Integer Programming - II

Outline

- Some Perspectives on Solving Integer Programming Problems
- The Branch-and-Bound Algorithm and Its Application to Binary Integer Programming
- A Branch-and-Bound Algorithm for Mixed Integer Programming

Some Perspectives on Solving Integer Programming Problems

- It may seem that IP problems should be relatively easy to solve.
- After all, *linear programming* problems can be solved extremely efficiently, and the only difference is that IP problems have far fewer solutions to be considered.
- In fact, *pure* IP problems with a bounded feasible region are guaranteed to have just a *finite* number of feasible solutions.

3

Some Perspectives on Solving Integer Programming Problems

- Unfortunately, there are two fallacies in this line of reasoning.
- One is that having a finite number of feasible solutions ensures that the problem is readily solvable.
- Finite numbers can be astronomically large.
- For example, consider the simple case of BIP problems.
- With n variables, there are 2^n solutions to be considered (where some of these solutions can subsequently be discarded because they violate the functional constraints). Thus, each time n is increased by 1, the number of solutions is *doubled*.

4

Some Perspectives on Solving Integer Programming Problems

- This pattern is referred to as the exponential growth of the difficulty of the problem.
- With *n* = 10, there are more than 1,000 solutions (1,024); with *n* = 20, there are more than 1,000,000; with *n* = 30, there are more than 1 billion; and so forth.
- Therefore, even the fastest computers are incapable of performing exhaustive enumeration (checking each solution for feasibility and, if it is feasible, calculating the value of the objective value) for BIP problems with more than a few dozen variables, let alone for *general* IP problems with the same number of integer variables.

5

Some Perspectives on Solving Integer Programming Problems

- The best algorithms today are capable of solving *many* pure BIP problems with a few hundred variables and *some* considerably larger ones (including certain problems with several tens of thousands of variables).
- Nevertheless, because of *exponential growth*, even the best algorithms cannot be guaranteed to solve every relatively small problem (less than a hundred binary or integer variables).
- Depending on their characteristics, certain relatively small problems can be much more difficult to solve than some much larger ones.

6

Some Perspectives on Solving Integer Programming Problems

- The second fallacy is that removing some feasible solutions (the noninteger ones) from a linear programming problem will make it easier to solve.
- To the contrary, it is only because all these feasible solutions are there that the guarantee can be given (see Sec. 5.1) that there will be a corner-point feasible (CPF) solution [and so a corresponding basic feasible (BF) solution] that is optimal for the overall problem.
- *This* guarantee is the key to the remarkable efficiency of the simplex method.
- As a result, linear programming problems generally are *much* easier to solve than IP problems.

7

Some Perspectives on Solving Integer Programming Problems

- Consequently, most successful algorithms for integer programming incorporate the simplex method (or dual simplex method) as much as they can by relating portions of the IP problem under consideration to the corresponding linear programming problem (i.e., the same problem except that the integer restriction is deleted).
- For any given IP problem, this corresponding linear programming problem commonly is referred to as its LP relaxation.
- We will look at algorithms that illustrate how a sequence of LP relaxations for portions of an IP problem can be used to solve the overall IP problem effectively.

8

Some Perspectives on Solving Integer Programming Problems

- Since IP problems are in general much more difficult to solve than linear programming problems, sometimes it is tempting to use the approximate procedure of simply applying the simplex method to the LP relaxation and then *rounding* the noninteger values to integers in the resulting solution.
- This approach may be adequate for some applications, especially if the values of the variables are quite large so that rounding creates relatively little error.
- However, you should beware of two pitfalls involved in this approach.

9

Some Perspectives on Solving Integer Programming Problems

- One pitfall is that an optimal linear programming solution is *not necessarily feasible* after it is rounded.
- Often it is difficult to see in which way the rounding should be done to retain feasibility.
- It may even be necessary to change the value of some variables by one or more units after rounding.

10

Some Perspectives on Solving Integer Programming Problems

- To illustrate, consider the following problem:

Maximize $Z = x_2$

subject to

$$-x_1 + x_2 \leq \frac{1}{2}$$

$$x_1 + x_2 \leq 3\frac{1}{2}$$

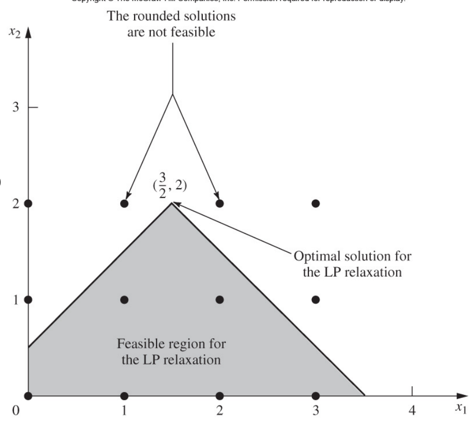
and

$$x_1 \geq 0, \qquad x_2 \geq 0$$

x_1, x_2 are integers.

Some Perspectives on Solving Integer Programming Problems

- As shown in the figure, the optimal solution for the LP relaxation is $x_1 = 1\frac{1}{2}$, $x_2 = 2$, but it is impossible to round the noninteger variable x_1 to 1 or 2 (or any other integer) and retain feasibility.
- Feasibility can be retained only by also changing the integer value of x_2.



- It is easy to imagine how such difficulties can be compounded when there are tens or hundreds of constraints and variables.

Some Perspectives on Solving Integer Programming Problems

- Even if an optimal solution for the LP relaxation is rounded successfully, there remains another pitfall.
- There is no guarantee that this rounded solution will be the optimal integer solution.
- In fact, it may even be far from optimal in terms of the value of the objective function.

13

Some Perspectives on Solving Integer Programming Problems

- This fact is illustrated by the following problem:

Maximize $Z = x_1 + 5x_2$

subject to

$$x_1 + 10\,x_2 \le 20$$
$$x_1 \qquad\qquad \le 2$$

and

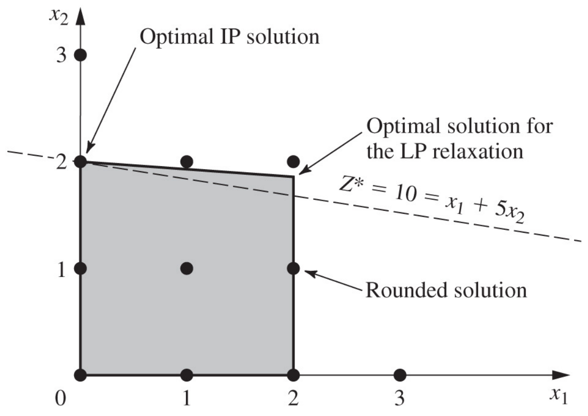
$$x_1 \ge 0, \qquad x_2 \ge 0$$

x_1, x_2 are integers.

14

Some Perspectives on Solving Integer Programming Problems

- Since there are only two decision variables, this problem can be depicted graphically as shown below:



15

Some Perspectives on Solving Integer Programming Problems

- Either the graph or the simplex method may be used to find that the optimal solution for the LP relaxation is $x_1 = 2,\ x_2 = \frac{9}{5}$, with $Z = 11$.
- If a graphical solution were not available (which would be the case with more decision variables), then the variable with the noninteger value $x_2 = \frac{9}{5}$ would normally be rounded in the feasible direction to $x_2 = 1$. The resulting integer solution is $x_1 = 2,\ x_2 = 1$, which yields $Z = 7$.
- Notice that this solution is far from the optimal solution $(x_1, x_2) = (0, 2)$, where $Z = 10$.

16

Some Perspectives on Solving Integer Programming Problems

- The most popular mode for IP algorithms is to use the *branch-and-bound technique* and related ideas to *implicitly enumerate* the feasible integer solutions, and we shall focus on this approach.
- We will next present the branch-and-bound technique in a general context and illustrate it with a basic branch-and-bound algorithm for BIP problems.
- Then, we will present another algorithm of the same type for general MIP problems.

17

The Branch-and-Bound Technique and Its Application to BIP

- Because any bounded *pure* IP problem has only a finite number of feasible solutions, it is natural to consider using some kind of *enumeration procedure* for finding an optimal solution.
- Unfortunately, as we discussed earlier, this finite number can be, and usually is, very large.
- Therefore, it is imperative that any enumeration procedure be cleverly structured so that only a tiny fraction of the feasible solutions actually need be examined.
- For example, dynamic programming provides one such kind of procedure for many problems having a finite number of feasible solutions (although it is not particularly efficient for most IP problems).
- Another such approach is provided by the *branch-and-bound technique*.
- This technique and variations of it have been applied with some success to a variety of OR problems, but it is especially well-known for its application to IP problems.

18

The Branch-and-Bound Technique and Its Application to BIP

- The basic concept underlying the branch-and-bound technique is to *divide and conquer*.
- Since the original “large” problem is too difficult to be solved directly, it is divided into smaller and smaller subproblems until these subproblems can be conquered.
- The dividing (*branching*) is done by partitioning the entire set of feasible solutions into smaller and smaller subsets.
- The conquering (*fathoming*) is done partially by *bounding* how good the best solution in the subset can be and then discarding the subset if its bound indicates that it cannot possibly contain an optimal solution for the original problem.

The Branch-and-Bound Technique and Its Application to BIP

- We shall now describe in turn these three basic steps—branching, bounding, and fathoming—and illustrate them by applying a branch-and-bound algorithm to the prototype example (the California Manufacturing Co. problem) presented before and repeated here (with the constraints numbered for later reference).

Maximize $Z = 9x_1 + 5x_2 + 6x_3 + 4x_4$

subject to

(1) $6x_1 + 3x_2 + 5x_3 + 2x_4 \leq 10$

(2) $x_3 + x_4 \leq 1$

(3) $-x_1 + x_3 \leq 0$

(4) $-x_2 + x_4 \leq 0$

and

(5) x_j is binary, for j = 1, 2, 3, 4.

Branching

- When you are dealing with binary variables, the most straightforward way to partition the set of feasible solutions into subsets is to fix the value of one of the variables (say, x_1) at $x_1 = 0$ for one subset and at $x_1 = 1$ for the other subset.
- Doing this for the prototype example divides the whole problem into the two smaller subproblems shown below.

21

Branching

Maximize $Z = 9x_1 + 5x_2 + 6x_3 + 4x_4$

Subproblem 1:

- Fix $x_1 = 0$ so the resulting subproblem is

Maximize $Z = 5x_2 + 6x_3 + 4x_4$

subject to

(1) $3x_2 + 5x_3 + 2x_4 \leq 10$

(2) $x_3 + x_4 \leq 1$

(3) $x_3 \leq 0$

(4) $-x_2 + x_4 \leq 0$

(5) x_j is binary, for $j = 2, 3, 4$.

Subproblem 2:

- Fix $x_1 = 1$ so the resulting subproblem is

Maximize $Z = 9 + 5x_2 + 6x_3 + 4x_4$

subject to

(1) $3x_2 + 5x_3 + 2x_4 \leq 4$

(2) $x_3 + x_4 \leq 1$

(3) $x_3 \leq 1$

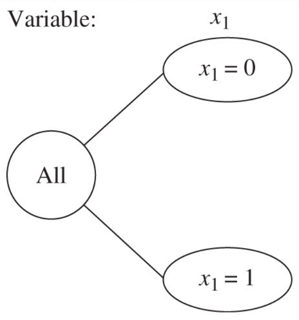
(4) $-x_2 + x_4 \leq 0$

(5) x_j is binary, for $j = 2, 3, 4$.

22

Branching

- The figure below portrays this dividing (branching) into subproblems by a *tree* with *branches* (arcs) from the *All* node (corresponding to the whole problem having *all* feasible solutions) to the two nodes corresponding to the two subproblems.



Branching

- This tree, which will continue "growing branches" iteration by iteration, is referred to as the solution tree (or enumeration tree) for the algorithm.
- The variable used to do this branching at any iteration by assigning values to the variable (as with x_1 above) is called the branching variable.
- Sophisticated methods for selecting branching variables are an important part of some branch-and-bound algorithms but, for simplicity, we always select them in their natural order—$x_1, x_2, \ldots, x_n$—throughout this discussion.

Branching

- Later, we will see that one of these subproblems can be conquered (fathomed) immediately, whereas the other subproblem will need to be divided further into smaller subproblems by setting $x_2 = 0$ or $x_2 = 1$.
- For other IP problems where the integer variables have more than two possible values, the branching can still be done by setting the branching variable at its respective individual values, thereby creating more than two new subproblems.
- However, a good alternate approach is to specify a range of values (for example, $x_j \leq 2$ or $x_j \geq 3$) for the branching variable for each new subproblem.
- This is the approach used for the second algorithm we will present.

25

Bounding

- For each of these subproblems, we now need to obtain a *bound* on how good its best feasible solution can be.
- The standard way of doing this is to quickly solve a simpler *relaxation* of the subproblem.
- In most cases, a relaxation of a problem is obtained simply by *deleting* ("relaxing") one set of constraints that had made the problem difficult to solve.
- For IP problems, the most troublesome constraints are those requiring the respective variables to be integer.
- Therefore, the most widely used relaxation is the *LP relaxation* that deletes this set of constraints.

26

Bounding

- To illustrate for the example, consider first the whole problem given in the prototype example (California Manufacturing Company).
- Its LP relaxation is obtained by replacing the last line of the model (x_j is binary, for j = 1, 2, 3, 4) by the constraints that $x_j \leq 1$ and $x_j \geq 0$ for j = 1, 2, 3, 4.
- Using the simplex method to quickly solve this LP relaxation yields its optimal solution.

  $(x_1,\ x_2,\ x_3,\ x_4) = \left(\frac{5}{6}, 1, 0, 1\right),$ with $\mathrm{Z} = 16\frac{1}{2}$.

27

Bounding

- Therefore, $\mathrm{Z} = 16\frac{1}{2}$ for all feasible solutions for the original BIP problem (since these solutions are a subset of the feasible solutions for the LP relaxation).
- In fact, as summarized below, this *bound* of $16\frac{1}{2}$ can be rounded down to 16, because all coefficients in the objective function are integer, so all integer solutions must have an integer value for Z.

  Bound for whole problem: $Z \leq 16$.

28

Bounding

- Now let us obtain the bounds for the two subproblems in the same way.
- Their LP relaxations are obtained from the models in the "Branching" discussion by replacing the constraints that x_j is binary for j = 2, 3, 4 by the constraints $0 \leq x_j \leq 1$ for j = 2, 3, 4.

29

Bounding

- Applying the simplex method then yields their optimal solutions (plus the fixed value of x_1) shown below.

  LP relaxation of subproblem 1: $(x_1, x_2, x_3, x_4) = (0, 1, 0, 1)$ with $Z = 9$

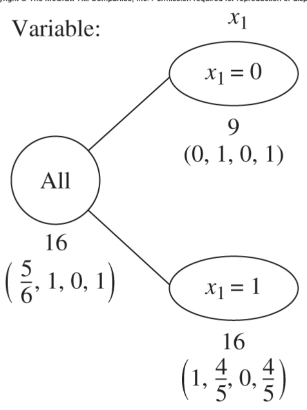
  LP relaxation of subproblem 2: $(x_1, x_2, x_3, x_4) = \left(1, \frac{4}{5}, 0, \frac{4}{5}\right)$ with $Z = 16\frac{1}{5}$

- The resulting bounds for the subproblems then are

  Bound for subproblem 1: $Z \leq 9$,

  Bound for subproblem 2: $Z \leq 16$.

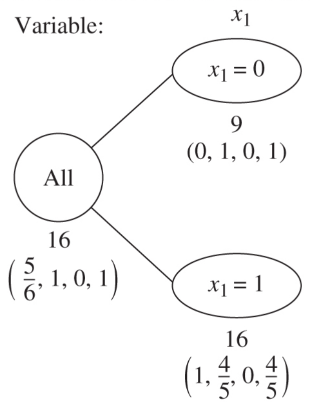
- The figure summarizes these results, where the numbers given just below the nodes are the bounds and below each bound is the optimal solution obtained for the LP relaxation.


Variable:
x_1
$x_1 = 0$
9
$(0, 1, 0, 1)$
All
16
$\left(\frac{5}{6}, 1, 0, 1\right)$
$x_1 = 1$
16
$\left(1, \frac{4}{5}, 0, \frac{4}{5}\right)$

30

Fathoming

- A subproblem can be conquered (fathomed), and thereby dismissed from further consideration, in the three ways described below.
- One way is illustrated by the results for subproblem 1 given by the $x_1 = 0$ node in the figure.
- Note that the (unique) optimal solution for its LP relaxation, (x_1, x_2, x_3, x_4) = (0, 1, 0, 1), is an *integer* solution.
- Therefore, this solution must also be the optimal solution for subproblem 1 itself.


Variable:
x_1
$x_1 = 0$
9
(0, 1, 0, 1)
All
16
$\left(\frac{5}{6}, 1, 0, 1\right)$
$x_1 = 1$
16
$\left(1, \frac{4}{5}, 0, \frac{4}{5}\right)$

31

Fathoming

- This solution should be stored as the first incumbent (the best feasible solution found so far) for the whole problem, along with its value of Z.
- This value is denoted by

  Z* = value of Z for current incumbent,

  so Z*= 9 at this point.
- Since this solution has been stored, there is no reason to consider subproblem 1 any further by branching from the $x_1 = 0$ node, etc.
- Doing so could only lead to other feasible solutions that are inferior to the incumbent, and we have no interest in such solutions.
- Because it has been solved, we fathom (dismiss) subproblem 1 now.

32

Fathoming

- The above results suggest a second key fathoming test.
- Since Z* = 9, there is no reason to consider further any subproblem whose *bound* $\leq$ 9, since such a subproblem cannot have a feasible solution better than the *incumbent*.
- Stated more generally, a subproblem is fathomed whenever its

  Bound $\leq$ Z*.

33

Fathoming

- This outcome does not occur in the current iteration of the example because subproblem 2 has a bound of 16 that is larger than 9.
- However, it might occur later for descendants of this subproblem (new smaller subproblems created by branching on this subproblem, and then perhaps branching further through subsequent “generations”).
- Furthermore, as new incumbents with larger values of Z* are found, it will become easier to *fathom* in this way.

34

Fathoming

- The third way of fathoming is quite straightforward.
- If the simplex method finds that a subproblem's LP relaxation has *no feasible solutions*, then the subproblem itself must have *no feasible solutions*, so it can be dismissed (fathomed).
- In all three cases, we are conducting our search for an optimal solution by retaining for further investigation only those subproblems that could possibly have a feasible solution better than the current incumbent.

35

Summary of Fathoming Tests

- A subproblem is *fathomed* (dismissed from further consideration) if

  Test 1: Its bound $\leq Z^*$,

  or

  Test 2: Its LP relaxation has no feasible solutions,

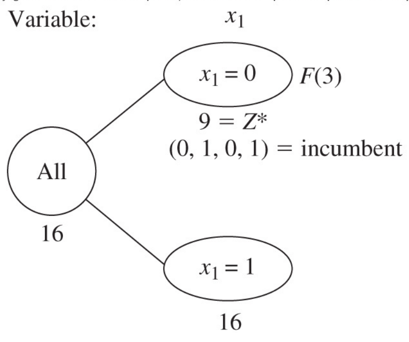
  or

  Test 3: The optimal solution for its LP relaxation is *integer*. (If this solution is better than the incumbent, it becomes the new incumbent, and test 1 is reapplied to all unfathomed subproblems with the new larger Z^*.)

36

Summary of Fathoming Tests

- The figure summarizes the results of applying these three tests to subproblems 1 and 2 by showing the current *solution tree*.
- Only subproblem 1 has been fathomed, by test 3, as indicated by *F*(3) next to the $x_1 = 0$ node.
- The resulting incumbent also is identified below this node.



37

Summary of Fathoming Tests

- The subsequent iterations will illustrate successful applications of all three tests.
- However, before continuing the example, we summarize the algorithm being applied to this BIP problem.
- This algorithm assumes that all coefficients in the objective function are integer and that the ordering of the variables for branching is x_1 , x_2, . . . , x_n.

38

Summary of the BIP Branch-and-Bound Algorithm

Initialization:

- Set $Z^* = -\infty$
- Apply the bounding step, fathoming step, and optimality test described below to the whole problem.
- If not fathomed, classify this problem as the one remaining “subproblem” for performing the first full iteration below.

Summary of the BIP Branch-and-Bound Algorithm

Steps for each iteration:

1. Branching: Among the *remaining* (unfathomed) subproblems, select the one that was created *most recently*. (Break ties according to which has the *larger bound*.) Branch from the node for this subproblem to create two new subproblems by fixing the next variable (the branching variable) at either 0 or 1.
2. Bounding: For each new subproblem, obtain its *bound* by applying the simplex method to its LP relaxation and rounding down the value of Z for the resulting optimal solution.
3. Fathoming: For each new subproblem, apply the three fathoming tests summarized above, and discard those subproblems that are fathomed by any of the tests.

Summary of the BIP Branch-and-Bound Algorithm

- Optimality test:
  - Stop when there are no remaining subproblems; the current incumbent is optimal.
  - Otherwise, return to perform another iteration.

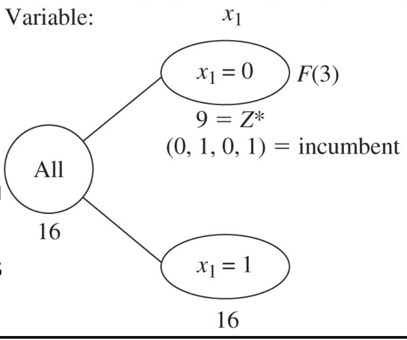
41

Completing the Example

- The pattern for the remaining iterations will be quite similar to that for the first iteration described above except for the ways in which fathoming occurs.
- Therefore, we shall summarize the branching and bounding steps fairly briefly and then focus on the fathoming step.

Iteration 2

- The only remaining subproblem corresponds to the $x_1 = 1$ node in the figure, so we shall branch from this node to create the two new subproblems given below.



42

Completing the Example

Maximize $Z = 9x_1 + 5x_2 + 6x_3 + 4x_4$

Subproblem 3:

- Fix $x_1 = 1$, $x_2 = 0$ so the resulting subproblem is

Maximize $Z = 9x_2 + 6x_3 + 4x_4$

subject to

(1) $5x_3 + 2x_4 \leq 4$
(2) $x_3 + x_4 \leq 1$
(3) $x_3 \leq 1$
(4) $x_4 \leq 0$
(5) x_j is binary, for j = 3, 4.

Subproblem 4:

- Fix $x_1 = 1$, $x_2 = 1$ so the resulting subproblem is

Maximize $Z = 14 + 6x_3 + 4x_4$

subject to

(1) $5x_3 + 2x_4 \leq 1$
(2) $x_3 + x_4 \leq 1$
(3) $x_3 \leq 1$
(4) $x_4 \leq 1$
(5) x_j is binary, for j = 3, 4.

43

Completing the Example

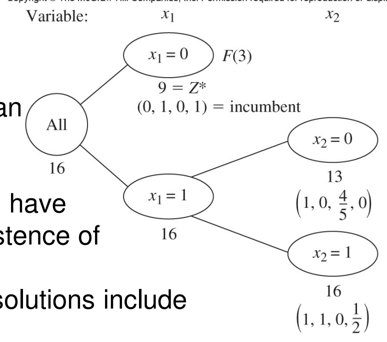
- The LP relaxations of these subproblems are obtained by replacing the constraints x_j is binary for j = 3, 4 by the constraints $0 \leq x_j \leq 1$ for j = 3, 4. Their optimal solutions (plus the fixed values of x_1 and x_2) are

  LP relaxation of subproblem 3: $(x_1, x_2, x_3, x_4) = \left(1, 0, \frac{4}{5}, 0\right)$ with $Z = 13\frac{4}{5}$

  LP relaxation of subproblem 4: $(x_1, x_2, x_3, x_4) = \left(1, 1, 0, \frac{1}{2}\right)$ with $Z = 16$
- The resulting bounds for the subproblems then are

  Bound for subproblem 3: $Z \leq 13$

  Bound for subproblem 4: $Z \leq 16$
- Note that both these bounds are larger than Z* = 9, so fathoming test 1 fails in both cases.
- Test 2 also fails, since both LP relaxations have feasible solutions (as indicated by the existence of an optimal solution).
- Test 3 fails as well because both optimal solutions include variables with noninteger values.
- The figure shows the resulting solution tree at this point.
- The lack of an *F* to the right of either new node indicates that both remain unfathomed.



44

Completing the Example

Iteration 3

- So far, the algorithm has created four subproblems.
- Subproblem 1 has been fathomed, and subproblem 2 has been replaced by (separated into) subproblems 3 and 4, but these last two remain under consideration.
- Because they were created simultaneously, but subproblem 4 ($x_1 = 1$, $x_2 = 1$) has the larger *bound* (16 > 13), the next branching is done from the (x_1, x_2) = (1, 1) node in the solution tree, which creates the following new subproblems (where constraint 3 disappears because it does not contain x_4).

45

Completing the Example

Maximize $Z = 9x_1 + 5x_2 + 6x_3 + 4x_4$

Subproblem 5:

- Fix $x_1 = 1$, $x_2 = 1$, $x_3 = 0$ so the resulting subproblem is

Maximize $Z = 14 + 4x_4$

subject to

(1) $2x_4 \leq 1$

(2),(4) $x_4 \leq 1$ (twice)

(5) x_4 is binary.

Subproblem 6:

- Fix $x_1 = 1$, $x_2 = 1$, $x_3 = 1$ so the resulting subproblem is

Maximize $Z = 20 + 4x_4$

subject to

(1) $2x_4 \leq 1$

(2) $x_4 \leq 0$

(4) $x_4 \leq 1$

(5) x_4 is binary.

If we form their LP relaxations by replacing constraint 5 by

(5) $0 \leq x_4 \leq 1$

the following results are obtained:

46

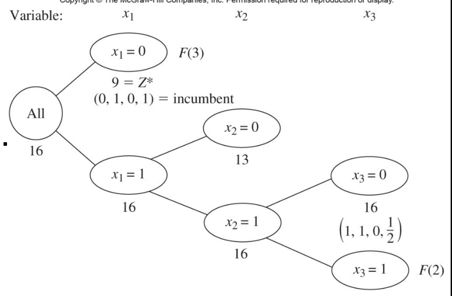
Completing the Example

LP relaxation of subproblem 5: $(x_1, x_2, x_3, x_4) = \left(1, 1, 0, \frac{1}{2}\right)$, with $Z = 16$

LP relaxation of subproblem 6: No feasible solutions

Bound for subproblem 5: $Z \leq 16$

- Note how the combination of constraints 1 and 5 in the LP relaxation of subproblem 6 prevents any feasible solutions.
- Therefore, this subproblem is fathomed by test 2.



- However, subproblem 5 fails this test, as well as test 1 (16 > 9) and test 3 ($x_4 = \frac{1}{2}$ is not integer), so it remains under consideration.
- We now have the solution tree shown in the figure.

47

Completing the Example

Iteration 4

- The subproblems corresponding to nodes (1, 0) and (1, 1, 0) in the previous figure remain under consideration, but the latter node was created more recently, so it is selected for branching from next.
- Since the resulting branching variable x_4 is the last variable, fixing its value at either 0 or 1 actually creates a single solution rather than subproblems requiring fuller investigation.
- These single solutions are:

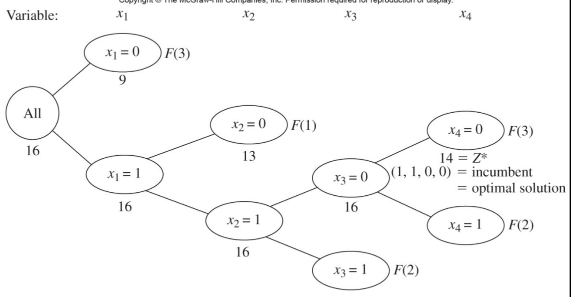
$x_4 = 0$: $(x_1, x_2, x_3, x_4) = (1, 1, 0, 0)$ is feasible, with $Z = 14$.

$x_4 = 1$: $(x_1, x_2, x_3, x_4) = (1, 1, 0, 1)$ is infeasible.

48

Completing the Example

- Formally applying the fathoming tests, we see that the first solution passes test 3 and the second passes test 2.
- Furthermore, this feasible first solution is better than the incumbent (14 > 9), so it becomes the new incumbent, with Z* = 14.
- Because a new incumbent has been found, we now reapply fathoming test 1 with the new larger value of Z* to the only remaining subproblem, the one at node (1, 0).



Subproblem 3:

Bound $= 13 \leq Z* = 14$.

- Therefore, this subproblem now is fathomed.
- We now have the solution tree shown in the figure.
- Note that there are *no remaining* (unfathomed) subproblems.
- Consequently, the optimality test indicates that the current incumbent

  $(x_1, x_2, x_3, x_4) = (1, 1, 0, 0)$

  is optimal, so we are done.

A Branch-and-Bound Algorithm for Mixed Integer Programming

- We shall now consider the general MIP problem, where *some* of the variables (say, *I* of them) are restricted to integer values (but not necessarily just 0 and 1) but the rest are ordinary continuous variables.
- For notational convenience, we shall order the variables so that the first *I* variables are the *integer-restricted* variables.

A Branch-and-Bound Algorithm for Mixed Integer Programming

- Therefore, the general form of the problem being considered is

  Maximize $Z = \sum_{j=1}^{n} c_j x_j$

  subject to

  $\sum_{j=1}^{n} a_{ij} x_j \leq b_i,$ for i =1, 2, …, m,

  and

  $x_j \geq 0,$ for j = 1, 2, . . . , n

  x_j is integer, for j = 1, 2, . . . , I ; $\mathrm{I} \leq n.$
- (When I = n, this problem becomes the pure IP problem.)
- We shall describe a basic branch-and-bound algorithm for solving this problem that, with a variety of refinements, has provided a standard approach to MIP.

51

A Branch-and-Bound Algorithm for Mixed Integer Programming

- This algorithm is quite similar in structure to the BIP algorithm presented.
- Solving *LP relaxations* again provides the basis for both the *bounding* and *fathoming* steps.
- In fact, only four changes are needed in the BIP algorithm to deal with the generalizations from *binary* to *general* integer variables and from *pure* IP to *mixed* IP.

52

A Branch-and-Bound Algorithm for Mixed Integer Programming

- One change involves the choice of the *branching* variable.
- Before, the *next* variable in the natural ordering—x_1, x_2, . . . , x_n—was chosen automatically.
- Now, the only variables considered are the *integer-restricted* variables that have a *noninteger* value in the optimal solution for the LP relaxation of the current subproblem.
- Our rule for choosing among these variables is to select the *first* one in the natural ordering. (Production codes generally use a more sophisticated rule.)

53

A Branch-and-Bound Algorithm for Mixed Integer Programming

- The second change involves the values assigned to the branching variable for creating the new smaller subproblems.
- Before, the *binary* variable was fixed at 0 and 1, respectively, for the two new subproblems.
- Now, the *general* integer-restricted variable could have a very large number of possible integer values, and it would be inefficient to create *and* analyze *many* subproblems by fixing the variable at its individual integer values.
- Therefore, what is done instead is to create just *two* new subproblems (as before) by specifying two *ranges* of values for the variable.

54

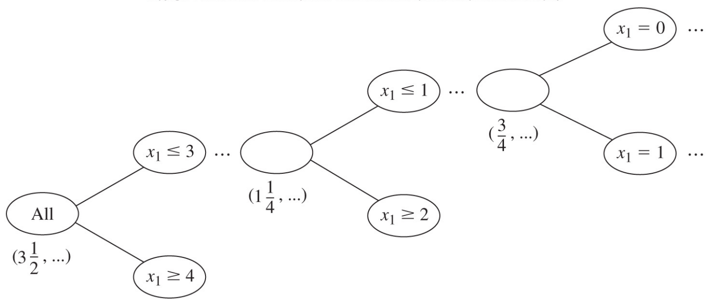
A Branch-and-Bound Algorithm for Mixed Integer Programming

- To spell out how this is done, let x_j be the current branching variable, and let x_j^* be its (noninteger) value in the optimal solution for the LP relaxation of the current subproblem.
- Using square brackets to denote
  $[x_j^*]$ greatest integer $\leq x_j^*$
  we have for the range of values for the two new subproblems
  $x_j \leq [x_j^*]$ and $x_j \geq [x_j^*] + 1$
  respectively.
- Each inequality becomes an *additional constraint* for that new subproblem.
- For example, if $x_j^* = 3\frac{1}{2}$, then
  $x_j \leq 3$ and $x_j \geq 4$
  are the respective additional constraints for the new subproblem.

55

A Branch-and-Bound Algorithm for Mixed Integer Programming

- When the two changes to the BIP algorithm described above are combined, an interesting phenomenon of a *recurring branching variable* can occur.
- To illustrate, as shown in the figure, let $j = 1$ in the above example where $x_j^* = 3\frac{1}{2}$, and consider the new subproblem where $x_1 \leq 3$.



56

A Branch-and-Bound Algorithm for Mixed Integer Programming

- When the LP relaxation of a descendant of this subproblem is solved, suppose that $x_1^* = 1\frac{1}{4}$.
- Then x_1 *recurs* as the branching variable, and the two new subproblems created have the additional constraint $x_1 \leq 1$ and $x_1 \geq 2$, respectively (as well as the previous additional constraint $x_1 \leq 3$).
- Later, when the LP relaxation for a descendant of, say, the $x_1 \leq 1$ subproblem is solved, suppose that $x_1^* = \frac{3}{4}$.
- Then x_1 *recurs* again as the branching variable, and the two new subproblems created have $x_1 = 0$ (because of the new $x_1 \leq 0$ constraint and the nonnegativity constraint on x_1) and $x_1 = 1$ (because of the new $x_1 \geq 1$ constraint and the previous $x_1 \leq 1$ constraint).

57

A Branch-and-Bound Algorithm for Mixed Integer Programming

- The third change involves the *bounding step.*
- Before, with a *pure* IP problem and integer coefficients in the objective function, the value of Z for the optimal solution for the subproblem's LP relaxation was *rounded down* to obtain the bound, because any feasible solution for the subproblem must have an *integer* Z.
- Now, with some of the variables *not* integer-restricted, the bound is the value of Z *without* rounding down.

58

Summary of the MIP Branch-and-Bound Algorithm

- The fourth (and final) change to the BIP algorithm to obtain our MIP algorithm involves fathoming test 3.
- Before, with a *pure* IP problem, the test was that the optimal solution for the subproblem's LP relaxation is *integer*, since this ensures that the solution is feasible, and therefore optimal, for the subproblem.
- Now, with a *mixed* IP problem, the test requires only that the *integer-restricted* variables be *integer* in the optimal solution for the subproblem's LP relaxation, because this suffices to ensure that the solution is feasible, and therefore optimal, for the subproblem.
- Incorporating these four changes into the summary presented in the preceding section for the BIP algorithm yields the following summary for the new algorithm for MIP.

59

Summary of the MIP Branch-and-Bound Algorithm

Initialization:

- Set $Z^* = -\infty$
- Apply the bounding step, fathoming step, and optimality test described below to the whole problem.
- If not fathomed, classify this problem as the one remaining subproblem for performing the first full iteration below.

60

Summary of the MIP Branch-and-Bound Algorithm

Steps for each iteration:

1. Branching: Among the *remaining* (unfathomed) subproblems, select the one that was created *most recently*. (Break ties according to which has the *larger bound*.) Among the *integer-restricted* variables that have a *noninteger* value in the optimal solution for the LP relaxation of the problem, choose the *first one* in the natural ordering of the variables to be the *branching variable*. Let x_j be this variable and x_j^* its value in this solution. Branch from the node for the subproblem to create two new subproblems by adding the respective constraints $x_j \leq \left[x_j^*\right]$ and $x_j \geq \left[x_j^*\right] + 1$.
2. Bounding: For each new subproblem, obtain its *bound* by applying the simplex method (or the dual simplex method when reoptimizing) to its LP relaxation and using the value of Z for the resulting optimal solution.
3. Fathoming: For each new subproblem, apply the three fathoming tests given below, and discard those subproblems that are fathomed by any of the tests.

Summary of the MIP Branch-and-Bound Algorithm

Test 1: Its bound ≤ Z*, where Z* is the value of Z for the current *incumbent*.

Test 2: Its LP relaxation has no feasible solutions.

Test 3: The optimal solution for its LP relaxation has *integer* values for the *integer-restricted* variables. (If this solution is better than the incumbent, it becomes the new incumbent, and test 1 is reapplied to all unfathomed subproblems with the new larger Z*.)

- Optimality test:
  - Stop when there are no remaining subproblems; the current *incumbent* is optimal.
  - Otherwise, perform another iteration.

An MIP Example

- We will now illustrate this algorithm by applying it to the following MIP problem:

Maximize $Z = 4x_1 - 2x_2 + 7x_3 - x_4$

subject to

$$\begin{aligned} x_1 \quad\quad + 5x_3 \quad\quad &\leq 10 \\ x_1 + x_2 - x_3 \quad\quad &\leq 1 \\ 6x_1 - 5x_2 \quad\quad\quad &\leq 0 \\ -x_1 \quad\quad + 2x_3 - 2x_4 &\leq 3 \end{aligned}$$

and

$x_j \geq 0$, for j = 1, 2, 3, 4
x_j is an integer, for j = 1, 2, 3.

- Note that the number of integer-restricted variables is *I* = 3, so x_4 is the only continuous variable.

63

An MIP Example

Initialization:

- After setting Z*= $-\infty$, we form the LP relaxation of this problem by *deleting* the set of constraints that x_j is an integer for j = 1, 2, 3.
- Applying the simplex method to this LP relaxation yields its optimal solution below.

LP relaxation of whole problem: $(x_1, x_2, x_3, x_4) = \left(\frac{5}{4}, \frac{3}{2}, \frac{7}{4}, 0\right)$, with $Z = 14\frac{1}{4}$

- Because it has *feasible* solutions and this optimal solution has *noninteger* values for its integer-restricted variables, the whole problem is not fathomed, so the algorithm continues with the first full iteration below.

64

An MIP Example

Subproblem 1:

- Original problem plus additional constraint

  $x_1 \leq 1$

Subproblem 2:

- Original problem plus additional constraint

  $x_1 \geq 2$

- Deleting the set of integer constraints again and solving the resulting LP relaxations of these two subproblems yield the following results.

LP relaxation of subproblem 1: $(x_1, x_2, x_3, x_4) = \left(1, \frac{6}{5}, \frac{9}{5}, 0\right)$, with $Z = 14\frac{1}{5}$

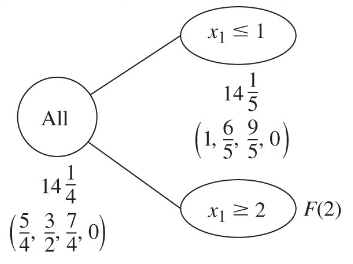
Bound for subproblem 1: $Z \leq 14\frac{1}{5}$.

LP relaxation of subproblem 2: No feasible solutions.

65

An MIP Example

- This outcome for subproblem 2 means that it is fathomed by test 2.
- However, just as for the whole problem, subproblem 1 fails all fathoming tests.
- These results are summarized in the solution tree shown in the figure.



$x_1 \leq 1$
$14\frac{1}{5}$
$\left(1, \frac{6}{5}, \frac{9}{5}, 0\right)$
All
$14\frac{1}{4}$
$\left(\frac{5}{4}, \frac{3}{2}, \frac{7}{4}, 0\right)$
$x_1 \geq 2$
$F(2)$

66

An MIP Example

Iteration 2

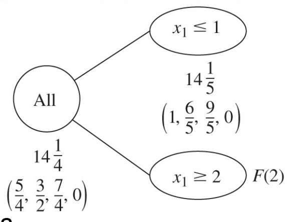
- With only one remaining subproblem, corresponding to the $x_1 \leq 1$ node in the figure, the next branching is from this node.
- Examining its LP relaxation's optimal solution given below, we see that this node reveals that the *branching variable* is x_2, because $x_2 = \frac{6}{5}$ is the first integer-restricted variable that has a noninteger value.
- Adding one of the constraints $x_2 \leq 1$ or $x_2 \geq 2$ then creates the following two new subproblems.

  Subproblem 3:

  - Original problem plus additional constraints $x_1 \leq 1, \quad x_2 \leq 1.$

  Subproblem 4:

  - Original problem plus additional constraints $x_1 \leq 1, \quad x_2 \geq 2.$


All
$14\frac{1}{4}$
$\left(\frac{5}{4}, \frac{3}{2}, \frac{7}{4}, 0\right)$
$x_1 \leq 1$
$14\frac{1}{5}$
$\left(1, \frac{6}{5}, \frac{9}{5}, 0\right)$
$x_1 \geq 2$
$F(2)$

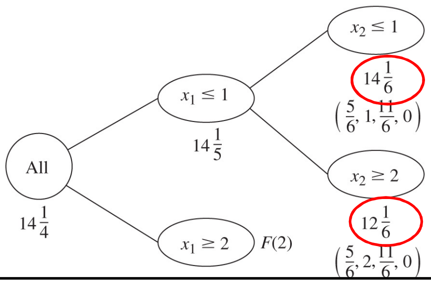
An MIP Example

- Solving their LP relaxations gives the following results.

  LP relaxation of subproblem 3: $(x_1, x_2, x_3, x_4) = \left(\frac{5}{6}, 1, \frac{11}{6}, 0\right)$, with $Z = 14\frac{1}{6}$.
  Bound for subproblem 3: $Z \leq 14\frac{1}{6}$.

  LP relaxation of subproblem 4: $(x_1, x_2, x_3, x_4) = \left(\frac{5}{6}, 2, \frac{11}{6}, 0\right)$, with $Z = 12\frac{1}{6}$.
  Bound for subproblem 4: $Z \leq 12\frac{1}{6}$.
- Because both solutions exist (feasible solutions) and have noninteger values for integer-restricted variables, neither subproblem is fathomed. (Test 1 still is not operational, since $Z^* = -\infty$ until the first incumbent is found.)
- The solution tree at this point is given in the figure.

(subproblem 3, with $14\frac{1}{6} > 12\frac{1}{6}$ is selected for the next branching)



An MIP Example

Iteration 3

- With two remaining subproblems (3 and 4) that were created simultaneously, the one with the larger bound (subproblem 3, with $14\frac{1}{6} > 12\frac{1}{6}$) is selected for the next branching.
- Because $x_1 = \frac{5}{6}$ has a noninteger value in the optimal solution for this subproblem's LP relaxation, x_1 becomes the branching variable. (Note that x_1 now is a *recurring* branching variable, since it also was chosen at iteration 1.) This leads to the following new subproblems.

69

An MIP Example

Subproblem 5:

- Original problem plus additional constraints

$x_1 \leq 1$
$x_2 \leq 1$
$x_1 \leq 0$ (so x_1= 0)

Subproblem 6:

- Original problem plus additional constraints

$x_1 \leq 1$
$x_2 \leq 1$
$x_1 \geq 1$ (so x_1= 1)

- The results form solving their LP relaxations are given below.

LP relaxation of subproblem 5: $(x_1, x_2, x_3, x_4) = \left(0, 0, 2, \frac{1}{2}\right)$, with $Z = 13\frac{1}{2}$.

Bound for subproblem 5: $Z \leq 13\frac{1}{2}$.

LP relaxation of subproblem 6: No feasible solutions.

70

An MIP Example

- Subproblem 6 is immediately fathomed by test 2.
- However, note that subproblem 5 also can be fathomed.
- Test 3 passes because the optimal solution for its LP relaxation has integer values $(x_1 = 0, x_2 = 0, x_3 = 2)$ for all three integer-restricted variables. (It does not matter that $x_4 = \frac{1}{2}$, since x_4 is not integer-restricted.)
- This *feasible* solution for the original problem becomes our first incumbent:

  Incumbent = $\left(0, 0, 2, \frac{1}{2}\right)$, with $Z^* = 13\frac{1}{2}$.

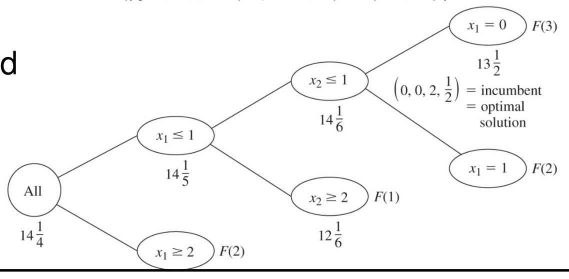
71

An MIP Example

- Using this Z* to reapply fathoming test 1 to the only other subproblem (subproblem 4) is successful, because its bound $12\frac{1}{6} \leq Z^*$.
- This iteration has succeeded in fathoming subproblems in all three possible ways.
- Furthermore, there now are no remaining subproblems, so the current incumbent is optimal.

  Optimal solutions = $\left(0, 0, 2, \frac{1}{2}\right)$, with $Z = 13\frac{1}{2}$.

- These results are summarized by the final solution tree given in the figure.



72

References

Hillier&Lieberman

- Chapter 12: Integer Programming 12.5-12.7.

73