Column generation algorithms

From optimization

Authors: Kedric Daly (Spring 2015)
Stewards: Dajun Yue, Fengqi You

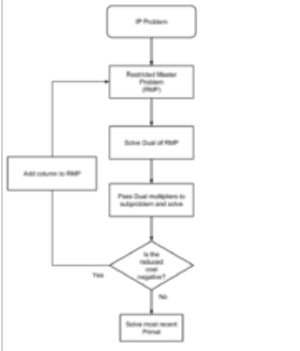
Contents

Introduction

Column generation algorithms are used for MILP problems. The formulation was initially proposed by Ford and Fulkerson in 1958[1]. The main advantage of column generation is that not all possibilities need to be enumerated. Instead, the problem is first formulated as a restricted master problem (RMP). This RMP has as few variables as possible, and new variables are brought into the basis as needed, similar to the simplex method[2]. By similar to the simplex method, it means that if a column with a negative reduced cost can be found, it is added to the RMP and this process is repeated until no more columns can be added to the RMP.

Formulation

The formulation of the column generation problem depends on the type of problem. One common example is the cutting stock problem. However, all cases involve taking the original problem and formulating the RMP as well as a subproblem. The solution of the RMP determines some of the parameters in the subproblem whereas the subproblem will be used to determine if there are any columns which can enter the basis. The subproblem does this by solving for the minimum reduced cost. If the reduced cost is negative, the solution can enter the basis as a new column. If the reduced cost is greater than or equal to zero, the lower bound for the optimal solution has been found, although this may not be an integer solution.



A simple column generation flowchart

Examples

Cutting Stock Problem (CSP)

In the cutting stock problem, the goal is to minimize the waste obtained from cutting rolls of fixed size (called "raws") while fulfilling customer orders.

For example, we may have steel rods of length L = 17m, with customer orders for twenty-five 3m length rods, twenty 5m length rods, and fifteen 9m length rods.
Let l_i be the length a customer demands. Thus,

$$\mathbf{l} = \begin{bmatrix} l_1 = 3m, & l_2 = 6m, & l_3 = 9m \end{bmatrix}^T$$

Let b_i be the demand for each piece of length li . Thus,

$$\mathbf{b} = \begin{bmatrix} b_1 = 25m, & b_2 = 20m, & b_3 = 18m \end{bmatrix}^T$$

Traditional IP formulation

The traditional integer programming formulation for the cutting stock problems involves minimizing the number of rolls that are cut in order to meet demand constraints as well as an overall size constraint.

Let N be the index of available rolls.
Let y_n be 1 if roll n is cut, and 0 otherwise.
Let x_i^n be the number of times item i is cut on roll n .
The IP formulation is then:

$$\begin{aligned} \min \quad & \sum_{n \in N} y_n \\ \text{s.t.} \quad & \sum_{n \in N} x_i^n \geq b_i \\ & \sum_{i=1}^{b} l_i x_i^n \leq L y_n \quad , \quad n \in N \\ & x_i^n \in \mathbb{Z}_+ \qquad\quad , \quad y_n \in \{0, \quad 1\} \end{aligned}$$

However, this formulation is inefficient and is difficult to solve to optimality for large numbers of variables[3]. Column generation algorithms can help solve this problem quickly by limiting the number of enumerations necessary.

Column Generation Formulation

For the column generation formulation, the different patterns the rods can be cut into are the main focus[4].

Let P be the set of all patterns that can be cut.
Let a_{ip} be the number of pieces of length li cut in pattern p.
Let x_p be the number of times pattern p is cut. Then the column generation RMP and dual are:

$$\begin{aligned} \min \quad & Z = \sum_{p \in P} x_p \\ \text{s.t.} \quad & \sum A\mathbf{x} \geq \mathbf{b} \\ & x_i \geq 0 \end{aligned}$$

$$\begin{aligned} \max \quad & Z^{dual} = \sum \mathbf{b}^T \boldsymbol{\pi} \\ \text{s.t.} \quad & A^T \boldsymbol{\pi} \leq \overrightarrow{1} \\ & \pi_i \geq 0 \end{aligned}$$

An initial set of columns must now be selected. This can be done simply by selecting "fake" columns where we know they will not end up in the solution, or by covering the basis. In this example, an identity matrix can be selected. A better basis-covering initial matrix would take $\lfloor L/l_i \rfloor$ because we can always cut at least that many bars from our raw. Thus, our initial A matrix is:

$$A = \begin{bmatrix} 5 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix}$$

Solving the dual of the RMP then yields the dual multiplier $\boldsymbol{\pi} = \begin{bmatrix} \frac{1}{5} & \frac{1}{2} & \frac{1}{2} \end{bmatrix}^T$. These values are then passed to the sub-problem to see if any columns will be added to A. The sub-problem is as follows:

$$\begin{aligned} c_r = \ z^{sub} = \ & \min \left(1 - \sum_{i=1}^{m} \pi_i a_{ip} \right) = \ 1 - \max \left(\sum_{i=1}^{m} \pi_i a_{ip} \right) \\ \text{s.t.} \quad & \mathbf{l}^T \mathbf{a_p} \leq L \\ & a_{ip} \in \mathbb{Z}_+ \qquad \forall i \end{aligned}$$

This sub-problem is a knapsack problem which has been studied extensively. Dynamic programming (e.g. branch-and-bound) can be used to solve this knapsack problem [5]. At the end of this sub-problem, we will compute the reduced cost, c_r , to determine whether or not we add the solution column to A. Similar to the simplex algorithm, if the reduced cost is negative, the column is added to the RMP, otherwise we are done adding columns, and the most recent primal solution will give us our lower bound solution to the RMP. Substituting the dual variables and other known quantities into the sub-problem gives us:

$$\begin{aligned} c_r = \ z^{sub} = \ & 1 - \max \left(\frac{1}{5} a_{1p} + \frac{1}{2} a_{2p} + \frac{1}{2} a_{3p} \right) \\ \text{s.t.} \quad & 3a_{1p} + 6a_{2p} + 7a_{3p} \leq 16 \\ & a_{ip} \in \mathbb{Z}_+ \qquad \forall i \end{aligned}$$

The solution of which gives $\mathbf{a_p} = \begin{bmatrix} 1 & 2 & 0 \end{bmatrix}^T$, with a reduced cost of $c_r = 1 - \left(\frac{1}{5}(1) + \frac{1}{2}(2) + \frac{1}{2}(0) \right) = -\frac{1}{5} \leq 0$. Since this reduced cost is negative, the column, $\mathbf{a_p} = \begin{bmatrix} 1 & 2 & 0 \end{bmatrix}^T$ is added to A in the RMP, and it will replace one of the columns in the basis. After adding the column,

$$A = \begin{bmatrix} 5 & 0 & 0 & 1 \\ 0 & 2 & 0 & 2 \\ 0 & 0 & 2 & 0 \end{bmatrix}$$

Solving the dual of the new RMP then yields the dual multiplier $\boldsymbol{\pi} = \begin{bmatrix} \frac{1}{5} & \frac{2}{5} & \frac{1}{2} \end{bmatrix}^T$. Again, these values are passed to the sub-problem and become the coefficients in the objective function. Solving the second iteration of the sub-problem yields $\mathbf{a_p} = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^T$ with a reduced cost of $c_r = 1 - \left(\frac{1}{5}(1) + \frac{2}{5}(1) + \frac{1}{2}(1) \right) = -\frac{1}{10} \leq 0$. Since this reduced cost is negative, the column $\mathbf{a_p} = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^T$ is added to A and the algorithm continues.

The new dual multipliers become $\boldsymbol{\pi} = \begin{bmatrix} \frac{1}{5} & \frac{2}{5} & \frac{2}{5} \end{bmatrix}^T$, and after substitution into the sub-problem, we find the solution column $\mathbf{a_p} = \begin{bmatrix} 5 & 0 & 0 \end{bmatrix}^T$ has a reduced cost of 0. Since this is not a negative reduced cost, this column is not added to A, and column generation stops. The optimal solution can then be found using the most recent

version of A and simply optimizing the RMP. The resulting solution for the RMP is $\mathbf{x} = \begin{bmatrix} 1\frac{1}{5} & 0 & 0 & 1 & 18 \end{bmatrix}^T$ which gives an objective value of $Z = \sum_{p \in P} x_p = 1\frac{1}{5} + 1 + 18 = 20\frac{1}{5}$.

The result is the lower bound of the integer solution for the CSP, and as in this case, is often not an integer. In the case of the CSP, simply rounding up is often enough to obtain a feasible integer solution, which in this case will be 21 raws in order to fill the orders.

Other Examples

Other applications of column generation include[6]:

- Human resource planning
- Vehicle routing
- Air crew scheduling

All of these applications still follow the basic format of column generation. An RMP is formulated and solved, with parameters being sent to a subproblem. The subproblem is then solved and if the reduced cost of the solution is negative, the column is added to the RMP and the cycle continues until the reduced cost is nonnegative. The formulation of each problem varies due to the different parameters, but the overall approach is the same.

Advantages and Disadvantages

Column generation algorithms are best used when there are a large number of variables, but not a large number of constraints by comparison. Enumerating all possibilities when there are a large number of variables, often due to many indices, takes a long time even with efficient solution methods. Column generation algorithms solve this by limiting what is enumerated, bringing columns into the basis only when needed. When columns are brought into the basis, it is also possible to remove whatever column was replaced by the entering column, which can help save memory while enumerating solutions. Saving time and memory is where column generation algorithms shine, although they are not without their drawbacks.

One of the main disadvantages of column generation is that it may be difficult to determine whether or not a problem can be formulated so that column generation will be beneficial. It is typically easier to come up with a standard MILP model than the column generation equivalent, since the column generation formulations are not always obvious. However, once this initial hurdle is overcome, column generation is a useful tool for solving MILP problems.

Conclusion

Column generation algorithms are most useful when dealing with large numbers of variables. They are effective because they avoid enumerating all possible elements of a traditional MILP formulation, and instead only evaluate variables as needed. This is accomplished by bringing columns into the RMP when the reduced cost is negative. The process repeats until a nonnegative reduced cost is reached, and then the most recent primal can be solved to obtain a bound for the MILP problem. While the initial formulation of MILPs using column generation algorithms may be difficult to see at first, if a formulation can be arrived at, use of a column generation algorithm has many potential time savings.

References

[1] L. R. Ford, Jr., D. R. Fulkerson, (1958) A Suggested Computation for Maximal Multi-Commodity Network Flows. Management Science 5(1):97-101. http://dx.doi.org/10.1287/mnsc.5.1.97

[2] Desrosiers, J., & Lübbecke, M. (2005). A Primer in Column Generation. In G. Desaulniers, J. Desrosiers & M. Solomon (Eds.), Column Generation (pp. 1-32): Springer US.

[3] (Nov 2012) Lecture 8: Column Generation [PDF document] Retrieved from http://ocw.nctu.edu.tw/upload/classbfs121109080773803.pdf

[4] Stein, C. (2007) Column Generation: Cutting Stock - A Very Applied Method [PDF Document]. Retrieved from http://www.columbia.edu/~cs2035/courses/ieor4600.S07/columngeneration.pdf

[5] Column Generation [PDF document]. Retrieved from http://systemsbiology.ucsd.edu/sites/default/files/Attachments/Images/classes/convex_presentations/ColGen.pdf

[6] Gan, H. (2008) Column Generation [PDF document]. Retrieved from http://www.more.ms.unimelb.edu.au/students/operationsresearch/lecturenotes/620362_ColGen.pdf

[7] Giovanni Righini. (April 2013) Column Generation [PDF document]. Retrieved from http://homes.di.unimi.it/righini/Didattica/ComplementiRicercaOperativa/MaterialeCRO/CG.pdf