Lecture 13: Optimization II: From flows to LPs

Recap: Max-Flow. [A very general problem encapsulating lots of interesting special cases.]

[[leave up]]

Input:
- Digraph $G=(V,E)$
- "source" $s \in V$, "target" $t \in V$
- positive integer "capacities" c_e on edges $e \in E$

"Feasible" outputs: flow $f : E \to \mathbb{R}$ s.t.

$$0 \le f(e) \le c_e \quad \forall e \in E$$

$$\sum_{u \to v} f(u,v) = \sum_{v \to w} f(v,w) \quad \forall v \in V \setminus \{s,t\}$$

Objective: max total flow, $\sum_{s \to v} f(s,v)$ [assuming no edges into s]

Ford-Fulkerson Alg.:
- $O(F^* m)$ time, F^* = opt. value
- Always finds an opt. solution with $f(e)$ integer $\forall e$.
- Also finds a minimum st-cut of value F^*

$R \subseteq V$, $s \in R$, $t \notin R$

$$\text{cutvalue}(R) = \sum_{\substack{u \to v \\ u \in R,\ v \notin R}} c(u,v)$$

"Obvious": if R is any cut, max flow $\le$ cutvalue(R)

Powerful: Always exists cut R^* with maxflow = cutvalue(R^*) [found by F.F.]

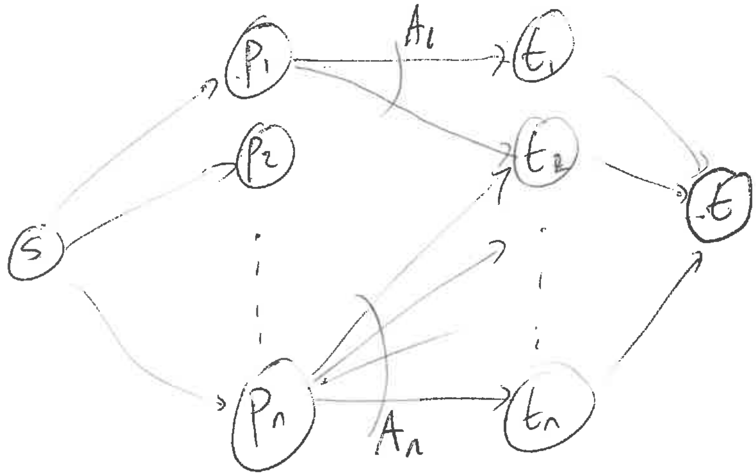
[R^* is like a certificate/proof that maxflow $\le$ something, R^* is an optimal certificate.]

Application: Task assignment [[matchings in bipartite graphs]] [[an original motivation for F.F.]]

Input:
- n tasks $t_1, \dots, t_n$
- n people $p_1, \dots, p_n$ [[not important that same # of people & tasks, just want to keep it simple]]
- each person p_i has a set $A_i \subseteq \{t_1, \dots, t_n\}$ of tasks they're capable of doing

Goal: Assign tasks (≤ 1 per person), maximize # assigned.

[[Not obviously a max-flow problem! Let's first make a graph...]]



[[Direct edges left-to-right, add fictitious source, sink.]]

$c_e = 1 \quad \forall$ edges

[[What do flows have to do with it?]]

Claim 1: If there's an assignment of $\ge F$ tasks, $\Rightarrow$ max-flow $\ge F$.

proof: For each (p_i, t_j) in assignment, push 1 unit of flow $s \to p_i \to t_j \to t$.

[[Caps okay:
- each $s \to p_j$ edge has ≤ 1 flow $\because$ people not overused.
- each $t_j - t$ " " " $\because$ tasks not multi-assigned]]

Claim 2: If there's an integer flow of val $\ge F$ $\Rightarrow$ can assign F tasks

proof: Integer flow $\Rightarrow$ $f(e) = 0$ or 1 for all e.

$\therefore \ge F$ saturated (flow-1) edges out of s.

Each must go to a different t_j ($\because$ each t_j has out-capacity 1)

$\Rightarrow$ assignment of $\ge F$ tasks.

$\therefore$ max # tasks = max integer flow = max flow,
& Ford-Fulkerson finds in $O(nm)$ time. ☺

⟦Many other scheduling tasks solvable with max-flow.
Including some with <u>costs</u>.⟧

<u>twist</u>: Min-<u>cost</u> max-flow:
Now each edge e has a <u>cost</u>, $\ell(e)$.
Goal: find the max-flow of minimal <u>cost</u>.

<u>Fact</u>: Also solvable by a Ford-Fulkerson-like alg. ⟦And faster ones, too⟧
⟦Main ideas: when selecting st-path to push flow on, choose one of cheapest cost. Then, in residual graph, your "back edges" get the <u>negative</u> of the cost: it's like you get money back if you reverse them.⟧

<u>Application</u>: In task assignment, every (p_i, t_j) pair could have a <u>payment</u> $\ell(p_i, t_j)$ ⟦to be paid if you make that assignment⟧
⟦Once you do a plain max-flow to determine most # of jobs assignable, F^*, can also determine the least payment needed to get those jobs done.⟧

<u>Facts</u>: As of 35 years ago, Goldberg-Tarjan had an $O(m \cdot n \cdot \log^{\text{const}}(mn))$-time min-cost max-flow alg. ⟦Practical, albeit quadratic time.⟧
As of 1 year ago: $O(m \log^{\text{const}}(mn))$-time alg! ⟦Probably not yet practical.⟧

~~Summary:~~ ⟦Summary: given any kind of optimization/scheduling problem, try to model as a (min cost)-max flow problem! Then $\exists$ good off-the-shelf algs.⟧

[[You can take this idea to the next level of generality via...]]

Linear Programming

[[IMHO the greatest problem solvable in polynomial time]]

- an extremely general task, including many (all?) optimization probs [[including max-flow]]
- solvable efficiently in theory & practice
- [[but no nearly-linear-time alg. currently known, unlike with flow problems]]

Example input: Find $x_1, x_2, \ldots$ // "n" real variables, here $n = 2$

(*) such that
$$x_1 + x_2 \le 7$$
$$-2x_1 + 3x_2 \ge -4$$
$$x_1 \ge 0$$
// "m" linear inequalities, ("constraints")

& maximizing $5x_1 + x_2$ ← linear "objective"

Solution: Max value is 27, achieved by $x_1 = 5$, $x_2 = 2$.

Rules: Objective is linear: $c_1x_1 + c_2x_2 + \cdots + c_nx_n$ for real consts $c_1, \ldots, c_n$.

Can't have x_1^2 or $2|x_1| - 5|x_2| \ldots$

Maximize or minimize [[Kind of the same, since maximizing $-f(x)$ is the same as minimizing $f(x)$]]

Constraints also linear ineqs: $a_1x_1 + \cdots + a_nx_n \ge b$

$\le$ or $=$ also OK [[$a \cdot x = b \iff a \cdot x \ge b$ & $a \cdot x \le b$ $\Updownarrow$ $(-a) \cdot x \ge -b$]]

Strict $>, <$ not OK.

FACTS
- Solvable in polynomial time. [[Proofs: not easy! We'll talk about how, but not get into details]]
- Industrial, practical "LP solvers" out there (CPLEX, Gurobi, CVX, ...) [[Can integrate w/ your fave programming language.]]

"LP is a modeling language". Captures...

- solving equations
- shortest paths
- MST
- (min cost) Max-Flow
- much more...

how?

Variables ~~x_1, x_2~~ $\to$ $f(e_1), f(e_2), \dots, f(e_n)$ $\forall$ edges $e_j \in G$.
[this is a variable name]

Constraints: $f(e_i) \geq 0 \quad \forall e_i$

$f(e_i) \leq C_{e_i} \quad \forall e_i$ — C_{e_i} a constant, from input

$$\sum_{u \to v} f(u,v) = \sum_{v \to w} f(v,w) \qquad \forall v \neq s,t.$$

$\rightsquigarrow$ $f^{in}(v) = f^{out}(v) \iff$
$f^{in}(v) - f^{out}(v) = 0$
$\iff \geq 0$ & ≤ 0.

Obj: Maximize $\displaystyle\sum_{s \to v} f(s,v)$.

[That's it. So once you know LPs solvable efficiently, so is Max-Flow.]

[Another wonderful property of LPs:] "Duality" $\equiv$ certificates of optimality.

[Recall little example] (*): I told you optimum val. is 27.
[Why should you believe me? :)]

I showed you $x_1 = 5, x_2 = 2$ achieves 27 $\rightsquigarrow$ a "certificate" OPT ≥ 27

[But doesn't immediately convince you you couldn't do better...]

"Dual certificate": "Mult. constr. 1 by $\frac{17}{5}$, constr. 2 by $\frac{1}{5}$, constr. 3 by 0, and add..."

$$\left\{\frac{17}{5}x_1 + \frac{17}{5}x_2 \leq \frac{119}{5}\right\} + \left\{\frac{8}{5}x_1 - \frac{12}{5}x_2 \leq \frac{16}{5}\right\} + \{0\} \rightsquigarrow \left\{5x_1 + x_2 \leq \frac{135}{5} = 27\right\}$$

This certifies that any feasible sol^n (x_1, x_2) has obj. ≤ 27 !

[[A miracle? No! Always happens, just like with max-flow = min-cut....]]

"LP Duality Thm": Given constrs. $3x_1 - 2x_2 + \sim\sim\sim \geq 7$

Obj: maximize $2x_1 - 5x_3 \cdots + 8x_n =: f(x_1, \ldots, x_n)$

if opt. value is F^*, there's always a "certificate" of form "mult 1st constr by λ_1, 2nd by λ_2, ..., m^{th} by λ_m, add up, and you'll get $f(x_1, \ldots, x_n) \leq F^*$".

LP solvers can also find these λ_i's efficiently!

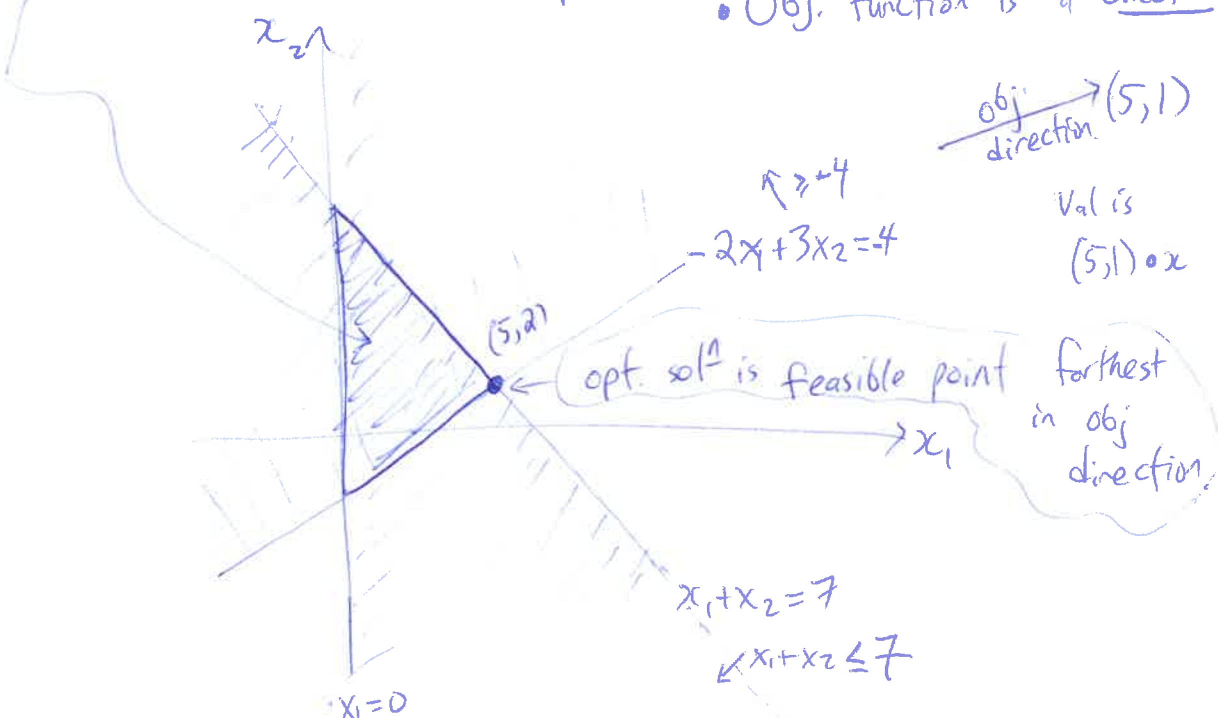
[[So they give you both the opt. sol^n, and a "certificate of optimality"!]]

[[A little bit about how (some) LP solvers work:]]

Geometric perspective:
- Candidate sol^{n}s $x = (x_1, \ldots, x_n)$ are points in $\mathbb{R}^n$!
- Each constraint is a "halfspace" x must be in [[if constr. has "=", it's a hyperplane]]
- "Feasible region" (all valid sol^{n}s) is a "polytope"
- Obj. function is a direction

example ⊛

$\mathbb{R}^2$



"Simplex algorithm" [[a decent heuristic, tho not poly-time]] tries to walk from corner to corner, always improving objective.