1999年度日本オペレーションズ・リサーチ学会
秋季研究発表会

Using ILOG Solver for column generation: A successful marriage of Constraint Programming and Linear Programming

03000500 アイログ(株) Amal de Silva

Introduction

In recent years Column Generation (Barnhart et al. 1996) has gained popularity as a technique to solve complex optimization problems in transportation. Column generation is used quite commonly for crew scheduling in the transportation industry (Airplanes, Buses, and Trains). In these problems there are number of activities that start and end at a fixed time. For Aircrew scheduling, an activity represents a flight leg, while for bus scheduling, an activity represents a bus trip. The objective is to assign these activities to crews such that each crew member carries out a number of activities in sequence in manner that it satisfies all the work rules (for example the union rules) and the total cost is minimized. For example, in Aircrew scheduling the objective is to determine an optimal set of pairings. A pairing is a sequence of flight legs that start at a base and ends at a base (See Desrochers and Soumis (1988) and Vance et al. (1997) for details). Similarly, in Bus Crew scheduling, the objective is to develop duties that are a sequence of bus trips that start from a depot and end at a depot. Since we are optimizing bus duties, the terms used will be for Bus crew scheduling. However, the concepts can be applied to airline crew pairings as well.

Column generation is carried out in two phases. In Phase 1 the columns are generated. Each column will represent a legal duty. The constraints that apply to individual duties are considered. For example, there may be a constraint that a person can work only for 8 hours in a duty. In this case it is possible consider an individual duty and figure out that if it breaks this 8 hour rule or not. It is not necessary to consider the complete solution to determine the legality of an individual duty.

In Phase 2, all the columns are brought together to find a feasible and optimal solution to the problem. In phase 1, duties are generated ignoring the requirement that each bus trip has to be covered by only one duty. Thus, this constraint has to be enforced in phase 2. This usually done by using the set partitioning formulation as:

$$\min \quad cx$$

$$s.t$$

$$A\underset{-}{x} = 1$$

$$\underset{-}{x} \text{ is binary}$$

A is a matrix where each row represents a piece of work (bus trip, flight leg). Each column (each element of vector $\underset{-}{x}$) represents a feasible duty. Column j has $a_{i,j} = 1$ if duty j goes through trip i and 0 otherwise. Also other constraints that apply to the whole solution have to be enforced in this phase. For example, a constraint might require that the total number of duties that start from a given depot have to be less than a given value. This constraint applies to the complete solution and it is not possible to consider an individual duty and conclude whether a duty satisfies the constraint or not. In this case, the constraint has to expressed as a Linear Programming constraint.

One major advantage of this model is that complex constraints that occur in real world crew scheduling problems can be accommodated easily. Columns in crew-scheduling systems for the transportation industry have complex non-linear constraints that cannot be easily included in an LP/MIP formulation. In a column generation framework it is easier to consider them as they have be enforced in the generation phase. That is, only columns that satisfy all the constraints (that apply to individual duties) are generated.

The major drawback of this model is that the number of feasible duties run into millions or billions and it is not possible to generate all possible columns. Column generation handles this by implicitly considering all the columns. As x is binary, the formulation given above need to be solved using the branch and bound algorithm. However, let us first consider the solution of the relaxed Linear Program. The basic column generation algorithm to solve the relaxed LP is as follows:

1. Generate a reasonable number of columns.
2. Solve the relaxed LP
3. Generate columns that price out, using dual values from the solution of relaxed LP. That is, for minimization problems, columns that have negative reduced cost. The reduced cost of a column is given by :

   $$c_j - \underset{-}{\pi} \cdot \underset{-j}{a}$$

   Where, c_j is the cost of duty j, while $\underset{-}{\pi}$ is the row vector of dual variables of the optimal dual solution in of current relaxed solution and $\underset{-j}{a}$ is column associated with duty j. If there are no columns that price out (has negative reduced cost) then the current solution is optimal. Thus, stop the algorithm.
4. Go to step 2.

Theoretically, it is possible to add only one column in step 3, but in practice, it may be better to add as many columns as possible.

In this paper we will discuss how ILOG Solver can be used to increase the efficiency of the generation of columns.

Using ILOG Solver for the Generation of columns

A great deal of research has been carried out in solving large set

partitioning Linear Programs (Phase 2)
(Hoffman and Padberg (1993)). However, less attention has been paid to the generation of columns (Phase 1), although from the author's experience this can constitute up to 80% of the computation time. The reason for this is possibly because Phase 1 has to deal with complex real world constraints that cannot be easily expressed as LP constraints and these constraints differ from one application to another.

Currently there are two techniques to carry out the generation of columns:

(1) Simple backtracking method. In this method, duties are generated by adding trips and then checking for legality. If it the duty becomes illegal after adding a new trip then this trip is removed from the duty and another trip is added. If the duty can still provide a legal duty, this trip is kept in the duty and a another trip is added. If by adding a trip the duty becomes legal, then the duty is stored and the algorithm continues. This algorithm will provide all the legal duties as it does an exhaustive search. Also in the column generation scheme, if the reduced cost of final column is not negative then it is thrown away. However, since it backtracks each time a trip added to duty triggers a failure, it is generally inefficient. As the complete enumeration of all legal duties would be time consuming, various heuristics have been developed to manage this phase.
(2) Multilabel Shortest Path algorithm: In this method a network is build such that a node represents a trip. If its possible for a duty have trip B after trip A then an arc between the nodes for Trip A and B is added. Thus, a path from a starting Node (a trip that can start a duty) and an ending node (A trip that can end a duty) will be a duty. Labels are kept in this network to store the state of the duty with respect to various constraints. The reader is referred to (Vance et al. (1997)) for more details of the algorithm. This algorithm is complex and it generally has to be used in conjunction with a backtracking algorithm. It is also difficult, if not impossible to implement complex constraints with this technique with out backtracking.

In this project, ILOG Solver was used for the column generation. ILOG Solver is a C++ constraint-programming library. It provides the primitives for declaring decision variables, stating constraints and solving the resulting problem. ILOG Solver's design is based on the following principles

- Problem models are separated from the search algorithms
- All constraints collaborate together to compute solutions
- The user can change the search strategy to generate the targeted solutions

A powerful feature of ILOG Solver is domain reduction through propagation. For each decision variable in ILOG Solver the domain of the feasible values are stored. ILOG Solver reduces the domains on-the fly and thus rapidly reduces the combinatorial explosion and avoids significant computational loads. When you reduce the domain of a decision variable, you refine the information known about this decision variable. ILOG Solver uses this information to deduce new information about the remaining decision variables. This is called propagation.

The advantage of using constraint programming for column generation is propagation. As trips are added to a duty, that is, as elements of the duty are instantiated, the domain of the other elements of the duty is reduced. For example, if the first element of the duty is instantiated to Trip A and Trip A can only connect to Trip B then the domain of the second element is reduced to Trip B. This effect will ripple through. Thus, the propagation will reduce the amount of backtracking required and can make the search algorithm significantly more efficient.

The other advantage of ILOG Solver is that it is very easy to add constraints to the algorithm. Hard coding the constraints can be time consuming and thus not suitable for developing systems in an industrial setting.

In this project, an expression for the reduced cost of a column, using the dual variables from the relaxed LP solved previously, was built and then a constraint was posted that required that all columns generated should have negative reduced cost. Thus, this ensures that the columns generated always had a negative reduced cost.

As the optimal of the relaxed LP can be fractional (that is, the value of X can be fractional), it was necessary to carry out a branch and bound to obtain the optimal integer solution. As the standard branching technique is inefficient in this framework, the branching rule developed by Ryan and Foster (see Vance et 1997, for more details) was used here. In this method, the branching was carried out by selecting two trips on fractional column and then requiring that the two trips should always be together in one branch. And in the other branch the restriction is that these two trips cannot be together. This type of constraints can be easily posted in ILOG Solver.

Conclusion

From this project, it is apparent that the combination of constraint programming and linear programming provide a powerful tool for rapidly building complex crew scheduling systems. In the development of software for the industry, the developers generally face tight deadlines. In such situations, tools like ILOG Solver and CPLEX provide the ability to rapidly prototype and develop applications.

References

Barnhart, C., Johnson, E.L., Nemhauser, G.L, Savelsberg, M,W.P and P.H. Vance(1996) "Branch and Price: Column Generation for Solving Huge Linear Programs", Operations Research, to Appear.

Desrocher, M., and F Soumis (1989), " A Column Generation Approach to the Urban Transit Crew Scheduling Problem", Transportation Science 23, 1-13.

Hoffman, K.L., and M. Padberg (1993), " Solving Airline Crew Scheduling Problems by Branch and Cut", Management Science, 39, 657-682.

ILOG Inc, "ILOG Solver white paper" (1997).

Vance, P.H., Atamturk, A., Barnhart, C., Gelman, E., Johnson, E.L., Krishna, A., Mahidhara, D., Nemhauser, G.L., Rebello, R., (1997) " A Heuristic Branch and Price Approach for Airline Crew Pairing Problem", Technical Report LEC 97-06, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, Georgia.