

Mixed Integer Programming

Theory and Algorithms

Sirong Luo

¹School of Statistics and Data Science
Shanghai University of Finance and Economics

Table of Contents

- 1 Mixed Integer Programming
- 2 Numerical Mixed Integer Programming Solvers
- 3 Relaxations and Duality
 - Linear Programming Relaxation
 - Lagrangian Relaxation
 - A Heuristic based on Lagrangian Relaxation for Clustering
- 4 Algorithms for Solving Mixed Integer Programs
 - Branch-and-Bound Method
 - Cutting-Plane Method

Mixed Integer Programming

- The types of optimization models that we have discussed so far, namely linear and quadratic programming, allow variables to take a continuum of values. In particular, the numerical solutions to these kinds of models may have fractional values. For instance, the solution to a portfolio construction model could suggest a plan to purchase 3205.76 shares of stock XYZ. In many cases it is natural to round this value and to interpret it as a suggestion to purchase 3205 or even 3200 shares of stock XYZ. However, if a variable in an optimization model is associated with choosing among two or more alternatives, for example, as in the capital budgeting problem described below, then a model that suggests taking fractions of each of the alternatives would be of limited value. Instead, a binary decision, namely "to choose" or "not to choose", needs to be made for each alternative.

Mixed Integer Programming

In general, an integer variable in an optimization model is a variable that is restricted to take integer values only. A mixed integer program is an optimization problem with the constraint that some of the variables must take integer values. In particular a mixed integer linear program is a problem of the form

$$\begin{aligned} \min_{\mathbf{x}} & \mathbf{c}^T \mathbf{x} \\ \text{s.t. } & \mathbf{Ax} = \mathbf{b} \\ & \mathbf{Dx} \geq \mathbf{d} \\ & x_j \in \mathbb{Z}, j \in J \end{aligned} \tag{8.1}$$

for some vectors $\mathbf{c} \in \mathbb{R}^n$, $\mathbf{b} \in \mathbb{R}^m$, $\mathbf{d} \in \mathbb{R}^p$, matrices $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{D} \in \mathbb{R}^{p \times n}$, and subset $J \subseteq \{1, \dots, n\}$ of the variables. When all variables are restricted to be integer, that is, when $J = \{1, \dots, n\}$, the problem (8.1) is called a *pure integer linear program*.

Mixed Integer Programming

An important case occurs when a model includes *binary* variables; that is, variables that are restricted to take the value 0 or 1. When all the variables in a mixed integer program are of this kind, it is called a binary program. As the examples below show, binary variables enable the modeling of important realistic features such as logical constraints, cardinality and threshold constraints, and others. However, this improvement in modeling power comes with a tradeoff in computational cost. The presence of a significant number of integer variables in an optimization problem can make it extremely difficult or impossible to solve unless there is a specific exploitable structure.

Mixed Integer Programming

Example 8.1 (Capital budgeting) Suppose we have a capital of 19 million dollars for long-term investment and have identified four investment opportunities with the following investment requirements and net present values (in million dollars):

	Investment 1	Investment 2	Investment 3	Investment 4
Required investment	7	10	6	3
Net present value	9	11	7	4

What investments should we choose to maximize our total net present value? Each investment is a "take it or leave it" opportunity: the investment must be funded entirely or not at all.

This problem can be formulated as the following binary linear programming model.

Mixed Integer Programming

Binary linear programming model for capital budgeting

Variables:

$$x_i = \begin{cases} 1 & \text{if investment } i \text{ is undertaken} \\ 0 & \text{otherwise} \end{cases} \quad \text{for } i = 1, \dots, 4.$$

Objective, in millions of dollars:

$$\max 9x_1 + 11x_2 + 7x_3 + 4x_4$$

Constraints:

$$\begin{aligned} 7x_1 + 10x_2 + 6x_3 + 3x_4 &\leq 19 && \text{(budget constraint)} \\ x_i &\in \{0, 1\} \text{ for } i = 1, \dots, 4 && \text{(binary variables).} \end{aligned}$$

Mixed Integer Programming

The optimal solution to the linear programming relaxation of this model, obtained by relaxing the binary constraints $x_i \in \{0, 1\}$, for $i = 1, \dots, 4$, to $0 \leq x_i \leq 1$, for $i = 1, \dots, 4$, is

$$\mathbf{x}^* = \begin{bmatrix} 1 \\ 0.3 \\ 1 \\ 1 \end{bmatrix}$$

This is not a feasible solution as x_2^* is not binary. If we round x_2^* to 0 we get a feasible solution. However, a better (and in fact the optimal) solution is

$$\mathbf{x}^* = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

Mixed Integer Programming

This could be counterintuitive as Investment 1 has the best "bang for the buck"; that is, has the highest ratio of net present value to investment requirement.

The presence of binary variables also readily enables the modeling of logical restrictions. For example, the logical restriction

If Investment 2 is made then Investment 4 must also be made can be modeled via the constraint

$$x_2 \leq x_4$$

Similarly, the logical constraint

If Investment 1 is made then Investment 3 must not be made can be modeled via the constraint

$$x_1 + x_3 \leq 1$$

Example 8.2 (Clustering) Clustering is a popular technique in data analysis. It is concerned with partitioning a collection of objects into subsets or "clusters" so that the objects within each cluster are more closely related with each other than with objects assigned to different clusters. Suppose we wish to partition a collection of N objects into $K < N$ clusters based on some kind of similarity measure:

$$\rho_{ij} = \text{similarity measure between objects } i, j$$

To give a financial flavor to this example, assume the objects to be clustered are N stocks and the similarity measure ρ_{ij} is the correlation between the returns of stocks i and j .

Mixed Integer Programming

We next describe a possible approach to the above clustering problem via binary programming. This approach is closely related to the popular K -median problem. Before diving into the binary programming formulation, we describe some of the main ideas. Assume the objects are indexed $1, \dots, N$ and are to be partitioned into the K clusters C_1, \dots, C_K . A key idea is to designate an element j_ℓ in each cluster C_ℓ as the centroid of cluster C_ℓ . This choice suggests the following natural measure of the similarity within cluster C_ℓ :

$$\sum_{i \in C_\ell} \rho_{i, j_\ell}$$

and in turn it gives the following overall measure of the quality of the clusters C_1, \dots, C_K :

$$\sum_{\ell=1}^K \sum_{i \in C_\ell} \rho_{i, j_\ell}$$

The following crucial observation is key in our formulation. The centroid j_ℓ represents the elements in cluster C_ℓ . Indeed, each cluster contains precisely the objects assigned to its centroid, and the clusters are completely determined by the choice of the centroids. These ideas are formalized in the following binary linear programming model.

Mixed Integer Programming

Binary linear programming model for clustering

Variables:

$$y_j = \begin{cases} 1 & \text{if } j \text{ is a centroid} \\ 0 & \text{otherwise} \end{cases} \quad \text{for } j = 1, \dots, N.$$

$$x_{ij} = \begin{cases} 1 & \text{if } i \text{ is represented by } j \\ 0 & \text{otherwise} \end{cases} \quad \text{for } i, j = 1, \dots, N.$$

Objective:

$$\max \sum_{j=1}^N \sum_{i=1}^N \rho_{ij} x_{ij}.$$

Mixed Integer Programming

Constraints:

$$\sum_{j=1}^N y_j = K \quad (\text{choose } K \text{ centroids})$$

$$\sum_{j=1}^N x_{ij} = 1, \quad \text{for } i = 1, \dots, N \quad (\text{each object must be represented by one centroid})$$

$$x_{ij} \leq y_j, \quad \text{for } i, j = 1, \dots, N \quad (i \text{ is represented by } j \text{ only if } j \text{ is a centroid})$$

$$x_{ij}, y_j \in \{0, 1\} \quad \text{for } i, j = 1, \dots, N \quad (\text{binary variables}).$$

Another correct formulation is obtained if we replace the third set of N^2 constraints

$$x_{ij} \leq y_j, \quad \text{for } i, j = 1, \dots, N$$

with the set of N constraints

$$\sum_{i=1}^N x_{ij} \leq N y_j, \quad \text{for } j = 1, \dots, N$$

Numerical Mixed Integer Programming Solvers

Excel Solver

The steps required for solving a mixed integer (or binary) linear program in Excel Solver are nearly identical to those for solving linear programs. The only new step is to state that some variables are integer (or binary).

Figure 8.1 displays a printout of an Excel spreadsheet implementation of the binary linear programming model for Example 8.1 as well as the dialog box obtained when we run the Excel add-in Solver. The spreadsheet model contains the three components of the binary program. The decision variables are in the range B7 : E7. The left-hand side of the budget constraint is in cell B9 and the objective function is in cell B10. The Excel formulas in cells B9 and B10 are SUMPRODUCT (B4:E4,B7:E7) and SUMPRODUCT(B5:E5,B7:E7) respectively. In addition to these components, notice the constraint

$$\text{\$ B\$7 : \$E\$7} = \text{binary}$$

in the Solver dialog box.

Numerical Mixed Integer Programming Solvers

	A	B	C	D	E
1					
2					
3		investment 1	investment 2	investment 3	investment 4
4	investment requirement	7	10	6	3
5	net present value	9	11	7	4
6					
7	choose?	0	1	1	1
8					
9	total investment requirements	19	<=	19	
10	total net present value	22			
11					
12					
13					
14					
15					
16					
17					
18					
19					
20					
21					
22					
23					
24					
25					
26					
27					
28					

Solver Parameters

Set Objective:

To: ☒ Max ☐ Min ☐ Value Of:

By Changing Variable Cells:

Subject to the Constraints:

☒ Make Unconstrained Variables Non-Negative


Select a Solving Method:

Buttons: Add, Change, Delete, Reset All, Load/Save, Options

Figure 8.1 Spreadsheet implementation and the Solver dialog box for the capital budgeting model

MATLAB CVX

Figure 8.2 displays a CVX script for the same problem.



The screenshot shows the MATLAB environment with the following code in the Command Window:

```

1 % Matlab CVX code for capital budgeting model
2 n = 4 ;
3 invest_required = [7;10;6;3] ;
4 npv = [9;11;7;4] ;
5 cvx_begin
6     cvx_solver gurobi
7     variable x(n) binary ;
8     maximize (npv'*x)
9     invest_required'*x <= 19 ;
10 cvx_end

```

Figure 8.2 MATLAB CVX code for capital budgeting model

Using either of these solvers we obtain the optimal solution:

$$\mathbf{x}^* = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

Relaxations and Duality

A relaxation of an optimization model

$$\begin{array}{ll}\min_{\mathbf{x}} & f(\mathbf{x}) \\ \text{s.t.} & \mathbf{x} \in \mathcal{X}\end{array}$$

is another optimization model

$$\begin{array}{ll}\min_{\mathbf{x}} & \tilde{f}(\mathbf{x}) \\ \text{s.t.} & \mathbf{x} \in \tilde{\mathcal{X}}\end{array}$$

that satisfies $\mathcal{X} \subseteq \tilde{\mathcal{X}}$ and $\tilde{f}(x) \leq f(x)$ for $x \in \mathcal{X}$. In other words, a relaxation is a less stringent optimization model obtained by "relaxing" some of the original constraints and "relaxing" its objective function. Relaxation plays a central role in a variety of algorithms for solving mixed integer programs. We next describe two widely used types of relaxations for mixed integer programming, namely linear programming and Lagrangian relaxations.

Linear Programming Relaxation

The linear programming relaxation of the mixed integer linear program (8.1) is the linear program obtained by dropping the integrality constraints; that is,

$$\begin{aligned} \min_{\mathbf{x}} & \mathbf{c}^T \mathbf{x} \\ \text{s.t. } & \mathbf{Ax} = \mathbf{b} \\ & \mathbf{Dx} \geq \mathbf{d}. \end{aligned} \tag{8.2}$$

Linear Programming Relaxation

Similarly, the linear programming relaxation of a mixed binary linear program

$$\begin{aligned} \min_{\mathbf{x}} \quad & \mathbf{c}^T \mathbf{x} \\ \text{s.t.} \quad & \mathbf{Ax} = \mathbf{b} \\ & \mathbf{Dx} \geq \mathbf{d} \\ & x_j \in \{0, 1\}, j \in J \end{aligned} \tag{8.3}$$

is the linear program

$$\begin{aligned} \min_{\mathbf{x}} \quad & \mathbf{c}^T \mathbf{x} \\ \text{s.t.} \quad & \mathbf{Ax} = \mathbf{b} \\ & \mathbf{Dx} \geq \mathbf{d} \\ & 0 \leq x_j \leq 1, j \in J. \end{aligned} \tag{8.4}$$

The proof of the following proposition is straightforward and we leave it as an exercise.

Proposition 8.3 *Consider the mixed integer program (8.1) and its linear programming relaxation (8.2). Then the following facts hold.*

- (a) The optimal value of the relaxation (8.2) is less than or equal to the optimal value of the mixed integer linear program (8.1).*
- (b) If the relaxation (8.2) is infeasible, then so is the mixed integer linear program (8.1).*
- (c) If the optimal solution \mathbf{x}^* of the relaxation (8.2) satisfies $x_j^* \in \mathbb{Z}$ for $j \in J$ then \mathbf{x}^* is also an optimal solution to the mixed integer linear program (8.1).*

Linear Programming Relaxation

The analogous facts also hold for the mixed binary linear program (8.3) and its linear relaxation (8.4). Proposition 8.3 suggests a possible avenue for solving (8.1): solve the (more tractable) linear programming relaxation (8.2). If this solution satisfies the relevant integrality constraints, then we have solved (8.1). If not, the lower bound obtained by solving (8.2) provides valuable information. For instance, if we can find a feasible solution to (8.1), then the quality of this solution can be assessed by comparing it with the lower bound obtained from solving (8.2). The sections below elaborate on this idea. In particular, Section 8.4 sketches algorithms that solve mixed linear integer programs by systematically solving a sequence of linear programming relaxations.

Linear Programming Relaxation

Proposition 8.3 also leads to the following somewhat counterintuitive conclusion about integer programming formulations. As noted in Example 8.2, there could be several correct and thus equivalent integer linear programming formulations to a given problem. Among them, it is generally better to have a formulation with a "tight" linear programming relaxation. Typically, a formulation with more constraints has a tighter linear programming relaxation and hence might be easier to solve.

Lagrangian Relaxation

The Lagrangian framework discussed in previous chapters can be extended to obtain relaxations of an optimization model. The intuitive idea is to obtain a relaxation of a model by shifting a set of "difficult" constraints to the objective. To be more precise, consider an optimization problem of the form

$$\begin{aligned} \min_{\mathbf{x}} \quad & \mathbf{c}^T \mathbf{x} \\ \text{s.t.} \quad & \mathbf{Ax} = \mathbf{b} \\ & \mathbf{x} \in \mathcal{X} \end{aligned} \tag{8.5}$$

where the combined set of constraints $\mathbf{Ax} = \mathbf{b}, \mathbf{x} \in \mathcal{X}$ is "difficult" but the set of constraints $\mathbf{x} \in \mathcal{X}$ is "easy". (We will discuss a concrete example of this situation in Section 8.3.3.)

Lagrangian Relaxation

A relaxation for (8.5) can be obtained as follows. Assume \mathbf{u} is a vector of suitable dimension and consider the following problem without the difficult constraints:

$$\begin{aligned} L(\mathbf{u}) := \min_{\mathbf{x}} & \mathbf{c}^\top \mathbf{x} + \mathbf{u}^\top (\mathbf{b} - \mathbf{Ax}) \\ \text{s.t. } & \mathbf{x} \in \mathcal{X} \end{aligned} \tag{8.6}$$

The problem (8.6) is a Lagrangian relaxation of (8.5). The following proposition is in the same spirit as Proposition 8.3. Again, its proof is straightforward and we leave it as an exercise.

Proposition 8.4 *Consider the optimization problem (8.5) and its Lagrangian relaxation (8.6) for some vector \mathbf{u} . Then the following facts hold.*

- (a) The optimal value $L(\mathbf{u})$ of the relaxation (8.6) is less than or equal to the optimal value of (8.5).*
- (b) If the optimal solution \mathbf{x}^* of the relaxation (8.6) satisfies $\mathbf{Ax}^* = \mathbf{b}$ then \mathbf{x}^* is also an optimal solution to (8.5).*

Lagrangian Relaxation

The *Lagrangian dual* of (8.5) is the problem of finding the best Lagrangian relaxation

$$\max_{\mathbf{u}} L(\mathbf{u})$$

where $L(\mathbf{u})$ is the optimal value of (8.6). We note that the function $\mathbf{u} \mapsto L(\mathbf{u})$ is concave; that is, $\mathbf{u} \mapsto -L(\mathbf{u})$ is convex. Thus the Lagrangian dual is a convex optimization problem.

Lagrangian Relaxation

The Lagrangian relaxation and Lagrangian dual also extend to problems where the set of difficult constraints involves both equalities and inequalities. We simply need to be a bit careful about the sign of the multipliers for the inequality constraints. Consider the optimization problem

$$\begin{aligned} \min_{\mathbf{x}} \quad & \mathbf{c}^T \mathbf{x} \\ \text{s.t.} \quad & \mathbf{Ax} = \mathbf{b} \\ & \mathbf{Dx} \geq \mathbf{d} \\ & \mathbf{x} \in \mathcal{X}. \end{aligned} \tag{8.7}$$

Lagrangian Relaxation

Given vectors \mathbf{u}, \mathbf{v} of suitable dimension with $\mathbf{v} \geq \mathbf{0}$ we obtain the following Lagrangian relaxation:

$$\begin{aligned} L(\mathbf{u}, \mathbf{v}) := \min_{\mathbf{x}} & \mathbf{c}^\top \mathbf{x} + \mathbf{u}^\top (\mathbf{b} - \mathbf{Ax}) + \mathbf{v}^\top (\mathbf{d} - \mathbf{Dx}) \\ \text{s.t.} \quad & \mathbf{x} \in \mathcal{X}. \end{aligned} \tag{8.8}$$

We have the following extended version of Proposition 8.4.

Lagrangian Relaxation

Proposition 8.5 Consider the optimization problem (8.7) and its Lagrangian relaxation (8.8) for some vectors \mathbf{u}, \mathbf{v} with $\mathbf{v} \geq \mathbf{0}$. Then the following facts hold.

(a) The optimal value $L(\mathbf{u}, \mathbf{v})$ of the relaxation (8.8) is less than or equal to the optimal value of (8.7).

(b) If the optimal solution \mathbf{x}^* of the relaxation (8.8) satisfies $\mathbf{A}\mathbf{x}^* = \mathbf{b}$, $\mathbf{D}\mathbf{x}^* \geq \mathbf{d}$, and $\mathbf{v}^\top (\mathbf{D}\mathbf{x}^* - \mathbf{d}) = 0$, then \mathbf{x}^* is also an optimal solution to (8.7).

The Lagrangian dual of (8.7) is

$$\begin{aligned} \max_{\mathbf{u}, \mathbf{v}} & L(\mathbf{u}, \mathbf{v}) \\ \text{s.t. } & \mathbf{v} \geq \mathbf{0} \end{aligned}$$

A Heuristic based on Lagrangian Relaxation for Clustering

We next describe a particularly successful application of Lagrangian relaxation for the clustering problem introduced in Example 8.2, namely,

$$\begin{aligned} Z := \max & \sum_{i=1}^N \sum_{j=1}^N \rho_{ij} x_{ij} \\ \text{s.t.} & \sum_{j=1}^N y_j = K \\ & \sum_{j=1}^N x_{ij} = 1 \quad \text{for } i = 1, \dots, N \\ & x_{ij} \leq y_j \quad \text{for } i, j = 1, \dots, N \\ & x_{ij}, y_j \in \{0, 1\} \quad \text{for } i, j = 1, \dots, N. \end{aligned} \tag{8.9}$$

A Heuristic based on Lagrangian Relaxation for Clustering

The above model can be solved by general-purpose solvers such as Excel Solver or CVX, or even commercial solvers like Gurobi or CPLEX, only for relatively small values of N . One of the main difficulties is that the model involves $N^2 + N$ binary variables and $N^2 + N + 1$ constraints. For a modest value of N like $N = 100$, the model becomes unmanageable if tackled by a standard solver. At the same time, in practical clustering problem instances N can easily range in the hundreds or thousands. A heuristic based on Lagrangian relaxation developed by Cornuéjols et al. (1977) can compute approximate solutions to (8.9) for virtually unlimited values of N . We next describe the main ideas behind this heuristic.

A Heuristic based on Lagrangian Relaxation for Clustering

Consider the following Lagrangian relaxation to (8.9): given a vector of multipliers $\mathbf{u} = [u_1 \ \cdots \ u_N]^\top$ let

$$\begin{aligned} L(\mathbf{u}) &:= \max_{\mathbf{x}, \mathbf{y}} \sum_{i=1}^N \sum_{j=1}^N \rho_{ij} x_{ij} + \sum_{i=1}^N u_i \left(1 - \sum_{j=1}^N x_{ij} \right) \\ \text{s.t. } &\sum_{j=1}^N y_j = K \\ &x_{ij} \leq y_j, \quad i, j = 1, \dots, N \\ &x_{ij}, y_j = 0 \text{ or } 1, \quad i, j = 1, \dots, N. \end{aligned} \tag{8.10}$$

This Lagrangian relaxation has moved the "difficult" constraints $\sum_{j=1}^N x_{ij} = 1$, with $i = 1, \dots, N$, to the objective via the multipliers \mathbf{u} and has kept the remaining constraints as the "easy" ones. This Lagrangian relaxation satisfies the following key properties:

A Heuristic based on Lagrangian Relaxation for Clustering

Property 1: $L(\mathbf{u}) \geq Z$, where Z is the optimal value of (8.9). This is an immediate consequence of Proposition 8.3.

Property 2: For a given \mathbf{u} , (8.10) is easy to solve. To see this, first notice that

$$\begin{aligned} L(\mathbf{u}) &:= \max_{\mathbf{x}, \mathbf{y}} \sum_{i=1}^N \sum_{j=1}^N (\rho_{ij} - u_i) x_{ij} + \sum_{i=1}^N u_i \\ \text{s.t. } &\sum_{j=1}^N y_j = K \\ &x_{ij} \leq y_j, \quad i, j = 1, \dots, N \\ &x_{ij}, y_j \in \{0, 1\}, \quad i, j = 1, \dots, N. \end{aligned}$$

A Heuristic based on Lagrangian Relaxation for Clustering

Given \mathbf{y} , this shows that x_{ij} should be set to its upper bound y_j or to its lower bound 0, depending on whether the objective coefficient $\rho_{ij} - u_i$ of x_{ij} is positive or negative. Therefore $L(\mathbf{u})$ can be rewritten as

$$\begin{aligned} L(\mathbf{u}) &= \max_{\mathbf{y}} \sum_{j=1}^N C_j y_j + \sum_{i=1}^N u_i \\ \text{s.t. } &\sum_{j=1}^N y_j = K \\ &y_j \in \{0, 1\}, \quad j = 1, \dots, N \end{aligned} \tag{8.11}$$

for $C_j := \sum_{i=1}^N \max(0, \rho_{ij} - u_i)$. Finally, observe that the solution to (8.11) is readily computable: Sort the C_j in decreasing order, say $C_{j_1} \geq C_{j_2} \geq \dots \geq C_{j_N}$. The optimal solution to (8.11) is obtained by setting $\bar{y}_{j_1} = \dots = \bar{y}_{j_K} = 1$ and the remaining \bar{y}_j s to zero. We get $L(\mathbf{u}) = \sum_{t=1}^K C_{j_t} + \sum_{i=1}^N u_i$.

A Heuristic based on Lagrangian Relaxation for Clustering

Property 3: Based on the optimal solution $\bar{\mathbf{y}}$ of $L(\mathbf{u})$ obtained in Property 2, one can get a heuristic (ad hoc) solution $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ for (8.9) and an assessment of how good it is.

- Each \mathbf{u} gives the upper bound $L(\mathbf{u}) \geq Z$ and the following heuristic feasible solution $\bar{\mathbf{x}}$ to (8.9). Let $\bar{\mathbf{y}}$ solve (8.11). Next, for each $i = 1, \dots, N$, assign i to the most similar centroid among the K centroids such that $\bar{y}_j = 1$. That is, let $j(i) = \operatorname{argmax}_{j: \bar{y}_j = 1} \rho_{ij}$ and let $\bar{\mathbf{x}}$ be as follows:

$$\bar{x}_{ij} = \begin{cases} 1 & \text{if } j = j(i) \\ 0 & \text{otherwise} \end{cases}$$

A Heuristic based on Lagrangian Relaxation for Clustering

- If $\sum_{i,j} \rho_{ij} \bar{x}_{ij}$ and $L(\mathbf{u})$ are close to each other, then we have a nearoptimal solution. To see this, observe that $\sum_{i,j} \rho_{ij} \bar{x}_{ij} \leq Z \leq L(\mathbf{u})$. Thus if $\sum_{i,j} \rho_{ij} \bar{x}_{ij}$ and $L(\mathbf{u})$ are close to each other, they must be close to the optimal value Z as well.
- To get the best upper bound $L(\mathbf{u})$ together with a heuristic solution of the above kind, solve

$$\min_{\mathbf{u}} L(\mathbf{u}).$$

This turns out to be a manageable convex optimization problem. In particular, it is amenable to a subgradient algorithm that we describe in Chapter 20.

Algorithms for Solving Mixed Integer Programs

The modeling power of mixed integer programming comes with some cost. Mixed integer programming belongs to the class of *NP-hard* computational problems (Conforti et al., 2014). In layman's terms, this means that, unlike convex optimization problems, which can be solved with fast and reliable numerical algorithms, the same cannot be expected for mixed integer programs. The algorithms that we describe next can in principle solve any mixed integer linear programs in finitely many steps. However, the NP-hardness of integer programming implies that for some problem instances the computational cost incurred by these algorithms could be insurmountable even for any foreseeable amount of computational power.

Algorithms for Solving Mixed Integer Programs

The two most popular generic methods for solving mixed integer linear programs are *cutting planes* and *branch and bound*. Both of these methods rely extensively on linear programming relaxations. A cutting plane is a new linear constraint to the linear programming relaxation that "cuts off" non-integer solutions without cutting off any feasible solution of the original mixed integer linear program. The method of cutting planes was proposed by Dantzig et al. (1954) in the context of the traveling-salesman problem, and by Gomory (1958, 1960) for pure integer linear programs and mixed integer linear programs, respectively. The method is based on solving a sequence of increasingly tighter linear programming relaxations by adding cutting planes until a solution to the mixed integer linear program is found.

Algorithms for Solving Mixed Integer Programs

On the other hand, Land and Doig (1960) proposed a "branch-and-bound" method to solve mixed integer linear programs. Branch and bound is an enumerative procedure based on dividing the original problem into a number of smaller problems (branching) and evaluating their quality based on their linear programming relaxations (bounding). Branch and bound was the most effective technique for solving mixed integer linear programs for multiple decades. However, in the 1990s, cutting planes made a resurgence. Current state-of-the-art integer programming solvers combine cutting planes and branch and bound into an overall procedure called "branch and cut", a term coined in Padberg and Rinaldi (1987).

Branch-and-Bound Method

The gist of the branch-and-bound method can be easily grasped via a couple of examples. Consider the following integer linear program (see Figure 8.3):

$$\begin{aligned} \max \quad & x_1 + x_2 \\ \text{s.t.} \quad & -x_1 + x_2 \leq 2 \\ & 8x_1 + 2x_2 \leq 19 \\ & x_1, x_2 \geq 0 \\ & x_1, x_2 \in \mathbb{Z}. \end{aligned} \tag{8.12}$$

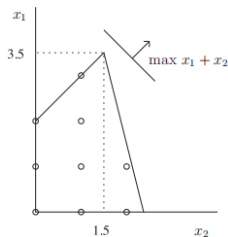


Figure 8.3 A two-variable integer program

Branch-and-Bound Methods

Step 1. Solve the linear programming relaxation of (8.12), namely

$$\begin{array}{ll}\max & x_1 + x_2 \\ \text{s.t.} & -x_1 + x_2 \leq 2 \\ & 8x_1 + 2x_2 \leq 19 \\ & x_1, x_2 \geq 0.\end{array}$$

The solution is $\bar{\mathbf{x}} = \begin{bmatrix} 1.5 & 3.5 \end{bmatrix}^T$ with objective value 5. Thus 5 is an upper bound on the optimal value of (8.12). The vector $\bar{\mathbf{x}} = \begin{bmatrix} 1.5 & 3.5 \end{bmatrix}^T$ is not a feasible solution to (8.12) since the entries of $\bar{\mathbf{x}}$ are fractional. How can we exclude this fractional solution while preserving the feasible integral solutions? One way is to branch: create two new linear programs, one with the additional constraint $x_1 \leq 1$, and the other with the additional constraint $x_1 \geq 2$. Clearly, any solution to the integer program must be feasible to one or the other of these two problems. We will solve both of these linear programs.

Step 2. Solve the first of the two new linear programs:

$$\begin{array}{ll}\max & x_1 + x_2 \\ \text{s.t.} & -x_1 + x_2 \leq 2 \\ & 8x_1 + 2x_2 \leq 19 \\ & x_1 \leq 1 \\ & x_1, x_2 \geq 0\end{array}$$

The solution is $\bar{\mathbf{x}} = \begin{bmatrix} 1 & 3 \end{bmatrix}^T$ with objective value 4. This is a feasible integral solution to (8.12). So now we have the upper bound 5 and the lower bound 4 on the optimal value of (8.12).

Step 3. Solve the second new linear program:

$$\begin{array}{ll}\max & x_1 + x_2 \\ \text{s.t.} & -x_1 + x_2 \leq 2 \\ & 8x_1 + 2x_2 \leq 19 \\ & x_1 \geq 2 \\ & x_1, x_2 \geq 0\end{array}$$

The solution is $\bar{\mathbf{x}} = \begin{bmatrix} 2 & 1.5 \end{bmatrix}^T$ with objective value 3.5. Because this value is worse than the lower bound of 4 that we already have, we do not need any further branching. We conclude that the vector $\bar{\mathbf{x}} = \begin{bmatrix} 1 & 3 \end{bmatrix}^T$ with objective value 4 found in Step 2 is an optimal solution to (8.12).

Branch-and-Bound Method

The solution of the above integer program by branch and bound required the solution of three linear programs. These problems can be arranged in a *branch-and-bound tree*, see Figure 8.4. Each node of the tree corresponds to one of the problems that were solved.

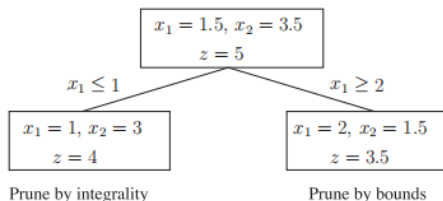


Figure 8.4 Branch-and-bound tree for (8.12)

We can stop the enumeration at a node of the branch-and-bound tree for three different reasons (when they occur, the node is said to be pruned).

Pruning by integrality occurs when the corresponding linear program has an optimal solution that is integral. This occurred in Step 2 in the above example.

Pruning by bounds occurs when the objective value of the linear program at that node is worse than the value of the best feasible solution found so far. This occurred in Step 3 in the above example.

Pruning by infeasibility occurs when the linear program at that node is infeasible. This did not occur in any of the steps in the above example.

Branch-and-Bound Method

We next illustrate the branch-and-bound method is a slightly modified instance that leads to a larger branch-and-bound tree. Consider the integer linear program:

$$\begin{aligned} \max & 3x_1 + x_2 \\ \text{s.t.} \quad & -x_1 + x_2 \leq 2 \\ & 8x_1 + 2x_2 \leq 19 \\ & x_1, x_2 \geq 0 \\ & x_1, x_2 \in \mathbb{Z} \end{aligned} \tag{8.13}$$

Figure 8.5 depicts the branch-and-bound tree for this problem.

Branch-and-Bound Method

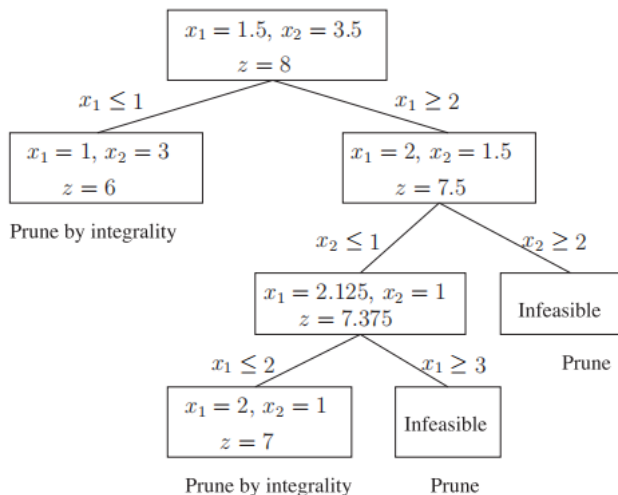


Figure 8.5 Branch-and-bound tree for (8.13)

Branch-and-Bound Method

Algorithm 8.1 sketches the branch-and-bound method for a general mixed integer linear program of the form (8.1). The branch-and-bound method keeps a list of linear programming problems obtained by relaxing the integrality requirements on the variables and imposing constraints such as $x_j \leq u_j$ or $x_j \geq l_j$. Each such linear program corresponds to a node of the branch-and-bound tree. It will be convenient to let N_i denote both a node and its corresponding linear program in the branch-and-bound tree. Let \mathbf{x}^i and z_i denote respectively the optimal solution and optimal value of the linear program N_i with the convention $z_i = \infty$ if N_i is infeasible. Let N_0 denote the root node of the branch-and-bound tree: it corresponds to the linear programming relaxation (8.2) of (8.1).

Throughout the algorithm we let \mathcal{L} denote the list of nodes that must still be solved. These are the nodes that have not been pruned nor branched on. Throughout the algorithm \mathbf{x}^* denotes the best feasible solution found so far and z_U its objective value. The value z_U is also the best upper bound on the optimal value z of (8.1) so far. Initially, the bound z_U can be derived from a heuristic solution to (8.1), or it can be set to $+\infty$ if no heuristic solution is available.

Algorithm 8.1 Branch-and-bound method

- 1: $\mathcal{L} := \{N_0\}$, $z_U := +\infty$, $\mathbf{x}^* := \emptyset$ (*initialization*)
 - 2: **if** $\mathcal{L} = \emptyset$ **then** HALT and **return** the vector \mathbf{x}^* **end if** (*termination*)
 - 3: choose and delete a node N_i from \mathcal{L} and solve it (*select next node to solve*)
 - 4: **if** $z_i \geq z_U$ **then** go to step 2 **end if** (*prune N_i*)
 - 5: **if** \mathbf{x}^i is feasible for (8.1) **then** (*update upper bound and prune*)
 $z_U := z_i$; $\mathbf{x}^* := \mathbf{x}^i$
 delete from \mathcal{L} all nodes N_k with $z_k \geq z_U$
 go to step 2
 - 6: **else** (*branch from N_i*)
 choose $j \in J$ such that $\bar{x}_j^i \notin \mathbb{Z}$
 branch on variable x_j , that is, construct two new linear programs
 N_i^1 : add the new constraint $x_j \leq \lfloor \bar{x}_j^i \rfloor$ to N_i
 N_i^2 : add the new constraint $x_j \geq \lceil \bar{x}_j^i \rceil$ to N_i
 add N_i^1, N_i^2 to \mathcal{L} and go to step 2
 - 7: **end if**
-

Branch-and-Bound Method

There are a variety of strategies for node selection (step 3) and for branching (step 6). Even more important to the success of branch and bound is the ability to prune the tree (steps 4 and 5). This will occur when z_U is a good upper bound and when z_i is a good lower bound. For this reason, it is crucial to have a formulation of (8.1) whose linear programming relaxation has an optimal value z_{LP} as close as possible to the optimal value z of (8.1).

Cutting-Plane Method

A *valid inequality* for a mixed integer linear program is a linear inequality that is satisfied by all feasible solutions. A *cutting plane* of a mixed integer linear program is a valid inequality that cuts off some solutions to its linear programming relaxation.

Cutting-Plane Method

As we noted in Proposition 8.3, if an optimal solution of the linear programming relaxation satisfies the integrality constraints of a mixed integer linear program, then it is an optimal solution to the mixed integer linear program. The gist of cutting-plane methods is the observation that, when the latter does not occur, the linear programming relaxation can be strengthened by adding a cutting plane that cuts off its optimal solution.

Gomory (1960) proposed the following approach for solving mixed integer linear programs. Assume the variables in the problem are non-negative and satisfy the equality constraint

$$\sum_{j \in J} a_j x_j + \sum_{j \notin J} a_j x_j = b \quad (8.14)$$

Cutting-Plane Method

Assume that b is not an integer and let f_0 be its fractional part, i.e. $b = \lfloor b \rfloor + f_0$, where $0 < f_0 < 1$. For $j \in J$, let $a_j = \lfloor a_j \rfloor + f_j$, where $0 \leq f_j < 1$. Replacing in (8.14) and moving sums of integer products to the right, we get

$$\sum_{j \in J: f_j \leq f_0} f_j x_j + \sum_{j \in J: f_j > f_0} (f_j - 1) x_j + \sum_{j \notin J} a_j x_j = k + f_0$$

where k is some integer. Using the fact that $k \leq -1$ or $k \geq 0$, we must have either

$$- \sum_{j \in J: f_j \leq f_0} \frac{f_j}{1 - f_0} x_j + \sum_{j \in J: f_j > f_0} \frac{1 - f_j}{1 - f_0} x_j - \sum_{j \notin J} \frac{a_j}{1 - f_0} x_j \geq 1$$

or

$$\sum_{j \in J: f_j \leq f_0} \frac{f_j}{f_0} x_j - \sum_{j \in J: f_j > f_0} \frac{1 - f_j}{f_0} x_j + \sum_{j \notin J} \frac{a_j}{f_0} x_j \geq 1$$

This is of the form $\sum_j c_j x_j \geq 1$ or $\sum_j d_j x_j \geq 1$, which implies $\sum_j \max(c_j, d_j) x_j \geq 1$ because the variables x_j are non-negative.

Cutting-Plane Method

Which is the larger of the two coefficients c_j and d_j in our case? The answer is easy since one coefficient is positive and the other is negative for each variable x_j . Therefore, we get

$$\sum_{j \in J: f_j \leq f_0} \frac{f_j}{f_0} x_j + \sum_{j \in J: f_j > f_0} \frac{1 - f_j}{1 - f_0} x_j + \sum_{j \notin J: a_j > 0} \frac{a_j}{f_0} x_j - \sum_{j \notin J: a_j < 0} \frac{a_j}{1 - f_0} x_j \geq 1 \quad (8.15)$$

Inequality (8.15) is valid for all $\mathbf{x} \geq 0$ that satisfy (8.14) with x_j integer for $j \in J$. It is called a *Gomory mixed integer cut*.

Cutting-Plane Method

We illustrate the use of Gomory cuts on problem (8.12). To that end, we first add slack variables x_3 and x_4 to turn the inequality constraints into equalities:

$$\begin{array}{ll}\max & x_1 + x_2 \\ \text{s.t.} & -x_1 + x_2 + x_3 = 2 \\ & 8x_1 + 2x_2 + x_4 = 19 \\ & x_1, x_2, x_3, x_4 \geq 0 \\ & x_1, x_2 \in \mathbb{Z}\end{array}$$

Solving the linear programming relaxation by the simplex method we get the optimal basis $B = \{1, 2\}$ and so the constraints of the linear programming relaxation can be written as

$$\begin{array}{l}x_1 - 0.2x_3 + 0.1x_4 = 1.5 \\ x_2 + 0.8x_3 + 0.1x_4 = 3.5 \\ x_1, x_2, x_3, x_4 \geq 0.\end{array}$$

Cutting-Plane Method

The corresponding basic solution is $x_3 = x_4 = 0, x_1 = 1.5, x_2 = 3.5$ with objective value $z = 5$. This solution is not integer. Let us generate the Gomory cut corresponding to the equation

$$x_1 - 0.2x_3 + 0.1x_4 = 1.5$$

We have $f_0 = 0.5, f_1 = f_2 = 0, a_3 = -0.2$ and $a_4 = 0.1$. Applying formula (8.15), we get the Gomory cut

$$\frac{0.2}{1 - 0.5}x_3 + \frac{0.1}{0.5}x_4 \geq 1, \quad \text{i.e.,} \quad 2x_3 + x_4 \geq 5$$

Since $x_3 = 2 + x_1 - x_2$ and $x_4 = 19 - x_1 - 2x_2$, we can express the above Gomory cut in terms of x_1, x_2 :

$$3x_1 + 2x_2 \leq 9$$

Branch-and-Bound Method

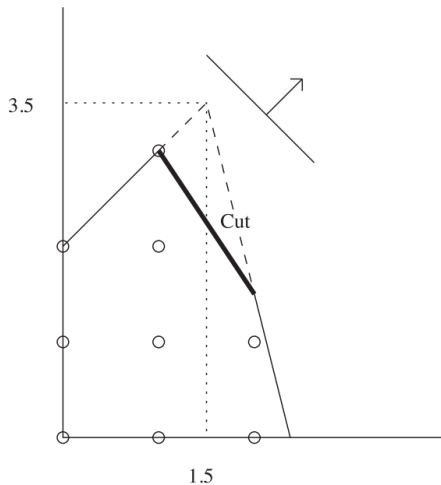


Figure 8.6 Formulation strengthened by a cut

Cutting-Plane Method

Adding this cut to the linear programming relaxation, we get the following strengthened linear programming relaxation (see Figure 8.6):

$$\begin{array}{ll}\max & x_1 + x_2 \\ \text{s.t.} & -x_1 + x_2 \leq 2 \\ & 8x_1 + 2x_2 \leq 19 \\ & 3x_1 + 2x_2 \leq 9 \\ & x_1, x_2 \geq 0.\end{array}$$

The optimal solution to this linear program is $x_1 = 1, x_2 = 3$ with objective value $z = 4$. Since x_1 and x_2 are integer, this is the optimal solution to (8.12).