Column Generation for Linear and Integer Programming

George L. Nemhauser

2010 Mathematics Subject Classification: 90
Keywords and Phrases: Column generation, decomposition, linear programming, integer programming, set partitioning, branch-and-price

1 The beginning – linear programming

Column generation refers to linear programming (LP) algorithms designed to solve problems in which there are a huge number of variables compared to the number of constraints and the simplex algorithm step of determining whether the current basic solution is optimal or finding a variable to enter the basis is done by solving an optimization problem rather than by enumeration.

To the best of my knowledge, the idea of using column generation to solve linear programs was first proposed by Ford and Fulkerson [16]. However, I couldn't find the term *column generation* in that paper or the subsequent two seminal papers by Dantzig and Wolfe [8] and Gilmore and Gomory [17,18]. The first use of the term that I could find was in [3], a paper with the title "A column generation algorithm for a ship scheduling problem".

Ford and Fulkerson [16] gave a formulation for a *multicommodity maximum flow* problem in which the variables represented path flows for each commodity. The commodities represent distinct origin-destination pairs and integrality of the flows is not required. This formulation needs a number of variables exponential in the size of the underlying network since the number of paths in a graph is exponential in the size of the network. What motivated them to propose this formulation? A more natural and smaller formulation in terms of the number of constraints plus the numbers of variables is easily obtained by using arc variables rather than path variables. Ford and Fulkerson observed that even with an exponential number of variables in the path formulation, the minimum reduced cost for each commodity could be calculated by solving a shortest path problem, which was already known to be an easy problem. Moreover the number of constraints in the path formulation is the number of arcs, while in the arc formulation it is roughly the (number of nodes)x(number of commodities) + number of arcs. Therefore the size of the basis in the path formulation is independent of the number of commodities and is significantly smaller when the number of commodities is large. This advantage in size they claimed might make it possible to solve instances with a large number of commodities with the simplex method. Modestly, they stated that they really had no idea whether the method would be practical since they had only solved a few small instances by hand.

It must have been so frustrating to try to do algorithmic research when it was so difficult to test if your ideas could yield practical algorithms. The value of some of these brilliant ideas proposed in the infancy of mathematical programming would not be proven for decades. Much of this early work was done at the RAND Corporation with its 'all star' team of applied mathematicians including Bellman (dynamic programming), Ford and Fulkerson (network flows), Dantzig (linear programming) and many others. As a sports fan, this reminds me of the great baseball teams of the New York Yankees, basketball teams of the Boston Celtics and soccer teams of Manchester United.

I was Ray Fulkerson's colleague at Cornell in the 1970s. I have no memory of him giving an opinion of the significance of the arc-path formulation of the multicommodity flow problem. Even if he thought this was a fundamental contribution, his modesty would have prevented him from saying so. However I think that this early work influenced his later contributions on blocking and anti-blocking pairs of polyhedra [15], which studies polyhedra associated with combinatorial optimization problems that frequently have an exponential number of variables and provided a basic theory of integral polyhedra.

Another way to derive Ford and Fulkerson's path formulation is to begin with the arc formulation and note that the arc capacity constraints link all of the variables while the flow balance constraints can be separated by commodity. For each commodity the extreme points of the flow balance constraints are the origin-destination simple paths for that commodity. Feasible solutions to the whole problem are convex combinations of these extreme flows that satisfy the arc capacity constraints. So if we begin with a so-called master LP that just contains a few of these extreme flows for each commodity and solve it to optimality, we can use an optimal dual solution to price out the extreme flows not yet considered by solving a shortest path problem for each commodity. This is precisely what Ford and Fulkerson proposed simply beginning with the path formulation.

This idea can be generalized to yield an algorithm for solving any LP by partitioning the constraints into a set of master constraints and a set of subproblem constraints. The resulting algorithm is what we call *Dantzig–Wolfe decomposition* [8]. I think it is rather odd that George Dantzig did not get his name attached to the simplex method but to this very important contribution still of surely lessor stature. Dantzig and Wolfe say:

> Credit is due to Ford and Fulkerson for their proposal for solving multicommodity network problems as it served to inspire the present development.

However the contribution of Dantzig–Wolfe decomposition is very significant in its own right since it does not depend on beginning with the exponential formulation. It could arise from an appropriate partition of the constraints into a small number that involved all of the variables and the rest that could be decomposed into individual subproblems involving only a relatively small subset of the variables. Think, for example, of a multiperiod problem with a few budget constraints involving variables from all of the periods and subproblems for each period, or a resource allocation problem involving a few constraints coupling all of the variables globally together with subproblems for each region. For these structures, and other similar ones, using Dantzig–Wolfe decomposition, a large LP can be decomposed into a master problem with a small number of constraints and an exponential number of variables corresponding to the extreme points of the subproblems, the solution of which represents convex combinations of these extreme points that satisfy the master constraints. Optimal dual solutions of the master problem provide prices to the subproblems, whose solutions yield new extreme point variables for the master.

2 Next steps – integer subproblems

The previous work relied only on LP. The multicommodity flow problem requires the generation of integer vectors that are incidence vectors of paths, but they can be found without the explicit imposition of integrality constraints.

The first column generation work that involved integer variables appears to have been done by Gilmore and Gomory [17]. They studied the *cutting stock* problem: given a positive integer number $d(i)$ of items of integer size $a(i)$, determine the minimum number of stock rolls of integer size b needed to pack all of the items. Gilmore and Gomory proposed a model in which there is an integer variable corresponding to every possible way to cut a roll. Since a solution to the cutting of a single roll is a solution of an *integer knapsack* problem (a single constraint integer program (IP)), which can have an exponential number of solutions, this model contains an exponential number of variables. However, when the LP relaxation of the model is solved over a subset of variables, optimality can be proved or new columns can be added to improve the solution by solving an integer knapsack problem with objective function specified by the dual variables in an optimal LP solution and constraint specified by the item and role sizes. The knapsack problem can be solved reasonably efficiently by dynamic programming or branch-and-bound even though it is NP-hard. The application of this work described in [18] appears to be the first use of column generation in a practical problem. Gilmore and Gomory's work on the cutting stock problem led to their work on the knapsack problem [19], and motivated Gomory's work on the group problem [20], which has had a significant impact on the field of integer programming.

Gilmore and Gomory only use the LP relaxation of their formulation of the cutting stock problem. They simply propose to round up the variables in an optimal LP solution to obtain a feasible solution to the IP. But this heuristic can be justified by the fact that, in general, the optimal LP solution value provides a very tight bound on the optimal number of rolls. In fact, it has been shown empirically in [29] that for a very large number of randomly generated instances the difference is always less than one. Carefully contrived instances with a difference greater than one are known [25, 30], but it is not known whether a difference of two or larger can be obtained. Although rounding up a fractional solution can increase the objective function by the number of items (number of basic variables), it has been observed in [4] that the increase is no more than 4 % of the number of items.

The whole point of this discussion is to emphasize that the Gilmore–Gomory formulation of the cutting stock problem provides a very tight relaxation. This is typically the case for such formulations leading to a tradeoff between a tight bound from an exponential formulation that can be challenging to solve and a compact (polynomial size) formulation with a much weaker bound. Although not stated by Gilmore and Gomory, and then lost in translation when the cutting stock problem is presented in basic operations research textbooks, there is a straightforward compact formulation of the cutting stock problem. Begin with an upper bound on the number of rolls required and a binary variable for each roll that is equal to one if the roll is used and zero otherwise. There are identical knapsack constraints for each potential roll with right-hand side b if its binary variable equals one, and zero otherwise and additional constraints requiring that the amount $d(i)$ of the ith item must be cut. The LP relaxation of this formulation is terrible. It gives no information since it is easy to show that the bound is the total amount to be cut divided by b. Furthermore if this LP relaxation is used in a branch-and-bound algorithm, the performance is terrible not only because of the weak bound, but also because of the symmetry of the formulation since all rolls are the same. In fact, a compact formulation similar to the one above was given by Kantorovich [23] who introduced the cutting stock problem in 1939!

The Gilmore–Gomory formulation applied to the *bin packing* specialization of the cutting stock problem in which $d(i) = 1$ for all i yields a set partitioning problem: given a ground set S and a set of subsets $S(j)$, $j = 1, \ldots, n$, find a minimum cardinality set of disjoint subsets whose union is S. In the bin packing problem S is the set of items and $S(j)$ is a subset that fits into a bin. $|S| = m$ is typically small, but n is exponential in m. This form of set partitioning and set covering (disjointness is not required) models arises in many combinatorial optimization problems. For example, in *node coloring* S is the set of nodes and $S(j)$ is a subset of nodes that is a stable set (a set of nodes that can receive the same color since no pair of them is joined by an edge). Thus column generation for the LP relaxation of the node coloring set partitioning formulation involves solving a minimum weight stable set problem, where the node weights correspond to the dual variables in an optimal LP solution. Note that the column generation formulation eliminates the symmetry possessed by a compact formulation in which there is a variable for each node-color pair. The absence of symmetry is a very important property of the exponential formulation since symmetry is a major nemesis of branch-and-bound algorithms.

These models appear in many practical applications as well. Perhaps the one that has received the most attention in the literature is *airline crew scheduling* [6, 21], but there are many other applications to all kinds of transportation routing problems, scheduling problems, districting problems, coloring problems, etc. In the crew scheduling problem S is a set of flights that need to be flown over a given time horizon, say a day or a week, and $S(j)$ is a subset of flights that can be flown by a single crew. The cost of using the subset $S(j)$ is $c(j)$. This cost function complicates the model introduced for bin packing and graph coloring since the objective function of total minimum cost is no longer a minimum cardinality objective function and a set of allowable flights is subject to complex rules concerning safety and other factors. Nevertheless, feasible subsets, which are called *pairings*, can be generated as constrained paths in a network and minimum cost constrained shortest paths for column generation can be generated as well.

The first published paper that appears to discuss such a model in detail is [5]. It reports on crew scheduling methods used by airlines in the 1960s, several of whom were already using a set partitioning model. Some were trying to solve the IP by optimization algorithms using branch-and-bound or cutting planes. They recognized that the algorithms could only deal with a small number of pairings. So pairings were generated up front and then a subset was heuristically chosen to include in the IP model. A significant improvement to the approach of a single round of pairing generation followed by a single round of optimization was proposed in [27]. Given a feasible solution, a better solution might be found by a neighborhood search that selects a small subset of flights, generates all of the pairings that only cover these flights and then solves a set partitioning problem defined by these flights and pairings. If an improvement is found, this solution replaces the current pairings that cover these flights. The neighborhood search can be iterated until no improvements are found. This quasi-column generation process was used by many airlines throughout the 1980s and even later [1]. Nevertheless it could only achieve a local optimum, and although the solution quality might be good, optimality could not be claimed. Other approaches solved the full LP relaxation by some form of column generation, but only provided a subset of columns to the IP solver. Even without an exponential number of columns these IP can be difficult to solve. Standard branching on binary variables is not very effective since the branch with the binary variable at zero hardly restricts the problem.

A branching rule proposed in [28], unrelated to column generation at the time, called *follow-on branching*, helped to alleviate this difficulty. In a simplified version of the rule, two adjacent arcs in the flight network associated with a fractional pairing are identified and then, on one branch, pairings that contain both of these flights are excluded, and on the other branch, pairings that contain one of them are excluded. It can be shown that such a pair of arcs exists in a fractional solution, and the fractional solution is excluded on both branches. This rule divides the solution space much more evenly than variable branching. As we shall see, generalizations of this rule are very useful when column generation is incorporated in a branch-and-bound search.

3 Branch-and-price: solving integer programs by column generation

If a tree search (branch-and-bound) algorithm for an IP with an implicit exponential number of variables is designed to produce an optimal solution or even one with a prescribed optimality tolerance, it is necessary to do column generation throughout the tree. To the best of our knowledge, the first appearance in the literature of column generation within branch-and-bound is in [13].

There are interesting challenges in applying column generation to problems associated with nodes within the search tree. Foremost is that standard branching on variables, besides being inefficient, can complicate column generation. Consider a set partitioning problem where we branch on a single binary variable corresponding to some subset. The branch where the variable is fixed to one does not create a problem since we now have a smaller set partitioning problem. But in the branch where the variable is set to zero we need to impose on the column generation solver a constraint saying that this subset is not feasible. Such constraints will significantly hamper the efficiency of the column generator.

However, a generalized version of the follow-on branching idea for crew scheduling makes it possible to preserve the efficiency of the column generation solver and also reasonably balances the solutions between the two newly created nodes. Consider a fractional column (subset) in an optimal solution of the LP relaxation. It can be shown that there are two elements in the column such that there is another fractional column containing only one of these elements. On one branch we exclude columns containing only one of these elements and on the other branch we exclude columns containing both. Not allowing only one of the elements to appear, i.e., both must appear together, amounts to combining the elements, while not allowing both to appear together involves adding a simple constraint. For example, in a node coloring problem where the elements are nodes and a feasible subset is a stable set, both appearing together is accomplished by replacing the two nodes by a super node with an edge from the super node to all other nodes that were connected to one or both or the original nodes, and not allowed to appear together is accomplished by adding an edge between the two nodes. We can think of this type of branching as branching on the variables from the original compact formulation instead of branching on the variables in the exponential set partitioning formulation. For example in the node coloring problem, the branching is on node variables. On one branch we require two nodes to have the same color and on the other the two nodes must get different colors. Early use of this branching rule are given in [10] for urban transit crew scheduling, [14] for vehicle routing, [2] for airline crew scheduling, [31] for bin packing, [11] for a survey of routing and scheduling applications and [26] for node coloring. Vanderbeck and Wolsey [34] studies column generation branching with general integer variables.

Barnhart et al. [7] unified this early literature by presenting a general methodology for column generation in IP and named the general technique *branch-and-price*. Vanderbeck [32] presents a general treatise on branching in column generation and gives some interesting new branching ideas in [33]. In the last decade there have been many successful applications of branch-and-price algorithms to practical problems and a completely different use in choosing neighborhoods for local search algorithms [22]. More information about column generation and branch-and-price algorithms can be found in Desrosiers and Lübbecke [12], who present a primer on column generation, in a chapter of a collection of articles on column generation [9], and Lübbecke and Desrosiers [24], who present a survey of techniques and applications of column generation in IP.

References

[1] R. Anbil, E. Gelman, B. Patty and R. Tanga (1991). Recent advances in crew pairing optimization at American Airlines. *Interfaces* 21, 62–74.

[2] R. Anbil, R. Tanga and E.L. Johnson (1992). A global optimization approach to crew scheduling. *IBM Systems Journal* 31, 71–78.

[3] L.E. Appelgren (1969). A column generation algorithm for a ship scheduling problem. *Transportation Science* 3, 53–68.

[4] D.L. Applegate, L.S. Buriol, B.L. Dillard, D.S. Johnson and P.W. Shor (2003). The cutting-stock approach to bin packing: theory and experiments. In *Proceedings of the Fifth Workshop on Algorithm Engineering and Experiments*, R.E. Ladner ed. SIAM, 2–15.

[5] J.P. Arabeyre, J. Fearnley, F.C. Steiger and W. Teather (1969). The airline crew scheduling problem: A survey. *Transportation Science* 3, 140–163.

[6] C. Barnhart, A. Cohn, E.L. Johnson, D. Klabjan. G.L. Nemhauser, and P.Vance (2002). Airline crew scheduling. In *Handbook in Transportation Science*, R.W. Hall ed. Kluwer, 517–560.

[7] C. Barnhart, E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh and P.H. Vance (1998). Branch-and-price: column generations for solving huge integer programs. *Operations Research* 46, 316–329.

[8] G.B. Dantzig and P. Wolfe (1960). Decomposition principle for linear programs. *Operations Research* 8, 101–111.

[9] G. Desaulniers, J. Desrosiers, and M. Solomon (2005). *Column Generation*, Springer.

[10] M. Desrochers and F. Soumis (1989). A column generation approach to the urban transportation problem. *Transportation Science* 23, 1–13.

[11] J. Desrosiers, Y. Dumas, M.M. Solomon and F. Soumis (1995). Time constrained routing and scheduling. In *Handbooks in Operations Research and Management Science 8, Network Routing*, M.E. Ball, T.L. Magnanti, C. Monma and G.L. Nemhauser eds. Elsevier, 35–140.

[12] J. Desrosiers and M.E. Lübbecke (2005). A primer in column generation. In *Column Generation*, G. Desaulniers, J. Desrosiers, and M. Solomon eds. Springer, 1–32.

[13] J. Desrosiers, F Soumis and M. Desrochers (1984). Routing with time windows by column generation. *Networks* 14, 545–565.

[14] Y. Dumas, J. Desrosiers and F. Soumis (1991). The pickup and delivery problem with time windows. *European Journal of Operations Research* 54, 7–22.

[15] L.R. Ford and D.R. Fulkerson (1958). A suggested computation for maximal multicommodity network flows. *Management Science* 5, 97–101.

[16] D.R. Fulkerson (1971). Blocking and anti-blocking pairs of polyhedra. *Mathematical Programming* 1, 168–194.

[17] P.C. Gilmore and R.E. Gomory (1961). A linear programming approach to the cutting-stock problem. *Operations Research* 9, 849–859.

[18] P.C. Gilmore and R.E. Gomory (1963). A linear programming approach to the cutting stock problem—Part II. *Operations Research* 11, 863–888.

[19] P.C. Gilmore and R.E. Gomory (1966). The theory and computation of knapsack functions. *Operations Research* 14, 1045–1074.

[20] R.E. Gomory (1965). On the relation between integer and non-integer solutions to linear programs. *Proceedings of the National Academy of Science* 53, 260–265.

[21] B. Gopalakrishnan and E.L. Johnson (2005). Airline crew scheduling: state-of-the-art. *Annals of Operations Research* 140, 305–337.

[22] M. Hewitt, G.L. Nemhauser and M.W.P. Savelsbergh (2012). Branch-and-price guided search for integer programs with an application to the multicommodity fixed charge network flow problem. To appear in *INFORMS Journal on Computing.*

[23] L.V. Kantorovich (1960). Mathematical methods of organizing and planning production. *Management Science* 6, 366–422. Translated from the Russian original 1939.

[24] M.E. Lübbecke and J. Desrosiers (2005). Selected topics in column generation. *Operations Research* 53, 1007–1023.

[25] O. Marcotte (1986). An instance of the cutting stock problem for which the rounding property does not hold. *Operations Research Letters* 4, 239–243.

[26] A. Mehrotra and M.A. Trick (1996). A column generation approach for exact graph coloring. *INFORMS Journal on Computing* 8, 344–354.

[27] J. Rubin (1973). A technique for the solution of massive set covering problems with application to airline crew scheduling. *Transportation Science* 7, 34–48.

[28] D.M. Ryan and B. Foster (1981). An integer programming approach to scheduling. In *Computer Scheduling of Public Transport Urban Passenger Vehicle and Crew Scheduling*, A. Wren ed. North-Holland, 269–280.

[29] G. Scheithauer and J. Terno (1995). The modified integer round-up property of the one-dimensional cutting stock problem. *European Journal of Operational Research* 84, 562–571.

[30] G. Scheithauer and J. Terno (1997). Theoretical investigations on the modified integer round-up property for the one-dimensional cutting stock problem. *Operations Research Letters* 20, 93–100.

[31] P.H. Vance, C. Barnhart, E.L. Johnson and G.L Nemhauser (1994). Solving binary cutting stock problems by column generation and branch-and-bound. *Computational Optimization and Applications* 3, 111–130.

[32] F. Vanderbeck (2000). On Dantzig–Wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm. *Operations Research* 48, 111–128.

[33] F. Vanderbeck (2011). Branching in branch-and-price: a generic scheme. *Mathematical Programming* 130, 249–294.

[34] F. Vanderbeck and L.A. Wolsey (1996). An exact algorithm for IP column generation. *Operations Research* Letters 19, 151–159.

George L. Nemhauser
Georgia Institute
of Technology
Atlanta GA, USA
george.nemhauser@isye.gatech.edu