Linear Programming

- Variables $x_j \in \mathbb{R}$ for $j \in \{1, \ldots, n\}$
- Maximize $\sum_j c_j x_j$
- Constraints $\sum_j a_{ij} x_j \leq b_i$ for $i = 1, \ldots, m$
- and $x_j \geq 0$ for all j.

Can be solved using the simplex algorithm.

Simplex has exponential running time in the worst case. (In practice it seems to work well on most problems.)

Complexity of Linear Programming

Khachian 1979: ellipsoid method with weakly polynomial running time.
(Running time is polynomial in the number of bits of the input, not on the RealRAM model).

Useless in practice.

Karmakar 1984: interior-point method

Also weakly polynomial, but quite useful in practice.

Ongoing arms race between simplex and interior-point methods.

The big open question: Is there a strongly polynomial algorithm for linear programming?

Max-flow and linear programming

We have $\textsc{MaxFlow} \leq_P \textsc{LP}$.

Variable x_e for the flow on edge e.

Constraints:

- $x_e \geq 0$
- $x_e \leq c(e)$
- Kirchhoff's law:
  For each vertex $u \in S \setminus \{s, t\}$:
  $\sum_{vu \in E} x_{vu} = \sum_{uv \in E} x_{uv}$.

Target:
Maximize $\sum_{sv \in E} x_{sv}$.

It's a linear program!

Min-cost flow

We can give each edge e a cost $\kappa(e)$.

The cost of a flow is

$$\text{cost}(f) = \sum_e \kappa(e) \cdot f(e).$$

In min-cost flow, we are asking for a flow with minimum cost among all flows of value at least ϕ.

New target: $\min \sum_e \kappa(e) x_e$

New constraint: $\sum_{su \in E} x_{su} \geq \phi$.

Variant: Instead of lower bound ϕ on flow, a lower bound $\ell(e)$ for each edge.

Why Max-Flow?

So why did we waste (?) so much time discussing max-flow instead of learning linear programming immediately?

- There is a strongly polynomial algorithm for max-flow!
- A strongly polynomial time algorithm for min-cost flow exists as well.
- In practice, max-flow problems are often solved by LP solvers, but for some applications we can do better.
- When all capacities are integers, then Ford-Fulkerson and other max-flow algorithms guarantee that the max-flow has integer value on each edge.
- This is essential for applications such as bipartite matching, project selection, disjoint paths, etc.

IP is hard

3-SAT $\leq_P$ IP

MonotoneSAT $\leq_P$ IP

. . . and so IP is NP-hard.

Still, IP solvers solve many practical IP problems, it is worth trying one for a problem at hand.

Integer (Linear) Programming

- Variables $x_j \in \mathbb{Z}$ for $j \in \{1, \ldots, n\}$
- Maximize $\sum_j c_j x_j$
- Constraints $\sum_j a_{ij} x_j \leq b_i$ for $i = 1, \ldots, m$
- and $x_j \geq 0$ for all j.

Commonly known as Integer Programming (IP) or as ILP.

Mixed integer linear programming means that some variables are in $\mathbb{R}$, others in $\mathbb{Z}$.

Writing max-flow as an IP, we can restrict the variables to be integers, and the solver will give us an integer solution. . .

However, . . .