Quiz.

1. The optimum of an integer program can be much worse than the optimum of the linear relaxation, even if the coefficients in the problem are rational and bounded.

A. Yes, it can
B. No, it can't

2. In branch and bound, you can eliminate some subproblems even when you don't know an integer solution yet.

A. Yes, you can
B. No, you can't

Branch and bound and branch and cut

Hannah Cairns

April 20, 2022

The problem: Mixed integer linear programs

In a **mixed integer linear program**, or MILP, we maximize a linear function under linear constraints. The twist is that some of the variables in the solution are required to be integers.

Here is a simple example.

$$\begin{aligned} \text{maximize} \quad & y \\ \text{under the constraints} \quad & 0.001\,y \leq x \\ & 0.001\,y \leq 1 - x \\ & x \in \mathbb{Z} \\ & y \in \mathbb{R} \end{aligned}$$

Integers are green and real numbers are blue.

The maximum possible value of y in this problem is 0.

- We can get the value 0 at $x = 0$, $y = 0$.
- We require x to be an integer, so $x \leq 0$ or $x \geq 1$. Therefore,

$$\begin{aligned} 0.001y &\leq x \\ 0.001y &\leq 1 - x \end{aligned} \qquad \Longrightarrow \qquad 0.001y \leq 0,$$

and $y \leq 0$ whenever the inequalities are satisfied.

$$\begin{aligned} \text{maximize} \quad & y \\ \text{under the constraints} \quad & 0.001\,y \leq x \\ & 0.001\,y \leq 1 - x \\ & x \in \mathbb{R} \\ & y \in \mathbb{R} \end{aligned}$$

But if we remove the requirement that x has to be an integer, the maximum value is five hundred.

- The point $x = 0.5, y = 500$ satisfies the constraints.
- If we multiply the two inequalities by 500 and add them up, we get the inequality $y \leq 500$, so there's no higher value.

What kinds of problems can be expressed as MILP?

- Routing and delivery problems (real amounts of stock are shipped in an integer number of trucks)
- Scheduling problems
- Lots of complicated things, including any problem in NP: the knapsack problem, satisfiability, the traveling salesman problem, etc.

So this is an NP-hard problem. Why would we want to translate our problems into this format?

Because if we relax the requirement that variables are integers, we get a **linear program**, or LP.

LPs can be solved in polynomial time (very quickly in practice), and this can be exploited to find solutions of the MILP.

Next: A review of linear programming.

A very fast review of linear programming

A linear program is an optimization problem where we maximize a linear function under linear constraints. For example:

$$\begin{aligned} \text{maximize} \quad & c^T x \\ \text{under the constraints} \quad & Ax \leq b \\ & x \in \mathbb{R}^n \end{aligned}$$

We are given vectors c and b, which have size n and m respectively, and an $m \times n$ matrix A.

This isn't the only way to write a linear program, but the other ways can all be put in this form.

The **feasible set** of a linear program is the set of points that satisfy the constraints, say $S := \{x \in \mathbb{R}^n : Ax \leq b\}$.

A linear program can be:

- **infeasible**, if the feasible set is empty,
- or **unbounded**, if the supremum of $c^T x$ over S is $+\infty$.

If it's not infeasible or unbounded, then it has an optimal solution.

An **extreme point** of S is a point in S that is not on any line between two other points in S.

If the feasible set of a program is bounded, then there is an optimal solution at an extreme point.

At every extreme point $x = (x_1, \ldots, x_n)$, there is a set of n rows of the matrix A which are linearly independent, and for which

$$\sum A_{ij} x_j = b_j \text{ for every row } i \text{ in the set.}$$

These equations have a unique solution.

Say that a set of n rows is a **feasible basis** if it's linearly independent and the solution is in the feasible set. We can find an optimal solution by searching through all the feasible bases, but this would take $\binom{m}{n}$ time.

The simplex method

The simplex method is a practical way to solve LPs.

It's a type of local search. A basis is a **neighbor** of another basis if they share all but one row.

The method: First, find a basis to start with.

Then:

1. Look at every feasible neighbor of the current basis to see if it's good.
2. If there's no good neighbor, we're at an optimum. Stop.
3. Otherwise, pick one and move to it. Go back to step 1.

Details have been omitted. The simplex algorithm is usually fast, although it can take exponential time in the worst case.

Duality

To prove that the maximum of the problem earlier was 500, we used a trick where we multiplied the inequalities in the constraints by some positive constants and added them up to bound the value.

This trick always works. For any bounded linear program, there are always coefficients $y_1, \ldots, y_m \geq 0$ so that the sum

$$\sum_{i=1}^{m} y_i \left(\sum_{j=1}^{n} A_{ij} x_j \right) \leq \sum_{i=1}^{m} y_i b_i$$

adds up to $c^T x$ on the left, and the optimal value on the right.

How can we find those special coefficients y? Of course, by solving another linear program!

The new program is called the **dual**, and the original one is called the **primal**. When the primal program asks for the maximum of $c^T x$ under the constraints $Ax \leq b$, the dual program is

$$\begin{aligned} \text{minimize} \quad & y^T b \\ \text{where} \quad & c = y^T A \\ & y \geq 0 \end{aligned}$$

For any x in the primal and any y in the dual, we have

$$c^T x = y^T A x \leq y^T b.$$

So we must have $\max c^T x \leq \min y^T b$.

The wonderful thing is that they are always equal. This is the **strong duality theorem**.

This means that, if I find an optimal solution x^* of the primal, then I can prove to you that it's optimal by giving you an optimal solution y^* of the dual.

You can verify my claim just by checking that x^* is in the primal, y^* is in the dual, and $c^T x^* = (y^*)^T b$.

Duality is a powerful way to transform a problem. For example, you can write the two problems below as linear programs, and they're dual to each other:

find the minimum cut necessary to
separate two points on a graph

$\Uparrow$

dual

$\Downarrow$

find the maximum flow between two points
if each edge can carry at most 1 current

So the minimum cut is the same as the maximum flow.

Back to integer programming

We can search for good solutions to a mixed integer linear program by exploiting the linear program structure.

Our overall strategy for doing that is:

- Find the optimal solution of the relaxed problem, and hope that it satisfies the integer constraints.
- If it doesn't, we have a non-integer solution to the linear program. Break the problem into subproblems in a way that:
  - excludes the non-integer solution, and
  - does not exclude any solution that satisfies the integer constraints.

  Then explore the subproblems.
- If we find an optimal solution in a subproblem, and it satisfies the integer constraints, then remember it and use it to reject any subproblems with a smaller optimum.

For example, consider the problem earlier. The optimum was at the non-integer point $x = 0.5, y = 500$.

If x is an integer, then either $x \leq 0$ or $x \geq 1$. So the solution of our original problem is the best of these two subproblems:

$$\begin{aligned} \text{maximize} \quad & y \\ \text{where} \quad & 0.001\,y \leq x \\ & 0.001\,y \leq 1 - x \\ & \boxed{x \leq 0} \\ & x \in \mathbb{Z}, y \in \mathbb{R}, \end{aligned}$$

$$\begin{aligned} \text{maximize} \quad & y \\ \text{where} \quad & 0.001\,y \leq x \\ & 0.001\,y \leq 1 - x \\ & \boxed{x \geq 1} \\ & x \in \mathbb{Z}, y \in \mathbb{R}. \end{aligned}$$

If we relax the first subproblem and solve it as a linear program, the optimum is

$$\text{value} = 0$$
$$x = 0$$
$$y = 0,$$

This satisfies the integer constraints, namely $x \in \mathbb{Z}$, so it's an optimal solution for the first subproblem! Its value is 0.

We have to keep going, though, because we have another subproblem and it might be better.

When we relax the second subproblem, the solution is

$$\begin{aligned} \text{value} &= 0 \\ x &= 0 \\ y &= 0, \end{aligned}$$

Hey, the optimal relaxed solution only has value 0! That isn't any better than the solution we already have, so we discard this subproblem.

We don't have any more subproblems, so we're done, and the answer is value $= 0, x = 0, y = 0$.

Branch and bound

This is the core idea of **branch and bound**.

You can break up the problem into subproblems by choosing a linear function which should be an integer, but isn't.

For example, if x_j is the relaxed optimum of the current subproblem, you'd want to pick integers n_j with

$$\sum n_j x_j = a, \qquad a \notin \mathbb{Z},$$

Then you'd create your subproblems by adding the constraint $\sum n_j x_j \leq \lfloor a \rfloor$ to one, and $\sum n_j x_j \geq \lceil a \rceil$ to the other one. That excludes the non-integer optimum from both the subproblems.

Is it that simple?

That's the basic idea, but to get it to work, you have to think about many details.

- This only works if some of the relaxed subproblems have integer extreme points. That can depend a lot on the formulation of the problem.
- How do you explore the tree? At first, you want to find a good integer solution so you can reject a lot of subproblems, so you might want to do a depth-first-ish search until you get one.
- How do you make life easier for the LP solver? As the primal problems get smaller, the dual problem gets bigger. If you use the simplex method on the dual problem, you can start the search for a subproblem at the solution to the parent problem, which is feasible and should already be close to optimal.

Branch and cut

In **branch and cut**, you do all the same stuff, but you get one more strategy for eliminating a non-integer optimum.

If an extreme point of the relaxed problem doesn't satisfy the constraints, and the feasible set is bounded, there is always some inequality $\sum \lambda_j x_j \leq a$ that cuts off that extreme point, but doesn't cut off any integer solution. This is called a **cut**.

Instead of branching, the algorithm can **pick an inequality that cuts off the bad optimum**, add it to the constraints, and solve the problem again.

It's not that simple.

How do you decide when and how to branch and what cuts to make? This is difficult, but here are some options:

- Use a general family of cuts due to Gomory. If you make enough cuts then you are guaranteed to get to the true optimum, eventually
- Find a paper about your specific problem which gives a good family of cuts, and choose one of them that is violated.
- Give $10,000 to Gurobi Optimization, LLC and they will do it for you

There is no known way to optimize a general mixed integer linear program in polynomial time. (If there was one, P would be NP.) But it's often possible to solve particular problems.

Solvers to try out:

- Open source: SCIP, cbc
- Closed source: Gurobi, Cplex, Xpress

And beware! Modeling is often not that great.

- Things might be optimal or close to optimal to start with, so that there's no room for improvement.
- It might be much more effective to look for simple efficiencies outside the model.
- Even if there are savings, they might not be enough to justify paying the modeling team.

See "Linear programming: some unsuccessful applications."

Thanks for listening!