Application Example of the Branch and Bound Algorithm

Consider the Capital Budgeting problem (which, as we already know, is structurally identical to the Knapsack problem, so we can use either name). Its formulation is this:

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

Subject to

$$\begin{aligned} \sum_{i=1}^{n} c_i x_i &\leq C \\ x_i &\in \{0,1\}, \qquad i = 1, \ldots, n \end{aligned}$$

Let us try to find the exact solution with the Branch and Bound (B&B) algorithm. In order to do it, we need to handle the following issues.

1. Transforming the formulation

In B&B we considered the optimization task

$$\max_{\mathbf{x} \in B} f(\mathbf{x}).$$

This does not have any constraint and seeks for the optimum over *all* binary vectors of dimension n. In contrast, our problem does have a constraint and seeks for the optimum only over those binary vectors that satisfy the constraint.

To handle this difference, we need to transform our constrained task into an unconstrained one. It can be done by the method that is often called *penalty function method*. It is based on the idea that we can substitute a constraint by modifying the objective function, such that we "penalize" those vectors that violate the constraint. In this way, we can do the search over all vectors (with the modified objective function), because when a global, unconstrained optimum is found, the "penalty" does not allow it to be taken at a vector that violates the constraint.

Specifically, in our case, we can do the following. The original objective function was

$$f(\mathbf{x}) = \sum_{i=1}^{n} p_i x_i.$$

Let the modified objective function be

$$\widetilde{f}(\mathbf{x}) = \begin{cases} \sum_{i=1}^{n} p_i x_i & \text{if} \quad \sum_{i=1}^{n} c_i x_i \leq C \\ \\ 0 & \text{if} \quad \sum_{i=1}^{n} c_i x_i > C. \end{cases}$$

We can observe that whenever a vector $\mathbf{x}$ satisfies the constraint, $\widetilde{f}(\mathbf{x}) = f(\mathbf{x})$ holds. On the other hand, if $\mathbf{x}$ violates the constraint, then $\widetilde{f}(\mathbf{x}) = 0$. Since the maximum we are looking for is positive[1], therefore, this positive maximum of $\widetilde{f}(\mathbf{x})$ cannot be taken on a vector that violates the constraint, as $\widetilde{f}(\mathbf{x}) = 0$ for those vectors, by definition. Thus, if we solve

$$\max_{\mathbf{x} \in B} \widetilde{f}(\mathbf{x})$$

by B&B, then the optimum provides us with the optimum of the Knapsack problem.

2. Finding a good upper bound function

In B&B we use an upper bound function $U_k(\mathbf{b})$ that provides an upper bound on the partial optimum $F_k(\mathbf{b})$ (see the Lecture Note about B&B).

What will be $F_k(\mathbf{b})$ is our case? If we fix the first k variables at values $b_1, \ldots, b_k \in \{0, 1\}$, then we get the following new Knapsack problem:

$$\max \sum_{i=k+1}^{n} p_i x_i + \sum_{i=1}^{k} p_i b_i$$

Subject to

$$\begin{aligned} \sum_{i=k+1}^{n} c_i x_i &\leq C - \sum_{i=1}^{k} c_i b_i \\ x_i &\in \{0, 1\}, \qquad i = k+1, \ldots, n \end{aligned}$$

$F_k(\mathbf{b})$ is the optimum of this task. How do we get the upper bound function $U_k(\mathbf{b})$? We can take the LP relaxation of this ILP, replacing the constraint $x_i \in \{0, 1\}$ by $0 \leq x_i \leq 1$. Thus, $U_k(\mathbf{b})$ is the optimum solution of the LP

[1] Assuming each item alone fits in the knapsack, since we can a priori exclude those that do not.

$$U_k(\mathbf{b}) = \max \sum_{i=k+1}^{n} p_i x_i + \sum_{i=1}^{k} p_i b_i$$

Subject to

$$\begin{aligned} \sum_{i=k+1}^{n} c_i x_i &\leq C - \sum_{i=1}^{k} c_i b_i \\ 0 \leq x_i &\leq 1, \qquad i = k+1, \ldots, n \end{aligned}$$

3. Fast computation of the upper bound function

The whole B&B approach makes sense only if we can compute the upper bound function significantly faster than the original task. In our case we face the problem: how do we quickly solve the above LP that defines $U_k(\mathbf{b})$? Fortunately, this LP has a special structure: it is the continuous version of the Knapsack problem. By continuous we mean that $x_i \in \{0, 1\}$ has been replaced by $0 \leq x_i \leq 1$. For such a continuous knapsack it is known that the optimum can be found by a fast greedy algorithm, as follows.

Order the variables according to decreasing p_i/c_i ratio. Let us call it preference ordering. The most preferred variable is the one with the highest p_i/c_i ratio. Consider the variables starting by the most preferred one, and assign them the value 1, as long as it is possible without violating the budget constraint. (Note that in the considered subproblem the budget is $C - \sum_{i=1}^{k} c_i b_i$. If it happens to be negative, then there is no solution to the subproblem, and we can represent it by $U_k(\mathbf{b}) = -1$, so that this branch of the search tree will surely be cut down.) When we cannot continue assigning 1 anymore without violating the budget, then assign a fractional value to the next variable, so that the remaining part of the budget, if any, is exactly used up. The remaining variables, if any, take 0 as their value. One can prove that this value assignment gives the optimum solution to the *continuous* Knapsack problem. Therefore, we have a fast way of computing $U_k(\mathbf{b})$.

Thus, we have all the ingredients to apply B&B. We aim at computing

$$\max_{\mathbf{x} \in B} \widetilde{f}(\mathbf{x})$$

and whenever the algorithm calls for $U_k(\mathbf{b})$, we can compute it by the fast algorithm described above.