8 Branch-and-bound and Cutting Plane methods

As I mentioned before, I am going to introduce two more ideas, which utilize (and certainly improve) LP relaxation. However, I am not going to explain them in details and only stick to very general ideas. There will not be too much math and much of the explanations will be in words.

Let us start by the branch-and-bound algorithm. The idea of branch-and-bound is in fact very straightforward: For any feasible optimization, suppose that the feasible region Ω can be divided into two parts Ω_1 and Ω_2, i.e. $\Omega = \Omega_1 \cup \Omega_2$. Further, suppose that we can optimize the objective function over Ω_1 and ω_2, and obtain the optimal points $\mathbf{x}_1, f_1$ and $\mathbf{x}_2, f_2$ for their corresponding optimal values and optimal points, respectively. Then, the optimal value for the original optimization is the minimum of f_1 and f_2. The optimal point is also its corresponding point between $\mathbf{x}_1$ and $\mathbf{x}_2$. The process of dividing the feasible region into two parts is called *branching*. One way of branching is to use hyperplanes and half-spaces.

Take a half-space $H = \{\mathbf{x} \mid \mathbf{a}^T\mathbf{x} + b \geq 0\}$, where $\mathbf{a}$ and b are arbitrary. Then, $\Omega_1 = \Omega \cap H$ and $\Omega_2 = \Omega \cap H^c = \Omega \backslash H$ can be the branches. An example is depicted in Figure 6. The main advantage of this type of branching is that after branching for ILPs, the resulting optimizations are still ILPs. There is no rigorous rule for choosing the half-space, but there are some **heuristic branching rules**. This means that we use some branching rules, which seem to be appropriate, but we do not have a mathematical argument in favor of them. The heuristic rules are often simply referred to as *heuristics*. The main problem with the integer optimizations is that they

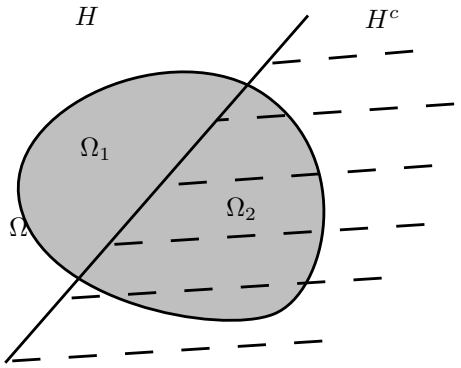

Figure 6: An example of branching.

cannot be in general exactly solved, even after branching. However, LP relaxation and rounding can provide upper and lower bounds for the ILP optimizations. Take a minimization example. The LP relaxation gives a lower-bound, while an approximate solution (obtained by rounding the LP solution or by the primal-dual method, or etc) gives an upper bound. Suppose that for an original problem, the LP relaxation and rounding give u and l as upper-bound and lower bound, respectively. Suppose that we perform linear branching and obtain two restricted optimizations, which are again ILPs and can be approximately solved by LP relaxation. Suppose that the LP relaxation over the branches give the bounds u_1, l_1 and u_2, l_2 respectively. Now, we can make some conclusions.

1. The value $\min(u, u_1, u_2)$ is a new upper bound for the original optimization. We can select the optimal rounded solution corresponding to the minimum as the "best solution so far".

2. If the lower bound l_2 is larger than (and not equal to) the upper bound $\min(u, u_1)$. The optimal solution is not in Ω_2.

The idea of the branch and bound technique is to successively break the feasible region into smaller pieces and use lower bounds of the LP relaxation to discard some areas (Item 2) above. Figure 7 shows an example. A simple way to implement the branch and bound technique is to use a stack of ILP optimization instances:

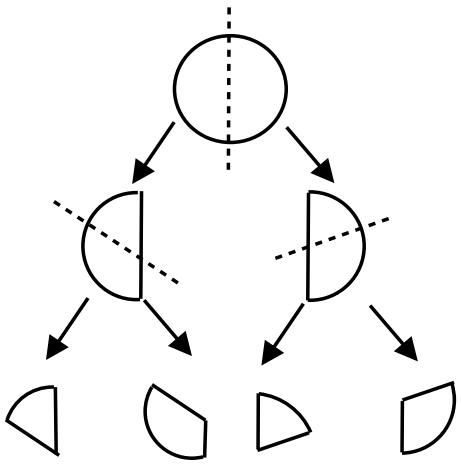

Figure 7: An example of successive branching.

1. In the beginning, the stack only includes the original problem. In the entire algorithm we save and update the best integer solution obtained so far $\mathbf{x}^*$.
2. At each iteration, we read one problem P and remove it from the stack.
3. We apply LP relaxation to P. Its optimal value gives a lower bound l and its rounded solution $\bar{\mathbf{x}}$ gives an upper bound.
4. We compare the rounded solution $\bar{\mathbf{x}}$ with the best solution so far $\mathbf{x}^*$ if its cost is smaller, we replace $\mathbf{x}^*$ by $\bar{\mathbf{x}}$.
5. We check the lower bound l. If is larger than the cost of $\mathbf{x}^*$, then our optimal solution cannot be in the feasible region of P. Hence, we do nothing.
6. If the lower bound is smaller than the optimal cost so far and the solution of the LP relaxation for P was not integral, we should make a closer look at the feasible region of P. Hence, we branch P and add the two resulting optimizations to the stack.

8.1 The cutting plane algorithm

Similar to the primal-dual method, the cutting plane algorithm can also be applied to continuous optimizations. It is a popular method in the convex optimization area. The idea of the cutting plane method is to successively restrict the feasible region of the LP relaxation by adding extra constraints. Each extra constraint restricts the feasible region to the intersection of our original feasible region and a half-space. This constraint is simply called a cutting plane. In the cutting plane method once an LP with the extra constraint is solved, it may help us to find better cutting planes.

Take a look at the example in Figure 8. The thin outer lines depict the original constraint and the inner lines are the cuts. Clearly, the feasible region of the LP problem is restricted by the cuts. However, if we consider the ILP problem, there is no difference between the Two regions. The cuts that do not change the feasible region of the ILP, but make the one for the LP relaxation smaller are called **valid cuts**. The inner valid cuts (thick ones) are in a sense more special than the others: If we try to change them such that the LP feasible region gets smaller, we will lose an integer point. So, they define the minimal LP region, which includes all feasible integer solutions. These cuts are called **strong cuts**. Notice that the inclined line defines a valid cut, but it is not strong as one can move it downward, without losing any integer solution.

The strong valid cuts always exist. If we find them and add them to the LP relaxation, we obtain the optimal integer solution. The problem is that finding the strong valid cuts is not simple.

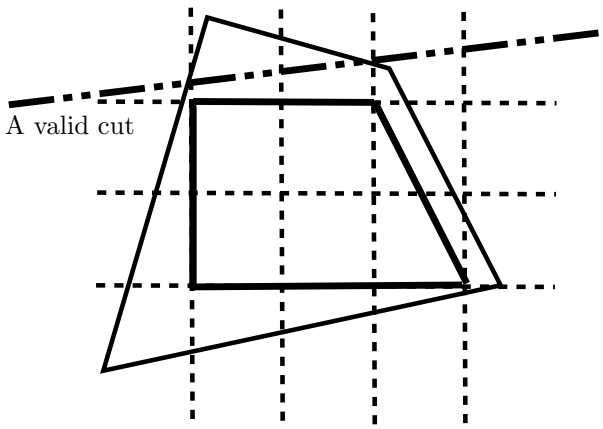

Figure 8: An example of the valid cutting planes. The thin outer lines depict the original constraints. The thick inner lines are the strong valid cuts. The dash-dotted line is a valid but not strong.

In practice, it is more likely to find a valid cut, such as the inclined line in Figure 8 than the strong ones. However, you may refine your LP in a recursive way: every time that you add a number of cuts and obtain a non-integral solution, you may add a new valid cut based on the LP solution. There is a simple way to generate such cuts from the solutions, which is known as the Gomory's method. I cannot explain this method as it requires to know first about the simplex method. The Gomory's cutting plane method is not so successful in general. However, it may replace (and improve) the LP relaxation step in the branch and bound method. The result is called the **branch and cut** method, which is a powerful tool to solve ILPs.