Optimizing Set-Covering Models when Column Generation requires Knapsack Based Subproblems with Unbounded Capacities

Daniel Porumbel[1] and Gilles Goncalves[1]

[1] Univ Lille Nord de France, F 59000 Lille, France; UArtois, LGI2A, F 62400, Béthune, France
{daniel.porumbel,gilles.goncalves}@univ-artois.fr

Numerous combinatorial optimization problems can be expressed in a set-covering model. For instance, the well-known *cutting-stock* problem is often solved using the famous Gilmore-Gomory model; very strong lower bounds can be obtained by optimizing the resulting linear program with column generation methods. The sub-problem is the knapsack problem with profits defined by the dual values. Similar sub-problems can arise in many other capacitated versions of various problems, *e.g.,* in logistics, one finds Capacitated Arc Routing, Capacitated Vehicle Routing, or *Capacitated p-Median*. The latter one is defined as follows: given a graph with n nodes, select $p < n$ nodes as facilities such that certain capacities (per facility) are respected; each facility serves a number of nodes and the objective is to minimize the sum of the distances from facilities to nodes. The subproblem in the associated set-covering model requires solving n knapsack problems problems [1]. Typically, such problems often use capacities that are polynomially bounded with respect to n; this makes the sub-problem solvable via dynamic programming in polynomial time.

We investigate the more difficult case of unbounded capacities. Such situations can easily arise in real-life applications that deal with continuous quantities of material; this leads to fractional weights, equivalent to considering unbounded capacities. If the sub-problem is a pure knapsack problem, certain state-of-the art knapsack algorithms (*e.g.* Combo[1]) can often still cope with the problem in reasonable time. However, slightly modified knapsack problems (many "capacitated" subproblems in logistics problems above) might take prohibitive time when the capacities are unbounded. We here focus on a version in which the capacity constraint can be (slightly) violated if a certain penalty is accepted. Solving a *single knapsack-based sub-problem can take prohibitive time*; in this case, classical column generation can require prohibitive time as well.

In column generation, many studies are concerned with stabilisation routines [2], *i.e.,* they speed-up the convergence by reducing the number of iterations (sub-problems to solve). We propose a different method in which the number of sub-problems is not necessarily reduced, but the sub-problems are simplified by manipulating the dual values (knapsack profits). Using ideas from the well-known ϵ-approximation algorithm for the knapsack problem [3], we observe that an unbounded knapsack-based sub-problem can be solved in pseudo-linear time if the profits are integer: one can use a dynamic programming algorithm indexed by profits. The classical column generation algorithm can not be applied—the profits (dual values) can be far from integer (or scalable to integers). The proposed method optimizes by moving on integer rays (defined with integer values) inside the polytope. For instance, it is usually possible to compute in polynomial time the intersection between the dual polytope boundary and the ray starting from 0 in the direction $[1\ 1\ \ldots 1]_n$ (regardless of the capacity). The same holds for any integer direction; using several rays leads to getting valid lower bounds. A valid upper bound can be derived from the facets (dual constraints) discovered when computing the intersection between rays and the boundary. We will test these ideas on various problem instances where knapsack constraints arise.

References

[1] L.A.N. Lorena and E.L.F. Senne A column generation approach to capacitated p-median problems *Computers & Operations Research*, 31(6):863–876, 2004.

[2] F. Clautiaux, C. Alves, J.V. de Carvalho, and J. Rietz New Stabilization Procedures for the Cutting Stock Problem Informs Journal on Computing DOI 10.1287/ijoc.1100.0415

[3] V. Vijay. Approximation Algorithms Springer-Verlag, 2003.

[1] www.diku.dk/~pisinger/codes.html