Approx. Algorithms using LPs.

Recall:

* to design an approx. algorithm we need a certificate of

upper bound on OPT for max prob

lower bound on OPT for min prob

* this certificate, unlike OPT itself, should be findable efficiently
* we then try to find a solution (usually by tinkering the certificate)

that competes with the certified bound on OPT.

to get an α-factor approx. for ($\alpha \leq 1$ for max prob)
max problem, want

quality(solution) $\geq \alpha \cdot$ (certified upper bound)

to get a C-factor approx. ($C \geq 1$)
min problem, want:

Cost(solution) $\leq C \cdot$ (certified lower bound)

e.g. maximal matchings as certificates for Vertex Cover

e.g. linear programming relaxation value as a certificate for Vertex Cover.

Given $G=(V,E)$, consider the linear program in variables x_u for each $u \in V$

$$0 \le x_u \le 1 \quad \forall u \in V$$

"Coverage constraint" $\forall \{u,v\} \in E, \; x_u + x_v \ge 1$

$$\min \sum_{u \in V} x_u$$

→ objective function

this is a linear program because

1) constraints are linear inequalities in vars

2) objective is linear fun in the vars

Fact: LPs of N variables & M constraints can be solved in time poly(MN).

Def (LP value): objective value of an optimal LP solution.

Why is LP value a certificate on OPT?

because integral solution (= true soln) is always feasible for the LP

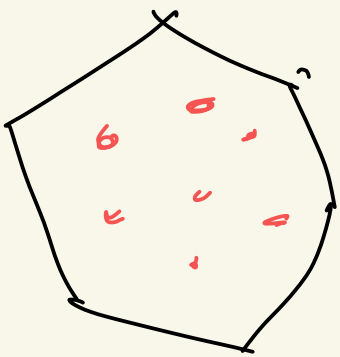
Lemma: Let S be a vertex cover of G. Let $x_u = 1$ if $u \in S$ & $x_u = 0$ otherwise. Then x satisfies the constraints of the LP.

Proof: Clearly $0 \le x_u \le 1$ $\forall u \in V$.

for any edge $\{u,v\}$ $x_u + x_v \ge 1$ since

at least one of u or v is in S.

Since true vertex covers are feasible solutions & LP minimizes over a larger space



red points = true VCs.

black polygon = Space that LP minimizes over

Corollary:

$$\text{LP-value}(G) \leq \text{OPT}(G).$$

Then, you saw a rounding that takes LP solution & finds a VC with cost $\leq 2 \cdot$ LP-Value.

This method of design of approx. algorithms is incredibly general.

1000s of applications.

Many cool ideas.

Today: 1) Application to SAT

2) "Randomized" Rounding

(CNF-)SAT

INPUT: A collection of OR-clauses

n variables with m clauses

OR-clause $= (x_i \vee \overline{x_j} \vee x_k \vee \dots)$

x_i: variable; $\overline{x_j}$: negated var

"literal" = variables or their negation.

"width of an OR-clause" = number of literals in it.

GOAL: ~~find a satisfying assignment~~

find an assignment that satisfies as many clauses as possible.

↳ "MAX-SAT"

MAX k-SAT : all clauses of width exactly k

$\leq$ k-SAT : width $\leq$ k

SAT : no width constraint.

Today: approx. algorithms for MAX-SAT.

BASELINE or easy algorithms

Lemma: For any assignment x, let $\bar{x}$ be its negation.

For any OR-clause, at least one of x & $\bar{x}$ satisfy it.

Proof: take any literal in the clause. Then it is true in at least one of x & $\bar{x}$.

□

Corollary: There is a (super-easy) $\frac{1}{2}$-approx. algorithm for MAX-SAT.

Proof:

Algo: 1) Take any $x \in \{0,1\}^n$

2) output x if it satisfies more clauses than $\bar{x}$, o/w output $\bar{x}$.

Claim: At least one of x & $\bar{x}$ satisfies $\geq \frac{m}{2}$.

Proof: For each clause C, let

$$C(x) = \begin{cases} 1 & \text{if } x \text{ sat } C \\ 0 & \text{o/w} \end{cases}$$

Then $C(x) + C(\bar{x}) \geq 1 \; \forall C$.

So: $\sum_C (C(x) + C(\bar{x})) \geq m$

So for one of x or $\bar{x}$

$$\sum_C C(x) \geq \frac{m}{2}.$$

□

Well, that was easy. Can we beat it? What was the certificate here?

Let's start small and do

MAX 1-SAT

All clauses have one literal.

So the input looks like:

For each i, some copies of x_i

some copies of $\overline{x_i}$

What's a good algorithm?

"Greedy": set x_i to true if more x_i-clauses than $\overline{x_i}$-clauses.

how well does this do?

If # x_i-clauses & $\overline{x_i}$-clauses are equal in number, then this algo satisfies exactly $\frac{m}{2}$

clauses.

Which seems same as the previous one. Or is it.... ?

What is OPT ???

Lemma: the greedy algorithm finds an optimal assignment for MAX 1-SAT.

Proof: WLOG assume that for each i, there are $\geq \frac{1}{2}$ frac of x_i-clauses. Note that this

is WLOG because if this is not true for some i, we simply treat $\overline{x_i}$ as a variable (& x_i as its negation).

Lesson: need a better certificate for OPT to beat $\frac{1}{2}$-approx for MAX 1-SAT!

Let's do MAX 2SAT now.

Here I'll first show you one more "trivial" algorithm

ALGO (RANDOM ASSIGNMENT)

Set $x_i = \begin{cases} 1 & \text{w.p. } \frac{1}{2} \\ 0 & \text{w.p. } \frac{1}{2} \end{cases}$

independently for each $1 \le i \le n$.

Our algorithm is now random.

We've learned enough to not be too scared of that! 

What is the fraction of clauses that this algorithm satisfies?

this is a random variable. So we will compute its expectation/average

Lemma: Fix any MAX k-SAT instance. Let X be the number of clauses satisfied by a random assignment x. Then

$$\mathbb{E}[X] = m \cdot (1 - 2^{-k})$$

$:= m \cdot \alpha_k$ today in short

Proof:

$$X_C = \begin{cases} 1 & \text{if } C(X) \text{ is true} \\ 0 & \text{o/w.} \end{cases}$$

Then $X = \sum_C X_C$

by linearity of expectation

$$\mathbb{E}[X] = \sum_C \mathbb{E}[X_C]$$

X_C is an indicator random

variable. So $\mathbb{E}[X_C] = \Pr[X_C = 1]$

C has k variables.

There are 2^k possible distinct values to this k-tuple.

All are equally likely in x.

All but 1 (the all literals false) assignment satisfy C.

So: $\Pr[X_C = 1] = \frac{2^k - 1}{2^k}$

$= 1 - 2^{-k}$. □

So random assignment is d_k-approx. for MAX-kSAT. (What's the certificate?)

For $k=1$, this is $\frac{1}{2}$. We know how to beat that. ☹

For $k=2$, this is $\frac{3}{4}$. Which is better than $\frac{1}{2}$. ☺

In fact this algorithm gets better as k grows.

What about MAX ≤2-SAT?

Now there are 1-clauses & 2-clauses

For 1-clauses greedy gets 1-approx, but no clear meaning of greedy for

2-clauses.

For 2-clauses, random assignment gets 3/4 but random assignment gets only 1/2 for 1-clauses.

Can we get best of both worlds?

Next: A 3/4 - approx. for MAX≤2-SAT.

Linear Program for MAX≤2-SAT

Key Message: Linear programming relaxations may not be obvious. Sometimes multiple good choices are possible. Requires thought.

Let's build towards the LP slowly.

A natural set of variables is

$y_1, y_2, \ldots, y_n$:

These "stand for" the truth assignment.

We want to encode the constraint that x_is take value in $\{0,1\}$.

But need to allow fractional values to make it a linear inequality constraint.

So: $$0 \leq y_i \leq 1 \quad \text{if } 1 \leq i \leq n \qquad \text{—(1)}$$

Next suppose we want to encode that a 1-clause C is satisfied.

$C = x_i$ or $C = \overline{x_i}$ for some i

(sat if $x_i = 1$) (sat if $x_i = 0$)

encoding: $y_i = 1$ $y_i = 0$.

$\equiv 1 - y_i = 1$

What about 2-clauses?

$C = x_i \vee x_j$ say.

Then: C is sat if $x_i = 1$ or $x_j = 1$ or both are 1.

linear relaxation: $y_i + y_j \geq 1$

$C = \overline{x_i} \vee x_j$

$(1 - y_i) + y_j \geq 1$

$C = x_i \vee \overline{x_j}$

$y_i + (1 - y_j) \geq 1$

$C = \overline{x_i} \vee \overline{x_j}$

$(1 - y_i) + (1 - y_j) \geq 1$

If we write the constraints above, one for every clause in the input, we encode the relaxation for the problem:

"find a satisfying assignment for the input 2-SAT instance"

But we have no reason to assume that the input formula is satisfiable

So our LP may not be feasible

(If we run a LP-solver, it may just output "no solution found")

Key Idea: Suppose for each C

We knew if C is satisfied in an optimal assignment.
Then, we could write the the above constraints only for the C we know are satisfied & ignore the others

But, we do not know this information (duh!)
So, we make variables for it!!!

Z_C : $= 1$ if C is sat

0 o/w.

We can now write the constraints in terms of Z_C

If C is satisfied:

1-clause, x_i: $y_i = Z_C$

$\overline{x_i}$ $(1-y_i) = Z_C$

2-clause: $x_i \vee x_j$ $y_i + y_j \geq Z_C$

$\overline{x_i} \vee x_j$ $(1-y_i) + y_j \geq Z_C$

⋮ ⋮

— (2)

Amazing thing: If C is not sat ($z_C = 0$), the above constraints are always satisfied by any $y \in [0,1]^n$.

So we can put the above constraints for every C.

But we now need to relax z_C to be in an interval too.

$$0 \leq z_C \leq 1 \quad \forall C. \quad \text{—(3)}$$

Finally we want to maximize the # of sat clauses:

$$\max \sum_C z_C. \quad \text{— (4)}$$

So our final LP

maximize (4)

s.t. constraints (1), (2), (3) are satisfied.

Lemma: Let x^* be an assignment that satisfies m' clauses. Then there are $y_1, \ldots, y_n, \{z_c\}_{c \in [m]}$ s.t. y, z satisfy (1),(2),(3) & $\sum_c z_c = m'$.

Proof: we give values to y & z using x^*.

$$y_i = x_i^* \quad \forall\ i$$

$z_c = 1$ if x^* satisfies clause c

easy to check $\sum_c z_c = m'$.

$0 \le y_i \le 1, 0 \le z_c \le 1$

for each clause, we can verify that constraints (2) hold.

↳ "feasibility" □

Corollary:

LP-value ≥ OPT

this gives us our certificate

Rounding → Randomized Rounding

Interpret y_i in $[0,1]$ to be probabilities.

Rounded truth assignment:

$\forall i \quad x_i^* = \begin{cases} 1 & \text{w.p. } y_i \\ 0 & \text{w.p. } 1-y_i \end{cases}$

independently

This is a randomized alg.

Let X be the random variable that counts the number of satisfied clauses by x^*.

As before $X = \sum_C X_C$

X_C: indicator of whether C is satisfied

Our goal:

Lemma: $\mathbb{E}[X] \geq \frac{3}{4} \text{LP-OPT}$

Idea: Let (y^*, z^*) be an optimal LP solution:

$$\text{LP-OPT} = \sum_C z_C^*$$

We will prove for every C

Claim: $\mathbb{E}[X_C] \geq \frac{3}{4} z_C^*$

We are then done by linearity of expectation.

Proof of Claim:

Let C be a 1-clause.

Say $C = x_i$.

Then, $\Pr[X_C = 1] = y_i^*$

So $\mathbb{E}[X_C] = y_i^*$.

On the other hand, we know LP solution satisfies

$$y_i^* \geq z_C^*.$$

So $\mathbb{E}[X_C] = z_C^* \geq \frac{3}{4} z_C^*$

done on this case.

The case of $C = \bar{x}_i$ is similar.

Next, more interestingly, let C be a 2-clause.

4-cases

$C = x_i \vee x_j$ (say)

$$\Pr[C \text{ is sat}]$$

$$= \Pr[x_i^* = 1 \text{ or } x_j^* = 1]$$

$$= 1 - \Pr[x_i^* = 0 \text{ and } x_j^* = 0]$$

$$= 1 - \Pr[x_i^* = 0] \cdot \Pr[x_j^* = 0]$$

→ independence of x_i^* & x_j^*

$$= 1 - (1 - y_i^*) \cdot (1 - y_j^*)$$

$$= y_i^* + y_j^* - y_i^* \cdot y_j^* \quad \text{— (*)}$$

We need to prove that (*) is
- least $\frac{3}{4} Z_c^*$

from LP constraints, we have

$$y_i^* + y_j^* \geq Z_c^*$$

Obs: $Z_c^* = \min\{1, y_i^* + y_j^*\}$ since otherwise increasing Z_c^* raises the LP value while satisfying all constraints

(*) subtracts something from $y_i^* + y_j^*$.
how large can this subtracted off be?

How large can this subtracted quantity be?

Lemma: for any a, b: we have $a \cdot b \leq \frac{(a+b)^2}{4}$

Proof: this is the "AM-GM" inequality but has a simple proof. Observe

$$4ab = (a+b)^2 - (a-b)^2$$

So: $$ab = \frac{1}{4}(a+b)^2 - \frac{1}{4}(a-b)^2$$

$$\leq \frac{1}{4}(a+b)^2 \text{ since}$$

2^{nd} term is ≥ 0.

So the subtracted off quantity

$$y_i^* y_j^* \leq \frac{(y_i^* + y_j^*)}{4}.$$

So: $\Pr[X_C = 1] = y_i^* + y_j^* - y_i^* y_j^*$

$$\geq y_i^* + y_j^* - \frac{(y_i^* + y_j^*)^2}{4}$$

Now: 2-cases

1) $y_i^* + y_j^* \leq 1$.

In this case $z_C^* = y_i^* + y_j^*$.

$$\Pr[X_C = 1] \geq y_i^* + y_j^* - \frac{(y_i^* + y_j^*)^2}{4}$$

$$= z_C^* - z_C^*/4$$

square of a <1 number is tinier

$\geq z_c^* - z_c^*/4$

$= \frac{3}{4} z_c^*$

2) $y_i^* + y_j^* > 1$

In this case $z_c^* = 1$.

& $(y_i^* + y_j^*) - \frac{(y_i^* + y_j^*)^2}{4}$

$\geq \min_{a \in [1,2]} a - a^2/4$

at $a=1$: $3/4$.

$a=2$: 1.

derivative: $1 - a/2 \geq 0$ if $a \in [1,2]$

So function $a - a^2/4$ is increasing on $[1,2]$.

Thus $a - a^2/4$ over $[1,2]$ is minimized at $a = 1$ & equals $3/4$.

Thus in this case: $Z_C^* = 1$

$$\Pr[X_C = 1] \geq 3/4$$

□