The L Shaped Algorithm

Jeremy Schiff

August 12, 2018

1 Introduction

Linear programming is one of the most used techniques for making decisions - human beings think in a linear manner and often find ourselves in a situation dominated by tradeoffs. By formulating a series of inequalities and some sort of goal, we are able to find a numerically optimal solution to guide the decision we ultimately make. Such problems are well-studied and can even be solved in a relatively quick "linear" time complexity in fixed dimension. However, this sort of formulation makes a few assumptions that are not in line with reality, in particular they assume that one has perfect knowledge as to future outcomes and can project exactly what would happen with each given decision. To resolve this, we can turn towards stochastic linear programming - a multi-stage process that introduces randomness into the calculus. In this paper, I consider the simplest form of stochastic linear programming: two-stage linear programming, examine the workings of the most popular method of solving such scenarios, and discuss the complexities found in this area. As an appendix, one can also find a MATLAB implementation of the algorithm and a pair of problems with known solutions and cuts so as to verify implementation correctness.

2 Problem Formulation: Two-Stage Linear Programming

Recall that single stage linear programs take the form of $\min_x c^T x$ such that $Ax \leq b$. In contrast, these two stage problems will take the following form: $\min_{x,y} c^T x + \sum_K p_k q_k^T y_k$ such that $Ax \leq b$ and $\forall k \; T_k x + W y_k \leq h_k$. (Almost always one also requires $x, y \geq 0$ in both cases, but the bound of 0 is not inherent to the problem structure, one simply needs a region bounded by below.)

Starting from left to right: the variable c_i represents the costs incurred for each x_i allocated in the eventual solution. This means it takes the form of a vector of length M where M is the number of types of things to initially allocate. As mentioned, x is the vector of length M that we are trying to solve for as it determines the optimal allocation in our program. The vector p is a probability vector that identifies the likelihood of each of the possible outcomes during recourse, inherently summing to a total of 1 with a length of P with P the number of possibilities. Correspondingly to c, the vector q_k determines the cost for each of the allocated y_ks. As a result, q is an L by P matrix where L is the number of different y coordinates necessary in problem allocation. Note that one does not find explicit values for y due to the randomness in the objective function, causing one to only be able to find an expected optimal for y, with actual values determined for each possibility. A and b take the same forms as in the linear case - A is an M by J matrix that determines the coefficients put on our J x coordinates by our M different bounds (that involve solely x_is) while b is a vector of length J. The final three variables are a bit complex to envision, but are the probabilistic analogues to A and b. Let's say that there are R bounds that involve some combination of x_i and y_i. Then, $T \in R^3$ is of dimensions R by M by P representing the coefficients placed on our M x_i variables along the R different bound equations indexed by the probabilistic outcome. Similarly, h is the analogue to b and gives one the constant in the bound equations and thus is of dimension R by P. Finally, W yields the y coefficients in our inequalities and is an L by R matrix. Note that one does not index by P as both the bound and x coefficients index by P, allowing for different ratios along bounds as needed. This problem structure is reasonably flexible, but cannot handle non-linear relationships between variables, more than two-stages (although a fairly intuitive extension exists), nor an infinite number of random outcomes.

3 Brief Description of the L Shaped Method

On a high level, the idea is to convert our program into its determinstic equivalent then perform Bender's decomposition on the resultant program. The problem with doing this approach naively is that it fails to accurately account for the probabilistic component of our program. To resolve this, we will add in a parameter θ to account for the costs of the process. The combination of this with the decomposition gives the algorithm its name as they result in a significant number of constraints that affect only x and θ, making the final constraint matrix have nonzero values in a pattern that looks like an L. Now, for a more detailed view of the steps:

1. Solve the one-stage linear programming problem given by $\min_x c^T x$ such that $Ax \leq b$. Store the optimal x and set $\theta = -\infty$. Proceed to step 2.

2. For each value of k, perform feasibility cuts by solving the program given by $\min_{y,v^+,v^-} v_1^+ + v_1^- + v_2^+ + v_2^- + ...v_i^+ + v_i^- + ...$ such that $Wy + v^+ - v^- \leq h_k - T_k x$ and $v^+, v^- \geq 0$. Note a few things about this - v^+ and v^- are vectors with size equal to the number of mixed x, y constraints and that intuitively, we are trying to minimize the residual that comes from picking the optimal y from a feasibility perspective. If the result of this program has $v^+, v^- = 0$ in each run, go to step 3. Otherwise continue by finding the simplex multiplier σ that results from the dual of the first violating problem (duality is a way of "swapping" constraints and costs of a linear program). We can add a feasibility cut by computing $D = \sigma^T T_k$ and $d = \sigma^T h_k$ with the former being x coefficients and the latter being a constant bound. If we have previously added an optimality cut, go to step 4. Otherwise, go to step 1.

3. Find optimality cuts using a Bender's decomposition approach for each possible outcome. This means that we optimize the linear program $\min q_k^T y$ such that $Wy \leq h_k - T_k x$ for all P possible values of k. In particular, we are interested in the simplex multiplier that results from the dual of this problem and will call that π_k. To check our solution's progress, we calculate the quantities $e = \sum_k p_k \pi_k h_k$ and $E = \sum_k p_k \pi_k T_k$. It is not immediately obvious without looking deeply, but e is a constant and E is one-dimensional and has the same size as x. Our solution x is optimal iff $\theta \geq e - Ex$. If not, we add a coefficient of -1 to the end of E so as to weigh θ then introduce E and e as the coefficients and bound respectively on a new "optimality cut" for our master problem as described in step 4.

4. Solve the one-stage linear programming problem (called the "master problem") given by $\min_{x,\theta} c^T x + \theta$ such that $Ax \leq b$ and that any feasibility and optimality cuts are satisfied. Return to step 2.

The details behind convergence is outside the scope of this paper. The idea behind the proof is to recognize that W has a finite number of bases, x is always cut off when we introduce a new constraint due to duality relationships, and that new constraints are always correct for similar reasons. These three concepts give that only a finite number of cuts are needed to generate what will always be an optimal solution.

4 Simple Example

We take $c = [1], A = [1], b = [1], p = [0.5, 0.5], q = [2, 1], T = [-2, -1], W = [-2], h = [-2, -2]$, $x, y \geq 0$. Written more simply, we have two cases with equal probability:
1. Minimize $x + 2y$ where $0 \leq x \leq 1$, $2x + 2y \geq 2$, $y \geq 0$
2. Minimize $x + y$ where $0 \leq x \leq 1$, $x + 2y \geq 2$, $y \geq 0$

The L-Shaped Method proceeds as follows on the simple problem given in the previous section:

1. (Master Problem) Solve $\min_x x$, $0 \leq x \leq 1$. Result is $x = 0$. Set $\theta = -\infty$

2. (Feasibility) Start with k=1: solve $\min_{v,y} +v^+ + v^-$, $-2y + v^+ + v^- \leq -2 - 2 * 0 \leq -2$ to find $v^+ + v^- = 0$. Proceed with k=2: solve $\min_{v,y} v^+ + v^-$, $-2y + v^+ + v^- \leq -2 - 1 * 0 \leq -2$ to find $v^+ + v^- = 0$. Both possibilities are feasible.

3. (Optimality) Start with k=1: solve $\min_y, 2y$, $-2y \leq -2 - 2 * 0 \leq -2$ to find dual multiplier 1. Proceed with k=2: solve $\min_y y$, $-2y \leq -2 - 2 * 0 \leq -2$ to find dual multiplier 0.5. This yields $e = 0.5 * 1 * -2 + 0.5 * 0.5 * -2 = -1.5$ and $E = 0.5 * 1 * -2 + 0.5 * 0.5 * -1 = -1.25$. As $\theta = -\infty < e - Ex = -1.5$, we instead add $-1.25x - \theta \leq -1.5$ to our constraints.

4. (Master Problem) Solve $\min_{x,\theta} x + \theta$, $0 \leq x \leq 1$, $-1.25x - \theta \leq -1.5$. Result is $(x, \theta) = (1, 0.25)$

5. (Feasibility) Start with k=1: solve $\min_{v,y} +v^+ + v^-$, $-2y + v^+ + v^- \leq -2 - 2 * 1 \leq -4$ to find $v^+ + v^- = 0$. Proceed with k=2: solve $\min_{v,y} v^+ + v^-$, $-2y + v^+ + v^- \leq -2 - 1 * 1 \leq -4$ to find $v^+ + v^- = 0$. Both possibilities are feasible.

6. (Optimality) Start with k=1: solve $\min_y, 2y$, $-2y \leq -2 - 2 * 1 \leq -4$ to find dual multiplier 1. Proceed with k=2: solve $\min_y, y$, $-2y \leq -2 - 2 * 1 \leq -4$ to find dual multiplier 0.5. This yields $e = 0.5 * 1 * -2 + 0.5 * 0.5 * -2 = -1.5$ and $E = 0.5 * 1 * -2 + 0.5 * 0.5 * -1 = -1.25$. As $\theta = .25 \geq e - Ex = -1.5 + 1.25 = -.25$, we can return our value of $x = 1$ as the optimal.

5 Discussion and Comparison

Unfortunately, computational results on the L shaped method are difficult to find which is in many ways characteristic of the literature on stochastic linear programming. While there are many results for linear programming decompositions and methods to exploit programs of a certain form, less easily identifiable information exists on two-stage linear programming results. This actually caused fairly significant issues during implementation as the papers are all written from the framework that the reader has considerable knowledge in numerical analysis and operations research terminology, much of which I was forced to "learn on the fly". In particular, the modified Bender's Decomposition is a very intricate process that is not shown very explicitly in most sources. For the purposes of this paper, however, the lack of literature on the topic prevents me from really comparing the implemented L Shaped Method with other algorithms in an experimental manner. (Indeed, this MATLAB implementation seems to be one of the few that one will be able to come across on Google even for what is supposed to be the most common method.) Furthermore, I have been unable to come across an actual estimate to bound the number of cuts one may have to perform - the best description I came across was in a presentation by Jim Luedtke from University of Wisconsin-Madison that suggests that even traditional Bender's Decomposition can have catastrophically poor performance akin to Simplex methods on normal linear programming, but usually converges quickly. With this in mind, I will attempt to briefly describe relative pros of algorithms that I have come across in my research.

1. With the L shaped method as the most common, by far the second most common algorithm used was to simply sample results through a Monte Carlo approach. This has the benefits of a fixed runtime, easy implementation, and ability to handle more diverse programs. Its clear downside, of course, is that it cannot guarantee an optimal or even "good" solution.

2. Northwestern's research page on the topic suggests that slightly modified versions of the L-Shaped algorithm compose the current state of the art approaches which is backed up in this more recent 2016 paper by Rei et al. on Bender Decompositions. Such algorithms are likely both more numerically stable and faster, but look quite daunting to implement.

3. While I could not find a clear example of this being used practically, there is some discussion as to using convex approximation to convert the expectation of the recourse function into something that can be handled more cleanly by other LP decompositions (ex: Dantzing-Wolfe) or even straightforward Bender Decompositions. My intuition is that these would be somewhere in between the Monte Carlo results and the L Shaped Method results in that they should be faster than the latter and more accurate than the former.

4. Finally, Ronald Hoppe describes an inner linearization method that he suggests is better under certain dimensionality constraints that are unlikely in practical problems. Unfortunately, no implementation was available online, so I was again unable to create an empirical comparison.

6 Conclusion

The L Shaped Method is a very useful algorithm to hold in one's optimization toolbox and represents an under-appreciated area within operations science research. Stochastic programming seems a fairly common extension one may wish to make on linear programs due to the importance of modeling uncertainty. Generally, it seems that the literature and algorithms need to be more accessible if such problems are going to be formulated in normal practice. Future inquiry into the topic may find it fruitful to consider either non-linear boundaries or infinite possible recourse scenarios to see what approaches are possible in what also seem valuable cases. As mentioned above, this report is coupled with a pair of toy problems in "Problems.m" as well as an implementation (to my knowledge the only publicly available MATLAB version) of the L Shaped Method itself in "L_Shaped_Method.m". The algorithm will automatically print out the cuts and guesses it makes so as to allow one to follow along with the steps. Further instructions on its use and syntax can be found within the file itself.