Lecture notes ?: Adding valid inequalities (cutting planes)

Vincent Conitzer

1 Introduction

In the branch-and-bound algorithm, every time we branch, the program splits into two programs with additional inequalities. We will now consider techniques that do not result in multiple programs, but still add valid additional constraints to the program that help us find the solution.

Let us consider again the integer 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 1 illustrates this integer program.

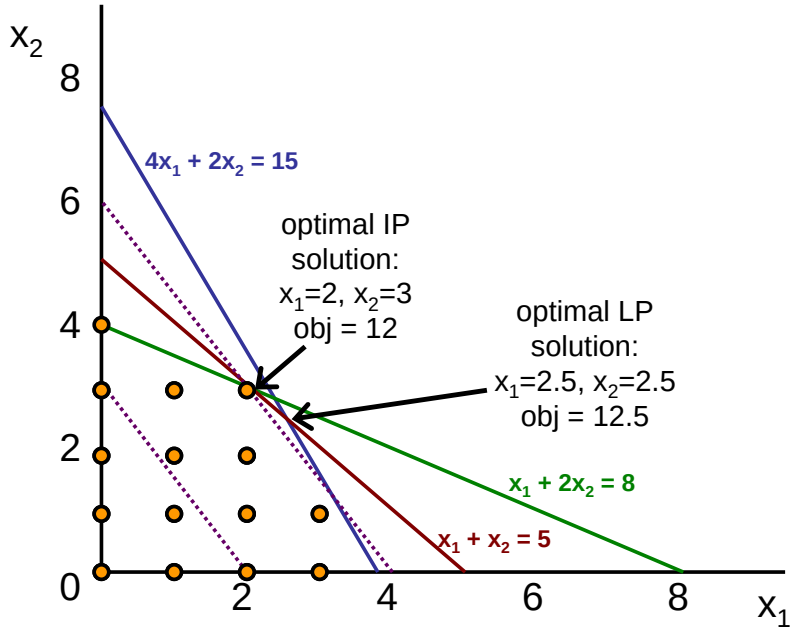

Figure 1: Graphical representation of the modified painting problem instance with integrality constraints.

It is not hard to find additional valid constraints. For example, we can simply add the first two constraints to obtain $5x_1+4x_2 \leq 23$. This is a valid constraint. However, it is not a very helpful one, because adding it does not change the feasible region for the LP relaxation. What we really want to do here is to add linear constraints that cut off part of the feasible region of the LP relaxation, but that do not cut off any *integer* solutions. If we add enough such constraints, then the feasible region for the LP relaxation will be cut down to the convex hull of all the integer feasible points. If we manage to add this many constraints, then solving the LP relaxation will give us an integer optimal solution. In fact, we usually do not need to add this many constraints: we only need to add enough constraints around the optimal points.

If we want to generate a valid linear inequality that is going to cut down the feasible region of the LP relaxation, we must somehow use the integrality constraints to come up with these inequalities. Below, we will see examples of various arguments for getting valid inequalities based on the integrality constraints, illustrated with the above integer program. These inequalities can generally be generated in very systematic ways, but the main purpose of the below is to illustrate the types of argument rather than systematic ways of generating them.

2 A simple rounding argument

Let us consider the constraint $4x_1 + 2x_2 \leq 15$. Dividing by 2, we get $2x_1 + x_2 \leq 7.5$. However, if x_1 and x_2 are set to integer values, then $2x_1 + x_2$ must be an integer as well. Hence, it follows that we must have $2x_1 + x_2 \leq 7$. If we simply add this constraint, then the solution $x_1 = 2, x_2 = 3$ (with objective value 12) becomes optimal for the LP relaxation). (To see why, adding $2x_1 + x_2 \leq 7$ to

the constraint $x_1 + x_2 \leq 5$ gives us $3x_1 + 2x_2 \leq 12$—but $3x_1 + 2x_2$ is the objective.)

Making this argument a little more general, let us consider the case where we have a constraint $a_1x_1 + \ldots + a_nx_n \leq b$, where all the a_j are integers (and all the x_j are required to take integer values). Let g be the greatest common divisor of all the a_j. Then, the following constraint must hold: $(a_1/g)x_1 + \ldots + (a_n/g)x_n \leq \lfloor b/g \rfloor$ (because all the a_j/g are integers). We note that this constraint is parallel to the original constraint, but it is stronger. It turns out that we cannot make it any stronger without cutting some integer points: there is always an integer point $(x_1, \ldots, x_n)$ such that $(a_1/g)x_1 + \ldots + (a_n/g)x_n = \lfloor b/g \rfloor$. (This follows from the fact that the a_j/g are relatively prime.)

In the above example, we could apply this argument directly to one of the existing constraints. In other examples, we first need to take a linear combination of existing constraints for this argument to be effective. For example, consider the following integer program:

maximize $x_1 + x_2$
subject to
$x_1 + 5x_2 \leq 20$
$5x_1 + x_2 \leq 20$
$x_1 \geq 0$, integer; $x_2 \geq 0$, integer

The optimal solution to the LP relaxation is $x_1 = 10/3, x_2 = 10/3$, for an objective value of $20/3$. However, the optimal integer feasible solution is $x_1 = 3, x_2 = 3$, for an objective value of 6. The rounding technique will not produce anything interesting when applied to either of the two constraints individually, because the greatest common divisor of 1 and 5 is 1, and 20 is already an integer. However, if we add the two constraints, we obtain $6x_1 + 6x_2 \leq 40$. Applying the rounding technique to this constraint, we get $x_1 + x_2 \leq \lfloor 40/6 \rfloor = 6$. Because $x_1 + x_2$ is also the objective, when we add this constraint, the optimal integer feasible solution $x_1 = 3, x_2 = 3$ (with objective value 6) becomes optimal for the LP relaxation.

3 Disjunctive constraints

Let us return to the original integer 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

When we discussed branch and bound, we took advantage of the fact that $x_1 \leq 2$ or $x_1 \geq 3$. We will now see how we can get a valid inequality out of this fact as well.

Multiplying the third constraint by 2 we obtain

$$2x_1 + 2x_2 \leq 10$$

which can be rewritten as

$$3x_1 + 2x_2 + (2 - x_1) \leq 12$$

Also, the first constraint can be rewritten as

$$3x_1 + 2x_2 + (x_1 - 3) \leq 12$$

Now, because either $x_1 \leq 2$ or $x_1 \geq 3$, it follows that

$$3x_1 + 2x_2 \leq 12$$

This is because if $x_1 \leq 2$, then

$$3x_1 + 2x_2 \leq 3x_1 + 2x_2 + (2 - x_1) \leq 12$$

On the other hand, if $x_1 \geq 3$, then

$$3x_1 + 2x_2 \leq 3x_1 + 2x_2 + (x_1 - 3) \leq 12$$

Again, adding the constraint $3x_1 + 2x_2 \leq 12$ is enough to make the optimal integer feasible solution optimal for the LP relaxation as well.

More generally, if for some integer k, some j', some $\alpha \geq 0$, and some $\beta \geq 0$, we have the constraints

$$(\sum_j a_j x_j) + \alpha(x_{j'} - (k+1)) \leq b$$

and

$$(\sum_j a_j x_j) + \beta(k - x_{j'}) \leq b$$

and additionally we know that $x_{j'} \leq k$ or $x_{j'} \geq k+1$, then it follows that

$$(\sum_j a_j x_j) \leq b$$

In fact, any valid inequality that is based on the fact that $x_{j'} \leq k$ or $x_{j'} \geq k+1$ can be obtained this way.

4 Modular arithmetic and Gomory cuts

Our final inequalities will be based on an equality constraint rather than an inequality constraint (this is not so restrictive because we can write linear programs in equality form). Suppose we have the equality constraint

$$\sum_j a_j x_j = b$$

Also, let us assume that all the x_j must be nonnegative integers. Given some d, let r_j be the remainder that results from dividing a_j by d, and similarly let s be the remainder that results from dividing b by d. Because the x_j are integers, it follows that

$$\sum_j r_j x_j = s \mod d$$

Then, because $s < d$ and $\sum_j r_j x_j \geq 0$ (because the x_j are nonnegative), this implies that $\sum_j r_j x_j \geq s$.

An interesting special case of this occurs when we set $d = 1$. In this case, the resulting inequality $\sum_j r_j x_j \geq s$ can be written as $\sum_j (a_j - \lfloor a_j \rfloor) x_j \geq b - \lfloor b \rfloor$. This is known as a *Gomory cut*.

To see an example of how to apply a Gomory cut, let us consider the painting example again. Suppose we solve the LP relaxation of this example using the simplex algorithm, thereby obtaining

the optimal dictionary:

maximize $12.5 - 0.5w_1 - w_3$
subject to
$x_1 = 2.5 - 0.5w_1 + w_3$
$w_2 = 0.5 - 0.5w_1 + 3w_3$
$x_2 = 2.5 + 0.5w_1 - 2w_3$
$x_1, x_2, w_1, w_2, w_3 \geq 0$

Of course, x_1 and x_2 are both set to 2.5 in the optimal solution. Let us consider the first inequality constraint, which can be written as follows:

$$x_1 + 0.5w_1 - w_3 = 2.5$$

We obtain the following Gomory cut from this:

$$0.5w_1 \geq 0.5$$

or equivalently

$$w_1 \geq 1$$

We can add this constraint to the program; if we then solve the LP relaxation with this added constraint, we obtain the optimal integer feasible solution.

It should be noted that it is *always* possible to apply a Gomory cut effectively in this way. This is because of the following reasons. If, in the optimal dictionary, a basic variable is set to a noninteger value, then let us get a Gomory cut from this constraint. That basic variable will not occur in the Gomory cut (since its coefficient is 1). Therefore, the Gomory cut will involve only nonbasic variables; moreover, the right-hand side will be positive (because we assumed that the basic variable was set to a noninteger value). Hence, the Gomory cut cuts off the current solution (because all the nonbasic variables are set to zero in the current solution, violating the Gomory cut), and we will make progress.