Randomised Algorithms

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

Thomas Sauerwald (tms41@cam.ac.uk)

Lent 2024

UNIVERSITY OF CAMBRIDGE

Outline

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, $O((1/\epsilon)^2 \cdot n^3)$.

Outline

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.: $(x_1 \vee x_3 \vee \overline{x_4}) \wedge (x_2 \vee \overline{x_3} \vee \overline{x_5}) \wedge \cdots$
- 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:

$$(x_1 \vee x_3 \vee \overline{x_4}) \wedge (x_1 \vee \overline{x_3} \vee \overline{x_5}) \wedge (x_2 \vee \overline{x_4} \vee x_5) \wedge (\overline{x_1} \vee x_2 \vee \overline{x_3})$$

$x_1 = 1$, $x_2 = 0$, $x_3 = 1$, $x_4 = 0$ 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 $x_1, x_2, \ldots, x_n$ and m clauses, the randomised algorithm that sets each variable independently at random is a randomised $8/7$-approximation algorithm.

Proof:

- For every clause $i = 1, 2, \ldots, m$, 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,

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

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

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

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

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

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 $x_1, x_2, \ldots, x_n$ 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 assigment 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 $x_1, x_2, \ldots, x_n$ 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 $1/(8m)$

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

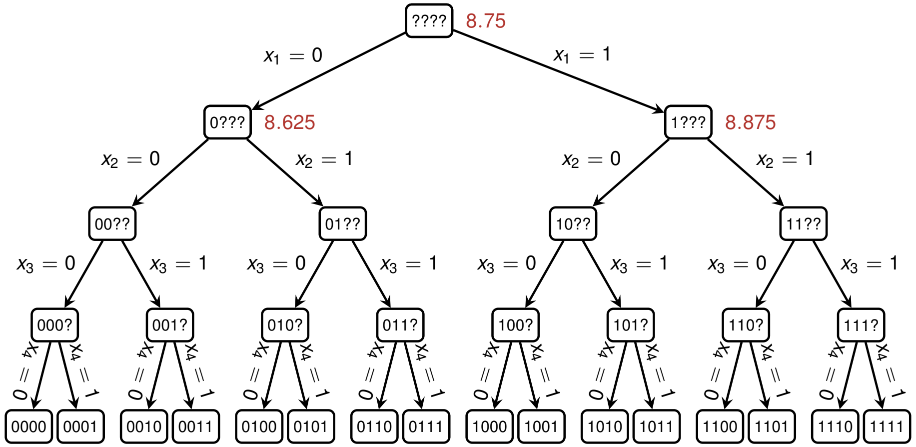
Y is defined as in the previous proof.

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

```
GREEDY-3-CNF(φ, n, m)
for j = 1, 2, ..., n
    Compute E[ Y | x_1 = v_1 ..., x_{j-1} = v_{j-1}, x_j = 1 ]
    Compute E[ Y | x_1 = v_1, ..., x_{j-1} = v_{j-1}, x_j = 0 ]
    Let x_j = v_j so that the conditional expectation is maximised
return the assignment v_1, v_2, ..., v_n
```

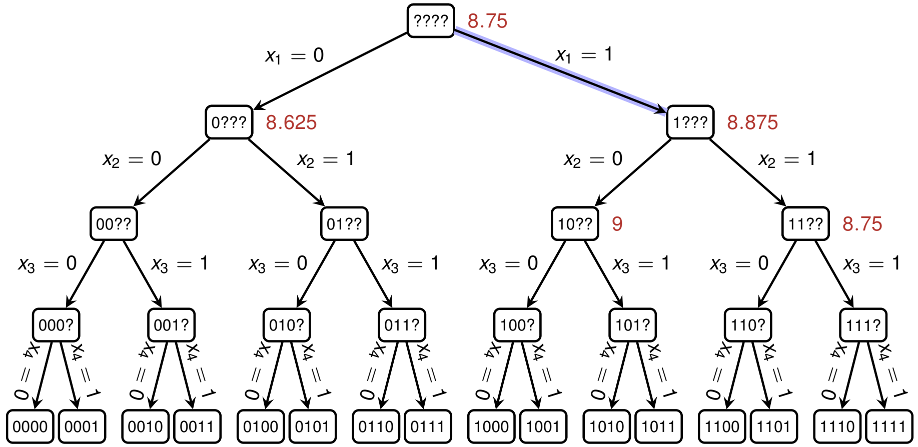
Run of GREEDY-3-CNF(φ, n, m)

$(x_1 \lor x_2 \lor x_3) \land (x_1 \lor \overline{x_2} \lor \overline{x_4}) \land (x_1 \lor x_2 \lor \overline{x_4}) \land (\overline{x_1} \lor \overline{x_3} \lor x_4) \land (x_1 \lor x_2 \lor \overline{x_4}) \land (\overline{x_1} \lor \overline{x_2} \lor \overline{x_3}) \land (\overline{x_1} \lor x_2 \lor x_3) \land (\overline{x_1} \lor \overline{x_2} \lor x_3) \land (x_1 \lor x_3 \lor x_4) \land (x_2 \lor \overline{x_3} \lor \overline{x_4})$



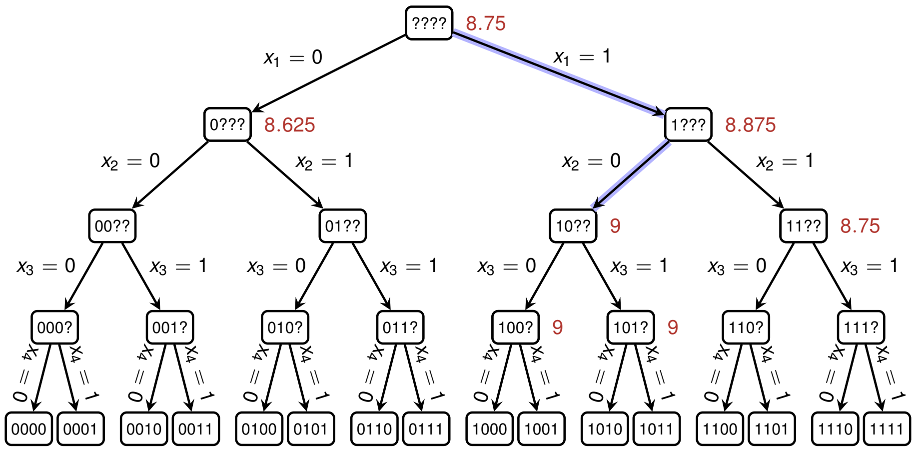
Run of GREEDY-3-CNF(φ, n, m)

$1 \land 1 \land 1 \land (\overline{x_3} \lor x_4) \land 1 \land (\overline{x_2} \lor \overline{x_3}) \land (x_2 \lor x_3) \land (\overline{x_2} \lor x_3) \land 1 \land (x_2 \lor \overline{x_3} \lor \overline{x_4})$



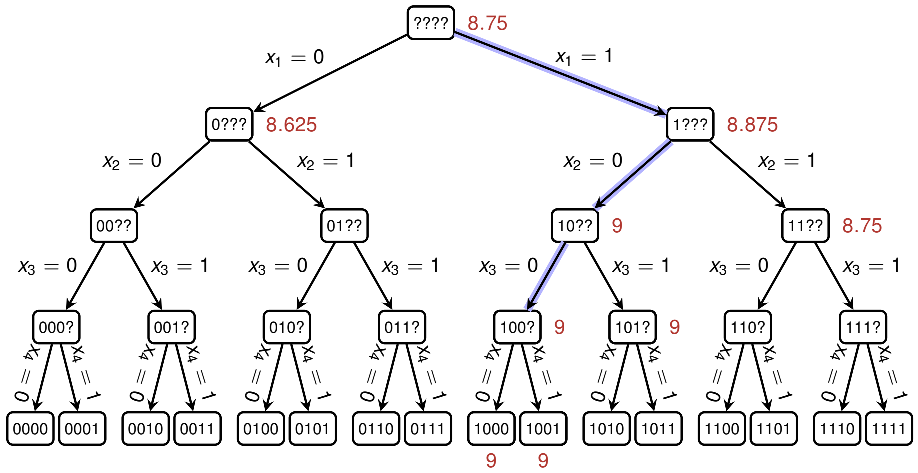
Run of GREEDY-3-CNF(φ, n, m)

$1 \land 1 \land 1 \land (\overline{x_3} \lor x_4) \land 1 \land 1 \land (x_3) \land 1 \land 1 \land (\overline{x_3} \lor \overline{x_4})$



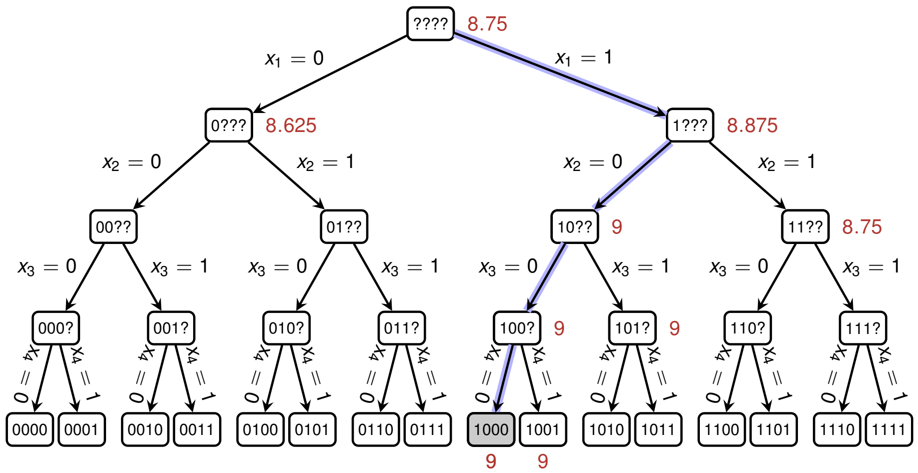
Run of GREEDY-3-CNF(φ, n, m)

$1 \land 1 \land 1 \land 1 \land 1 \land 1 \land 0 \land 1 \land 1 \land 1$



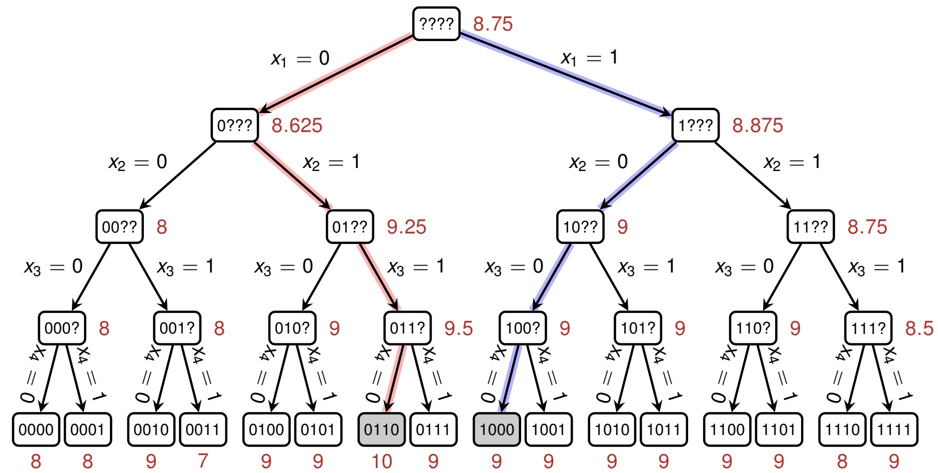
Run of GREEDY-3-CNF(φ, n, m)

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



Run of GREEDY-3-CNF(φ, n, m)

$(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee \overline{x_4}) \wedge (x_1 \vee x_2 \vee \overline{x_4}) \wedge (\overline{x_1} \vee \overline{x_3} \vee x_4) \wedge (x_1 \vee x_2 \vee \overline{x_4}) \wedge$
$(\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_3) \wedge (x_1 \vee x_3 \vee x_4) \wedge (x_2 \vee \overline{x_3} \vee \overline{x_4})$



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

Analysis of GREEDY-3-CNF(ϕ, n, m)

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

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

computable in $O(1)$

- **Step 2:** satisfies at least $7/8 \cdot m$ clauses
  - Due to the greedy choice in each iteration $j = 1, 2, \ldots, n$,

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

□

MAX-3-CNF: Concluding Remarks

Theorem 35.6

Given an instance of MAX-3-CNF with n variables $x_1, x_2, \ldots, x_n$ 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 polynomial-time 8/7-approximation.

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!

Source of Image: Stefan Szeider, TU Vienna

Outline

Randomised Approximation

MAX-3-CNF

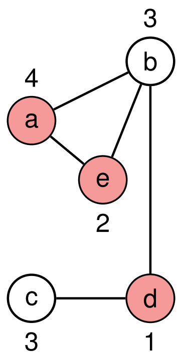
Weighted Vertex Cover

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 (still) 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)

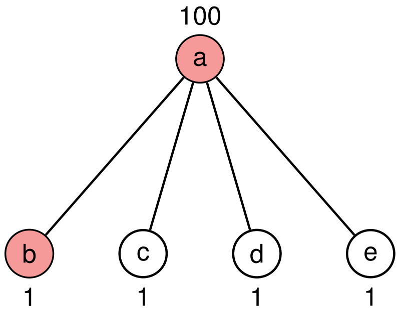
```
C = ∅
E' = G.E
while E' ≠ ∅
    let (u, v) be an arbitrary edge of E'
    C = C ∪ {u, v}
    remove from E' every edge incident on either u or v
return C
```

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

A Greedy Approach working for Unweighted Vertex Cover

APPROX-VERTEX-COVER(G)

```
C = ∅
E' = G.E
while E' ≠ ∅
    let (u, v) be an arbitrary edge of E'
    C = C ∪ {u, v}
    remove from E' every edge incident on either u or v
return C
```

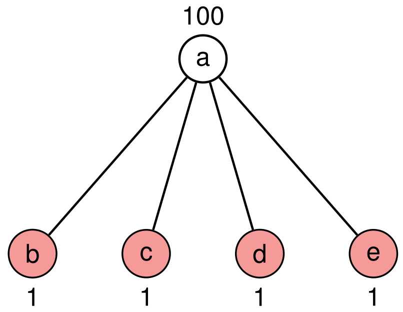


Computed solution has weight 101

A Greedy Approach working for Unweighted Vertex Cover

APPROX-VERTEX-COVER(G)

```
C = ∅
E' = G.E
while E' ≠ ∅
    let (u, v) be an arbitrary edge of E'
    C = C ∪ {u, v}
    remove from E' every edge incident on either u or v
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

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

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

Linear Program

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

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

The Algorithm

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

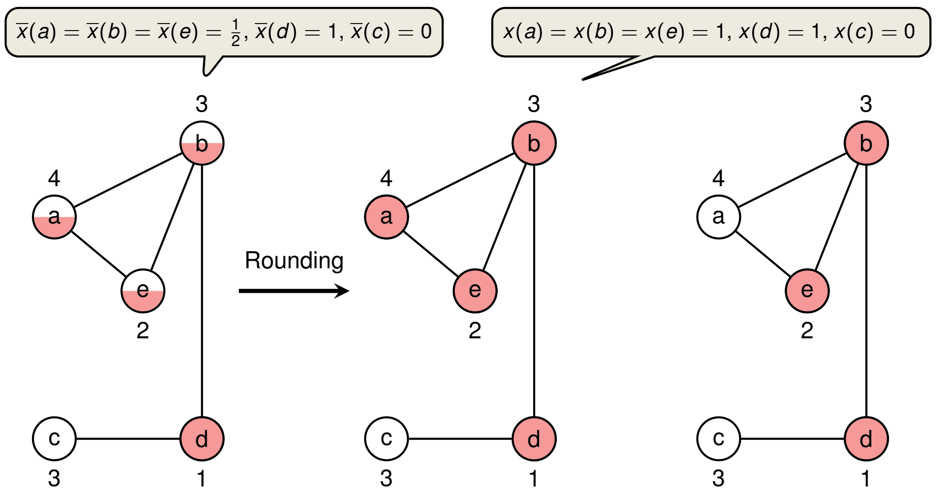
```
C = ∅
compute x̄, an optimal solution to the linear program
for each v ∈ V
    if x̄(v) ≥ 1/2
        C = C ∪ {v}
return C
```

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



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 $\overline{x}(u)$ and $\overline{x}(v)$ is at least $1/2 \Rightarrow C$ covers edge (u, v)
- **Step 2:** The computed set C satisfies $w(C) \leq 2z^*$:

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