8 Integer Programming

In short, integer programming is linear programming with the additional requirement that all variables need to take integral values. Integer programs (IPs) allow for modeling a very large class of discrete mathematical optimization problems. In particular, most classical combinatorial optimization problems can easily be modeled as IPs. However, this comes at a price. IPs are an $\mathcal{NP}$-hard problem class and, depending on the problem type at hand, finding optimal (or even feasible) solutions may take excessive time. Nevertheless, strong methods have been developed to deal with many IPs that we face in real-world applications. In this chapter, we provide a brief introduction to some of the key techniques used to solve IPs in practice. This exposition is less focussed on specific theoretical properties, but rather provides a glimpse into a broadly used and highly influential toolbox to solve IPs.

8.1 Introduction to integer programming

As mentioned, integer programs are defined analogously to linear programs with the only difference that all variables are required to take integer values, for example,

$$\begin{array}{lrcl} \max & c^\top x & & \\ & Ax & \leq & b \\ & x & \in & \mathbb{Z}^n \end{array},$$

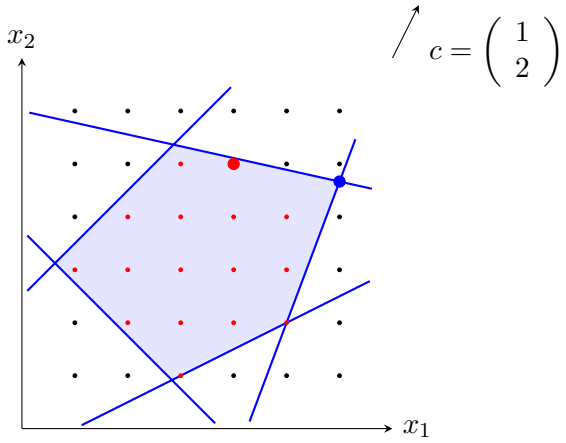
where $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, and $b \in \mathbb{R}^m$. As with linear programming, the above form is just one way to write integer programs. More generally, one can also use equalities or '$\geq$' constraints.

Figure 8.1 shows an example integer program in two dimensions. When forgetting about the integrality constraints, one obtains a linear program, which is called the *linear relaxation* of the IP. Unfortunately, the linear relaxation of an IP often does not shed much light on the integer program. Still, many methods for solving IPs repeatedly use linear relaxations of certain well-defined sub-problems. In particular branch & bound procedures and also branch & cut procedures heavily rely on linear relaxations, because they can be solved very quickly and can be used as upper bounds on underlying IPs.

We start with a recreational application of IPs, by showing how they can be used to solve the eight queens puzzle.

IP example: the eight queens puzzle

The eight queens puzzle is a famous problem linked to chess. The goal of the puzzle is to place eight chess queens on a standard 8×8 chessboard so that no two queens threaten each other, i.e., there is no more than one queen in each row, column, and diagonal. Figure 8.2 shows the squares



$$\begin{array}{lrcrcl} \max & x_1 & + & 2x_2 & & \\ \text{s.t.} & 2x_1 & + & 9x_2 & \leq & 54 \\ & -2x_1 & + & 2x_2 & \leq & 5 \\ & 4x_1 & + & 4x_2 & \geq & 15 \\ & x_1 & - & 2x_2 & \leq & 1 \\ & 8x_1 & - & 3x_2 & \leq & 34 \end{array}$$

Figure 8.1: Example of an integer program in two dimensions. The blue area shows the linear relaxation of the IP, i.e., the feasible solutions if one disregards the integrality constraints. The IP only optimizes over the integer points inside the relaxation, which are highlighted in red. The blue point shows the unique optimal solution of the linear relaxation, and the large red one shows the unique optimal solution of the IP.

threatened by a queen and Figure 8.3 shows an example placement of seven queens where there are no unthreatened squares left for an eighth queen.

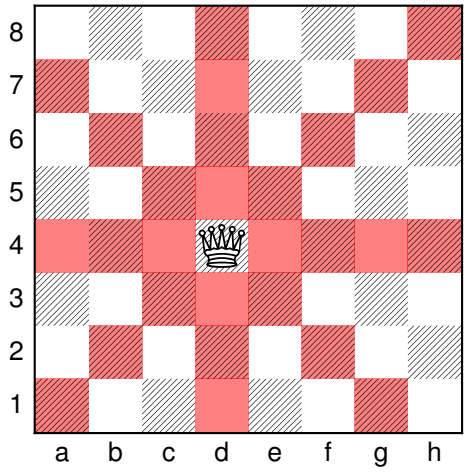


Figure 8.2: The squares in red are those where the shown queen can move to. Any other queen on one of those squares is said to be *threatened* by the shown queen.

To reformulate this combinatorial problem in terms of an integer program, we introduce a binary variable $x_{ij} \in \{0, 1\}$ for each square of the board, which is indexed by a pair $(i, j) \in [8] \times [8]$. Hence, each such binary variable will simply be an integer variable with linear constraints

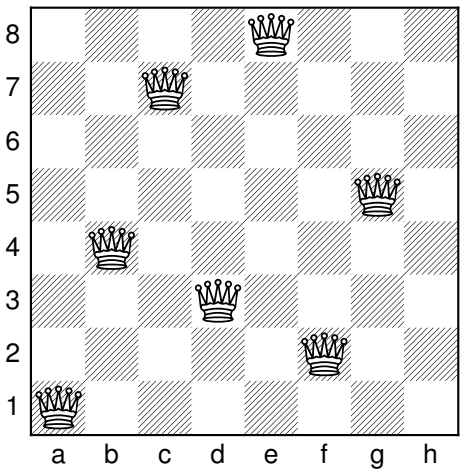


Figure 8.3: Example of a placement of seven queens on the board. Note that there is a queen in each of the files a-g and every square in the h file is threatened, so there is no valid square to place the eighth queen.

$0 \leq x_{ij} \leq 1$. Moreover, we think of $x_{ij} = 1$ as placing a queen on square (i, j), and of $x_{ij} = 0$ as not doing so.

The constraints now follow in a straightforward way from the statement of the puzzle: For each row, column, and diagonal of the chessboard, the sum of the corresponding variables is at most 1. The objective of the optimization problem is to maximize the sum of all components of the vector x, i.e., the number of queens placed on the board. Thus, we arrive at the following IP:

$$\begin{aligned}
\max \quad & \sum_{i=1}^{8}\sum_{j=1}^{8} x_{ij} & & \\
& \sum_{j=1}^{8} x_{ij} \leq 1 & & \forall i \in [8] \\
& \sum_{i=1}^{8} x_{ij} \leq 1 & & \forall j \in [8] \\
& \sum_{\substack{i,j\in[8]:\\ i+j=k}} x_{ij} \leq 1 & & \forall k \in \{2,3,\dots,16\} \\
& \sum_{\substack{i,j\in[8]:\\ i-j=k}} x_{ij} \leq 1 & & \forall k \in \{-7,-6,\dots,7\} \\
& x \in \{0,1\}^{8\times 8} \ . & &
\end{aligned} \tag{8.1}$$

Let us check that the optimal solutions of the IP (8.1) correspond one-to-one to the solutions of the eight queens puzzle. On the one hand, the optimal value does not exceed 8 because a

chessboard has 8 rows and there can be at most one queen per row. On the other hand, every solution of the puzzle corresponds to a feasible solution of the integer program attaining the value 8, which is the upper bound. Therefore, the optimal solutions of the integer program are precisely the solutions of the eight queens puzzle.

Figure 8.4 shows an optimal solution to the problem.

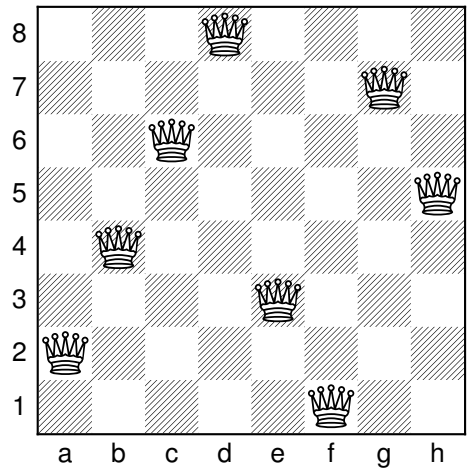


Figure 8.4: A solution to the eight queens puzzle.

It turns out that the eight queens puzzle has 12 distinct solutions that are not related by rotations or reflections, and 92 solutions in total.

Exercise 8.1

Construct an IP to find a solution to the eight queens puzzle that cannot be obtained from the one shown in Figure 8.4 through rotations and reflections.

8.2 Branch & bound

Branch & bound, and various variations thereof, is one of the most common solution approaches for integer programs. The principal idea is to intelligently explore the solution space and to try to cut off large parts of the solution space as soon as one can be sure that there is no optimal solution in them.

To illustrate the approach we consider the following IP with binary variables. The approach can easily be extended to integer variables with larger ranges.

$$\begin{array}{lrcrcrcrcl} \max & 75x_1 & + & 6x_2 & + & 3x_3 & + & 33x_4 & & \\ & 774x_1 & + & 76x_2 & + & 22x_3 & + & 42x_4 & \leq & 875 \\ & 67x_1 & + & 27x_2 & + & 794x_3 & + & 53x_4 & \leq & 875 \\ & & & & & & & x & \in & \{0,1\}^4 \end{array}$$

We start by replacing the integrality constraints $x \in \{0,1\}^4$ by $x \in [0,1]^4$ and solve the resulting linear relaxation. This leads to the solution shown in box 1 of Figure 8.5 (the number of the box is highlighted with a gray background), i.e.,

$$(x_1, x_2, x_3, x_4) = (1, 0.506, 0.934, 1) \ ,$$

with objective value $z = 113.837$. If this LP solution had been integral, we could have stopped right-away and return it as optimal solution. Indeed, the LP we solved optimized over a bigger solution set, obtained from the original IP by ignoring the integrality requirement on the variables. However, the values of x_2 and x_3 are not integral. We choose one of these variables, say x_2, and *branch* on this variable. More precisely, we create two sub-problems, one in which we force x_2 to be equal to 0 (box 9 in Figure 8.5) and one with $x_2 = 1$ (box 2). (If x_2 had been an integer variable with a larger range instead of a binary one, then we could have branched into two problems where one requires $x_2 \leq 0$ and the other one $x_2 \geq 1$.)

The optimal solution to the original problem is the better of the optimal solutions to the two newly created problems through branching. Branch & bound now recurses on the sub-problems, leading to a so-called branch & bound tree as shown in Figure 8.5. However, this tree is typically constructed in a depth-first order. Actually, the number of the boxes in Figure 8.5 highlights an order in which the sub-problems may be constructed and explored in a branch & bound procedure.

Let us start with the first 5 boxes. Here we successively branch on a fractional variable. In this case only the 5th box, where all variables are set to integral values, leads to a feasible integral solution. If it had happened that already an earlier sub-problem, say the one corresponding to box 4, had been integral, then we would not have branched further. Box 5 is the first feasible solution we found and we save it together with its value. We backtrack to the previous problem, the one corresponding to box 4, and go into the branch $x_3 = 1$. This leads to an infeasible linear program, which we ignore. We backtrack to the closest box above box 6 for which one of the two branches has not been explored yet. This is box 3, and we explore the other branch, i.e., $x_4 = 1$. In this case, the resulting LP is infeasible, and we can therefore stop exploring this branch further. Clearly, if the LP is infeasible, then the IP, which has integrality constraints on top of the LP, is infeasible, too.

A new important step happens at the next box we explore after backtracking. This is box 8, for which the optimal LP value is 42. Because this value is strictly smaller than the value of the best solution we found so far, which was found in box 5 and has value 81, we can stop exploring this box any further. Notice that coincidentally, the optimal solution to the LP of box 8 happens to be integral. However, this is irrelevant for this reasoning. Indeed, if even the linear relaxation cannot improve on the currently best solution, then there is no reason to branch further to obtain an integral solution. This step is called the *bounding* step of the branch & bound procedure, and explains the second part of its name.

We proceed as explained, and find the problem in box 10, whose optimal LP solution happens to be integral and improves on the best solution found previously. The last sub-problem we consider is the one in box 11, where we have an optimal LP value of only 86.72, which is less than the value of 108 of our best solution found so far, coming from box 10. Hence, no further exploration of problems stemming from box 11 is necessary. At this point, the branch & bound tree has been fully explored, and we return the best solution found, which is the one from box 10.

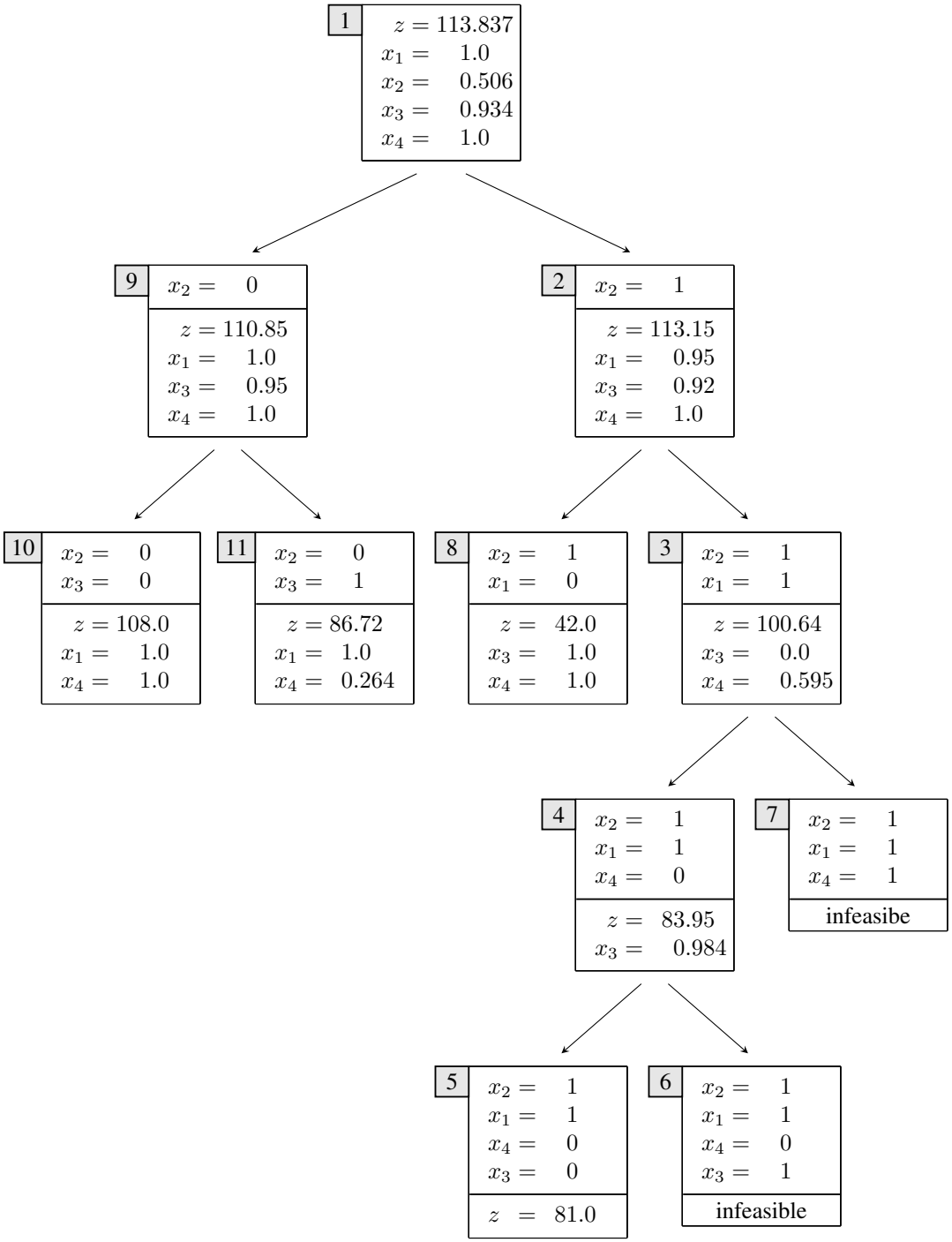


Figure 8.5: A branch & bound tree. Box 1 shows the optimal solution to the linear relaxation of the original problem. This solution is fractional, and we branch on the variable x_2 to obtain two new LP solutions shown in boxes 9 and 2, and so on. The lower part of each box shows the optimal LP solution to the problem when fixing variables as shown in the upper part of the box.

Remark 8.2: Incumbent

The best solution found so far is often called the *incumbent*.

Remark 8.3: Non-uniqueness of branch & bound tree

The tree constructed in this way is not unique, because there are various orders in which the branchings can be performed. First, one can choose the variable on which to branch, and second one can choose on which of the two children to continue first when doing a branching.

Remark 8.4: Speed-ups via strong incumbents

For a branch & bound procedure to be fast, it is crucial that a strong feasible solution is found quickly, because this can be used for the bounding step in the future, i.e., to discard certain sub-problems because even their linear relaxation has a worse objective than the value of the incumbent. This explains why we applied depth-first search to explore the branch & bound tree, with the hope to quickly find a feasible solution. Moreover, branch & bound procedures are typically combined with various heuristics to quickly obtain strong feasible solutions, which then helps to speed up the exploration of the solution space through branch & bound.

Remark 8.5: Speed-ups via strong relaxations

As we discussed, the reason why branch & bound procedures do not need to enumerate over all possible solutions, but can sometimes discard large parts of the solution space, is due to bounding. The bounding step is based on comparing solutions to LP relaxations with the value of the incumbent. For this to be effective, we do not only need strong incumbents, but it is also paramount to use a strong linear relaxation to start with. Our discussions on polyhedral relaxations of problems provides important ingredients and intuition on how to build such relaxations for various problems. The focus in Chapter 5 on polyhedral approaches for combinatorial optimization problems was on efficiently solvable problems, where we were able to describe the corresponding combinatorial polytope. For hard problems, there is little hope that we can get a good inequality description of their combinatorial polytope, because it is unlikely that they can be solved efficiently. Nevertheless, similar techniques and reasonings allow for obtaining descriptions that are good approximations of the convex hull of the solutions of the IP we are interested in.

8.3 Branch & cut

Branch & cut is an enhancement of branch & bound procedures through so-called cutting planes. More precisely, consider a sub-problem in the branch & bound procedure where we would normally branch, i.e., the sub-problem

(i) is not infeasible,
(ii) the value of its LP relaxation is strictly better than the value of the incumbent,
(iii) its LP solution is not integral.

Hence, for the example considered in the previous section, this could be a sub-problem corresponding to any of the boxes 1, 2, 3, 4, or 9 in Figure 8.5. Instead of branching at such a node, a branch & cut procedure may first add so-called *cutting planes*. These are linear inequalities that cut off the current optimal LP solution, but do not cut off any integral solution. Figure 8.6 shows again the feasible solutions to our introductory IP for branch & bound together with a cutting plane (in green) that cuts off the current optimal LP solution. By adding a cutting plane, the LP relaxation is strengthened. One then solves the strengthened LP relaxation, and continues with that one.

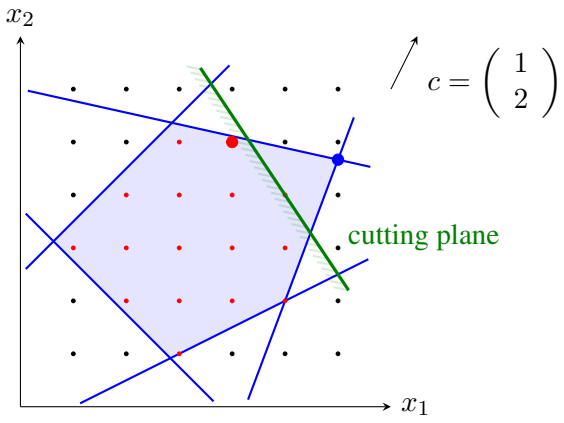


Figure 8.6: Example of a cutting plane. The cutting plane should cut off the current optimal solution to the linear relaxation but must not cut off any integral solutions.

The stronger LP, obtained after adding a cutting plane, leads to a better upper bound and makes it easier to use bounding, because the optimal value of the LP may have decreased after adding a cutting plane. Also, it is possible that the strengthened LP has an LP relaxation with an integral optimal solution, or one that is infeasible; in both cases, no further branching is necessary. In short, the goal of adding cutting planes is to speed up the procedure by reducing the total number of branchings, and hence sub-problems to be solved.

The study of cutting planes is a research area in its own. A detailed discussion of cutting planes is beyond the scope of this script. Still, to provide a glimpse into how cutting planes can be generated, we show how so-called Chvátal-Gomory cuts can be found through the simplex tableau.

8.3.1 Generating a Chvátal-Gomory cut through the simplex tableau

Chvátal-Gomory cuts are obtained by rounding the coefficients of linear inequalities that are valid for the linear relaxation. To exemplify how they can be constructed, consider again our introductory IP example for branch & bound, which we repeat below for convenience.

$$\begin{array}{rrcrcrcrcl} \max & 75x_1 & + & 6x_2 & + & 3x_3 & + & 33x_4 & & \\ & 774x_1 & + & 76x_2 & + & 22x_3 & + & 42x_4 & \leq & 875 \\ & 67x_1 & + & 27x_2 & + & 794x_3 & + & 53x_4 & \leq & 875 \\ & & & & & & & x & \in & \{0,1\}^4 \end{array}$$

We show how to add a cutting plane to the linear relaxation of this problem, i.e., this corresponds to box 1 in Figure 8.5. For this we solve the linear relaxation of the above problem with the Simplex Method. We first write the relaxation in canonical form:

$$\begin{array}{rrcrcrcrcl} \max & 75x_1 & + & 6x_2 & + & 3x_3 & + & 33x_4 & & \\ & 774x_1 & + & 76x_2 & + & 22x_3 & + & 42x_4 & \leq & 875 \\ & 67x_1 & + & 27x_2 & + & 794x_3 & + & 53x_4 & \leq & 875 \\ & x_1 & & & & & & & \leq & 1 \\ & & & x_2 & & & & & \leq & 1 \\ & & & & & x_3 & & & \leq & 1 \\ & & & & & & & x_4 & \leq & 1 \\ & & & & & & & x & \in & \mathbb{R}^4_{\geq 0} \ . \end{array}$$

By adding slack variables $y_1, \ldots, y_6$ to the above inequalities, where the i-th inequality from top down gets the slack variable y_i, we obtain the problem in standard form, with the following corresponding tableau.

	x_1	x_2	x_3	x_4	1
z	-75	-6	-3	-33	0
y_1	774	76	22	42	875
y_2	67	27	794	53	875
y_3	1	0	0	0	1
y_4	0	1	0	0	1
y_5	0	0	1	0	1
y_6	0	0	0	1	1

By applying phase II of the Simplex Method, we obtained the following optimal tableau.

	y_3	y_1	y_2	y_6	1
z	14.2289	0.0784	0.0016	29.623	113.8373
x_2	-10.2608	0.0133	-0.0004	-0.5386	0.506
x_3	0.2645	-0.0005	0.0013	-0.0484	0.9337
x_1	1	0	0	0	1
y_4	10.2608	-0.0133	0.0004	0.5386	0.494
y_5	-0.2645	0.0005	-0.0013	0.0484	0.0663
x_4	0	0	0	1	1

Hence, an optimal LP solution is

$$(x_1, x_2, x_3, x_4) = (1, 0.506, 0.9337, 1) \ ,$$

which corresponds to the solution that is shown in box 1 of Figure 8.5. This solution is fractional because x_2 and x_3 have fractional values. In particular, this implies that these two variables are basic; for otherwise, they would have value zero in the basic solution that corresponds to the tableau. Consider one of the two variables, say x_2. Its corresponding constraint in the optimal tableau reads

$$x_2 - 10.2608y_3 + 0.0133y_1 - 0.0004y_2 - 0.5386y_6 = 0.506 \ .$$

By rounding down the coefficients of the left-hand side to the next integer, we obtain the following weaker constraint with integral coefficients:

$$x_2 + \lfloor -10.2608 \rfloor y_3 + \lfloor 0.0133 \rfloor y_1 + \lfloor -0.0004 \rfloor y_2 + \lfloor -0.5386 \rfloor y_6 \leq 0.506 \ ,$$

i.e.,

$$x_2 - 11y_3 - y_2 - y_6 \leq 0.506 \ .$$

We now observe that in an integral solution, not just the variables x_1, x_2, x_3, x_4 need to be integral, but also the slack variables $y_1, \ldots, y_6$, because all the coefficients and right-hand sides of our problem are integral. (Note that this can be achieved for any rational LP/IP by scaling the constraints.) Because the above inequality has integral coefficients on the left-hand side, any integral solution will lead to an integral left-hand side value. Hence, we can round down the right-hand side coefficient to the next integer and obtain an inequality that is valid for any feasible *integral* solution to the original problem. The resulting inequality is

$$x_2 - 11y_3 - y_2 - y_6 \leq \lfloor 0.506 \rfloor = 0 \ . \tag{8.2}$$

Moreover, the basic optimal solution we computed must violate this constraint, because it sets the non-basic variables to 0. Hence, even after rounding its coefficients, the constraint remains tight for this solution. Moreover, we strictly decrease the right-hand side through the rounding step, because we assumed it to be fractional. Consequently, the constraint must be violated by the basic optimal solution corresponding to the tableau. Thus, the cutting plane we found, given by (8.2), indeed cuts off the computed basic optimal LP solution, and we can add it to the LP.