Lecture notes ?: Constraint and column generation, and the cutting stock problem

Vincent Conitzer

1 Introduction

So far, we have assumed that we can explicitly write down the entire linear program. In many settings, this is not feasible: there may be an exponential number of constraints, or an exponential number of variables. (We will see an example shortly.) Let us first consider the case where we have an exponential number of constraints. Here, one natural approach to take is the following. We start by writing down only some of the constraints of the linear program (perhaps the ones that we think are the most likely to be binding). Let us call the true linear program L, and the linear program with the restricted set of constraints L'. We then find an optimal solution x for L'. There are two possibilities: either x is not feasible for the original problem L, because it violates at least one of the constraints in L that we had not written down in L'; or x is optimal for L. (x cannot be feasible but suboptimal for L, because the optimal value for L can be at most the optimal value for L'.) Now, suppose we have some algorithm (in this context, sometimes called an *oracle*) for L which, when given a solution x, either finds a constraint in L that x violates, or otherwise tells us that there are no violated constraints. We run this oracle on our solution x; if it returns a violated constraint, then we add this constraint to L', and we solve it again. Then, we run the oracle on the new solution to obtain another violated constraint, *etc.*—until the oracle tells us that there are no more violated constraints, in which case we know that our current solution is optimal for L. This approach is called *constraint generation*. Often, we seek to generate not just any constraint that is violated, but rather the constraint that is the most violated (which would appear to be more likely to be binding in the optimal solution to L).

It may seem wasteful to solve L' from scratch every time. Fortunately, we do not need to do this. In particular, when we solve L' we can also easily obtain an optimal solution for the dual of L'. When we add a constraint to L', the current dual solution is still feasible: all that has changed in the dual is that a new variable has been added, which is currently set to 0. So, we can start from the current dual solution to obtain the next optimal dual solution (from which we can then easily read off the next optimal primal solution).

Now, what about a problem with exponentially many variables? Such a problem is the dual of a problem with exponentially many constraints, so if there is a good oracle for constraint generation in the dual, we can use that. This is called *column generation*.

2 The cutting stock problem

We now consider the classic example of a problem in which column generation is helpful, the *cutting stock problem*. In the cutting stock problem, we operate a factory which produces long rolls of paper of a fixed width W. However, we can cut the paper as it comes out, for example producing two rolls of width $W/2$. (We are only allowed to cut in this vertical direction, not in a horizontal or diagonal

direction.) We have certain orders for paper that we need to fill. Each order i has a width w_i and a length l_i. We are allowed to stitch multiple rolls of paper together if they have the same width. For example, if we have a single order of length l and width $W/2$, we can produce one roll of length $l/2$ (and width W), cut it into two rolls of length $l/2$ and width $W/2$, and stitch them together to obtain one roll of length l and width $W/2$. We are not allowed to stitch in the other direction: for example, we cannot combine two rolls of width $W/4$ into a roll of width $W/2$ (if we did, then the paper would look ugly everywhere). Our goal is to fill all of our orders while using the minimum amount of paper. That is, we want the total paper (of width W) that we produce to be as short as possible.

Since we can only cut vertically, we may as well cut the paper as it is coming out of the machine. At any point in time, we are cutting the paper into a certain combination of widths. We call such a combination a *pattern*. For example, if $W = 11$, one pattern is to cut the paper into widths 5, 5, and 1. It may be that we do not have orders of width 1 so that the produced roll of width 1 is simply waste. In this case, we will simply say that the pattern is $\{5, 5\}$ (and it is implicit that the remaining 1 is wasted). So, a number will never occur in a pattern unless we have an order of that width.

To have a concrete example, suppose that $W = 11$, and we have three orders:

- $w_1 = 5$, $l_1 = 20$;
- $w_2 = 4$, $l_2 = 10$;
- $w_3 = 2$, $l_3 = 9$.

An optimal solution here is to use the pattern $\{5, 5\}$ for a length of 5, and the pattern $\{2, 4, 5\}$ for a length of 10. This will give us a roll of width 5 of length $2 \cdot 5 + 10 = 20$, a roll of width 4 of length 10, and a roll of width 2 of length 10—enough to fill all our orders. (We are producing a little too much of width 2, but we can simply throw the excess away.) This gives us an objective of 15 (the total length of width W paper that we produce).

More generally, suppose we have enumerated *all* different patterns, indexed by j. In general, there are many such patterns; some of them are clearly dominated by other patterns (for example, $\{4, 5\}$ is dominated by $\{2, 4, 5\}$, since it does not hurt to produce additional width 2 paper), but even without considering the dominated patterns there are many patterns. For example, undominated patterns for the (tiny) above instance include $\{5, 5\}, \{5, 4, 2\}, \{5, 2, 2, 2\}, \{4, 4, 2\}, \{4, 2, 2, 2\}, \{2, 2, 2, 2, 2\}$. So, in general, we do not want to explicitly list all of the patterns. We temporarily ignore this difficulty and write the linear program as if we can enumerate all the patterns j. Let x_j be the amount of pattern j that we produce, and let a_{ij} be the number of times that the width of order i occurs in pattern j. For example, in the above instance, if pattern 1 is $\{5, 5\}$, then $a_{11} = 2$. We will assume that no two orders have the same width (if they do, we can combine them into a single order). We obtain the following linear program:

minimize $\sum_{j=1}^{n} x_j$
subject to
$(\forall i \in \{1, \ldots, m\})$ $\sum_{j=1}^{n} a_{ij} x_j \geq l_i$
$(\forall j \in \{1, \ldots, n\})$ $x_j \geq 0$

It should be noted that the w_i parameters do not occur in the program. However, they are taken into account implicitly, because they determine what patterns are possible. The dual is:

maximize $\sum_{i=1}^{m} l_i y_i$
subject to
$(\forall j \in \{1, \ldots, n\})\ \sum_{i=1}^{m} a_{ij} y_i \leq 1$
$(\forall i \in \{1, \ldots, m\})\ y_i \geq 0$

We can interpret this dual as follows. Suppose there is an external market at which one unit of paper of width w_i (that is, a roll of width w_i and length 1) sells for y_i. Also suppose that each unit of width W that we produce costs us 1. Then, if the constraints in the dual are met, we should quit our business, because we cannot beat the market prices: no matter what pattern j we use for a unit of our width W paper, the total market price for the pieces of paper that we produce, which is $\sum_i a_{ij} y_i$, is at most 1, which is the cost of our paper. Hence, filling all the orders in the market must be cheaper than filling them all ourselves, and hence $\sum_i l_i y_i$ (the cost of filling all the orders in the market, and also the dual's objective) must be a lower bound on the primal objective. Of course, in reality, there is no external market—this is just a lower bounding argument. Strong duality tells us that there exist feasible prices y_i such that there is a feasible solution x_j to the primal that has the same cost as the (imaginary) market, $\sum_j x_j = \sum_i l_i y_i$.

Let us return to the example instance from above, for which we already know an optimal primal solution $(5 \cdot \{5, 5\} + 10 \cdot \{2, 4, 5\})$, and find an optimal solution to the dual, using complementary slackness. The constraint on order 3 (width 2 paper) is not binding, so we know that the corresponding dual variable y_3 must be 0. Furthermore, indexing pattern $\{5, 5\}$ by 1 and pattern $\{2, 4, 5\}$ by 2, we know that $\sum_i a_{i1} y_i = 2y_1 = 1$ and $\sum_i a_{i2} y_i = y_1 + y_2 + y_3 = 1$. Because $y_3 = 0$, we have $y_1 = y_2 = 0.5$ as an optimal solution to the dual.

Now we return to the issue that in general, there are too many patterns to write down the linear program explicitly. We will use column generation to address this. We start with some arbitrary set of patterns; say pattern 1 is $\{5, 5\}$ and pattern 2 is $\{4, 4, 2\}$ (note that pattern 2 was something else before). We solve the example with this restricted set of patterns. It is not hard to see that the optimal solution to the primal is now $x_1 = 10, x_2 = 9$, for an objective of 19 (which is not as good as the 15 that we can obtain in the unrestricted problem instance). Using complementary slackness, we can obtain $y_2 = 0$ (because we have an excess of width 4 paper), $y_1 = 0.5, y_3 = 1$. Bizarrely, the price for width 5 paper is now lower than that for width 2 paper. This is merely an artifact of the patterns that we started with: given these patterns, width 5 paper is in a sense easier to obtain than width 2 paper.

We now want to add a new pattern. To do so, we find the most violated constraint in the dual (among the set of all constraints in the real problem instance). That is, we want to find a pattern j that maximizes $\sum_i a_{ij} y_i$, a pattern that is most valuable with respect to the current prices. We can phrase this problem in general as the following integer program:

maximize $\sum_{i=1}^{m} y_i a_i$
subject to
$\sum_{i=1}^{m} w_i a_i \leq W$
$(\forall i \in \{1, \ldots, m\})\ a_i \geq 0$, integer

Here, a_i is the number of times that we include width w_i in our pattern. Somewhat confusingly, the a_i are the variables of this problem, whereas the y_i are parameters (defined by the current dual solution). The w_i, which now appear explicitly, are parameters as well. This is a type of KNAPSACK problem (it is slightly unusual because we can use the same width multiple times). While KNAPSACK problems are in general NP-hard, they are among the easiest NP-hard problems and can in practice be solved quite fast. For our particular situation, it is easy to see that the optimal solution is to set $a_1 = 0, a_2 = 0, a_3 = 5$—that is, add the pattern $\{2, 2, 2, 2, 2\}$.

Now, we solve the linear program again with the third pattern added—pattern 1 is $\{5, 5\}$, pattern

2 is $\{4, 4, 2\}$, and pattern 3 is $\{2, 2, 2, 2, 2\}$. The optimal solution becomes $x_1 = 10, x_2 = 5, x_3 = 0.8$, for an objective value of 15.8. This is getting closer to the true optimal value of 15, but we are not quite there yet. Again, we can obtain the current dual solution using complementary slackness. Because all of the primal variables are set to nonzero values, all of the dual constraints must be binding. This results in the following optimal dual prices: $y_1 = 0.5, y_2 = 0.4, y_3 = 0.2$.

Now, again, we solve the KNAPSACK instance to find a pattern that maximizes the total value in terms of the dual prices. Patterns $\{5, 4, 2\}$ and $\{5, 2, 2, 2\}$ both give a total value of 1.1. If we add pattern $\{5, 4, 2\}$, we have both of the patterns that occur in the optimal solution that we identified in the beginning, so that the next time that we solve the linear program, we get a solution with value 15. Of course, at this point, we do not yet know that this is an optimal solution to the unrestricted problem instance. However, when we solve the KNAPSACK instance with the optimal dual prices $y_1 = y_2 = 0.5, y_3 = 0$, we find that every pattern has value at most 1; hence, there are no more violated constraints in the dual, and we have reached optimality.