Methods and Models for Combinatorial Optimization

Cover inequalities

L. De Giovanni M. Di Summa

We introduce a class of inequalities, known as *cover inequalities*, that can be used to improve the formulation of the knapsack problem. We will then see how these inequalities can be applied to more general integer linear programming problems.

1 Cover inequalities for the knapsack problem

In the *knapsack problem* we are given n items of weight $a_1, \dots, a_n$ and profit $p_1, \dots, p_n$ respectively, and a bag (the knapsack) of maximum capacity β. We have to determine a subset of the n items that can be put in the bag without exceeding the maximum capacity, maximizing the total profit. In the following we assume that $a_1, \dots, a_n$ and β are integer numbers.[1] If we define binary variables $x_1, \dots, x_n$, where $x_i = 1$ if and only if item i is selected, we can formulate the knapsack problem as the following integer linear programming problem:

$$\max \sum_{i=1}^{n} p_i x_i \tag{1}$$

$$\text{s.t.} \sum_{i=1}^{n} a_i x_i \leq \beta, \tag{2}$$

$$0 \leq x_i \leq 1, \quad i = 1, \dots, n, \tag{3}$$

$$x_i \in \mathbb{Z}, \quad i = 1, \dots, n. \tag{4}$$

The feasible region of the continuous relaxation, given by constraints (2)–(3), is usually much larger than the ideal formulation. It is then reasonable to introduce inequalities to strengthen the formulation: these inequalities must be satisfied by all the integer solutions, but should hopefully yield a continuous relaxation closer to the ideal formulation.

[1] We recall that from the perspective of implementation we always work with rational numbers. Starting from rational numbers $a_1, \dots, a_n$ and β, we can obtain integer numbers by multiplying them by their least common denominator. The resulting problem is equivalent to the original one, as we are only changing the unit of measurement of weight.

1.1 Cover inequalities

We call *cover* any subset of the n items that exceeds the knapsack capacity: in other words, a cover is a subset $C \subseteq \{1, \ldots, n\}$ such that

$$\sum_{i \in C} a_i > \beta.$$

Since $a_1, \ldots, a_n$ and β are all integer numbers, the above condition can be restated as follows:

$$\sum_{i \in C} a_i \geq \beta + 1.$$

Given a cover C, since it is impossible to put all the elements of C in the knapsack, at most $|C| - 1$ of them can be selected. This implies that the following inequality is satisfied by all integer solutions:

$$\sum_{i \in C} x_i \leq |C| - 1.$$

This inequality is called *cover inequality*.

If all possible cover inequalities are added to the original formulation of the knapsack problem, we obtain a better formulation. (However, it can be shown that this is not the ideal formulation yet.) Unfortunately, the number of cover inequalities is exponentially large, as in the worst case there is one cover inequality for every non-empty subset C of $\{1, \ldots, n\}$. Even though in most practical cases only some of the subsets of $\{1, \ldots, n\}$ are actually covers, the number of covers is too large. The idea is then to add dynamically the cover inequalities only when they are really needed.

1.2 Separation of cover inequalities

Suppose that we have solved the continuous relaxation (1)–(3) (for instance with the simplex method), thus obtaining a solution $\bar{x}$ that is not an integer vector. We want to verify if there is a cover inequality violated by $\bar{x}$, with the purpose of adding this inequality to the formulation and solving the new linear programming problem. The problem of finding a cover inequality violated by $\bar{x}$, or deciding that none exists, is the *separation problem* for the cover inequalities. Note that since the number of cover inequalities is exponentially large, we cannot simply enumerate them and look for a violated one.

Deciding if there is a cover inequality violated by $\bar{x}$ means deciding if there exists a subset $C \subseteq \{1, \ldots, n\}$ such that

(i) $\sum_{i \in C} a_i \geq \beta + 1$;

(ii) $\sum_{i \in C} \bar{x}_i > |C| - 1$.

Condition (i) ensures that C is a cover, while condition (ii) says that $\bar{x}$ does not satisfy the corresponding cover inequality.

To model the above problem, we introduce binary variables $z_1, \dots, z_n$, where $z_i = 1$ if and only if item i is in the cover C. Condition (i) can be rewritten as follows:

$$\sum_{i=1}^{n} a_i z_i \geq \beta + 1.$$

Since $\sum_{i=1}^{n} z_i = |C|$, condition (ii) can be rewritten as follows:

$$\sum_{i=1}^{n} \bar{x}_i z_i > \sum_{i=1}^{n} z_i - 1, \qquad \text{i.e.,} \qquad \sum_{i=1}^{n} (1 - \bar{x}_i) z_i < 1.$$

Consider the following integer linear programming problem:

$$\max \ \sum_{i=1}^{n} (1 - \bar{x}_i) z_i \tag{5}$$

$$\text{s.t.} \ \sum_{i=1}^{n} a_i z_i \geq \beta + 1, \tag{6}$$

$$0 \leq z_i \leq 1, \quad i = 1, \dots, n, \tag{7}$$

$$z_i \in \mathbb{Z}, \quad i = 1, \dots, n. \tag{8}$$

(Note that the $\bar{x}_i$'s are known values and not variables.) The constraints ensure that the z_i variables define a cover. Let $\bar{z}$ be the optimal (integer) solution of this problem. If the optimal value is smaller than 1, then $\sum_{i=1}^{n} (1 - \bar{x}_i)\bar{z}_i < 1$, which, as seen above, means that the cover inequality defined by $\bar{z}$ is violated by $\bar{x}$. Otherwise, if the optimal value is ≥ 1, then there is no cover inequality violated by $\bar{x}$, as in this case $\sum_{i=1}^{n} (1 - \bar{x}_i) z_i \geq 1$ for any vector z defining a cover.

We can then decide if there is a cover inequality violated by $\bar{x}$ (and find it) by solving the integer linear programming problem (5)–(8). Note that this problem is very similar to the original knapsack problem (1)–(4), and therefore it would probably be better to solve directly the original problem rather than solving a similar problem just to find a new inequality to include in the formulation (and then iterate this procedure!). Nonetheless, we will see below that this approach is much more promising in more general situations.

2 Cover inequalities for general binary problems

Consider a general integer linear programming problem in which all variables are binary:

$$\max \ c^T x \tag{9}$$

$$\text{s.t.} \ Ax \leq b, \tag{10}$$

$$0 \leq x_i \leq 1, \quad i = 1, \dots, n, \tag{11}$$

$$x_i \in \mathbb{Z}, \quad i = 1, \dots, n. \tag{12}$$

A crucial observation is that every single constraint of the system $Ax \leq b$ can be seen as a knapsack-type constraint of the form (2). In other words, if we replace the system $Ax \leq b$ with any of its constraints, and remove all the others, we obtain a relaxation of the problem that looks exactly like a knapsack problem. It is then possible to add to the formulation the cover inequalities associated with each of the knapsack problems obtained this way (i.e., removing all constraints but one). The number of possible cover inequalities will be huge, but we can employ the approach described above to add the inequalities only when they are really needed.

Suppose that we have solved the continuous relaxation (9)–(11), thus obtaining a non-integer solution $\bar{x}$. For each knapsack problem obtained by removing all the constraints of the system $Ax \leq b$ except one, we ask ourselves whether there is a cover inequality violated by $\bar{x}$. To this purpose, it is sufficient to solve a problem of the form (5)–(8). If the optimal value of this problem is smaller than 1, then we have found a cover inequality violated by $\bar{x}$ that can be added to the formulation; otherwise there is no cover inequality violated by $\bar{x}$. We can then solve the new continuous relaxation (including the cover inequalities that have been added) and iterate this procedure until the current solution satisfies all cover inequalities. The resulting algorithm is the following:

Generation of cover inequalities

1. Solve the continuous relaxation (9)-(11) and let $\bar{x}$ be the optimal solution obtained.
2. If $\bar{x}$ is integer, then STOP: optimal integer solution.
3. For each constraint of the system $Ax \leq b$, solve the corresponding integer linear programming problem (5)-(8) (where $a_1, \ldots, a_n$ and β are the coefficients and the right-hand side of the constraint). Let $\bar{z}$ be the optimal integer solution obtained.
4. If the optimal value of problem (5)-(8) is smaller than 1, then $\bar{z}$ gives a cover inequality violated by $\bar{x}$, defined by $C = \{i : \bar{z}_i = 1\}$.
5. If for all the problems (5)-(8) solved at the previous step the optimal value is ≥ 1, then there is no cover inequality violated by $\bar{x}$: STOP.
6. Add to the formulation all the cover inequalities found above, solve the new linear programming problem, let $\bar{x}$ be its optimal solution and go to step 2.

With the above algorithm we have to solve many problems of the form (5)–(8), which are integer linear programming problems and therefore hard in general. However, problem (5)-(8) is one of the simplest integer linear programming problems and therefore, although

in principle an exponential time might be needed to solve it, in practice an optimal solution can be found in a reasonable amount of time.

As seen above, the algorithm terminates when the current solution does not violate any cover inequality.[2] Of course it is possible to stop the algorithm before its natural termination if we think that the inequalities that we have added are sufficient to give a good formulation of the problem. In both cases, if $\bar{x}$ is not integer, we can apply the Branch-and-Bound method. The fact that some cover inequalities have been added allows us to start from a better formulation and usually makes the Branch-and-Bound method terminate in a shorter time. This kind of approach, in which the formulation is strengthened with valid inequalities and then Branch-and-Bound is applied, is called *cut-and-branch*. If valid inequalities are added at every node of the Branch-and-Bound tree, then the method is called *branch-and-cut*.

[2]Note however that when this happens, $\bar{x}$ might still be non-integer, because the cover inequalities are not sufficient to describe the ideal formulation of the problem (which is not known).