# Lecture notes 1: Introduction to linear and (mixed) integer programs

Vincent Conitzer

## 1 An example

We will start with a simple example. Suppose we are in the business of selling reproductions of two different paintings. We can sell any number of reproductions of painting 1 for \$3 each, and any number of reproductions of painting 2 for \$2 each. (This is reasonable if we are a relatively small firm that does not have a significant effect on market prices.) Unfortunately, we have only a limited amount of paint: we have 16 units of blue paint, 8 units of green paint, and 5 units of red paint. A reproduction of painting 1 requires 4 units of blue, 1 unit of green, and 1 unit of red. A reproduction of painting 2 requires 2 units of blue, 2 units of green, and 1 unit of red. How many reproductions of each painting should we create to maximize our revenue?

Let $x_1$ denote the number of reproductions of painting 1 that we create, and $x_2$ the number of reproductions of painting 2. Of course, we must have $x_1 \geq 0$ and $x_2 \geq 0$. Additionally, there is a constraint for each color of paint. For example, due to our limited amount of blue paint, we must have $4x_1 + 2x_2 \leq 16$. Similarly, we must have $x_1 + 2x_2 \leq 8$ and $x_1 + x_2 \leq 5$, due to our limited amounts of green and red paint, respectively. Our total revenue will be $3x_1 + 2x_2$, which we want to maximize. We can summarize all of this as follows:

**maximize** $3x_1 + 2x_2$
**subject to**
$4x_1 + 2x_2 \leq 16$
$x_1 + 2x_2 \leq 8$
$x_1 + x_2 \leq 5$
$x_1 \geq 0; x_2 \geq 0$

This is a *linear program*, consisting of an *objective* that we seek to maximize, and *constraints* that we must satisfy. (It is linear because the objective and the constraints are linear in the variables; for example, if there had been a term $x_1x_2$ or $x_1^2$, it would not have been a linear program.) Because this particular linear program has only two variables, we can solve it graphically by inspection (Figure 1).

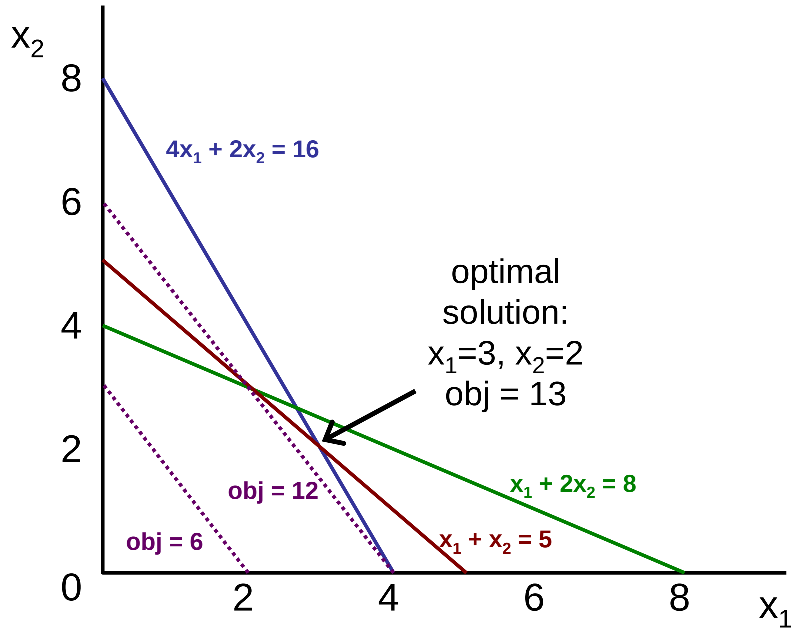


Figure 1: Graphical representation of the painting problem instance.

Each constraint is represented by a line segment; any point on or below that line segment satisfies the constraint. In order for a point to be *feasible*, that is, satisfy all the constraints, it must be below *all* of the line segments representing constraints (and $x_1$ and $x_2$ must be nonnegative). The dotted line segments represent the objective function: each dotted line segment constitutes a set of points that all have the same objective value. As we move in the northeast direction (more precisely, perpendicularly to the dotted lines), the objective value increases. It is now easy to see that the feasible point that maximizes the objective value (that is, the optimal point) is the point where the lines $4x_1 + 2x_2 = 16$ and $x_1 + x_2 = 5$ (corresponding to the blue and red constraints, respectively) intersect; this point is $x_1 = 3, x_2 = 2$, and it achieves the optimal revenue of 13. (The point where $4x_1 + 2x_2 = 16$ and $x_1 + 2x_2 = 8$ intersect is not feasible, because that point lies above the line $x_1 + x_2 = 5$.)

Now suppose we modify the problem slightly, by reducing the available blue paint from 16 to 15 units. This results in the following linear program:

**maximize** $3x_1 + 2x_2$
**subject to**
$4x_1 + 2x_2 \leq 15$
$x_1 + 2x_2 \leq 8$
$x_1 + x_2 \leq 5$
$x_1 \geq 0; x_2 \geq 0$

This corresponds to the picture in Figure 2.

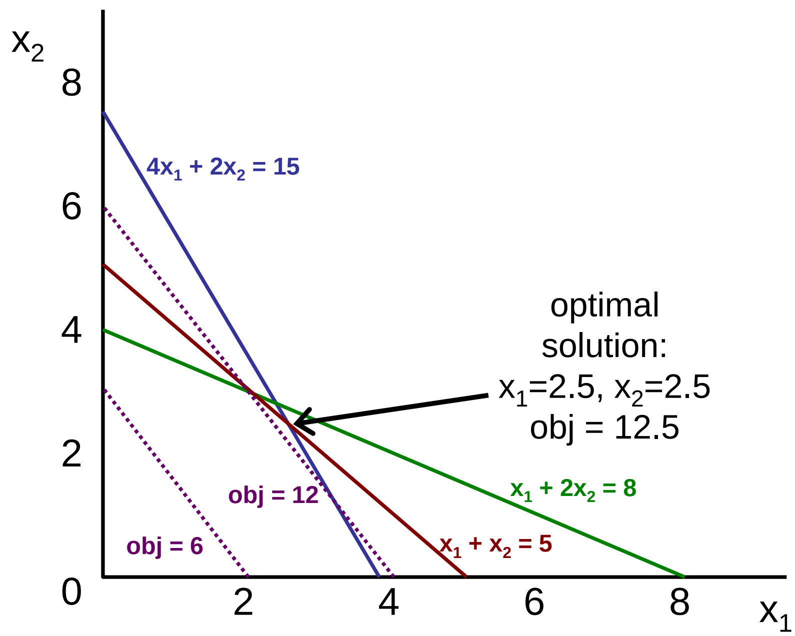


Figure 2: Graphical representation of the modified painting problem instance.

The optimal point is still at the intersection of the line segments corresponding to the blue and red constraints; it is the point $x_1 = 2.5, x_2 = 2.5$ (with an objective value of 12.5). Now, in the context of our painting example, this is not a very reasonable solution: how are we going to sell half a painting? Realistically, we need additional *integrality* constraints: $x_1$ and $x_2$ should both be integers. This type of constraint is fundamentally different from the inequality constraints that we have considered so far. Adding the integrality constraints results in the following *integer (linear) program*:

**maximize** $3x_1 + 2x_2$
**subject to**
$4x_1 + 2x_2 \leq 15$
$x_1 + 2x_2 \leq 8$
$x_1 + x_2 \leq 5$
$x_1 \geq 0$, integer; $x_2 \geq 0$, integer

Figure 3 illustrates this integer program.

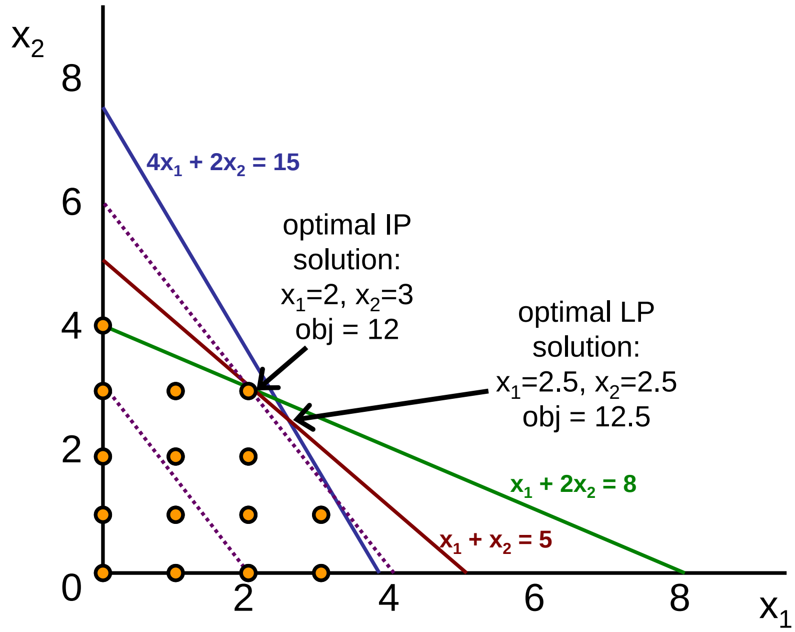


Figure 3: Graphical representation of the modified painting problem instance with integrality constraints.

The dots are the feasible points, that is, the combinations of values that meet all of the constraints (including the integrality constraints). By inspection, the optimal feasible point is now $x_1 = 2, x_2 = 3$, with an objective value of 12. Hence, the integrality constraints come at a cost, because without them, we could have achieved an objective value of 12.5. In contrast, in the original linear program (with 16 units of blue paint), if we had added integrality constraints, this would not have come at any cost, because the optimal solution to that linear program (without integrality constraints) already had integer values.

In general, there may be an integrality constraint on some variables, but not on others. For example, it may be the case that $x_1$ does not need to take an integer value, but $x_2$ does. Perhaps painting 1 is an abstract painting of which we can easily sell just a part, whereas painting 2 depicts a scene of which we cannot sell just a part. (I am sure that such a statement betrays a significant lack of artistic sensibility on my part, for which I apologize.) This results in the following *mixed integer (linear) program*:

**maximize** $3x_1 + 2x_2$
**subject to**
$4x_1 + 2x_2 \leq 15$
$x_1 + 2x_2 \leq 8$
$x_1 + x_2 \leq 5$
$x_1 \geq 0$; $x_2 \geq 0$, integer

The feasible points are now given by horizontal line segments, as illustrated in Figure 4.

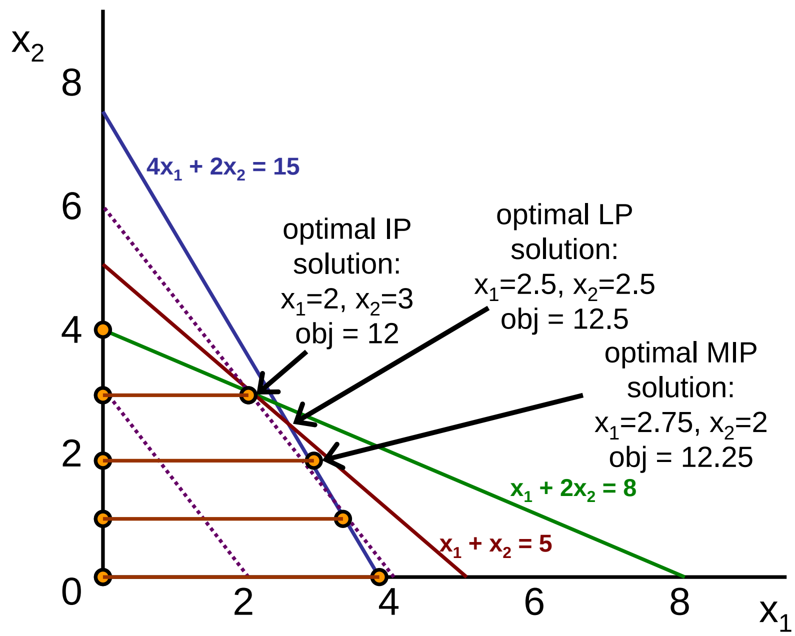


Figure 4: Graphical representation of the modified painting problem instance with one integrality constraint.

By inspection, the optimal feasible point is now the point on the blue line ($4x_1 + 2x_2 = 15$) for which $x_2 = 2$, that is, the point $x_1 = 2.75, x_2 = 2$.

This graphical method of solving linear programs obviously does not scale to problem instances with larger numbers of variables (due to my inability to draw, for me, it does not even scale to three variables). Fortunately, we will see some algorithms for this type of problem soon.

## 2 Abstract linear/integer programs

When we changed the amount of available blue paint from 16 to 15, we did not really change the form of the program. It is often useful to write a linear program in an abstract form, replacing the numbers in the problem by *parameters*. For example, we can let $c_j$ denote the selling price of a reproduction of the $j$th painting, $b_i$ the available amount of paint color $i$, and $a_{ij}$ the amount of paint $i$ required for a reproduction of painting $j$. In general, we can have $n$ different paintings, and $m$ different paint colors. This results in the following abstract program for the painting problem:

**maximize** $c_1x_1 + \ldots + c_nx_n$
**subject to**
$a_{11}x_1 + \ldots + a_{1n}x_n \leq b_1$
$\vdots$
$a_{m1}x_1 + \ldots + a_{mn}x_n \leq b_m$
$x_1 \geq 0$ (, integer); $\ldots$; $x_n \geq 0$ (, integer)

It is common to use $m$ for the number of constraints (other than the nonnegativity and integrality constraints), with $i$ indicating a particular constraint; and $n$ for the number of variables, with $j$ indicating a particular variable. **Warning:** when writing a program in abstract form like this, it is extremely important not to confuse parameters with variables. We should remember that, when we are solving a specific instance of the painting problem, we will be given exact numbers for the $c_j, b_i$, and $a_{ij}$ (the parameters), but we will be asked to solve for the optimal $x_j$ (the variables of the program). In fact, if, say, the $a_{ij}$ were also variables of the program (as well as the $x_j$), then it would no longer be linear.

The example about reproducing paintings above illustrates many of the key phenomena in (integer) linear programs. Nevertheless, in general linear programs, we can do a few more things. First of all, parameters may be negative. If all parameters are nonnegative (as in the painting example above), then it is trivial to find a feasible solution: setting all the variables to 0 will always work. If we allow negative parameters, however, then finding a feasible solution is not trivial; in fact, no feasible solution may exist.

However, we will now show that if the parameters are allowed to be negative, then the linear program

**maximize** $c_1x_1 + \ldots + c_nx_n$
**subject to**
$a_{11}x_1 + \ldots + a_{1n}x_n \leq b_1$
$\vdots$
$a_{m1}x_1 + \ldots + a_{mn}x_n \leq b_m$
$x_1 \geq 0; \ldots; x_n \geq 0$

is in fact fully general. (For simplicity, we will just consider linear programs, so that we do not consider any integrality constraints.)

First of all, instead of maximizing an objective, we often wish to minimize an objective—for example, we may wish to minimize some sort of cost. However, if our goal is to minimize $c_1x_1 + \ldots + c_nx_n$, this is equivalent to maximizing $-c_1x_1 - \ldots - c_nx_n$. Similarly, if we have a constraint of the form $a_{i1}x_1 + \ldots + a_{in}x_n \geq b_i$, then we can equivalently write $-a_{i1}x_1 - \ldots - a_{in}x_n \leq -b_i$.

We may also have an equality constraint $a_{i1}x_1 + \ldots + a_{in}x_n = b_i$. We can replace this equality by two inequalities, $a_{i1}x_1 + \ldots + a_{in}x_n \leq b_i$ and $a_{i1}x_1 + \ldots + a_{in}x_n \geq b_i$ (and the latter can be converted to less-than-or-equal form as before).

Another possibility is that we have a variable $x_j$ that can take negative values—that is, there is no nonnegativity constraint. If so, we can introduce two new variables, $x'_j \geq 0$ and $x''_j \geq 0$, and replace $x_j$ with $x'_j - x''_j$ everywhere, which can take any value.

Finally, we may have both variables and constants on both sides of the inequality, that is, inequalities of the form $a_{i1}x_1 + \ldots + a_{in}x_n + b_i \geq a'_{i1}x_1 + \ldots + a'_{in}x_n + b'_i$. We can replace this simply by $(a_{i1} - a'_{i1})x_1 + \ldots + (a_{in} - a'_{in})x_n \geq (b'_i - b_i)$.

As a result, we can write any linear program in the above *standard form*. Sometimes, it is useful to write linear programs in other forms. We can turn any inequality constraint into an equality constraint by introducing a so-called *slack variable*. That is, we can replace the constraint $a_{i1}x_1 + \ldots + a_{in}x_n \leq b_i$ by the constraint $a_{i1}x_1 + \ldots + a_{in}x_n + w_i = b_i$ (where $w_i \geq 0$). The slack variable $w_i$ indicates "by how much" the original constraint $i$ is satisfied.

Throughout, we will sometimes write programs in nonstandard form, with the understanding that the program can easily be converted back to standard form if we so desire.

## 3 Additional terminology

Any assignment of values to the variables is called a *solution*. A solution is *feasible* if it respects all the constraints, and *optimal* if it maximizes the objective among feasible solutions. A program may have no feasible solutions at all, for example, if the constraints $x_1 \le 1$ and $-x_1 \le -2$ are both present. A program with no feasible solutions is called *infeasible*. It may also be the case that for any real value $r$, there exists a feasible solution with objective value at least $r$. For example, it may be the case that $c_1 = 1$ and $a_{i1} = 0$ for all $i$, and there exists a feasible solution. In that case, we can start with the feasible solution and then make $x_1$ as large as we like, thereby making the objective as large as we like. If this is the case, then we say that the program is *unbounded*.

## 4 A quick preview

In what follows, we will study a number of example problems that can be modeled as linear or integer programs. We will then study other properties of linear and integer programs, as well as algorithms for solving them.

As we will see, linear programs are, in various senses, significantly easier to solve than (mixed) integer programs. For example, given any linear program together with an optimal solution to that problem, there is always a succinct proof (or *certificate*) of that solution's optimality. Whether such simple proofs exist for (mixed) integer programs is not known (but if they do, it would imply P=coNP). In fact, linear programs can be solved in polynomial time (although the most common algorithm for solving linear programs, the *simplex algorithm*, is in fact not a polynomial-time algorithm), whereas solving (mixed) integer programs is NP-hard. Because of this, algorithms for solving integer programs typically build on algorithms for solving linear programs, often calling the latter as subroutines.