This paper is the second integer programming related paper by John E. Mitchell I reviewed. This paper gives more advances methods to solve integer programming problem. While in the first paper “Cutting Plane Algorithms For Integer Programming”, Mitchell mainly talked about Gomory and Chvatal-Gomory cut, in this paper he mixed the cutting plane methods with branch-and-bound algorithm to further speed-up the convergence rate to the optimal solution.

Paper Summary

In this paper Mitchell used the same example problem with his first paper I reviewed, so I was quite familiar with the problem already. The branch-and-cut algorithm was basically combination of cutting plane and branch and bound methods. The sequence of the branch-and-cut algorithm was solving the linear programming relaxation of the problem, deciding either branching or cutting was going to be utilized if necessary, solving the new linear programming relaxation based on the previous step decision, checking the new solution, making the solution as a new incumbent solution if it is an integer and better than any other solutions, fathoming the solution if it is integer but not better than the current incumbent solution, fathoming the solution even if it is not an integer but it is worse than the incumbent solution, otherwise repeating the step two of the sequence.

According to Mitchell, the secret of branch-and-cut algorithm successes is preprocessing to eliminate unnecessary constraints, determine any fixed variables, and simplify the problem. The preprocessing is mainly related to the step five through eight in the sequence. Both Chvatal-Gomory and Gomory methods can be used in branch-and-cut algorithm. Mitchell acknowledge that H. P. Crowder, E. L. Johnson, and M. Padberg developed the breakthrough of the branch-and-cut algorithm by using extensive and efficient preprocessing, good primal heuristics, and cutting planes derived from knapsack problems with binary variables. Primal heuristic is used to convert the fraction solution to a relaxation into a good integral solution. The valid inequality is derived from the facial structure of the knapsack polytope. Therefore, for example if R ⊆ N with

$$\sum_{i \in R} a_i > b$$

then,

$$\sum_{i \in R} x_i \cdot |R| - 1$$
is valid inequality. By defining R as a minimal such set that deleting any member of R leaves the sum of coefficients smaller than b, the inequality defines a facet of the corresponding knapsack polytope.

Other useful inequalities are lift-and-project introduced by Balas. In this inequalities, given the feasible region for a binary programming problem \( S = \{ x : Ax \leq b, x_i = 0, 1 \forall i \} \), each variable can be used to generate a set of disjunctive inequalities.

Mitchell also talks about cut and branch algorithm, which requires cutting only at the root node of the tree. This algorithm requires far greater time to solve than solving the relaxation at the root. According to Mitchell, the advantage of this algorithm however, includes:

- All generated cuts are valid throughout the tree, since the are valid at the root
- Bookkeeping is reduced, since the relaxation is identical at each node
- No need to generate cutting planes at other nodes

Mitchell suggests using lifting cut approaches to deal with potential memory requirement problem. Since a cut is added at one node of the branch-and-cut tree may not be valid for another sub problem, then he suggest to make a global cut throughout the tree by lifting the cut if necessary. Lifting the cut can also be regarded as rotating the constraint. Basically this method suggests putting a coefficient in front of a cut, and incorporates the cut in the original constraints. Then one can adjust the value of the coefficient for any sub problem.

**Conclusion**

Mitchell’s paper talks quite thoroughly about branch-and-cut algorithm such as what kind of cuts are suitable for algorithm, the negative and positive advantage of cut, the new improvement of the algorithm, and the potential problem of the algorithm. This paper is suitable only for a reader who has a fair knowledge about cutting plane methods, since Mitchell does not provide thorough explanation about detail in his paper. However, in my opinion, this paper is easier to read than Mitchell’s paper about cutting planes that I reviewed earlier. Even though Mitchell still uses many references in this paper, they are not too much to confuse the readers. It helps that Mitchell uses the same example with his earlier paper to visualize the problems, so readers who have read the earlier paper (like me) can follow the example quite easily. Fortunately, Mitchell organizes and expresses his idea quite well in this paper. Comparing with Mitchell’s earlier paper “Cutting Planes Algorithm For Solving Integer Programming”, this paper is a big improvement.