leq 16.5$$

If we round  $x_1$  to 6, then

$$-6 + 10 = 4 \text{ not } \leq 3.5$$

$$6 + 10 = 16 \leq 16.5$$

Thus, either way we go, rounding up or down, we result in an infeasible solution. The only way we can make  $x_1$  an integer is if we also change  $x_2$ .

*Pitfall #2:* Even if the rounded solution is feasible, there is no guarantee it will be optimal or that it will even be reasonably accurate (reasonably close to the optimal). This is illustrated by the following problem.

$$\begin{aligned} \max Z &= x_1 + 5x_2 \\ \text{s.t.} \\ x_1 + 10x_2 &\leq 20 \\ x_1 &\leq 2 \\ x_1 \geq 0, x_2 &\geq 0 \\ x_1, x_2 &\text{ integers} \end{aligned}$$

This problem is illustrated in Fig. 1.

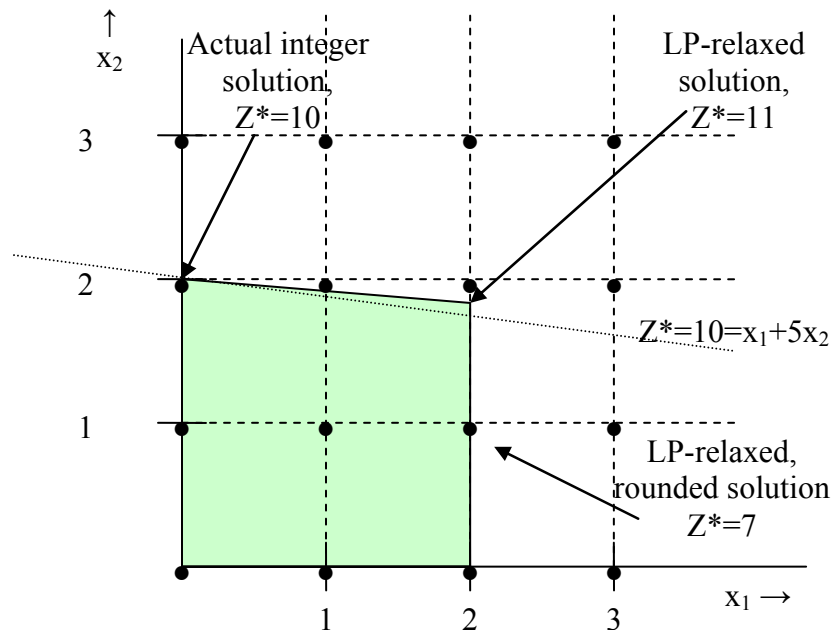


Fig. 1

In Fig. 1, the dots are the possible integer solutions, and the shaded region is the feasible region. The LP-relaxed solution is  $(2, 9/5)$  where  $Z^*=11$ . If we rounded, then we would get  $(2,1)$  where  $Z^*=7$ . But it is easy to see that the point  $(0,2)$  is on the  $Z^*=10$  line. Because  $(2,0)$  is integer and feasible, and higher than  $Z^*=7$ , we see that the rounding approach has failed us miserably.

### 3.0 Some other methods

There are two broad classes of methods to solving IPs:

- Cutting plane methods
- Tree-search methods

#### 3.1 Cutting plane methods

Cutting plane methods generate additional constraints. There are several kinds of cutting plane methods. One of the simplest to understand is the fractional integer programming algorithm [2, ch 13]. In this method, we shrink the feasible region the minimum amount possible so that all corner points are integer. If we are successful in doing this, then the solution to the corresponding relaxed LP will be integer and thus the solution to the original integer program. Figure 2 illustrates.

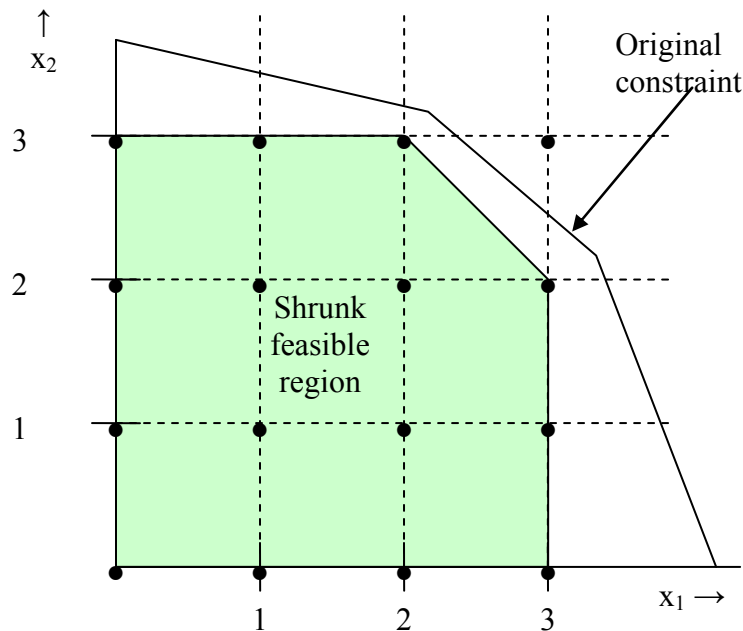


Fig. 2

Another very popular cutting plane method, particularly for MIP, is Benders decomposition [2, Sec 15.2]. To apply this method, the variables must be separable. If they are, one can set up a Master problem to solve for one set of variables and subproblems to solve for the other set. Then the algorithm iterates between Master and subproblems.

### 3.1 Tree-search methods

The essence of tree-search algorithms are that they conceptualize the problem as a huge tree of solutions, and then they try to do some smart things to avoid searching the entire tree. Such a tree is shown in Fig. 3.

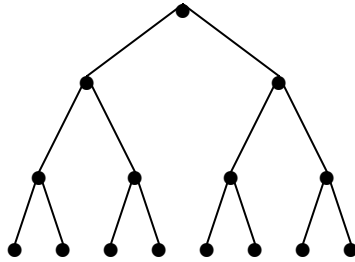


Fig. 3

The common features of tree-search algorithms are [2, App C]:

1. They are easy to understand;
2. They are easy to program on a computer;
3. The upper bound on the number of steps the algorithm needs to find the solution is  $O(k^n)$ , where  $n$  is the number of decision variables (this means running time increases exponentially with the number of variables);
4. They lack mathematical structure.

The most popular tree-search method today is called branch and bound. We will study this method in the next section.

## 4.0 Branch and bound (B&B)

Two definitions are necessary.

- Predecessor
- Successor

Problem  $P_j$  is predecessor to Problem  $P_k$ , and Problem  $P_k$  is successor to Problem  $P_j$ , if they are identical with the exception that one continuous-valued variable in Problem  $P_j$  is constrained to be integer in Problem  $P_j$ .

How to constrain a continuous-valued variable to be integer?

Consider the following problem [3, ch. 23]. We will call it  $P_0$ .

$$\begin{aligned} \max \quad & \zeta = 17x_1 + 12x_2 \\ \text{s.t.} \end{aligned}$$

$$\begin{aligned} P_0 \quad & 10x_1 + 7x_2 \leq 40 \\ & x_1 + x_2 \leq 5 \\ & x_1, x_2 \geq 0 \end{aligned}$$

Solution using CPLEX yields

$$P_0 \text{ Solution: } x_1 = 1.667, x_2 = 3.333, \quad \zeta = 68.333$$

Observe this solution as the corner point labeled “solution” in Fig. 4.

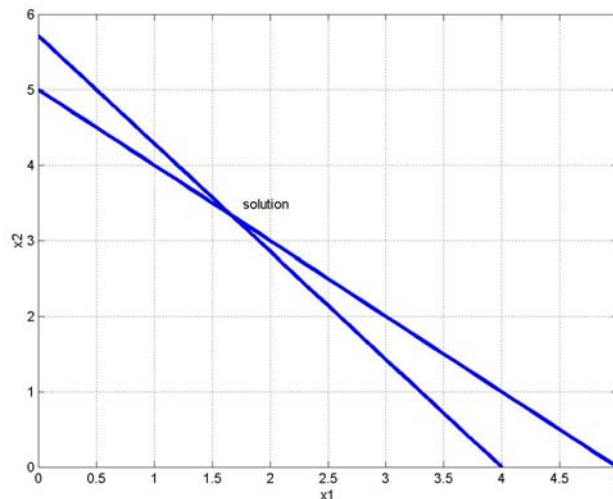


Fig. 4

What if we now pose a problem  $P_1$  to be exactly like  $P_0$  except that we will constrain  $x_1 \leq 1$ ? Here it is:

$$\begin{aligned}
 & \max \quad \zeta = 17x_1 + 12x_2 \\
 & \text{s.t.} \\
 P_1 \quad & 10x_1 + 7x_2 \leq 40 \\
 & x_1 + x_2 \leq 5 \\
 & x_1 \leq 1 \\
 & x_1, x_2 \geq 0
 \end{aligned}$$

What do you expect the value of  $x_1$  to be in the optimal solution?

Because the solution without the constraint  $x_1 \leq 1$  wanted 1.667 of  $x_1$ , we can be sure that the solution with the constraint  $x_1 \leq 1$  will want as much of  $x_1$  as it can get, i.e., it will want  $x_1 = 1$ .

But now let's ask, in terms of constraining a continuous-valued variable to be integer: what have we done here?

1. We solved a *predecessor* problem as a relaxed LP and obtained optimal values (but non-integer) for the variables.
2. Then we constructed a *successor* problem by indirectly imposing integrality on one variable via constraining it to be less than or equal to the integer *just* less than the value of that variable in the optimal solution to the predecessor problem.

Let's use CPLEX to solve  $P_1$  to see if it works. The CPLEX solution is:

$$P_1 \text{ Solution: } \quad x_1 = 1.0, x_2 = 4.0, \quad \zeta = 65.0$$

It worked in that  $x_1$  did in fact become integer. In fact,  $x_2$  became integer as well, but this is by coincidence, i.e., in general, the "trick" of imposing integrality on a variable, as we have done above, is not guaranteed to also impose integrality on the remaining variables.

A related way to think of this is that we just made a new corner point, as illustrated in Fig. 5, that had to be the next “best” (largest  $\zeta$  in this case) corner point to the optimal without the constraint  $x_1 \leq 1$ .

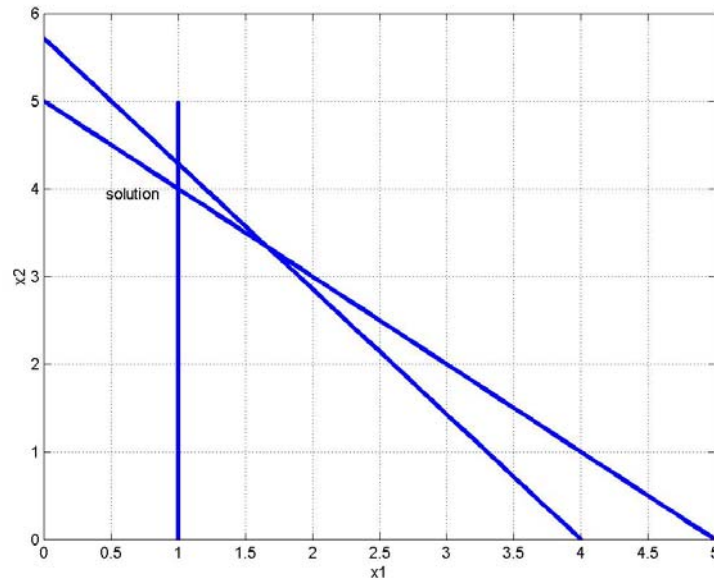


Fig. 5

So we understand now how to constrain a continuous-valued variable to be integer.

But here is another question for you... What if we want to solve the following IP:

$$\begin{aligned}
 \max \quad & \zeta = 17x_1 + 12x_2 \\
 \text{s.t.} \quad & \\
 IP_1 \quad & 10x_1 + 7x_2 \leq 40 \\
 & x_1 + x_2 \leq 5 \\
 & x_1, x_2 \geq 0 \\
 & x_1, x_2 \text{ integers.}
 \end{aligned}$$

This problem is identical to  $P_0$  except we require  $x_1, x_2$  to be integer. The question is: Is the  $P_1$  solution we obtained, which by chance is a *feasible* solution to  $IP_1$ , also the *optimal* solution to  $IP_1$ ?

The answer to this question requires a criterion, and here it is:

**IP Optimality Criterion:** A solution to an IP is optimal if the corresponding objective value is better than the objective value corresponding to every other feasible solution to that IP.

So is the  $P_1$  solution the optimal solution to  $IP_1$ ?

Answer: We do not know. Why?

→ Because we have not yet explored the entire solution space.

What part of the solution space remains? The part associated with  $x_1=2$ . So let's constrain  $x_1 \geq 2$ . This results in problem  $P_2$ .

$$\begin{aligned}
 & \max \quad \zeta = 17x_1 + 12x_2 \\
 & \text{s.t.} \\
 P_2 \quad & 10x_1 + 7x_2 \leq 40 \\
 & x_1 + x_2 \leq 5 \\
 & x_1 \geq 2 \\
 & x_1, x_2 \geq 0
 \end{aligned}$$

Using CPLEX to solve  $P_2$  results in:

$$P_2 \text{ Solution: } x_1 = 2.0, x_2 = 2.857, \quad \zeta = 68.8286$$

This time, we were not so fortunate to obtain a feasible solution to  $IP_1$ , since  $x_2$  is not integer. And so the  $P_2$  solution is not feasible to  $IP_1$ , and therefore it is certainly no optimal to  $IP_1$ . The situation is illustrated in Fig. 6.

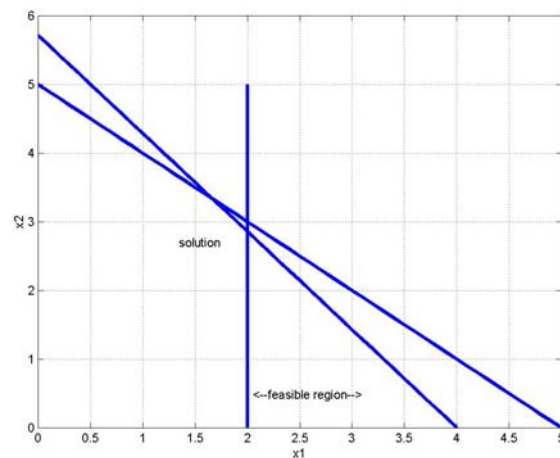


Fig. 6

But does the  $P_2$  solution tell us anything useful?

YES! Compare the objective function value of  $P_2$ , which is 68.8286, with the objective function value of  $P_1$ , which is 65. Since we are maximizing, the objective function value of  $P_2$  is better. But the  $P_2$  solution is not feasible. However, we can constrain  $x_2$  appropriately so that we get a feasible solution. Whether such a feasible solution will have better objective function value we do not know. *What we do know is, because the objective function value of  $P_2$  (68.8286) is better than the objective function value of  $P_1$  (65), it is worthwhile to check it.* Although the objective function value of successor problems to  $P_2$  can only get worse (lower), they might be better than  $P_1$ , and if we can find a successor (or a successor's successor,...) that is feasible, it might be better than our best current feasible solution, which is  $P_1$ .

On the other hand, if  $P_2$  had resulted in an objective function lower than that of  $P_1$ , what would we have done?

→ We would not have evaluated any more successor problems to  $P_2$ . Why? Because successor nodes add constraints, and it is impossible for the objective to get better by adding constraints. Either the objective will get worse, or it will stay the same. But it will not get better. Therefore if a predecessor node is already not as good as the current best feasible solution, there is no way a successor node will be, and we might as well terminate evaluation of successors to that predecessor node.

But in this case, the objective function value of  $P_2$  is better than that of our current best feasible solution, so let's solve  $P_2$  with the additional constraint  $x_2 \leq 2$ . Call this problem  $P_3$ , given below.

$$\max \quad \zeta = 17x_1 + 12x_2$$

s.t.

$$10x_1 + 7x_2 \leq 40$$

$$P_3 \quad x_1 + x_2 \leq 5$$

$$x_1 \geq 2$$

$$x_2 \leq 2$$

$$x_1, x_2 \geq 0$$

Using CPLEX to solve  $P_3$  results in:

$$P_3 \text{ Solution: } x_1 = 2.6, x_2 = 2.0, \quad \zeta = 68.2$$

Note that  $x_1$  has reverted back to non-integer. We could have expected this since we forced  $x_2$  to change, requiring a new corner point, as shown in Fig. 7.

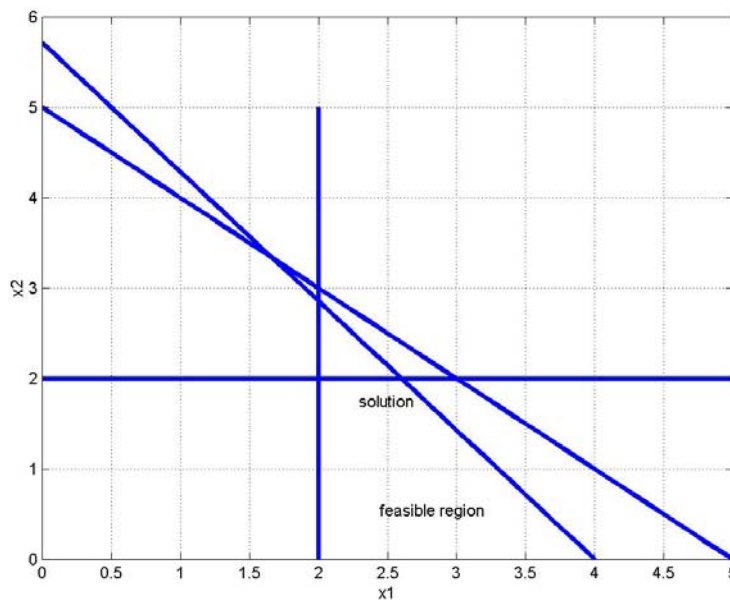


Fig. 7

Now we need to force a feasible solution, and we do so by imposing  $x_1 \leq 2$ , as indicated in problem  $P_4$  below.

$$\begin{aligned}
 & \max \quad \zeta = 17x_1 + 12x_2 \\
 & \text{s.t.} \\
 & \quad 10x_1 + 7x_2 \leq 40 \\
 P_4 \quad & \quad x_1 + x_2 \leq 5 \\
 & \quad x_1 \leq 2 \\
 & \quad x_2 \leq 2 \\
 & \quad x_1, x_2 \geq 0
 \end{aligned}$$

Using CPLEX to solve  $P_4$ , we obtain:

$$P_4 \text{ Solution : } x_1 = 2.0, x_2 = 2.0, \quad \zeta = 58.0$$

This solution is displayed in Fig. 8.

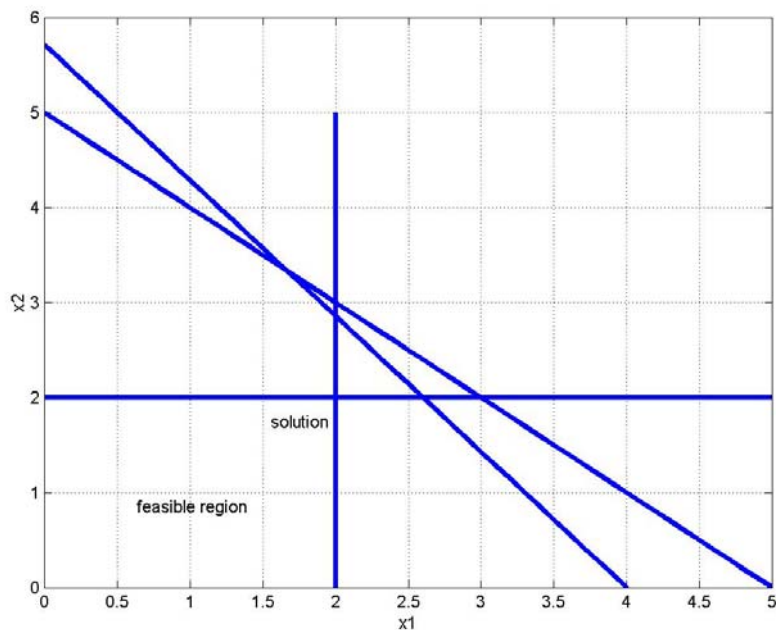


Fig. 8

This solution is feasible! Wonderful. However, comparing the objective value function to our “best so-far” value of 65, we see that this value, 58, is worse. So the  $P_4$  solution is not of interest to us since we already have one that is better.

What next?

Let's go back to problem  $P_3$  where we had this:

$$P_3 \text{ Solution: } x_1 = 2.6, x_2 = 2.0, \quad \zeta = 68.2$$

In  $P_4$ , we pushed  $x_1$  to 2. Let's explore the space associated with pushing  $x_1$  to 3. So we need a new problem for this, where we require  $x_1 \geq 3$ . Let's call this  $P_5$ , given below.

$$\begin{aligned} & \max \quad \zeta = 17x_1 + 12x_2 \\ & \text{s.t.} \\ & \quad 10x_1 + 7x_2 \leq 40 \\ P_5 \quad & \quad x_1 + x_2 \leq 5 \\ & \quad x_1 \geq 3 \\ & \quad x_2 \leq 2 \\ & \quad x_1, x_2 \geq 0 \end{aligned}$$

Using CPLEX to solve it, we obtain

$$P_5 \text{ Solution: } x_1 = 3.0, x_2 = 1.4286, \quad \zeta = 68.1429$$

This solution is not integer, but it has an objective function value better than our current best feasible solution (65), and so let's branch again to a successor node with  $x_2 \leq 1$ . Let's call this  $P_6$ , given below.

$$\begin{aligned} & \max \quad \zeta = 17x_1 + 12x_2 \\ & \text{s.t.} \\ & \quad 10x_1 + 7x_2 \leq 40 \\ P_6 \quad & \quad x_1 + x_2 \leq 5 \\ & \quad x_1 \geq 3 \\ & \quad x_2 \leq 1 \\ & \quad x_1, x_2 \geq 0 \end{aligned}$$

Using CPLEX to solve it, we obtain

$$P_6 \text{ Solution: } x_1 = 3.3, x_2 = 1.0, \quad \zeta = 68.1$$

Again, the solution is not integer, but it has an objective function value better than our current best feasible solution (65), and so let's branch again to a successor node with  $x_1 \leq 3$ . Let's call this  $P_7$ , given below:

$$\begin{array}{ll}
 \max & \zeta = 17x_1 + 12x_2 \\
 \text{s.t.} & \\
 & 10x_1 + 7x_2 \leq 40 \\
 P_7 & x_1 + x_2 \leq 5 \\
 & x_1 \leq 3 \\
 & x_2 \leq 1 \\
 & x_1, x_2 \geq 0
 \end{array}$$

Using CPLEX to solve it, we obtain

$$P_7 \text{ Solution: } x_1 = 3.0, x_2 = 1.0, \quad \zeta = 63.0$$

This solution is feasible! However, comparing the objective value function to our "best so-far" value of 65, we see that this value, 63, is worse. So the  $P_7$  solution is not of interest to us since we already have one that is better.

So now what?

Let's go back to problem  $P_6$  where we had this:

$$P_6 \text{ Solution: } x_1 = 3.3, x_2 = 1.0, \quad \zeta = 68.1$$

In  $P_7$ , we pushed  $x_1$  to 3. Let's explore the space associated with pushing  $x_1$  to 4. So we need a new problem for this, where we require  $x_1 \geq 4$ . Let's call this  $P_8$ , given below.

$$\begin{array}{ll}
\max & \zeta = 17x_1 + 12x_2 \\
\text{s.t.} & \\
& 10x_1 + 7x_2 \leq 40 \\
P_8 & x_1 + x_2 \leq 5 \\
& x_1 \geq 4 \\
& x_2 \leq 1 \\
& x_1, x_2 \geq 0
\end{array}$$

Using CPLEX to solve it, we obtain

$$P_8 \text{ Solution: } x_1 = 4.0, x_2 = 0, \quad \zeta = 68.0$$

This solution is feasible! Not only that, but when we compare the objective value function to our “best so-far” value of 65, we see that this value, 68, is better. So the  $P_8$  solution becomes our new “best solution.” And we will use 68 as our bound against which we will test other solutions.

What other solutions do we need to test?

Let’s go back to problem  $P_2$  where we had this:

$$P_2 \text{ Solution: } x_1 = 2.0, x_2 = 2.857, \quad \zeta = 68.8286$$

In  $P_3$ , we pushed  $x_2$  to 2. Let’s explore the space associated with pushing  $x_2$  to 3. So we need a new problem for this, where we require  $x_1 \geq 2$ . Let’s call this  $P_9$ , given below.

$$\begin{array}{ll}
\max & \zeta = 17x_1 + 12x_2 \\
\text{s.t.} & \\
& 10x_1 + 7x_2 \leq 40 \\
P_9 & x_1 + x_2 \leq 5 \\
& x_1 \geq 2 \\
& x_2 \geq 3 \\
& x_1, x_2 \geq 0
\end{array}$$

Using CPLEX to solve, we learn that Problem  $P_9$  is infeasible.

We have one more branch to check...

Let's go back to problem  $P_5$  where we had this:

$$P_5 \text{ Solution: } x_1 = 3.0, x_2 = 1.4286, \quad \zeta = 68.1429$$

In  $P_6$ , we pushed  $x_2$  to 1. Let's explore the space associated with pushing  $x_2$  to 2. So we need a new problem for this, where we require  $x_2 \geq 2$ . Let's call this  $P_{10}$ , given below.

$$\begin{array}{ll} P_{10} & \max \quad \zeta = 17x_1 + 12x_2 \\ & \text{s.t.} \\ & 10x_1 + 7x_2 \leq 40 \\ & x_1 + x_2 \leq 5 \\ & x_1 \geq 3 \\ & x_2 \geq 2 \\ & x_1, x_2 \geq 0 \end{array}$$

Using CPLEX to solve, we learn that Problem  $P_{10}$  is infeasible.

The tree-search is illustrated in Fig. 9 below.

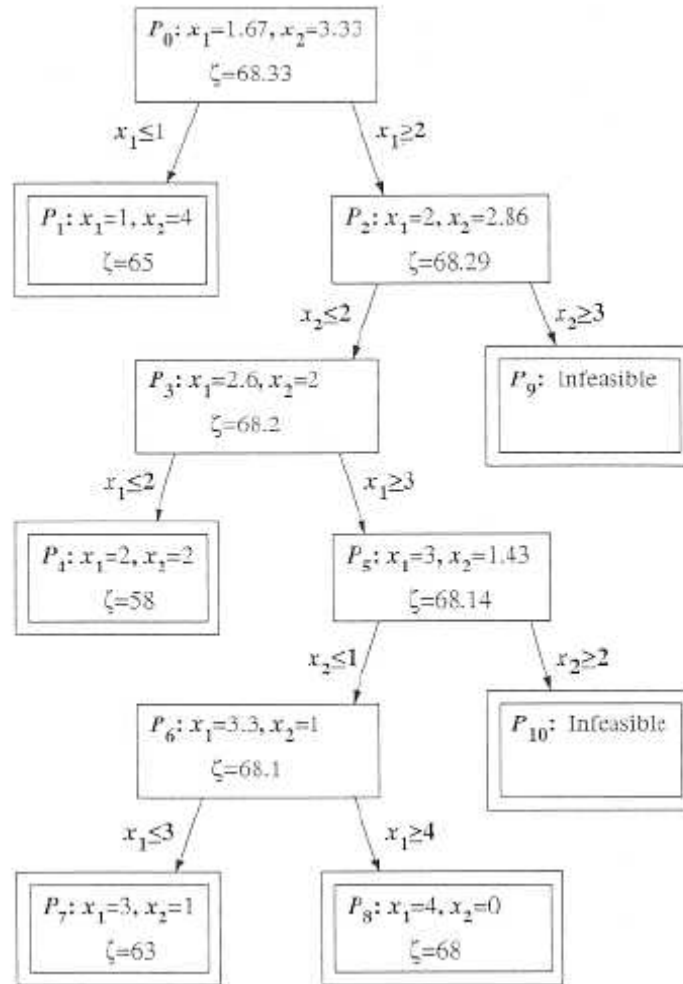


Fig. 9 [3]

There are two central ideas in the B&B method.

1. *Branch*: It uses LP-relaxation to decide how to branch. Each branch will add a constraint to the previous LP-relaxation to enforce integrality on one variable that was not integer in the predecessor solution.
2. *Bound*: It maintains the best integer feasible solution obtained so far as a bound on tree-paths that should still be searched.
  - a. If any tree node has an objective value less optimal than the identified bound, no further searching from that node is necessary, since adding constraints can never improve an objective.

- b. If any tree node has an objective value more optimal than the identified bound, then additional searching from that node is necessary.

In the B&B method, the first thing to do is to solve the problem as an LP-relaxation. If the solution results so that all integer variables are indeed integer, the problem is solved.

On the other hand, if the LP-relaxation results in a solution that contains a particular variable  $x_k = [x_k] + s_k$ , where  $0 < s_k < 1$ , and  $[x_k]$  is an integer, then we branch to two more linear programs

- The predecessor problem except with the additional constraint:  $x_k = [x_k]$
- The predecessor problem except with the additional constraint  $x_k = [x_k] + 1$

## 5.0 Other algorithmic issues on B&B

In section 4.0, we identified the essence of the B&B algorithm:

Branch: force integrality on one variable by adding a constraint to an LP-relaxation.

Bound: Continue branching only if the objective function value of the current solution is better than the objective function value of the best feasible solution obtained so far.

It is our primary goal to have understood this.

However, you should be aware that there are several other issues related to B&B that need to be considered before implementation. Below is a brief summary of two of these issues [4, ch. 6].

## 5.1 Node selection

The issue here is this:

➔ Following examination of a node in which we conclude we need to branch, should we go deeper, or should we broader?

Consider, in the example of Section 4.0, the situation we were in at the node corresponding to Problem  $P_3$ . Reference to Fig. 9 above will be useful at this point. The solution we obtained was

$$P_3 \text{ Solution: } x_1 = 2.6, x_2 = 2.0, \quad \zeta = 68.2$$

And so it was clear to us, because our current best feasible solution had an objective function value of 65, and  $68.2 > 65$ , we needed to branch further on this node.

However, consider  $P_3$ 's predecessor node,  $P_2$ . Its solution was

$$P_2 \text{ Solution: } x_1 = 2.0, x_2 = 2.857, \quad \zeta = 68.8286$$

To get to  $P_3$ , we had added the constraint  $x_2 \leq 2$ . But there was something else we could have done, and that was added the constraint  $x_2 \geq 3$ . This was the other branch. We had to pick one or the other, and we decided on  $x_2 \leq 2$ .

But once at  $P_3$ , after evaluating it, we had a tough decision to make.

- Depth: Do we continue from  $P_3$ , requiring  $x_1 \leq 2$ , for example? or
- Breadth: Do we go back to  $P_2$  to examine its other branch,  $x_2 \geq 3$ ?

For high-dimensional IPs, it is usually the case that feasible solutions are more likely to occur deep in a tree than at nodes near the root. Finding multiple feasible solutions early in B&B is important because it tightens the bound (in the above example, it increases the bound), and therefore enables termination of branching at more nodes (and therefore decreases computation). One can see that this will be the case if we consider the bound before we find a feasible solution: the bound is infinite! ( $+\infty$  for maximization problems and  $-\infty$  for minimization problems).

## 5.2 Branching variable selection

In the example of Section 4.0, branching variable selection was not an issue. At every node, there was only, at most, one non-integer variable, and so there was no decision to be made. This was because our example problem had only two variables,  $x_1$  and  $x_2$ .

However, if we consider having any more than 2 decision variables, even 3, we run into this issue, and that is:

➔ Given we are at a node that needs to branch, and there are 2 or more non-integer variables, which one do we branch on?

This is actually a rich research question that, so far, has not been solved generically but rather, is addressed for individually for individual problem types by pre-specifying an *ordering* of the variables that are required to be integer. Good orderings become apparent after running the algorithm many times for many different conditions. Sometimes, good orderings are apparent based on some physical understanding of the problem, e.g., perhaps the largest unit should be chosen first.

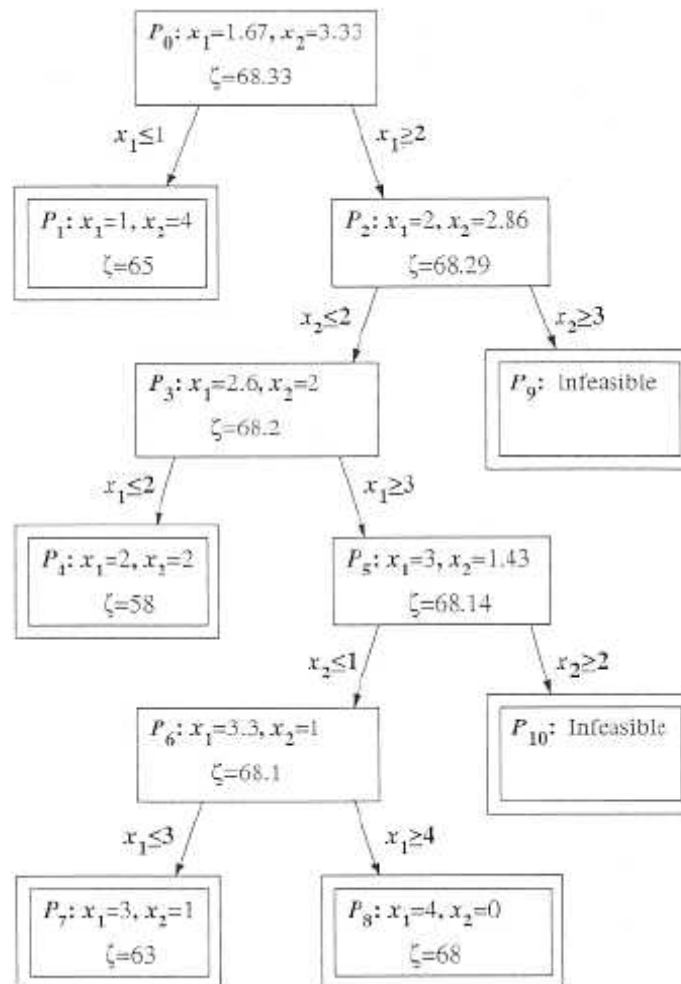
## 5.3 Mixed integer problems

Our example focused on a pure-integer program (PIP). However, the UC is a mixed integer problem (MIP). Do we need to consider this more deeply?

Reconsider our example problem, except now consider it as a mixed integer problem as follows:

$$\begin{aligned}
 & \max \quad \zeta = 17x_1 + 12x_2 \\
 & \text{s.t.} \\
 IP_2 \quad & 10x_1 + 7x_2 \leq 40 \\
 & x_1 + x_2 \leq 5 \\
 & x_1, x_2 \geq 0 \\
 & x_1 \text{ integer.}
 \end{aligned}$$

Recall our tree-search diagram, repeated here for convenience. You should be able to see that the solution is obtained as soon as we solved  $P_1$  and  $P_2$ .



The other possible MIP would be to require  $x_2$  integer only, as follows:

$$\begin{array}{ll}
 \max & \zeta = 17x_1 + 12x_2 \\
 \text{s.t.} & \\
 IP_3 & 10x_1 + 7x_2 \leq 40 \\
 & x_1 + x_2 \leq 5 \\
 & x_1, x_2 \geq 0 \\
 & x_2 \text{ integer.}
 \end{array}$$

Recalling the solution to  $P_0$ ,

$$P_0 \text{ Solution: } x_1 = 1.667, x_2 = 3.333, \quad \zeta = 68.333$$

we see that we can branch by setting either  $x_2 \leq 3$  or  $x_2 \geq 4$ , resulting in

$x_2 \leq 3$ :

$$\text{Solution: } x_1 = 1.9, x_2 = 3.0, \quad \zeta = 68.3$$

$x_2 \geq 4$ :

$$\text{Solution: } x_1 = 1.0, x_2 = 4.0, \quad \zeta = 65$$

And so we know the solution is the one for  $x_2 \leq 3$ .

The conclusion here is that we can very easily solve MIP within our LP-relaxation branch and bound scheme by simply allowing the non-integer variables to remain relaxed.

## 6.0 Using CPLEX to solve MIPs directly

Here is the sequence I used to call CPLEX's MIP-solver to solve our example problem above.

1. Created the problem statement within a text file called mip.lp as follows:

```
maximize
  17 x1 + 12 x2
subject to
  10 x1 + 7 x2 <= 40
  x1 + x2 <= 5
Bounds
  0<= x1 <= 1000
  0<= x2 <= 1000
Integer
  x1 x2
end
```

3. Used WinSCP to port the file to pluto.
4. Used Putty to log on to pluto.
5. Typed `cplex101` to call cplex.
6. Typed `read mip.lp` to read in problem statement.
7. Typed `mipopt` to call the MIP-solver. The result was as follows:

Tried aggregator 1 time.  
MIP Presolve modified 3 coefficients.  
Reduced MIP has 2 rows, 2 columns, and 4 nonzeros.  
Presolve time = 0.00 sec.  
MIP emphasis: balance optimality and feasibility.  
Root relaxation solution time = 0.00 sec.

Nodes				Cuts/			
Node	Left	Objective	IInf	Best Integer	Best Node	ItCnt	Gap
0	0	68.3333	2		68.3333	2	
*	0+	0		65.0000	68.3333	2	5.13%
*		68.0000	0	68.0000	Cuts: 3	4	0.00%

Mixed integer rounding cuts applied: 1

MIP - Integer optimal solution: Objective = 6.8000000000e+01  
Solution time = 0.00 sec. Iterations = 4 Nodes = 0

8. Typed display solution variables - to display solution variables.  
The result was:

Variable Name	Solution Value
x1	4.000000

All other variables in the range 1-2 are 0.

HW 8, Due Friday 4/18/08:

Using branch-and-bound to solve problem 1 and problem 2. For both problems:

- a. Solve them using successive LP-relaxations, where each LP-relaxation is solved using the CPLEX (or Matlab) LP-solver.
- b. Solve them using a MIP-solver. For this one, you should be aware that Matlab does have a MIP-solver, but I am not sure how good it is. I am quite sure the CPLEX solver is very good.

Problem 1:

$$\text{Max } z = 5x_1 + 2x_2$$

$$\text{s.t. } 3x_1 + x_2 \leq 12$$

$$x_1 + x_2 \leq 5$$

$$x_1, x_2, \geq 0; x_1, x_2 \text{ integer}$$

Problem 2:

$$\text{Max } z = 3x_1 + x_2$$

$$\text{s.t. } 5x_1 + 2x_2 \leq 10$$

$$4x_1 + x_2 \leq 7$$

$$x_1, x_2, \geq 0; x_2 \text{ integer}$$

---

[1] F. Hillier and G. Lieberman, "Introduction to Operations Research," fourth edition, Holden-Day, 1986.

[2] T. Hu, "Integer Programming and Network Flows," Addison-Wesley, 1970.

[3] R. Vanderbei, "Linear Programming, Foundations and Extensions," third edition, Springer, 2008.

[4] G. Nemhauser, A. Kan, and M. Todd, editors, "Handbook in Operations Research and Management Science, Volume 1: Optimization," North Holland, Amsterdam, 1989.