13 Branch and Bound

Lectures 13 through 15 will be concerned with three conceptually different approaches for optimization problems that are computationally hard: an exact method, which finds optimal solutions but has an exponential worst-case running time; heuristic methods, which need not offer guarantees regarding running time or solution quality, but often provide a good tradeoff between the two in practice; and approximation algorithms, which run in polynomial time and return solutions with a guaranteed bound on the degree of suboptimality.

Branch and bound is a general method for solving optimization problems, especially in the context of non-convex and combinatorial optimization. Suppose for concreteness that we want to

$$\begin{aligned} &\text{minimize} && f(x) \\ &\text{subject to} && x \in X \end{aligned}$$

for some feasible region X. Branch and bound uses an algorithmic technique known as *divide and conquer*, which splits a problem into smaller and smaller subproblems until they become easy to solve. For the above minimization problem, it works by splitting X into $k \geqslant 2$ sets $X_1, \ldots, X_k$ such that $\bigcup_{i=1,\ldots,k} X_i = X$. This step is called *branching*, since its recursive application defines a tree structure, the so-called *search tree*, whose vertices are the subsets of X. Once optimal solutions have been found for the subsets $X_1, \ldots, X_k$, it is easy to obtain a solution for X, because $\min_{x \in X} f(x) = \min_{i=1,\ldots,k} \min_{x \in X_i} f(x)$.

Of course, branching as such doesn't make the problem any easier to solve, and for an NP-hard problem there might be no way around exploring an exponential number of vertices of the search tree. In practice we might hope, however, that we will be able to *prune* large parts of the tree that cannot contain an optimal solution. The procedure that allows us to do this is known as *bounding*. It tries to find lower and upper bounds on the optimal solution, i.e., functions ℓ and u such that for all $X' \subseteq X$, $\ell(X') \leqslant \min_{x \in X'} f(x) \leqslant u(X')$. Then, if $\ell(Y) \geqslant u(Z)$ for two sets $Y, Z \subseteq X$, Y can be discarded. A particular situations where this happens is when Y does not contain any feasible solutions, and we assume that $\ell(Y) = \infty$ by convention in this case.

For the upper bound, it suffices to store the value $U = f(x)$ of the best feasible solution $x \in X$ found so far. A good way to obtain a lower bound for a set $Y \subseteq X$ is by letting $\ell(Y) = \min_{x \in Y'} f(x)$ for some set $Y' \supseteq Y$ for which minimization of f is computationally tractable. It is easy to see that this indeed provides a lower bound. Moreover, if minimization over Y' yields a solution $x \in Y$, then this solution is optimal for Y. The branch and bound method stores U and a list L of active sets $Y \subseteq X$ for which no optimal solution has been found so far, corresponding to vertices in the search tree that still need to be explored, along with their associated lower bounds. It then proceeds as follows:

1. Initialize: set $U = \infty$, $L = \{X\}$.
2. Branch: pick a set $Y \in L$, remove it from L, and split it into $k \geqslant 2$ sets $Y_1, \ldots, Y_k$.
3. Bound: for $i = 1, \ldots, k$, compute $\ell(Y_i)$. If this yields $x \in X$ such that $\ell(Y_i) = f(x) < U$, then set U to $f(x)$. If $\ell(Y_i) < U$, but no $x \in X$ as above is found, then add Y_i to L.
4. If $L = \emptyset$, then stop. The optimum objective value is U. Otherwise go to Step 2.

To apply the method in a concrete setting, we need to specify how a set Y to branch on is chosen and how it is split into smaller sets, and how lower bounds are computed. These decisions are of course critical for the practical performance of the procedure.

13.1 Dakin's Method

Dakin's method applies the branch and bound idea to integer programs. An obvious way to obtain lower bounds in this case is by solving the LP relaxation, i.e., the linear program obtained by dropping the integrality constraints.

Assume that we are branching on a set $Y \in L$, and that the LP relaxation corresponding to Y has optimal solution y. If $y \in Y$, then y is optimal for Y. Otherwise, there is some i such that y_i is not integral, and we can split Y into two sets $Y^1 = \{x \in Y : x_i \leqslant \lfloor y_i \rfloor\}$ and $Y^2 = \{x \in Y : x_i \geqslant \lceil y_i \rceil\}$. Note that $Y^1 \cup Y^2 = Y$, as desired. Moreover, this branching rule forces the solution away from its current value $y \notin Y$. While this does not guarantee that y_i becomes integral in the next step, and may even force another variable away from its integral value, it works remarkably well in practice. It is worth noting that we do not have to start from scratch when solving the LP relaxation for Y_i: it was obtained by adding a constraint to an LP that is already solved, and the dual simplex method often finds a solution satisfying the additional constraint very quickly. In order to minimize the number of solved LPs that have to be stored to implement this approach, it makes sense to branch on a set obtained in the previous step whenever possible, i.e., to traverse the search tree in a depth-first manner.

Example 13.1. Assume that we want to

$$\begin{aligned} \text{minimize} \quad & x_1 - 2x_2 \\ \text{subject to} \quad & -4x_1 + 6x_2 \leqslant 9 \\ & x_1 + x_2 \leqslant 4 \\ & x_1 \geqslant 0, x_2 \geqslant 0 \\ & x_1, x_2 \in \mathbb{Z}. \end{aligned}$$

An illustration is shown in Figure 13.1. Let $f(x) = x_1 - 2x_2$, and $X = \mathbb{Z}^2 \cap \tilde{X}$ where

$$\tilde{X} = \{ x \in \mathbb{R}^2 : -4x_1 + 6x_2 \leqslant 9, x_1 + x_2 \leqslant 4, x_1 \geqslant 0, x_2 \geqslant 0 \}.$$

We start with $U = \infty$ and $L = \{X\}$. By solving the LP relaxation for X, we find that $\ell(X) = \min_{x \in \tilde{X}} f(x) = f(x^0) = -7/2$ for $x^0 = (3/2, 5/2)$. Set X is the only candidate for

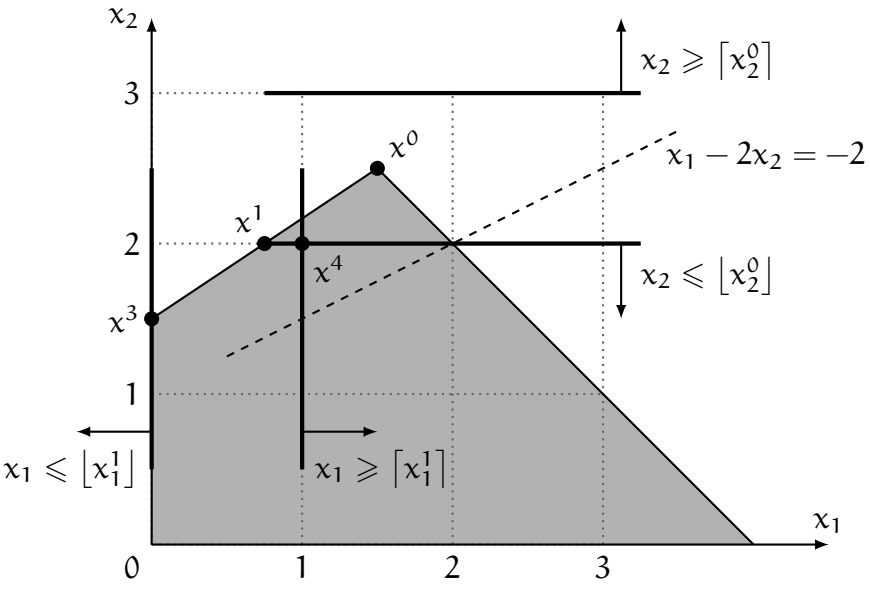

Figure 13.1: Illustration of Dakin's method, applied to the IP of Example 13.1

branching and can for example be split into $X^1 = \mathbb{Z}^2 \cap \tilde{X}^1$ and $X^2 = \mathbb{Z}^2 \cap \tilde{X}^2$, where

$$\tilde{X}^1 = \{x \in \tilde{X} : x_2 \leqslant 2\} \quad \text{and} \quad \tilde{X}^2 = \{x \in \tilde{X} : x_2 \geqslant 3\}.$$

We then bound X^1 and X^2 by solving the corresponding LP relaxations, and obtain $\ell(X^1) = \min_{x \in \tilde{X}^1} f(x) = f(x^1) = -13/4$ for $x^1 = (3/4, 2)$, and $\tilde{X}^2 = \emptyset$. We thus set $L = \{X^1\}$. We now branch by splitting X^1 into $X^3 = \mathbb{Z}^2 \cap \tilde{X}^3$ and $X^4 = \mathbb{Z}^2 \cap \tilde{X}^4$, where

$$\tilde{X}^3 = \{x \in \tilde{X}^1 : x_1 \leqslant 0\} \quad \text{and} \quad \tilde{X}^4 = \{x \in \tilde{X}^1 : x_1 \geqslant 1\},$$

and bounding X^3 and X^4 to obtain $\ell(X^3) = \min_{x \in \tilde{X}^3} f(x) = f(x^3) = -3$ for $x^3 = (0, 3/2)$ and $\ell(X^4) = \min_{x \in \tilde{X}^4} f(x) = f(x^4) = -3$ for $x^4 = (1, 2)$. Since $x^4 \in X$, we can set $U = f(x^4) = -3$. Then, $\ell(X^3) \geqslant U$, so we can discard X^3 and are done.

13.2 The Traveling Salesman Problem

Recall that in the traveling salesman problem (TSP) we are given a matrix $A \in \mathbb{N}^{n \times n}$ and are looking for a permutation $\sigma \in S_n$ that minimizes $a_{\sigma(n)\sigma(1)} + \sum_{i=1}^{n-1} a_{\sigma(i)\sigma(i+1)}$. Matrix entry a_{ij} can be interpreted as a cost associated with edge $(i, j) \in E$ of a graph $G = (V, E)$, and we are then trying to find a *tour*, i.e., a cycle in G that visits every vertex exactly once, of minimum overall cost. We have seen that the TSP is NP-hard, but we could try to encode it as an integer program and solve it using branch and bound. Consider variables

$$x_{ij} \in \{0, 1\} \quad \text{for } i, j = 1, \ldots, n, \tag{13.1}$$

encoding whether the tour traverses edge (i, j). There are various ways to ensure that these variables indeed encode a tour, i.e., that $x_{ij} = 1$ if and only if $\sigma(n) = i$ and

$\sigma(1) = j$, or $\sigma(k) = i$ and $\sigma(k+1) = j$ for some $k \in \{1, \ldots, n-1\}$. Of course, there has to be exactly one edge entering and one edge leaving every vertex, i.e.,

$$\begin{aligned} \sum_{i=1}^{n} x_{ij} &= 1 \quad \text{for } j = 1, \ldots, n, \\ \sum_{j=1}^{n} x_{ij} &= 1 \quad \text{for } i = 1, \ldots, n. \end{aligned} \tag{13.2}$$

The so-called cut-set formulation additionally requires that there are at least two edges across every cut $S \subseteq V$, whereas the subtour elimination formulation makes sure that no set $S \subsetneq V$ contains more than $|S|-1$ edges. The problem with both of these formulations is of course that they require an exponential number of constraints, one for each set $S \subseteq V$.

A polynomial formulation can be obtained by introducing, for $i = 1, \ldots, n$, an auxiliary variable $t_i \in \{0, \ldots, n-1\}$ indicating the position of vertex i in the tour. If $x_{ij} = 1$, it holds that $t_j = t_i + 1$. If $x_{ij} = 0$, on the other hand, then $t_j \geqslant t_i - (n-1)$. This can be written more succinctly as

$$t_j \geqslant t_i + 1 - n(1 - x_{ij}) \quad \text{for all } i \geqslant 1,\ j \geqslant 2,\ i \neq j. \tag{13.3}$$

Since values satisfying (13.3) exist for every valid tour, adding this constraint does not affect solutions corresponding to valid tours. On the other hand, it suffices to rule out subtours, i.e., cycles of length less than $|V|$. To see this, consider a solution that satisfies (13.3), and assume for contradiction that it consists of two or more subtours. Summing the constraints over the edges in a subtour that does not contain vertex 1 leads to the condition that $0 \geqslant k$, where k is the number of edges in the subtour, a contradiction.

A minimum cost tour can thus be found by minimizing $\sum_{i,j} x_{ij} a_{ij}$ subject to (13.1), (13.2), and (13.3). This integer program has a polynomial number of variables and constraints and can be solved using Dakin's method, which bounds the optimum by relaxing the integrality constraints (13.1). There are, however, other relaxations that are specific to the TSP and can provide better bounds.

Observe, for example, that the integer program obtained by relaxing the subtour elimination constraints (13.3) is an instance of the assignment problem (9.1). It can be solved efficiently in practice using the network simplex method, which yields a solution consisting of one or more subtours. If there is more than one subtour, then taking the set $\{e_1, \ldots, e_k\} \subseteq E$ of edges of one or more of the subtours and disallowing each of them in turn splits the feasible set Y into $Y_1, \ldots, Y_k$, where $Y_i = \{x \in Y : x_{uv} = 0, e_i = (u, v)\}$. Clearly, the optimal TSP tour cannot contain all edges of a subtour, so it must be contained in one of the sets Y_i. Moreover, adding a constraint of the form $x_{ij} = 0$ is equivalent to setting the corresponding cost a_{ij} to a large enough value, so the new problem will still be an instance of the assignment problem. Note that none of the sets Y_i contains the optimal solution of the current relaxation, so $\ell(Y_i) \geqslant \ell(Y)$ for all i, and $\ell(Y_i) > \ell(Y)$ if the optimal solution of the current relaxation was unique.