Let us rewrite these equations in an equivalent but somewhat altered form:

\[
\begin{align*}
-3(0) - 3(1) + (0) + 2(1) + 3(0) & \leq -2, \quad -3(0) - 3(1) + (0) + 2(0) + 3(1) \leq -2, \\
-5(0) - 3(1) - 2(0) - 1(1) + (0) & \leq -4, \quad -5(0) - 3(1) - 2(0) - 1(0) + (1) \leq -4.
\end{align*}
\]

For \(x_4 = 1\), the first constraint is infeasible by 1 unit and the second constraint is feasible, giving 1 total unit of infeasibility. For \(x_5 = 1\), the first constraint is infeasible by 2 units and the second by 2 units, giving 4 total units of infeasibility. Thus \(x_4 = 1\) appears more favorable, and we would subdivide based upon that variable. In general, the variable giving the least total infeasibilities by this approach would be chosen next. Reviewing the example problem the reader will see that this approach has been used in our solution.

### 9.8 CUTTING PLANES

The cutting-plane algorithm solves integer programs by modifying linear-programming solutions until the integer solution is obtained. It does not partition the feasible region into subdivisions, as in branch-and-bound approaches, but instead works with a single linear program, which it refines by adding new constraints. The new constraints successively reduce the feasible region until an integer optimal solution is found.

In practice, the branch-and-bound procedures almost always outperform the cutting-plane algorithm. Nevertheless, the algorithm has been important to the evolution of integer programming. Historically, it was the first algorithm developed for integer programming that could be proved to converge in a finite number of steps. In addition, even though the algorithm generally is considered to be very inefficient, it has provided insights into integer programming that have led to other, more efficient, algorithms.

Again, we shall discuss the method by considering the sample problem of the previous sections:

\[
z^* = \max 5x_1 + 8x_2,
\]

subject to:

\[
\begin{align*}
x_1 + x_2 + s_1 &= 6, \\
5x_1 + 9x_2 + s_2 &= 45, \\
x_1, x_2, s_1, s_2 &\geq 0.
\end{align*}
\]

\(s_1\) and \(s_2\) are, respectively, slack variables for the first and second constraints.

Solving the problem by the simplex method produces the following optimal tableau:

\[
\begin{array}{c|cccc}
\hline
(-z) & -\frac{5}{4}s_1 & -\frac{3}{4}s_2 & = & -41\frac{1}{4}, \\
x_1 & +\frac{9}{4}s_1 & -\frac{1}{4}s_2 & = & \frac{9}{4}, \\
x_2 & -\frac{5}{4}s_1 & +\frac{1}{4}s_2 & = & \frac{15}{4}, \\
\hline
x_1, x_2, s_1, s_2 & \geq 0.
\end{array}
\]

Let us rewrite these equations in an equivalent but somewhat altered form:

\[
\begin{align*}
-2s_1 - s_2 + 42 &= \frac{3}{4} - \frac{3}{4}s_1 - \frac{1}{4}s_2, \\
+2s_1 - s_2 &= \frac{1}{4} - \frac{1}{4}s_1 - \frac{3}{4}s_2, \\
-2s_1 - 3 &= \frac{3}{4} - \frac{3}{4}s_1 - \frac{1}{4}s_2,
\end{align*}
\]

\(x_1, x_2, s_1, s_2 \geq 0\).

These algebraic manipulations have isolated integer coefficients to one side of the equalities and fractions to the other, in such a way that the constant terms on the righthand side are all nonnegative and the slack variable coefficients on the righthand side are all nonpositive.
In any integer solution, the lefthand side of each equation in the last tableau must be integer. Since \( s_1 \) and \( s_2 \) are nonnegative and appear to the right with negative coefficients, each righthand side necessarily must be less than or equal to the fractional constant term. Taken together, these two observations show that both sides of every equation must be an integer less than or equal to zero (if an integer is less than or equal to a fraction, it necessarily must be 0 or negative). Thus, from the first equation, we may write:

\[
\frac{3}{4} - \frac{3}{4} s_1 - \frac{1}{4} s_2 \leq 0 \quad \text{and integer,}
\]

or, introducing a slack variable \( s_3 \),

\[
\frac{3}{4} - \frac{3}{4} s_1 - \frac{1}{4} s_2 + s_3 = 0, \quad s_3 \geq 0 \quad \text{and integer. (C_1)}
\]

Similarly, other conditions can be generated from the remaining constraints:

\[
\frac{1}{4} - \frac{1}{4} s_1 - \frac{3}{4} s_2 + s_4 = 0, \quad s_4 \geq 0 \quad \text{and integer (C_2)}
\]

\[
\frac{3}{4} - \frac{3}{4} s_1 - \frac{1}{4} s_2 + s_5 = 0, \quad s_5 \geq 0 \quad \text{and integer (C_3)}
\]

Note that, in this case, \((C_1)\) and \((C_3)\) are identical.

The new equations \((C_1), (C_2),\) and \((C_3)\) that have been derived are called cuts for the following reason: Their derivation did not exclude any integer solutions to the problem, so that any integer feasible point to the original problem must satisfy the cut constraints. The linear-programming solution had \( s_1 = s_2 = 0 \); clearly, these do not satisfy the cut constraints. In each case, substituting \( s_1 = s_2 = 0 \) gives either \( s_3, s_4, \) or \( s_5 < 0 \). Thus the net effect of a cut is to cut away the optimal linear-programming solution from the feasible region without excluding any feasible integer points.

The geometry underlying the cuts can be established quite easily. Recall from (11) that slack variables \( s_1 \) and \( s_2 \) are defined by:

\[
\begin{align*}
  s_1 &= 6 - x_1 - x_2, \\
  s_2 &= 45 - 5x_1 - 9x_2.
\end{align*}
\]

Substituting these values in the cut constraints and rearranging, we may rewrite the cuts as:

\[
\begin{align*}
  2x_1 + 3x_2 &\leq 15, \quad (C_1 \text{ or } C_3) \\
  4x_1 + 7x_2 &\leq 35. \quad (C_2)
\end{align*}
\]

In this form, the cuts are displayed in Fig. 9.21. Notethat they exhibit the features suggested above. In each case, the added cut removes the linear-programming solution \( x_1 = \frac{9}{4}, \ x_2 = \frac{15}{4} \), from the feasible region, at the same time including every feasible integer solution.

The basic strategy of the cutting-plane technique is to add cuts (usually only one) to the constraints defining the feasible region and then to solve the resulting linear program. If the optimal values for the decision variables in the linear program are all integer, they are optimal; otherwise, a new cut is derived from the new optimal linear-programming tableau and appended to the constraints.

Note from Fig. 9.21 that the cut \( C_1 = C_3 \) leads directly to the optimal solution. Cut \( C_2 \) does not, and further iterations will be required if this cut is appended to the problem (without the cut \( C_1 = C_3 \)). Also note that \( C_1 \) cuts deeper into the feasible region than does \( C_2 \). For problems with many variables, it is generally quite difficult to determine which cuts will be deep in this sense. Consequently, in applications, the algorithm frequently generates cuts that shave very little from the feasible region, and hence the algorithm’s poor performance.

A final point to be considered here is the way in which cuts are generated. The linear-programming tableau for the above problem contained the constraint:

\[
x_1 + \frac{9}{4} s_1 - \frac{1}{4} s_2 = \frac{9}{4}.
\]
Suppose that we round *down* the fractional coefficients to integers, that is, \( \frac{9}{4} \) to 2, \( -\frac{1}{4} \) to \(-1\), and \( \frac{9}{4} \) to 2. Writing these integers to the left of the equality and the remaining fractions to the right, we obtain as before, the equivalent constraint:

\[
x_1 + 2s_1 - s_2 - 2 = \frac{1}{4} - \frac{1}{4}s_1 - \frac{3}{4}s_2.
\]

By our previous arguments, the cut is:

\[
\frac{1}{4} - \frac{1}{4}s_1 - \frac{3}{4}s_2 \leq 0 \quad \text{and integer.}
\]

Another example may help to clarify matters. Suppose that the final linear-programming tableau to a problem has the constraint

\[
x_1 + \frac{1}{6}x_6 - \frac{7}{6}x_7 + 3x_8 = \frac{5}{6}.
\]

Then the equivalent constraint is:

\[
x_1 - 2x_7 + 3x_8 - 4 = \frac{5}{6} - \frac{1}{6}x_6 - \frac{5}{6}x_7,
\]

and the resulting cut is:

\[
\frac{5}{6} - \frac{1}{6}x_6 - \frac{5}{6}x_7 \leq 0 \quad \text{and integer.}
\]

Observe the way that fractions are determined for negative coefficients. The fraction in the cut constraint determined by the \( x_7 \) coefficient \( -\frac{7}{6} = -1\frac{1}{6} \) is *not* \( \frac{1}{6} \), but rather it is the fraction generated by rounding down to \(-2\); i.e., the fraction is \(-1\frac{3}{6} = (-2) = \frac{5}{6} \).

Tableau 2 shows the complete solution of the sample problem by the cutting-plane technique. Since cut \( C_1 = C_3 \) leads directly to the optimal solution, we have chosen to start with cut \( C_2 \). Note that, if the slack variable for any newly generated cut is taken as the basic variable in that constraint, then the problem is in the proper form for the dual-simplex algorithm. For instance, the cut in Tableau 2(b) generated from the \( x_1 \) constraint

\[
x_1 + \frac{7}{2}s_1 - \frac{1}{2}s_2 = \frac{7}{4} \quad \text{or} \quad x_1 + 2s_1 - s_2 - 2 = \frac{1}{2} - \frac{1}{3}s_1 - \frac{2}{3}s_2
\]

is given by:

\[
\frac{1}{2} - \frac{1}{3}s_1 - \frac{2}{3}s_2 \leq 0 \quad \text{and integer.}
\]

Letting \( s_4 \) be the slack variable in the constraint, we obtain:

\[
-\frac{1}{3}s_1 - \frac{2}{3}s_2 + s_4 = -\frac{1}{3}.
\]
Since $s_1$ and $s_2$ are nonbasic variables, we may take $s_4$ to be the basic variable isolated in this constraint (see Tableau 2(b)).

By making slight modifications to the cutting-plane algorithm that has been described, we can show that an optimal solution to the integer-programming problem will be obtained, as in this example, after adding only a finite number of cuts. The proof of this fact by R. Gomory in 1958 was a very important theoretical break-through, since it showed that integer programs can be solved by some linear program (the associated linear program plus the added constraints). Unfortunately, the number of cuts to be added, though finite, is usually quite large, so that this result does not have important practical ramifications.