

Randomised Algorithms

Lecture 9: Approximation Algorithms: MAX-3-CNF and Vertex-Cover

Thomas Sauerwald (tms41@cam.ac.uk)

Lent 2025



UNIVERSITY OF
CAMBRIDGE

Randomised Approximation

MAX-3-CNF

Weighted Vertex Cover

Approximation Ratio for Randomised Approximation Algorithms

Approximation Ratio

A **randomised** algorithm for a problem has **approximation ratio** $\rho(n)$, if for any input of size n , the **expected** cost (value) $\mathbf{E}[C]$ of the returned solution and optimal cost C^* satisfy:

$$\max\left(\frac{\mathbf{E}[C]}{C^*}, \frac{C^*}{\mathbf{E}[C]}\right) \leq \rho(n).$$

not covered here (non-examinable)

Randomised Approximation Schemes

An **approximation scheme** is an approximation algorithm, which given any input and $\epsilon > 0$, is a $(1 + \epsilon)$ -approximation algorithm.

- It is a **polynomial-time approximation scheme** (PTAS) if for any fixed $\epsilon > 0$, the runtime is polynomial in n . For example, $O(n^{2/\epsilon})$.
- It is a **fully polynomial-time approximation scheme** (FPTAS) if the runtime is polynomial in both $1/\epsilon$ and n . For example, $O((1/\epsilon)^2 \cdot n^3)$.

Randomised Approximation

MAX-3-CNF

Weighted Vertex Cover

MAX-3-CNF Satisfiability

Assume that no literal (including its negation) appears more than once in the same clause.

MAX-3-CNF Satisfiability

- **Given:** 3-CNF formula, e.g.: $(x_1 \vee x_3 \vee \bar{x}_4) \wedge (x_2 \vee \bar{x}_3 \vee \bar{x}_5) \wedge \dots$
- **Goal:** Find an assignment of the variables that satisfies as many clauses as possible.

Relaxation of the **satisfiability** problem. Want to compute how “close” the formula to being satisfiable is.

Example:

$$(x_1 \vee x_3 \vee \bar{x}_4) \wedge (x_1 \vee \bar{x}_3 \vee \bar{x}_5) \wedge (x_2 \vee \bar{x}_4 \vee x_5) \wedge (\bar{x}_1 \vee x_2 \vee \bar{x}_3)$$

$x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 0$ and $x_5 = 1$ satisfies 3 (out of 4 clauses)

Idea: What about assigning each variable uniformly and independently at random?

Analysis

Theorem 35.6

Given an instance of MAX-3-CNF with n variables x_1, x_2, \dots, x_n and m clauses, the randomised algorithm that sets each variable independently at random is a **randomised $8/7$ -approximation algorithm**.

Proof:

- For every clause $i = 1, 2, \dots, m$, define a **random variable**:

$$Y_i = \mathbf{1}\{\text{clause } i \text{ is satisfied}\}$$

- Since each literal (including its negation) appears at most once in clause i ,

$$\mathbf{P}[\text{clause } i \text{ is not satisfied}] = \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{8}$$

$$\Rightarrow \mathbf{P}[\text{clause } i \text{ is satisfied}] = 1 - \frac{1}{8} = \frac{7}{8}$$

$$\Rightarrow \mathbf{E}[Y_i] = \mathbf{P}[Y_i = 1] \cdot 1 = \frac{7}{8}.$$

- Let $Y := \sum_{i=1}^m Y_i$ be the number of satisfied clauses. Then,

$$\mathbf{E}[Y] = \mathbf{E}\left[\sum_{i=1}^m Y_i\right] = \sum_{i=1}^m \mathbf{E}[Y_i] = \sum_{i=1}^m \frac{7}{8} = \frac{7}{8} \cdot m. \quad \square$$

Linearity of Expectations

maximum number of satisfiable clauses is m

Interesting Implications

Theorem 35.6

Given an instance of MAX-3-CNF with n variables x_1, x_2, \dots, x_n and m clauses, the randomised algorithm that sets each variable independently at random is a polynomial-time randomised $8/7$ -approximation algorithm.

Corollary

For any instance of MAX-3-CNF, there exists an assignment which satisfies at least $\frac{7}{8}$ of all clauses.

There is $\omega \in \Omega$ such that $Y(\omega) \geq \mathbf{E}[Y]$

Probabilistic Method: powerful tool to show existence of a non-obvious property.

Corollary

Any instance of MAX-3-CNF with at most 7 clauses is satisfiable.

Follows from the previous Corollary.

Expected Approximation Ratio

Theorem 35.6

Given an instance of MAX-3-CNF with n variables x_1, x_2, \dots, x_n and m clauses, the randomised algorithm that sets each variable independently at random is a polynomial-time randomised $8/7$ -approximation algorithm.

One could prove that the probability to satisfy $(7/8) \cdot m$ clauses is at least $1/(8m)$

$$\mathbf{E}[Y] = \frac{1}{2} \cdot \mathbf{E}[Y \mid x_1 = 1] + \frac{1}{2} \cdot \mathbf{E}[Y \mid x_1 = 0].$$

Y is defined as in the previous proof.

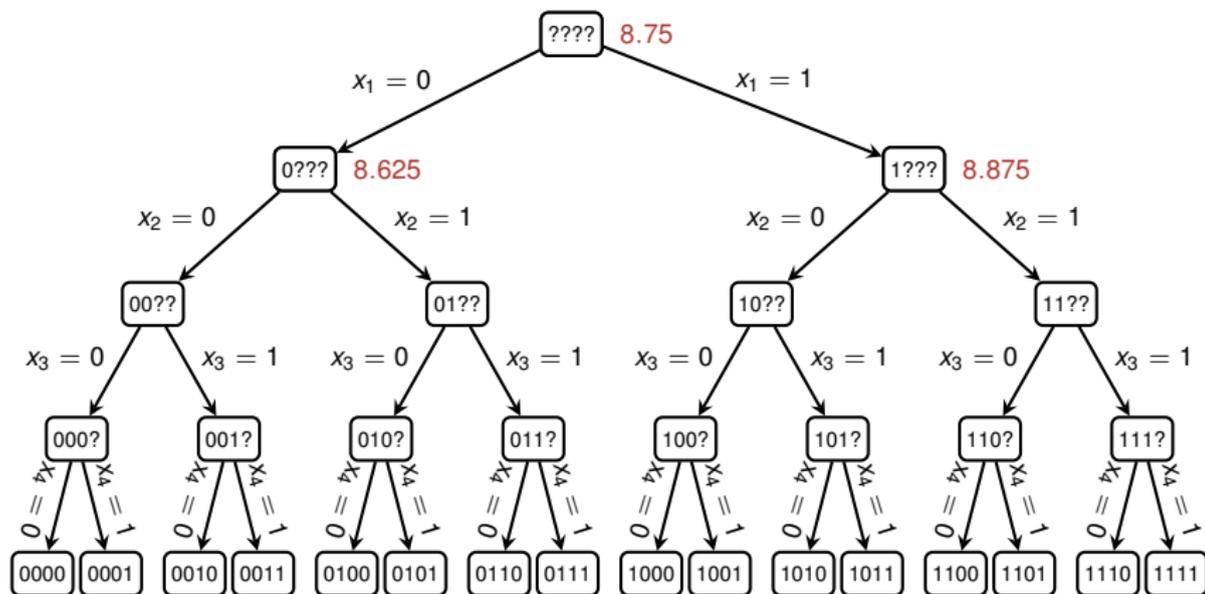
One of the two conditional expectations is at least $\mathbf{E}[Y]$

GREEDY-3-CNF(ϕ, n, m)

- 1: **for** $j = 1, 2, \dots, n$
- 2: Compute $\mathbf{E}[Y \mid x_1 = v_1, \dots, x_{j-1} = v_{j-1}, x_j = 1]$
- 3: Compute $\mathbf{E}[Y \mid x_1 = v_1, \dots, x_{j-1} = v_{j-1}, x_j = 0]$
- 4: Let $x_j = v_j$ so that the conditional expectation is maximised
- 5: **return** the assignment v_1, v_2, \dots, v_n

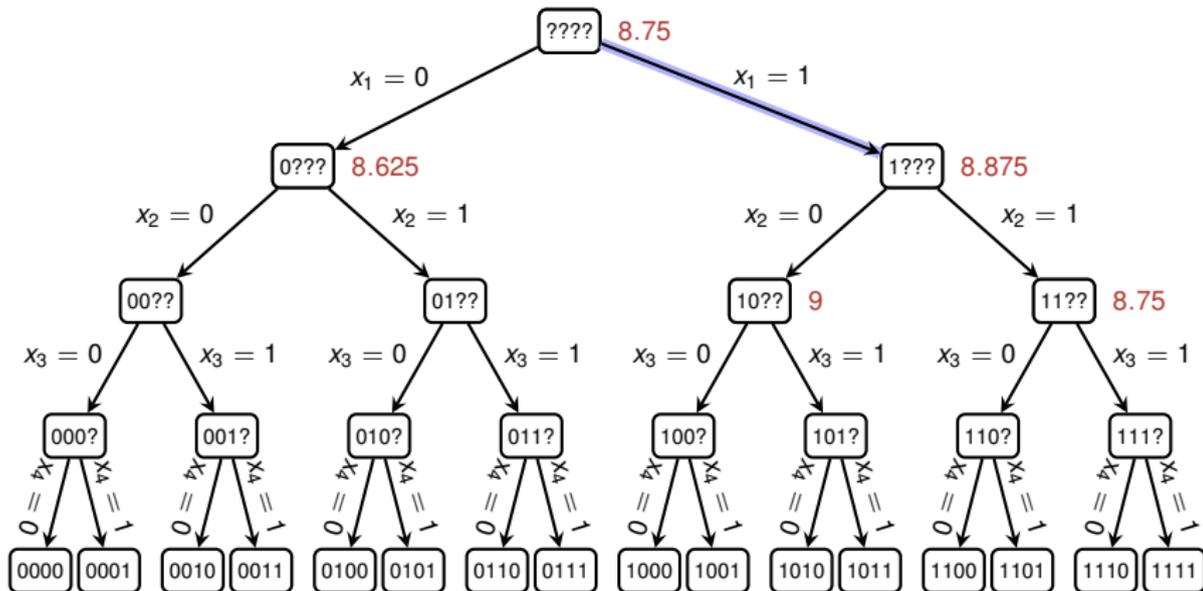
Run of GREEDY-3-CNF(φ, n, m)

$$(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \bar{x}_2 \vee \bar{x}_4) \wedge (x_1 \vee x_2 \vee \bar{x}_4) \wedge (\bar{x}_1 \vee \bar{x}_3 \vee x_4) \wedge (x_1 \vee x_2 \vee \bar{x}_4) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_3) \wedge (x_1 \vee x_3 \vee x_4) \wedge (x_2 \vee \bar{x}_3 \vee \bar{x}_4)$$



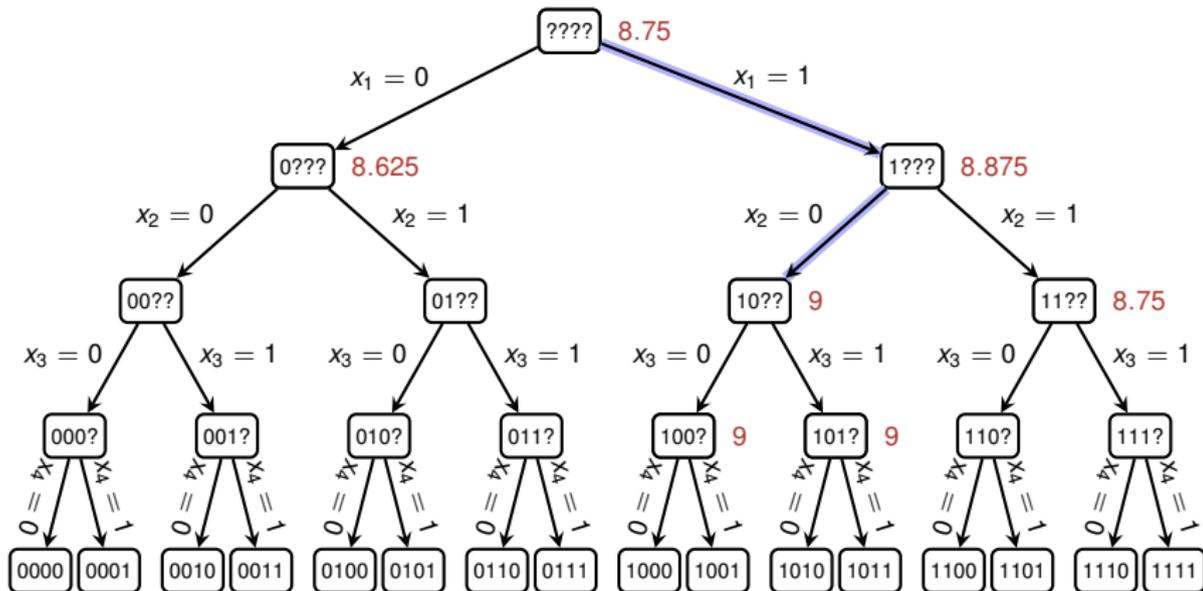
Run of GREEDY-3-CNF(φ, n, m)

$$1 \wedge 1 \wedge 1 \wedge (\overline{x_3} \vee x_4) \wedge 1 \wedge (\overline{x_2} \vee \overline{x_3}) \wedge (x_2 \vee x_3) \wedge (\overline{x_2} \vee x_3) \wedge 1 \wedge (x_2 \vee \overline{x_3} \vee \overline{x_4})$$



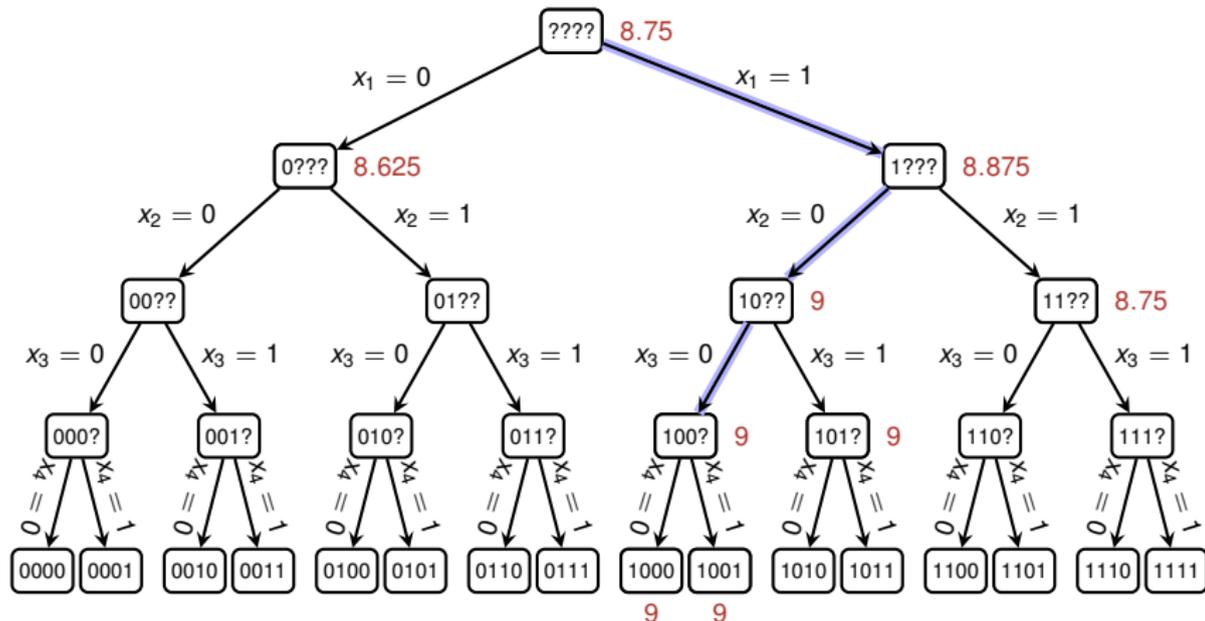
Run of GREEDY-3-CNF(φ, n, m)

$$1 \wedge 1 \wedge 1 \wedge (\overline{x_3} \vee x_4) \wedge 1 \wedge 1 \wedge (x_3) \wedge 1 \wedge 1 \wedge (\overline{x_3} \vee \overline{x_4})$$



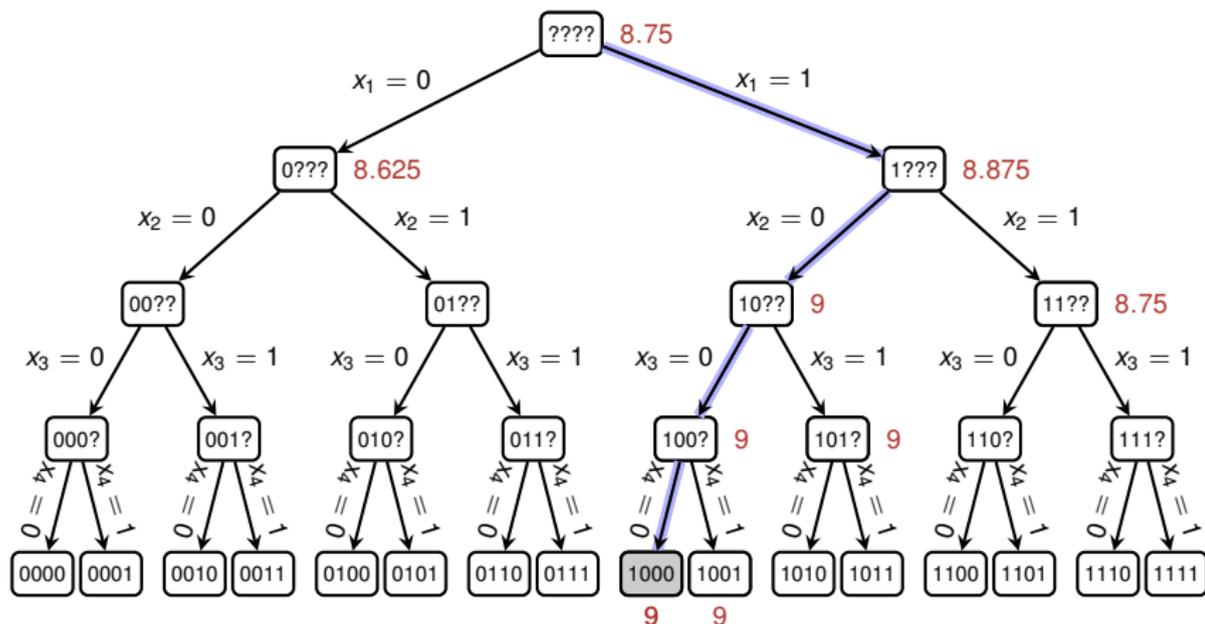
Run of GREEDY-3-CNF(φ, n, m)

$1 \wedge 1 \wedge 1 \wedge 1 \wedge 1 \wedge 1 \wedge 0 \wedge 1 \wedge 1 \wedge 1$



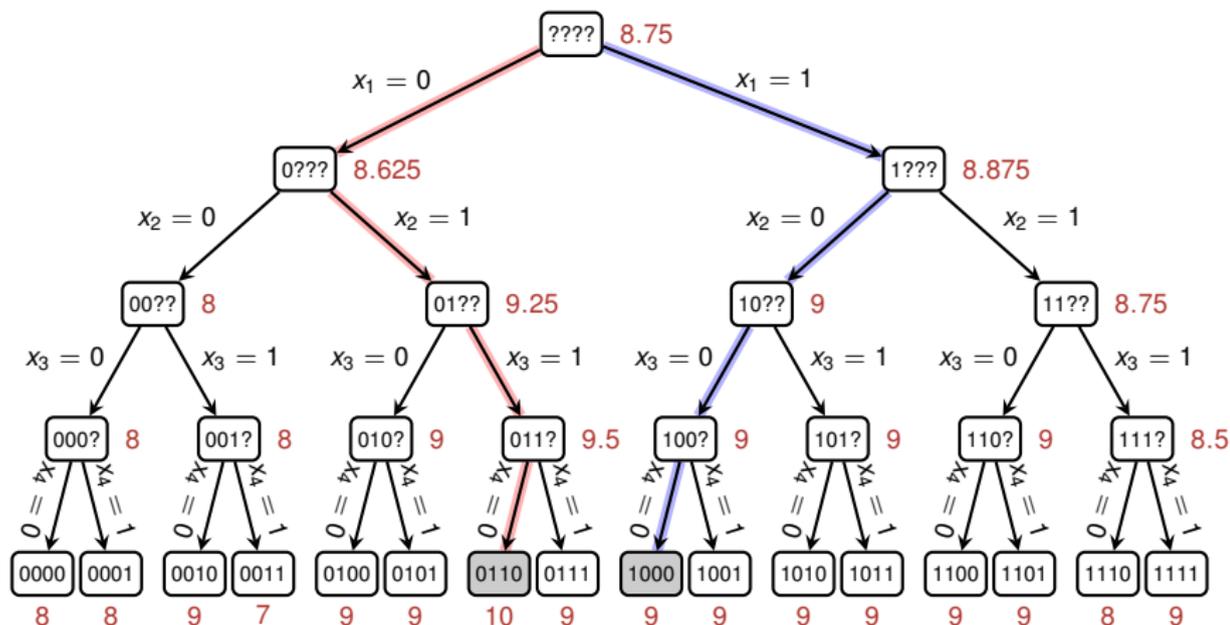
Run of GREEDY-3-CNF(φ, n, m)

$1 \wedge 1 \wedge 1 \wedge 1 \wedge 1 \wedge 1 \wedge 0 \wedge 1 \wedge 1 \wedge 1$



Run of GREEDY-3-CNF(φ, n, m)

$$(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \bar{x}_2 \vee \bar{x}_4) \wedge (x_1 \vee x_2 \vee \bar{x}_4) \wedge (\bar{x}_1 \vee \bar{x}_3 \vee x_4) \wedge (x_1 \vee x_2 \vee \bar{x}_4) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_3) \wedge (x_1 \vee x_3 \vee x_4) \wedge (x_2 \vee \bar{x}_3 \vee \bar{x}_4)$$



Returned solution satisfies 9 out of 10 clauses, but the formula is satisfiable.

This algorithm is deterministic.

Theorem

GREEDY-3-CNF(ϕ, n, m) is a polynomial-time $8/7$ -approximation.

Proof:

▪ **Step 1:** polynomial-time algorithm

- In iteration $j = 1, 2, \dots, n$, $Y = Y(\phi)$ averages over 2^{n-j+1} assignments
- A smarter way is to use **linearity of (conditional) expectations**:

$$\mathbf{E} [Y \mid x_1 = v_1, \dots, x_{j-1} = v_{j-1}, x_j = 1] = \sum_{i=1}^m \mathbf{E} [Y_i \mid x_1 = v_1, \dots, x_{j-1} = v_{j-1}, x_j = 1]$$

▪ **Step 2:** satisfies at least $7/8 \cdot m$ clauses

- Due to the greedy choice in each iteration $j = 1, 2, \dots, n$,

$$\begin{aligned} \mathbf{E} [Y \mid x_1 = v_1, \dots, x_{j-1} = v_{j-1}, x_j = v_j] &\geq \mathbf{E} [Y \mid x_1 = v_1, \dots, x_{j-1} = v_{j-1}] \\ &\geq \mathbf{E} [Y \mid x_1 = v_1, \dots, x_{j-2} = v_{j-2}] \\ &\vdots \\ &\geq \mathbf{E} [Y] = \frac{7}{8} \cdot m. \quad \square \end{aligned}$$

computable in $O(1)$

MAX-3-CNF: Concluding Remarks

— Theorem 35.6 —

Given an instance of MAX-3-CNF with n variables x_1, x_2, \dots, x_n and m clauses, the randomised algorithm that sets each variable independently at random is a **randomised $8/7$ -approximation algorithm**.

— Theorem —

GREEDY-3-CNF(ϕ, n, m) is a **deterministic poly-time $8/7$ -approxim.**

— Theorem (Hastad'97) —

For any $\epsilon > 0$, there is **no polynomial time $8/7 - \epsilon$ approximation algorithm** of MAX3-CNF unless P=NP.

Essentially there is nothing smarter than just guessing!

Yes, my research has finally concluded...

So you said you have been studying the field of algorithms for MAX-3-SAT?

...the best approach is to **randomly guess** a solution.



Source of Image: Stefan Szeider, TU Vienna

Outline

Randomised Approximation

MAX-3-CNF

Weighted Vertex Cover

The Weighted Vertex-Cover Problem

Vertex Cover Problem

- **Given:** Undirected, **vertex-weighted** graph $G = (V, E)$
- **Goal:** Find a **minimum-weight** subset $V' \subseteq V$ such that if $\{u, v\} \in E(G)$, then $u \in V'$ or $v \in V'$.

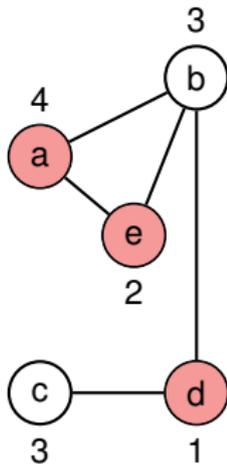
This is an NP-hard problem.



Question: How can we deal with graphs that have **negative** weights?

Applications:

- Every **edge** forms a **task**, and every **vertex** represents a **person/machine** which can execute that task
- **Weight** of a vertex could be **salary** of a person
- Perform all tasks with the **minimal amount of resources**



A Greedy Approach working for Unweighted Vertex Cover

APPROX-VERTEX-COVER(G)

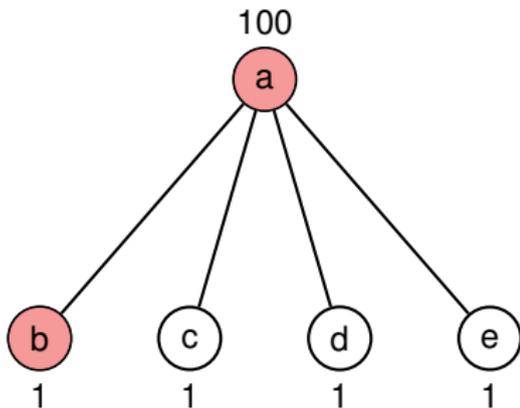
```
1  $C = \emptyset$ 
2  $E' = G.E$ 
3 while  $E' \neq \emptyset$ 
4     let  $(u, v)$  be an arbitrary edge of  $E'$ 
5      $C = C \cup \{u, v\}$ 
6     remove from  $E'$  every edge incident on either  $u$  or  $v$ 
7 return  $C$ 
```

This algorithm is a 2-approximation for **unweighted graphs!**

A Greedy Approach working for Unweighted Vertex Cover

APPROX-VERTEX-COVER(G)

- 1 $C = \emptyset$
- 2 $E' = G.E$
- 3 **while** $E' \neq \emptyset$
- 4 let (u, v) be an arbitrary edge of E'
- 5 $C = C \cup \{u, v\}$
- 6 remove from E' every edge incident on either u or v
- 7 **return** C

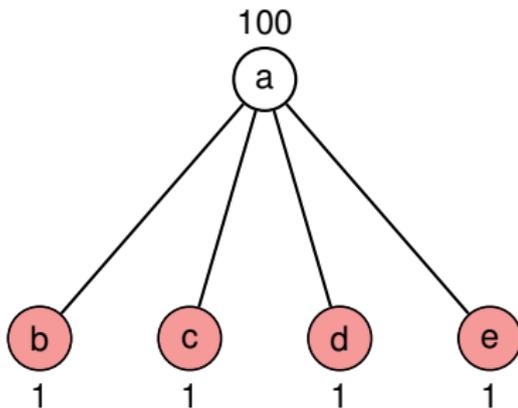


Computed solution has weight 101

A Greedy Approach working for Unweighted Vertex Cover

APPROX-VERTEX-COVER(G)

- 1 $C = \emptyset$
- 2 $E' = G.E$
- 3 **while** $E' \neq \emptyset$
- 4 let (u, v) be an arbitrary edge of E'
- 5 $C = C \cup \{u, v\}$
- 6 remove from E' every edge incident on either u or v
- 7 **return** C



Optimal solution has weight 4

Invoking an (Integer) Linear Program

Idea: Round the solution of an associated linear program.

0-1 Integer Program

$$\begin{array}{ll} \text{minimize} & \sum_{v \in V} w(v)x(v) \\ \text{subject to} & x(u) + x(v) \geq 1 \quad \text{for each } (u, v) \in E \\ & x(v) \in \{0, 1\} \quad \text{for each } v \in V \end{array}$$

optimum is a lower bound on the optimal weight of a minimum weight-cover.

Linear Program

$$\begin{array}{ll} \text{minimize} & \sum_{v \in V} w(v)x(v) \\ \text{subject to} & x(u) + x(v) \geq 1 \quad \text{for each } (u, v) \in E \\ & x(v) \in [0, 1] \quad \text{for each } v \in V \end{array}$$

Rounding Rule: if $x(v) \geq 1/2$ then round up, otherwise round down.

The Algorithm

APPROX-MIN-WEIGHT-VC(G, w)

```
1  $C = \emptyset$ 
2 compute  $\bar{x}$ , an optimal solution to the linear program
3 for each  $v \in V$ 
4     if  $\bar{x}(v) \geq 1/2$ 
5          $C = C \cup \{v\}$ 
6 return  $C$ 
```

Theorem 35.7

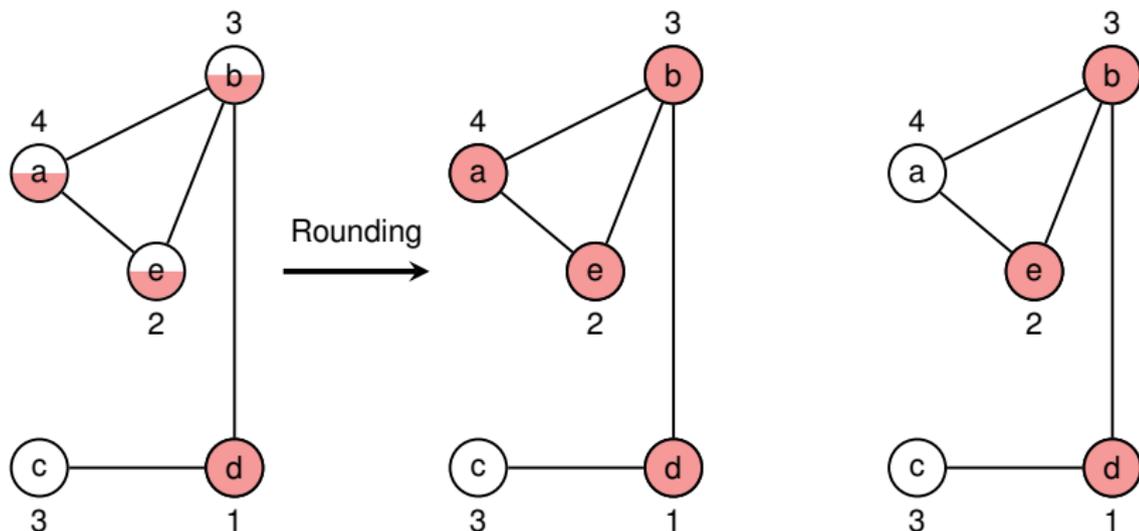
APPROX-MIN-WEIGHT-VC is a polynomial-time 2-approximation algorithm for the minimum-weight vertex-cover problem.

is polynomial-time because we can solve the linear program in polynomial time

Example of APPROX-MIN-WEIGHT-VC

$$\bar{x}(a) = \bar{x}(b) = \bar{x}(e) = \frac{1}{2}, \bar{x}(d) = 1, \bar{x}(c) = 0$$

$$x(a) = x(b) = x(e) = 1, x(d) = 1, x(c) = 0$$



fractional solution of LP
with weight = 5.5

rounded solution of LP
with weight = 10

optimal solution
with weight = 6

Approximation Ratio

Proof (Approximation Ratio is 2 and Correctness):

- Let C^* be an optimal solution to the minimum-weight vertex cover problem
- Let z^* be the value of an optimal solution to the linear program, so

$$z^* \leq w(C^*)$$

- Step 1:** The computed set C covers all vertices:
 - Consider any edge $(u, v) \in E$ which imposes the constraint $x(u) + x(v) \geq 1$
 \Rightarrow at least one of $\bar{x}(u)$ and $\bar{x}(v)$ is at least $1/2 \Rightarrow C$ covers edge (u, v)
- Step 2:** The computed set C satisfies $w(C) \leq 2z^*$:

$$w(C^*) \geq z^* = \sum_{v \in V} w(v) \bar{x}(v) \geq \sum_{v \in V: \bar{x}(v) \geq 1/2} w(v) \cdot \frac{1}{2} = \frac{1}{2} w(C). \quad \square$$

