Optimization engineering techniques for the exact solution of NP-hard combinatorial optimization problems


In optimization engineering, the engineer might have different approaches to solve a problem and he/she asks himself/herself, “Which is the best exact approach for finding the optimal solution?” The best results are often obtained by using hybrid algorithms, i.e., by combining different approaches because the most effective algorithm to be used for finding the optimal solution of a given problem strongly depends on the specific instance to be solved. In this paper, the design of effective exact enumeration algorithms for finding the optimal solution of a given NP-hard combinatorial optimization problem is considered using combining the commonly used approaches, i.e., dynamic programming, branch-and-bound, branch-and-cut. To illustrate the above-mentioned approach, the 0-1 knapsack problem and asymmetric traveling salesman problem are considered. The mathematical models of these problems are given by an integer linear program. I think this is a good paper to see the importance of combining different approaches to improve the solution methodology to an optimization problem.

The proposed method for the 0-1 knapsack problem (KP) is to combine dynamic programming with tight upper bounds, obtaining a new algorithm. Tight upper bounds are derived by imposing cardinality constraints. These additional constraints on the maximum and minimum cardinality of an optimal solution are generated from extended covers and are relaxed with the original capacity constraint leading to a new 0-1 knapsack problem. The new problem tends to be much easier to solve than the original one, since the LP upper bounds for this problem are generally tight. An optimal solution to the transformed problem yields an upper bound for the original problem, but generally it also produces a feasible solution to the original problem, thus solving the original KP to optimality. The enumeration part of this algorithm is based on the dynamic programming recursion, but the initial problem is chosen as a collection of items which fit together well with respect to some heuristic algorithms. Moreover, when the number of states in the dynamic programming gets too high, the lower bound is improved by pairing states with items not in the algorithm. This usually results in a tightening of the lower bound and thus in additional fathoming of states. The average performances of the hybrid, the branch-and-bound and dynamic programming approaches have been experimentally evaluated by using five classes of randomly generated data instances. The hybrid approach is clearly superior to all the branch-and-bound and dynamic programming approaches, being able to solve all the instances in less than 0.1 seconds. The computation results are listed as a table in the paper.
The method for the asymmetric traveling salesman problem (ATSP) presents a different approach, a hybrid algorithm, to combine different algorithms. The exact algorithms to solve ATSP are based on branch-and-bound and branch-and-cut algorithms. In the branch-and-bound algorithm, several lower bounding procedures for ATSP are designed, by considering different substructures of the problem, each associated with a subset of constraints defining a well-structured relaxation. The solution of this relaxation gives a valid lower bound for ATSP. In the branch-and-cut algorithms, at each node of the branch-decision tree a lower bound is obtained by solving an LP relaxation of the corresponding ATSP, containing only a subset of the constraints. In order to obtain an algorithm, which is able to produce good results for all the instances, a hybrid algorithm, which selects the type of approach to be used for each specific instance according to its characteristics, has been proposed. In particular, both lower bounds using these two algorithms are computed at the root node of the branch-decision-tree: if the percentage difference between the two bounds is "small" (i.e., less than 1%), then branch-and-bound algorithm is executed, otherwise branch-and-cut algorithm is chosen. The results show that the hybrid approach is clearly superior to the branch-and-bound and branch-and-cut algorithms, being able to solve all the considered instances in less than 1000 seconds. These results are available as a list in the paper.

In conclusion, we can see in above-mentioned problems how combining different approaches in integer programming can enhance the efficiency of the solution. Therefore, this paper is a good example of finding various algorithms to combine different methods in order to improve the result and the computation complexity of a certain optimization problem. It also displays that a hybrid approach can have very different characteristics from one problem to the other. The 0-1 knapsack problem and asymmetric traveling salesman problem uses different hybrid algorithms. Therefore, the reader should note that the best approach to be used for finding the optimal solution of a problem strongly depends on the specific problem, and often on the specific instance, to be solved. If our goal is to design an exact enumerative algorithm for a given NP-hard combinatorial optimization problem, which is not only based on mathematical and algorithmic tools, but it is also effective, i.e., it has relatively short average computing times, and is stable with respect to the different instances of the problem, then the best results are often obtained by using hybrid algorithms, i.e., by combining different approaches, according to what can be called an optimization engineering technique.