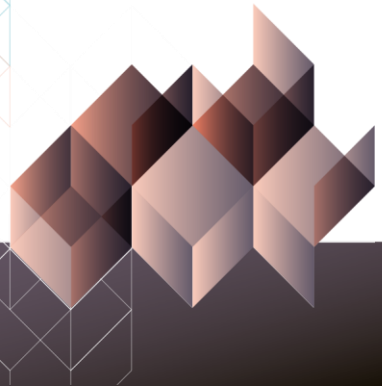
HØGSKOLEN
I BERGEN

BERGEN UNIVERSITY COLLEGE

Random Sampling and Randomized Rounding of LPs

Chapter 5



General

- Sometimes helpful that algorithms make random choices (Quicksort is an example)
  - Flip a coin
  - Flip a biased coin
  - Draw a value uniformly from a given interval

Performance guarantee will be the expected value of the solution produced relative to the value of an optimal solution

Derandomization

HØGSKOLEN I BERGEN
BERGEN UNIVERSITY COLLEGE

- Remove random choices

Get a deterministic version of the algorithm with same performance guarantee

Randomization gains us simplicity in our algorithm design and analysis

MAX SAT

- n boolean variables, $x_1, \ldots, x_n$ (true/false)
- m clauses, $C_1, \ldots, C_m$ of some number of variables and their negations
  - Example; $x_3 \vee \overline{x_5} \vee x_{11}$
- Nonnegative weight w_j for each clause C_j

- $(x_1 \vee \overline{x_5} \vee x_{11}), (x_1 \vee \overline{x_7}), \ldots$

C_1 C_2

MAX SAT

- Goal: Find an assignment of true/false to the variables that maximizes the weight of the satisfied clauses

- Terminology
  - A variable x_i or a negated variable $\overline{x_i}$ is a literal
  - A variable x_i is called a positive literal
  - A variable $\overline{x_i}$ is called a negative literal
  - Number of literals in a clause is its length or size
  - Denote length of C_j by l_j
  - Clauses of length one is called unit clauses

Example

- $C_1 = 4(x_1), C_2 = 3(\overline{x_1} \vee x_2),$
  $C_3 = 2(\overline{x_1} \vee x_3), C_4 = 1(\overline{x_2} \vee \overline{x_3})$

x_1	x_2	x_3	C_1	C_2	C_3	C_4	Total
0	0	0					
0	0	1					
0	1	0					
0	1	1					
1	0	0					
1	0	1					
1	1	0					
1	1	1					

Example, filled in

- $C_1 = 4(x_1), C_2 = 3(\overline{x_1} \vee x_2),$
  $C_3 = 2(\overline{x_1} \vee x_3), C_4 = 1(\overline{x_2} \vee \overline{x_3})$

x_1	x_2	x_3	C_1	C_2	C_3	C_4	Total
0	0	0	0	3	2	1	6
0	0	1	0	3	2	1	6
0	1	0	0	3	2	1	6
0	1	1	0	3	2	0	5
1	0	0	4	0	0	1	5
1	0	1	4	0	2	1	7
1	1	0	4	3	0	1	8
1	1	1	4	3	2	0	9

Best

Assumptions

- No literals are repeated in a clause (does not affect the satisfiability of the clause)
- A most one of x_i or $\overline{x_i}$ appears in a clause (otherwise it is trivially satisfied)
- All clause are distinct (can sum weight of identical clauses)

Algorithm

- Set each x_i to true independently with probability 1/2

Theorem 5.1

- Setting each x_i to true with probability ½ independently gives a randomized ½-approximation algorithm for the maximum satisfiability problem

Proof

- Random variables
  - $Y_j = \begin{cases} 1 \; if \; C_j \; is \; satisfied \\ 0 \qquad\quad otherwise \end{cases}$
  - $W = \sum_{j=1}^{m} w_j Y_j$
- Expectation
  - $E[W] = \sum_{j=1}^{m} w_j E[Y_j] = \sum_{j=1}^{m} w_j \Pr[clause \; C_j \; satisfied]$
- Probability that C_j is not satisfied (every positive literal set to false and every negative literal is set to true)

$$Pr[clause \; C_j \; satisfied] = \left(1 - \left(\frac{1}{2}\right)^{l_j}\right) \geq \frac{1}{2}$$

- Hence $E[W] \geq \frac{1}{2}\sum_{j=1}^{m} w_j \geq \frac{1}{2} OPT$
  Upper limit on OPT is that all clauses are satisfied

Observation

- If $l_j \geq k$ for all clauses, then the algorithm is a $\left(1 - \left(\frac{1}{2}\right)^k\right)$ –approximation algorithm for such instances
- Performance guarantee is better on instances consisting of long clauses

- If $l_j = 3$ for all clauses we have MAX E3SAT and we have performance guarantee of 7/8

Theorem 5.2

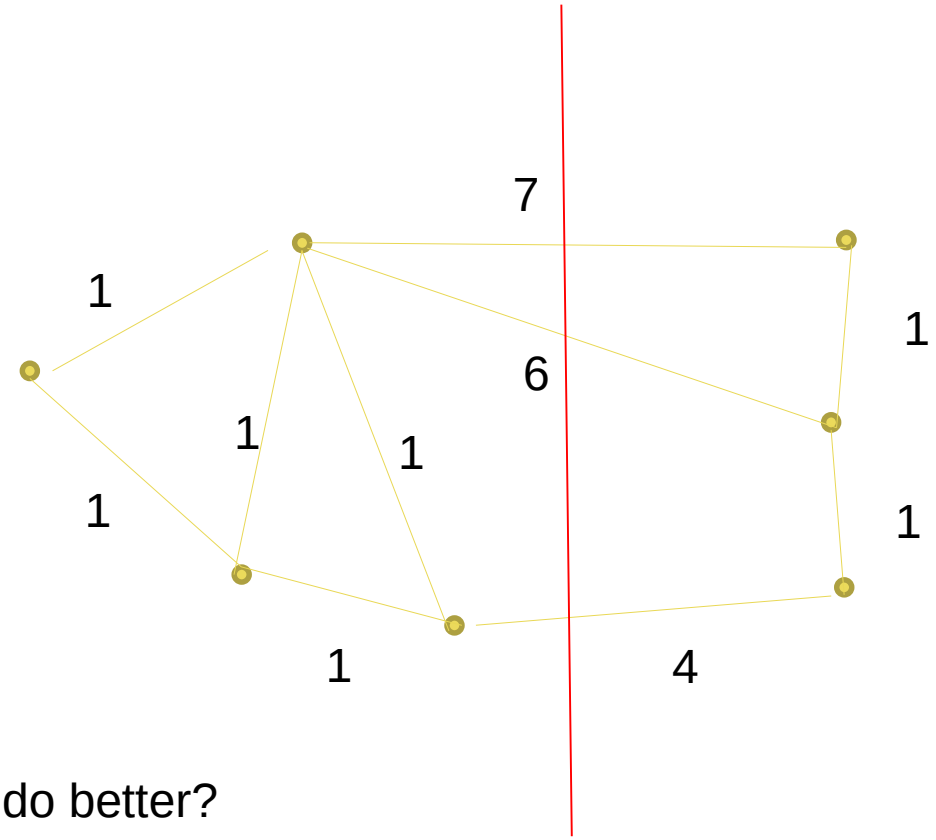
- If there is a $\left(\frac{7}{8}+\varepsilon\right)$-approximation algorithm for MAX E3SAT for any constant $\epsilon > 0$, then P = NP

MAX CUT

- Undirected graph $G = (V, E)$
- Nonnegative weight $w_{ij} > 0$ for each edge $(i, j) \in E$

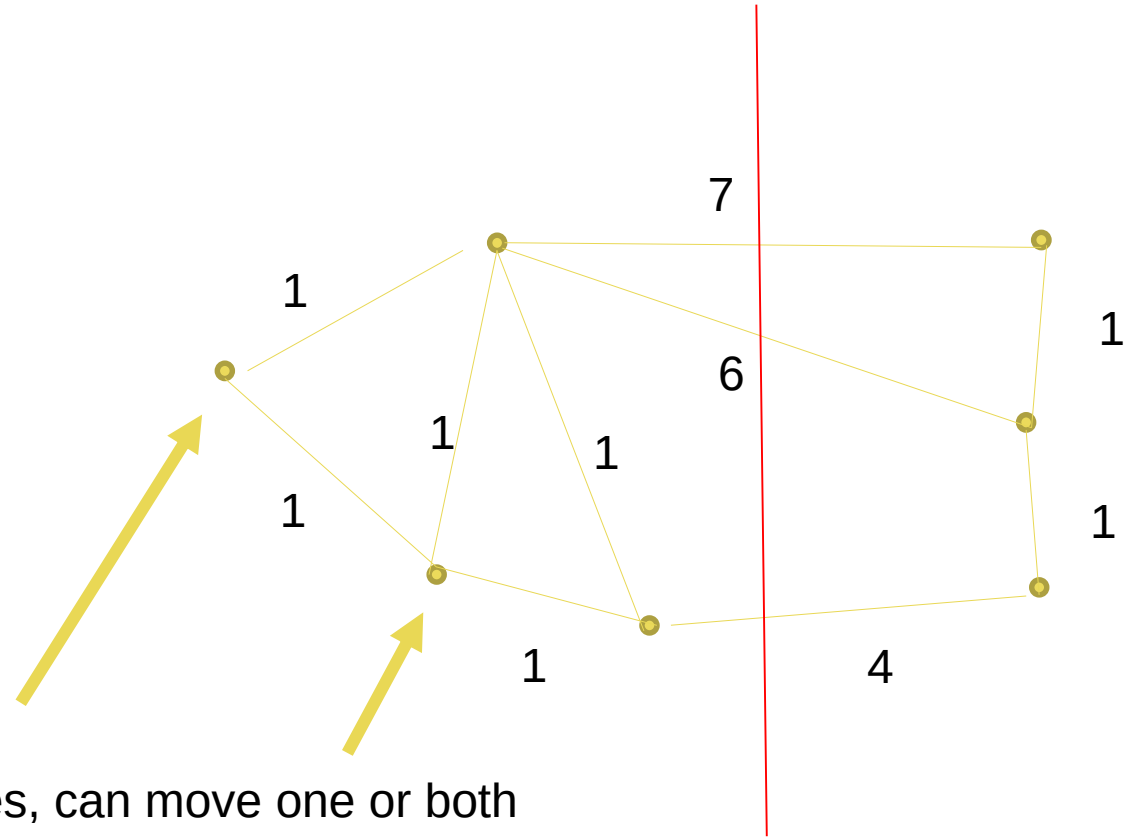
- Goal: Partition the vertices in to parts U and $W = V - U$ so as to maximize the weight of the edges where the two endpoints are in different parts.

Example



Possible to do better?

Example



Yes, can move one or both (best) of the vertices to the right side

Algorithm

- Place each vertex $v \in V$ into U independently with probability ½

Theorem 5.3

- If we place each vertex $v \in V$ into U independently with probability ½ , then we obtain a ½ -approximation algorithm for the maximum cut problem

Derandomization

- Obtain a deterministic algorithm whose solution value is as good as the expected value of the randomized version

Derandomization, MAX SAT

HØGSKOLEN I BERGEN
BERGEN UNIVERSITY COLLEGE

- Want to set x_1 and preserve the expected value of the algorithm
  - The best way will maximize the expected value of the resulting solution
- Formally, if $E[W|x_1 \leftarrow true] \geq E[W|x_1 \leftarrow false]$, we set x_1 true and false otherwise
- $E[W] = E[W|x_1 \leftarrow true]\Pr[x_1 \leftarrow true] + E[W|x_1 \leftarrow false]\Pr[x_1 \leftarrow false] = \frac{1}{2}(E[W|x_1 \leftarrow true + E[W|x_1 \leftarrow false])$

- Assume we have set variables $x_1, \ldots, x_i$ to truth values $b_1, \ldots, b_i$
- How shall we set variable x_{i+1}?
- Same way such that we maximize the expected value given the previous settings

- $E[W|x_1 \leftarrow b_1, \ldots, x_i \leftarrow b_i]$

$$= \sum_{j=1}^{m} w_j E\left[Y_j | x_1 \leftarrow b_1, \ldots, x_i \leftarrow b_i\right]$$

$$= \sum_{j=1}^{m} w_j Pr\left[clause\ C_j\ satisfied | x_1 \leftarrow b_1, \ldots, x_i \leftarrow b_i\right]$$

- Probability that clause C_j is satisfied given that $x_1 \leftarrow b_1, \ldots, x_i \leftarrow b_i$
  - 1 if C_j is already satisfied by $x_1 \leftarrow b_1, \ldots, x_i \leftarrow b_i$
  - $1 - (\frac{1}{2})^k$ where k is the number of literals in the clause that remains unset.
- Example: $x_3 \vee \overline{x_5} \vee x_{11}$
  - $\Pr(clause\ satisfied \mid x_1 \leftarrow true, x_2 \leftarrow false, x_3 \leftarrow true) = 1$
  - $\Pr(clause\ satisfied \mid x_1 \leftarrow true, x_2 \leftarrow false, x_3 \leftarrow false) = 1 - \left(\frac{1}{2}\right)^2 = \frac{3}{4}$ (We know that x_3 is false, but there are still two possibilities to satisfy the clause ($x_5 \leftarrow false$ or $x_{11} \leftarrow true$)

Flipping Biased Coins

- Will set p_i true with some probability $p \neq \frac{1}{2}$.
- First assume we have no unit clauses $\overline{x_i}$
- Set each to true independently with probability $p > \frac{1}{2}$.

Lemma 5.4

- If each x_i is set to true with probability $p > \frac{1}{2}$ independently, then the probability that any given clause is satisfied is at least $\min(p, 1 - p^2)$ for MAX SAT instances with no negated unit clauses.
- Proof
  Positive unit clause satisfied with probability *p*
  Clause with length at least two, satisfied with probability $1 - p^a(1 - p^b)$ where a is the number of negated variables and b is the number of unnegated variables. Since $p > \frac{1}{2} > 1 - p$ *and* $a + b \geq 2$, *we have the probability for the clause is true is at least* $1 - p^2$ *in this case*

Allowing unit negated clauses

- Get a better bound on OPT than $\sum_{i=1}^{m} w_j$
  - Unit clauses x_i and $\overline{x_i}$ can not both be true
  - Assume that for every i the weight of the unit clause x_i is at least the weight of the unit clause $\overline{x_i}$ (if not we could negate all occurrences of x_i)
  - Let v_i be the weight of the unit clause $\overline{x_i}$ if exists in the instance and 0 otherwise
- Then $OPT \leq \sum_{j=1}^{m} w_j - \sum_{j=1}^{n} v_i$

Theorem 5.7

- We can obtain a randomized $\frac{1}{2}(\sqrt{5}-1)$-approximation algorithm for MAX SAT

- Comment
  - $\frac{1}{2}(\sqrt{5}-1) \approx 0.62$ compared to 0.50 with $p = \frac{1}{2}$

Randomized Rounding

HØGSKOLEN I BERGEN
BERGEN UNIVERSITY COLLEGE

1. Set up an integer programming formulation with 0-1 integer variables
2. Set up the corresponding LP relaxation, which can be solved in poly time
3. Interpret the fractional parts of decision variables as a probability to round the variable up

MAX SAT

- Integer programming formulation
  - Variables y_i for each variable x_i, where $y_i = 1$ means $x_i = true$, 0 otherwise
  - Variables z_j for each clause C_j, where $z_j = 1$ means C_j is satisfied, 0 otherwise
  - For each clause C_j, let P_j be the indices of the variables x_i that occur positively in the clause and let N_j be the indices of the variables x_i that occur negated in the clause

Integer Program

- maximize $\sum_{j=1}^{m} w_j z_j$
- subject to
- $\sum_{i \in P_j} y_i + \sum_{i \in N_j} (1 - y_i) \geq z_j, \quad \forall C_j = \bigvee_{i \in P_j} x_i \vee \bigvee_{i \in N_j} \overline{x_i}\,,$

$$y_i \in \{0,1\}, \quad i = 1, \dots n,$$

$$z_j \in \{0,1\}, \quad j = 1, \dots, m$$

Example

- Find the IP formulation of the following MAX SAT problem

  $C_1 = 4(x_1), C_2 = 3(\overline{x_1} \vee x_2), C_3 = 2(\overline{x_1} \vee x_3), C_4 = 1(\overline{x_2} \vee \overline{x_3})$

  Solution next slide

IP formulation

- $C_1 = 4(x_1), C_2 = 3(\overline{x_1} \vee x_2), C_3 = 2(\overline{x_1} \vee x_3), C_4 = 1(\overline{x_2} \vee \overline{x_3})$
- maximize $4z_1 + 3z_2 + 2z_3 + z_4$
- subject to
  - $y_1 \geq z_1 \quad \rightarrow y_1 - z_1 \geq 0$
  - $y_2 + (1 - y_1) \geq z_2 \quad \rightarrow -y_1 + y_2 - z_2 \geq -1$
  - $y_3 + (1 - y_1) \geq z_3 \quad \rightarrow -y_1 + y_3 - z_3 \geq -1$
  - $(1 - y_2) + (1 - y_3) \geq z_4 \quad \rightarrow -y_2 - y_3 - z_4 \geq -2$
  - $y_i \in \{0,1\}, i = 1, 2, 3$
  - $z_j \in \{0,1\}, j = 1, \ldots, 4$

LP Relaxation

- Maximize $\sum_{j=1}^{m} w_j z_j$
- Subject to
- $\sum_{i \in P_j} y_i + \sum_{i \in N_j} (1 - y_i) \geq z_j, j = 1, \ldots, m \; \forall C_j = \bigvee_{i \in P_j} x_i \vee \bigvee_{i \epsilon N_j} \overline{x_i}$

$$0 \leq y_i \leq 1, \qquad i = 1, \ldots, n$$
$$0 \leq z_j \leq 1, \qquad j = 1, \ldots, m$$

Setting x_i

- If Z^*_{LP} is the optimal value of the LP relaxation, then $Z^*_{LP} \geq Z^*_{IP} = OPT$
- Let (y^*, z^*) be an optimal solution to LP relaxation
- Set x_i to true with probability y^*_i independently

Theorem 5.10

- Randomized rounding gives a randomized $(1 - \frac{1}{e})$-approximation algorithm for MAX SAT

- Comment: $(1 - \frac{1}{e}) \approx 0.63$

Matlab

- Show how we solve LP in Matlab
- From documentation

Equation

Finds the minimum of a problem specified by

$$\min_x f^T x \text{ such that } \begin{array}{r} A \cdot x \le b, \\ Aeq \cdot x = beq, \\ lb \le x \le ub. \end{array}$$

Need to fit our problem to this form

f, x, b, beq, lb, and ub are vectors, and A and Aeq are matrices.

Syntax

```
x = linprog(f,A,b)
x = linprog(f,A,b,Aeq,beq)
x = linprog(f,A,b,Aeq,beq,lb,ub)
```

Max ↔ Min

- Change sign for coefficients in objective function
- Example
  - maximize $3x_1 - 2x_2 + 4x_3$

  - minimize $-3x_1 + 2x_2 - 4x_3$
  - (Can go from min to max the same way)

$\geq \quad \leftrightarrow \quad \leq$

- Multiply inequalities by -1
- Example
  - $-3x_1 + 4x_2 \geq -12$

  - $3x_1 - 4x_2 \leq 12$

Example Matlab

- Example

maximize $-3x_1 - 4x_2$
subject to
$-3x_1 + 4x_2 \geq -12$
$-x_1 - 2x_2 \leq -4$
$x_1 \geq 1$
$x_2 \geq 0$

Allready in right form

Bounds are ok

minimize $3x_1 + 4x_2$
subject to
$3x_1 - 4x_2 \leq 12$
$-x_1 - 2x_2 \leq -4$
$x_1 \geq 1$
$x_2 \geq 0$

Solve MAX SAT with Matlab

HØGSKOLEN I BERGEN
BERGEN UNIVERSITY COLLEGE

- $4(x_1) \wedge 3(\overline{x_1} \vee x_2) \wedge 2(\overline{x_1} \vee x_3) \wedge 1(\overline{x_2} \vee \overline{x_3})$
- IP
- maximize $4z_1 + 3z_2 + 2z_3 + z_4$
- subject to
  - $y_1 \geq z_1 \quad \rightarrow y_1 - z_1 \geq 0$
  - $y_2 + (1 - y_1) \geq z_2 \quad \rightarrow -y_1 + y_2 - z_2 \geq -1$
  - $y_3 + (1 - y_1) \geq z_3 \quad \rightarrow -y_1 + y_3 - z_3 \geq -1$
  - $(1 - y_2) + (1 - y_3) \geq z_4 \quad \rightarrow -y_2 - y_3 - z_4 \geq -2$
  - $y_i \in \{0,1\}, i = 1,2,3$
  - $z_j \in \{0,1\}, j = 1, \ldots, 4$

Chose Better of two solutions

- Observation
  - Randomized rounding. Satisfies a clause with probability
    - $\left[1-\left(1-\frac{1}{l_j}\right)^{l_j}\right] z_j^*$
    - Best with short clauses
  - Unbiased randomized algorithm. Satisfies a clause with probability
    - $1-2^{-l_j} \geq (1-2^{-l_j}) z_j^*$

Theorem 5.11

HØGSKOLEN I BERGEN
BERGEN UNIVERSITY COLLEGE

- Choosing the better of the two solutions given by the randomized rounding algorithm and the unbiased randomized algorithm yields a randomized ¾-approximation algorithm for MAX SAT

- Comments
  - Randomized rounding algorithm: ≈ 0.63
  - Unbiased randomized algorithm: $= 0.50$

Not in Curriculum

- Chapter 5.6, 5.7, 5.8 and 5.9

Chapter 5.10

HØGSKOLEN I BERGEN
BERGEN UNIVERSITY COLLEGE

- Chernoff Bounds
  - Important theorem for analyzing randomized rounding algorithms
  - Theorem says
    It is very likely that the sum of n independent 0-1 variables is not far away from the expected value of the sum.

More formally

- Theorem 5.23. Let $X_1, \ldots, X_n$ be n independent 0-1 random variables, not necessarily identically distributed. Then for $X = \sum_{i=1}^{n} X_i$ and $\mu = E[X], L \leq \mu \leq U$, and $\delta > 0$,

$$\Pr[X \geq (1+\delta)U] < \left(\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right)^{U} \text{ and}$$

$$\Pr[X \leq (1-\delta)L] < \left(\frac{e^{-\delta}}{(1-\delta)^{(1-\delta)}}\right)^{L}$$

Chapter 5.11

- Not in curriculum

Color Dense 3-Colorable Graphs

- A graph is dense if for some constant α the number of edges in the graph is at least $\alpha \binom{n}{2}$

  Comment: $\binom{n}{2}$ is the maximum number of possible edges in an undirected graph.
- In a δ-dense graph every node has at least δn neighbours
- A graph is k-colorable if each of the nodes can be assigned exactly one of k colors in such a way that no edge has its two endpoints assigned the same color

- NP-complete to decide whether a graph is 3-colorable

Theorem 5.31

- It is NP-hard to decide if a graph can be colored with only three colors, or needs at least five colors.

Algorithm for 3-colorable graphs

HØGSKOLEN I BERGEN
BERGEN UNIVERSITY COLLEGE

- Will with high probability color any δ-dense 3-colorable graph in poly time
- Randomized algorithm, but no approximation algorithm
- Analysis is a useful application of Chernoff bounds

Ideas

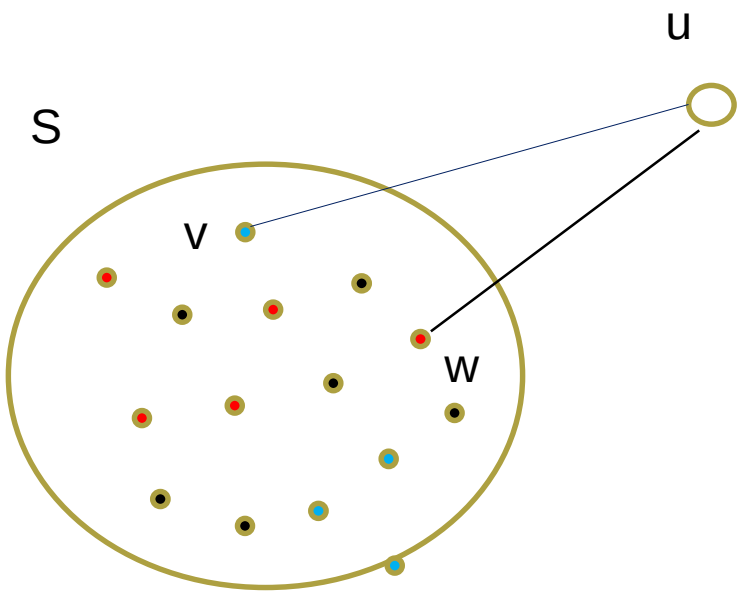
- Find a “small” random sample of the vertices
- For each correct coloring of the sample
  - Try to extend it to a correct coloring for the whole graph

Comment: If the graph is 3-colorable, it must be possible to extend at least one of the colorings of the sample to a coloring of the whole graph

Algorithm

- Select a random subset $S \subseteq V$ of $O((\ln n)/\delta)$ vertices by including each vertex in the subset with probability $(3c \ln n)/(\delta n)$ for some constant c.
  - Comment: Possible to show with high probability that
    - S is small
    - every vertex has at least one neighbor in S
- Since S is small, it is possible to enumerate all correct colorings of S in poly time. Try to extend them one by one into a coloring for the whole graph (possible to do in poly time because every vertex has a neighbor in S)

Why does it work?



If we find a vertex u with two neighbors v and w in S of different colors, then the color of u is determined and we can add u to S.
(If we find neighbors with three different colors, it is impossible to color u)

Why does it work? (2)

HØGSKOLEN I BERGEN
BERGEN UNIVERSITY COLLEGE

- All neighbors of u in S have the same color
  - Try one of the two available colors on u and add u to S. New vertices can be colored according to the rule on the previous slide
  - Backtrack if necessary

Why does it work? (3)

HØGSKOLEN I BERGEN
BERGEN UNIVERSITY COLLEGE

- The book explains it more formally by defining a MAX SAT problem