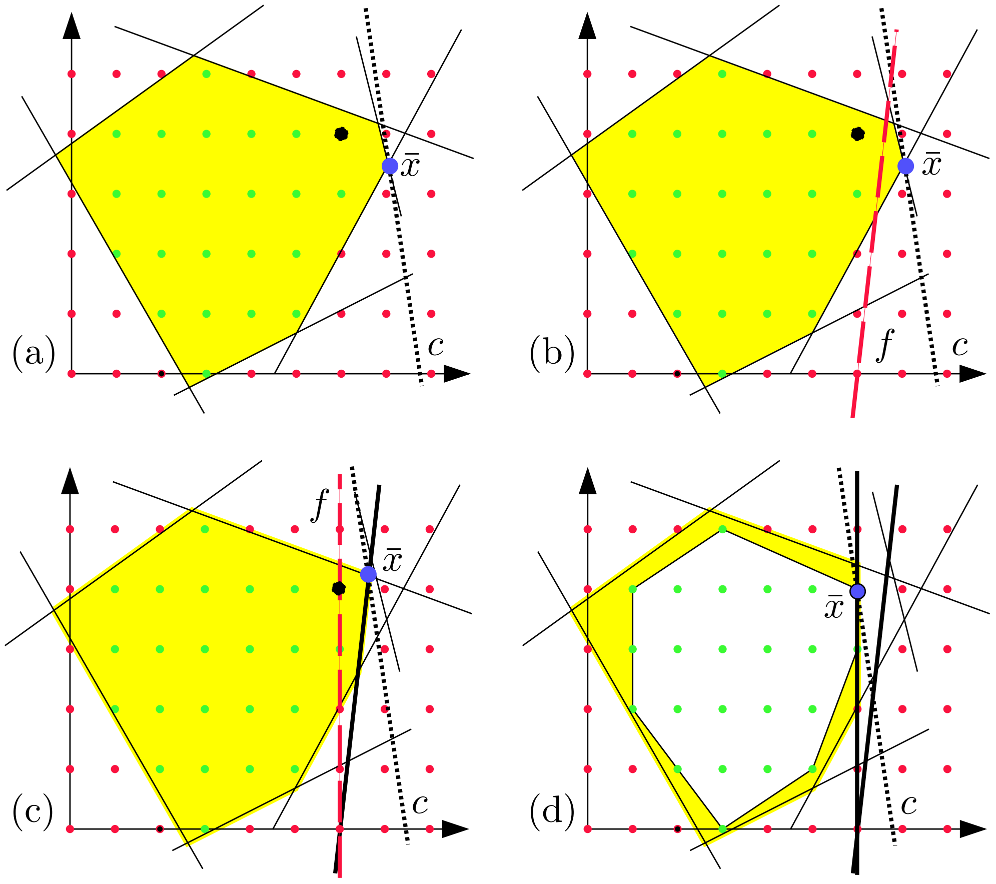
Solving the ILP using branch-and-cut

Solving ILPs is a main topic in combinatorial optimization. We will take a brief look at the *branch-and-cut* approach.

Branch-and-cut makes use of two techniques:

- **Cutting planes:** to solve an ILP, one considers relaxations of the problem (usually the LP-relaxation) and repeatedly "cuts" away parts of the polytope (by adding new constraints) in the hope of obtaining an integer solution. To find cutting planes one has to solve the *separation problem*, that means find a violated, *valid* inequality of the ILP.
- **Branch-and-bound:** an enumeration tree of all possible variable settings is partially traversed, computing local upper bounds and global lower bounds, which are used to avoid parts of the tree that cannot produce the optimal value.

Solving the ILP using branch-and-cut (2)



Adding cutting planes. The last cutting plane is facet-defining and leads to the optimal integer solution.

Solving the ILP using branch-and-cut (3)

The convex hull of all feasible incidence vectors (the green points in the previous figure) forms the *problem polytope P* (the inner, white polytope in the previous figure). If one considers the inequalities of the ILP, they generally describe a larger polytope (the yellow polytope), although this polytope does not contain an infeasible integer point (the red points).

If the solution $\bar{x}$ of the LP-relaxation is integral, it corresponds to a feasible incidence vector that represents an optimal solution. Otherwise we search for a *valid* inequality $fx \leq f_0$ that “cuts off” the solution $\bar{x}$, i. e.,

$$\begin{aligned} fy &\leq f_0 \qquad \text{for all } y \in P \text{ and} \\ f\bar{x} &> f_0 \ . \end{aligned}$$

The set $\{x \mid fx = f_0\}$ is called a *cutting plane*.

Solving the ILP using branch-and-cut (4)

The search for a cutting plane is called the *separation problem*. Any cutting plane found is added to the linear program and the linear program is solved again.

There is a special class of cutting planes we are interested in, namely the *facets* of the problem polytope. As we know, facets of a d-dimensional polytope are faces of dimension $d - 1$. They are in a sense the “best” cutting planes since they directly bound the problem polyhedron. Hence, it is important to *identify the classes of facet-defining inequalities*.

Solving the ILP using branch-and-cut (5)

We generally compute cutting planes for two reasons:

1. A class of contraints in our ILP formulation is too large (i.e., exponentially large) to write down (we relax the LP-relaxation further).

2. The inequalities in the original ILP formulation are not sufficient to yield an integer solution. Hence we search for valid inequalities that cut off those fractional solutions.

For example consider the number of mixed cycles in the MWT problem. It grows exponentially with the size of the graph. Instead of writing them all down, we relax the problem first by omitting them, and then compute cutting planes by checking whether they are violated. [Indeed we do something slightly different, since the mixed cycle inequalities do *not* always describe facets of the MWT-polytope.]

ILP using branch-and-cut (6)

How often do we have to repeat that? Are we guaranteed to find a solution? The following theorem by Grötschel, Lovász, and Schrijver gives a partial answer:

Theorem. For any proper class of polyhedra, the optimization problem is polynomially solvable if and only if the separation problem is polynomially solvable.

This basically says, that for NP-complete problems there is little hope to guarantee a quick termination of the cutting plane algorithm.

ILP using branch-and-cut (7)

Either we just do not know enough classes of facet-defining inequalities, or we cannot solve the separation problem for one of these classes in polynomial time.

Hence we repeat the cutting step either until an integer solution is found that fulfills all constraints (which would be optimal for the original problem), or until we get stuck, that means we cannot find a violated inequality for *any* of our classes of valid inequalities.

ILP using branch-and-cut

In this case we **branch**. That is, we choose a variable x_i and solve two sub-cases, namely the case $x_i = 0$ and the case $x_i = 1$. Repeated application produces a enumeration tree of possible cases.

We call an upper bound for the original ILP *local*, if it is obtained from considering a subproblem in the enumeration tree.

If the solution found for a subproblem is feasible for the original problem and has a higher score than any solution found so far, then it is recorded and its value becomes the new *global lower bound* for the original objective function. Remember that the feasibility of a new solution has to be carefully checked, since we fixed a number of variables on our way to the subproblem.

ILP using branch-and-cut (9)

Subsequently, we only pursue subproblems whose local upper bound is greater or equal to the global lower bound.

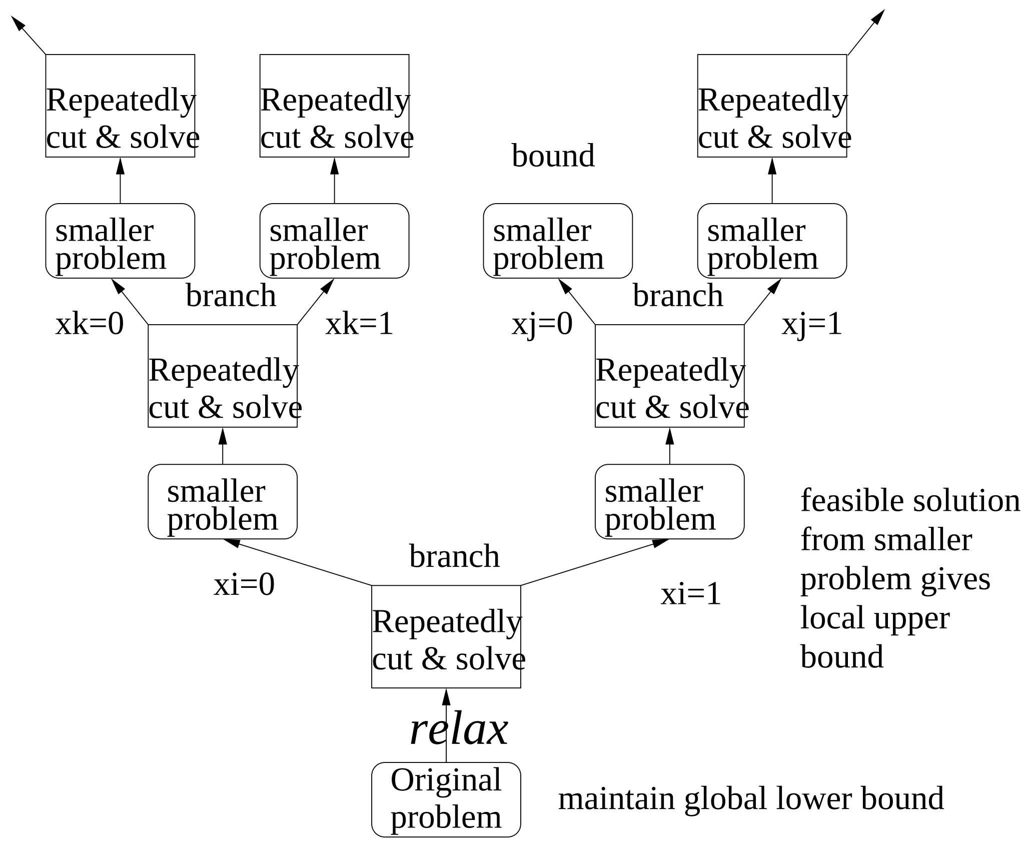
This is an example of the *branch-and-bound* paradigm for solving hard combinatorial problems.

Branch-and-bound is a divide-and-conquer approach to solving a problem by dividing it into smaller problems. The *local* solution of a subproblem gives rise to a lower bound for the best possible score for the original problem, if the solution of the subproblem also solves the original problem.

The highest such local score attained so far gives rise to a *global* lower bound for the original problem and is used to skip parts of the enumeration tree in which the current global lower bound can't be beaten.

ILP using branch-and-cut

General strategy:



The details are a bit more involved and are skipped here. We now take the initial MWT problem as an example to point out the important steps in branch-and-cut algorithms.