The Maximum Satisfiability problem

K. Subramani[1]

[1] Lane Department of Computer Science and Electrical Engineering
West Virginia University

April 7, 2014

Outline

Outline

Outline

Outline

Satisfiability (SAT)

Satisfiability (SAT)

Definition (Satisfiability (SAT))

Given a CNF formula $\phi = C_1 \wedge C_2 \ldots C_m$, over the variables $\{x_1, x_2, \ldots, x_n\}$, is there an assignment of $\{\textbf{true}, \textbf{false}\}$ values to the literals, such that each clause C_i is satisfied?

Satisfiability (SAT)

Definition (Satisfiability (SAT))

Given a CNF formula $\phi = C_1 \wedge C_2 \dots C_m$, over the variables $\{x_1, x_2, \dots, x_n\}$, is there an assignment of $\{\mathbf{true}, \mathbf{false}\}$ values to the literals, such that each clause C_i is satisfied?

Note

SAT was the first naturally occurring problem to be proven **NP-complete** *(Stephen Cook, 1971).*

Satisfiability (SAT)

Definition (Satisfiability (SAT))

Given a CNF formula $\phi = C_1 \wedge C_2 \dots C_m$, over the variables $\{x_1, x_2, \dots, x_n\}$, is there an assignment of $\{\textbf{true}, \textbf{false}\}$ values to the literals, such that each clause C_i is satisfied?

Note

SAT was the first naturally occurring problem to be proven **NP-complete** *(Stephen Cook, 1971).*
Applications of SAT are too numerous to mention; logic, verification, AI, optimization, ...
kSAT, $k \geq 3$ is **NP-complete***; $k \leq 2$ is in* **P***. HornSAT is also in* **P***.*

Satisfiability (SAT)

Definition (Satisfiability (SAT))

Given a CNF formula $\phi = C_1 \wedge C_2 \ldots C_m$, over the variables $\{x_1, x_2, \ldots, x_n\}$, is there an assignment of $\{\mathbf{true}, \mathbf{false}\}$ values to the literals, such that each clause C_i is satisfied?

Note

SAT was the first naturally occurring problem to be proven **NP-complete** *(Stephen Cook, 1971). Applications of SAT are too numerous to mention; logic, verification, AI, optimization, ...*
kSAT, $k \geq 3$ is **NP-complete***; $k \leq 2$ is in* **P***. HornSAT is also in* **P***.*

Example

$$\begin{aligned} \phi \quad = \quad & (x_1, \bar{x_4}, \bar{x_7}) \\ & (x_2, \bar{x_1}) \\ & (x_3, \bar{x_1}, \bar{x_4}) \end{aligned}$$

MaxSAT

MaxSAT

Definition (Maximim Satisfiability (MaxSAT))

Given a CNF formula $\phi = C_1 \wedge C_2 \ldots C_m$, over the variables $\{x_1, x_2, \ldots, x_n\}$, find an assignment of $\{\textbf{true}, \textbf{false}\}$ values to the literals, such that the number of clauses satisfied is

MaxSAT

Definition (Maximim Satisfiability (MaxSAT))

Given a CNF formula $\phi = C_1 \wedge C_2 \ldots C_m$, over the variables $\{x_1, x_2, \ldots, x_n\}$, find an assignment of $\{\mathbf{true}, \mathbf{false}\}$ values to the literals, such that the number of clauses satisfied is *maximized.*

MaxSAT

Definition (Maximim Satisfiability (MaxSAT))

Given a CNF formula $\phi = C_1 \wedge C_2 \ldots C_m$, over the variables $\{x_1, x_2, \ldots, x_n\}$, find an assignment of $\{\mathbf{true}, \mathbf{false}\}$ values to the literals, such that the number of clauses satisfied is *maximized.*

Note

MaxSAT

Definition (Maximim Satisfiability (MaxSAT))

Given a CNF formula $\phi = C_1 \wedge C_2 \ldots C_m$, over the variables $\{x_1, x_2, \ldots, x_n\}$, find an assignment of $\{\mathbf{true}, \mathbf{false}\}$ values to the literals, such that the number of clauses satisfied is *maximized.*

Note

MaxSAT

Definition (Maximim Satisfiability (MaxSAT))

Given a CNF formula $\phi = C_1 \wedge C_2 \ldots C_m$, over the variables $\{x_1, x_2, \ldots, x_n\}$, find an assignment of $\{\mathbf{true}, \mathbf{false}\}$ values to the literals, such that the number of clauses satisfied is *maximized.*

Note

1. *Clearly, SAT* $\leq$ *MaxSAT.*

MaxSAT

Definition (Maximim Satisfiability (MaxSAT))

Given a CNF formula $\phi = C_1 \wedge C_2 \dots C_m$, over the variables $\{x_1, x_2, \dots, x_n\}$, find an assignment of $\{\textbf{true}, \textbf{false}\}$ values to the literals, such that the number of clauses satisfied is *maximized.*

Note

1. *Clearly, SAT $\leq$ MaxSAT.*
2. *Max2SAT and MaxHornSAT are both* **NP-hard**.

MaxSAT

Definition (Maximim Satisfiability (MaxSAT))

Given a CNF formula $\phi = C_1 \wedge C_2 \ldots C_m$, over the variables $\{x_1, x_2, \ldots, x_n\}$, find an assignment of $\{\textbf{true}, \textbf{false}\}$ values to the literals, such that the number of clauses satisfied is *maximized.*

Note

1. *Clearly, SAT $\leq$ MaxSAT.*
2. *Max2SAT and MaxHornSAT are both* **NP-hard***.*
3. *In the weighted version, each clause has a weight associated with it.*

MaxSAT

Definition (Maximim Satisfiability (MaxSAT))

Given a CNF formula $\phi = C_1 \wedge C_2 \ldots C_m$, over the variables $\{x_1, x_2, \ldots, x_n\}$, find an assignment of $\{\mathbf{true}, \mathbf{false}\}$ values to the literals, such that the number of clauses satisfied is *maximized.*

Note

1. *Clearly, SAT $\leq$ MaxSAT.*
2. *Max2SAT and MaxHornSAT are both* **NP-hard***.*
3. *In the weighted version, each clause has a weight associated with it. The goal in this case is to maximize the* sum of the weights *of the satisfied clauses.*

MaxSAT

Definition (Maximim Satisfiability (MaxSAT))

Given a CNF formula $\phi = C_1 \wedge C_2 \ldots C_m$, over the variables $\{x_1, x_2, \ldots, x_n\}$, find an assignment of $\{\textbf{true}, \textbf{false}\}$ values to the literals, such that the number of clauses satisfied is *maximized.*

Note

1. *Clearly, SAT $\leq$ MaxSAT.*
2. *Max2SAT and MaxHornSAT are both* **NP-hard**.
3. *In the weighted version, each clause has a weight associated with it. The goal in this case is to maximize the* sum of the weights *of the satisfied clauses.*
4. *We will focus on the cardinality version.*

MaxSAT

Definition (Maximim Satisfiability (MaxSAT))

Given a CNF formula $\phi = C_1 \wedge C_2 \ldots C_m$, over the variables $\{x_1, x_2, \ldots, x_n\}$, find an assignment of $\{\textbf{true}, \textbf{false}\}$ values to the literals, such that the number of clauses satisfied is *maximized.*

Note

1. *Clearly, SAT $\leq$ MaxSAT.*
2. *Max2SAT and MaxHornSAT are both* **NP-hard**.
3. *In the weighted version, each clause has a weight associated with it. The goal in this case is to maximize the* sum of the weights *of the satisfied clauses.*
4. *We will focus on the cardinality version. All our arguments carry over to the weighted case with almost no change.*

The variable setting algorithm

The variable setting algorithm

Variable-setting

The variable setting algorithm

Variable-setting

The variable setting algorithm

Variable-setting

1. **for** $(i = 1$ **to** $n)$

The variable setting algorithm

Variable-setting

1. **for** ($i = 1$ **to** n)
2. Flip a fair coin.

The variable setting algorithm

Variable-setting

1. **for** ($i = 1$ **to** n)
2.   Flip a fair coin.
3.   **If** (the coin turns up “heads”)

The variable setting algorithm

Variable-setting

1. **for** ($i = 1$ **to** n)
2. Flip a fair coin.
3. **If** (the coin turns up “heads”)
4. Set x_i to **true**.

The variable setting algorithm

Variable-setting

1. **for** ($i = 1$ **to** n)
2. Flip a fair coin.
3. **If** (the coin turns up “heads”)
4. Set x_i to **true**.
5. **else**

The variable setting algorithm

Variable-setting

1. **for** ($i = 1$ **to** n)
2. Flip a fair coin.
3. **If** (the coin turns up "heads")
4. Set x_i to **true**.
5. **else**
6. Set x_i to **false**.

The variable setting algorithm

Variable-setting

1. **for** ($i = 1$ **to** n)
2. Flip a fair coin.
3. **If** (the coin turns up “heads”)
4. Set x_i to **true**.
5. **else**
6. Set x_i to **false**.
7. **endif**

The variable setting algorithm

Variable-setting

1. **for** ($i = 1$ **to** n)
2. Flip a fair coin.
3. **If** (the coin turns up “heads”)
4. Set x_i to **true**.
5. **else**
6. Set x_i to **false**.
7. **endif**
8. **endfor**

The variable setting algorithm

Variable-setting

for ($i = 1$ **to** n)
    Flip a fair coin.
    If (the coin turns up “heads”)
        Set x_i to **true**.
    else
        Set x_i to **false**.
    endif
endfor
Return the number of satisfied clauses.

Analysis

Analysis

Lemma

Let k denote the width of the clausal system,

Analysis

Lemma

Let k denote the width of the clausal system, i.e., the number of literals in the clause with the fewest number of literals.

Analysis

Lemma

Let k denote the width of the clausal system, i.e., the number of literals in the clause with the fewest number of literals. The expected number of clauses satisfied by the above algorithm is $OPT \cdot (1 - \frac{1}{2^k})$.

Analysis

Lemma

Let k denote the width of the clausal system, i.e., the number of literals in the clause with the fewest number of literals. The expected number of clauses satisfied by the above algorithm is $OPT \cdot (1 - \frac{1}{2^k})$.

Proof.

Analysis

Lemma

Let k denote the width of the clausal system, i.e., the number of literals in the clause with the fewest number of literals. The expected number of clauses satisfied by the above algorithm is $OPT \cdot (1 - \frac{1}{2^k})$.

Proof.

1. Let p_i denote the probability that clause C_i is satisfied.

Analysis

Lemma

Let k denote the width of the clausal system, i.e., the number of literals in the clause with the fewest number of literals. The expected number of clauses satisfied by the above algorithm is $OPT \cdot (1 - \frac{1}{2^k})$.

Proof.

1. Let p_i denote the probability that clause C_i is satisfied.
2. Clearly, $p_i \geq$

Analysis

Lemma

Let k denote the width of the clausal system, i.e., the number of literals in the clause with the fewest number of literals. The expected number of clauses satisfied by the above algorithm is $OPT \cdot (1 - \frac{1}{2^k})$.

Proof.

1. Let p_i denote the probability that clause C_i is satisfied.
2. Clearly, $p_i \geq (1 - \frac{1}{2^k})$.

Analysis

Lemma

Let k denote the width of the clausal system, i.e., the number of literals in the clause with the fewest number of literals. The expected number of clauses satisfied by the above algorithm is $OPT \cdot (1 - \frac{1}{2^k})$.

Proof.

1. Let p_i denote the probability that clause C_i is satisfied.
2. Clearly, $p_i \geq (1 - \frac{1}{2^k})$.
3. Let X_i denote an indicator variable.

Analysis

Lemma

Let k denote the width of the clausal system, i.e., the number of literals in the clause with the fewest number of literals. The expected number of clauses satisfied by the above algorithm is $OPT \cdot (1 - \frac{1}{2^k})$.

Proof.

1. Let p_i denote the probability that clause C_i is satisfied.
2. Clearly, $p_i \geq (1 - \frac{1}{2^k})$.
3. Let X_i denote an indicator variable. X_i is set to 1, if clause C_i is satisfied under the variable setting algorithm and 0 otherwise.

Analysis

Lemma

Let k denote the width of the clausal system, i.e., the number of literals in the clause with the fewest number of literals. The expected number of clauses satisfied by the above algorithm is $OPT \cdot (1 - \frac{1}{2^k})$.

Proof.

1. Let p_i denote the probability that clause C_i is satisfied.
2. Clearly, $p_i \geq (1 - \frac{1}{2^k})$.
3. Let X_i denote an indicator variable. X_i is set to 1, if clause C_i is satisfied under the variable setting algorithm and 0 otherwise.
4. Let $X = \sum_{i=1}^{m} X_i$.

Analysis

Lemma

Let k denote the width of the clausal system, i.e., the number of literals in the clause with the fewest number of literals. The expected number of clauses satisfied by the above algorithm is $OPT \cdot (1 - \frac{1}{2^k})$.

Proof.

1. Let p_i denote the probability that clause C_i is satisfied.
2. Clearly, $p_i \geq (1 - \frac{1}{2^k})$.
3. Let X_i denote an indicator variable. X_i is set to 1, if clause C_i is satisfied under the variable setting algorithm and 0 otherwise.
4. Let $X = \sum_{i=1}^{m} X_i$. Clearly, we are interested in X.

Analysis

Lemma

Let k denote the width of the clausal system, i.e., the number of literals in the clause with the fewest number of literals. The expected number of clauses satisfied by the above algorithm is $OPT \cdot (1 - \frac{1}{2^k})$.

Proof.

1. Let p_i denote the probability that clause C_i is satisfied.
2. Clearly, $p_i \geq (1 - \frac{1}{2^k})$.
3. Let X_i denote an indicator variable. X_i is set to 1, if clause C_i is satisfied under the variable setting algorithm and 0 otherwise.
4. Let $X = \sum_{i=1}^{m} X_i$. Clearly, we are interested in X.
5. However, since X is a random variable, we focus on $\mathbf{E}[X]$.

□

Analysis (contd.)

Analysis (contd.)

Proof.

Analysis (contd.)

Proof.

1. Observe that,

Analysis (contd.)

Proof.

1. Observe that,

$$\mathbf{E}[X] \quad =$$

Analysis (contd.)

Proof.

1. Observe that,

$$\mathbf{E}[X] = \mathbf{E}[\sum_{i=1}^{m} X_i]$$

Analysis (contd.)

Proof.

1. Observe that,

$$\begin{aligned}\mathbf{E}[X] &= \mathbf{E}[\sum_{i=1}^{m} X_i] \\ &= \sum_{i=1}^{m} \mathbf{E}[X_i]\end{aligned}$$

Analysis (contd.)

Proof.

1. Observe that,

$$\begin{aligned} \mathbf{E}[X] &= \mathbf{E}[\sum_{i=1}^{m} X_i] \\ &= \sum_{i=1}^{m} \mathbf{E}[X_i] \\ &= \sum_{i=1}^{m} p_i \end{aligned}$$

Analysis (contd.)

Proof.

1. Observe that,

$$\begin{aligned} \mathbf{E}[X] &= \mathbf{E}[\sum_{i=1}^{m} X_i] \\ &= \sum_{i=1}^{m} \mathbf{E}[X_i] \\ &= \sum_{i=1}^{m} p_i \\ &\geq \sum_{i=1}^{m} (1 - \frac{1}{2^k}) \end{aligned}$$

Analysis (contd.)

Proof.

1. Observe that,

$$\begin{aligned} \mathbf{E}[X] &= \mathbf{E}[\sum_{i=1}^{m} X_i] \\ &= \sum_{i=1}^{m} \mathbf{E}[X_i] \\ &= \sum_{i=1}^{m} p_i \\ &\geq \sum_{i=1}^{m} (1 - \frac{1}{2^k}) \\ &= m \cdot (1 - \frac{1}{2^k}) \end{aligned}$$

Analysis (contd.)

Proof.

1. Observe that,

$$
\begin{aligned}
\mathbf{E}[X] &= \mathbf{E}[\sum_{i=1}^{m} X_i] \\
&= \sum_{i=1}^{m} \mathbf{E}[X_i] \\
&= \sum_{i=1}^{m} p_i \\
&\geq \sum_{i=1}^{m} (1 - \frac{1}{2^k}) \\
&= m \cdot (1 - \frac{1}{2^k}) \\
&\geq OPT \cdot (1 - \frac{1}{2^k}).
\end{aligned}
$$

□

Randomized Rounding

Randomized Rounding

The LP-based approach

Randomized Rounding

The LP-based approach

1. Let C_j^+ denote the set of literals that appear in uncomplemented form in clause C_j.

Randomized Rounding

The LP-based approach

1. Let C_j^+ denote the set of literals that appear in uncomplemented form in clause C_j. Likewise, let C_j^- denote the set of literals that appear in complemented form in clause C_j.

Randomized Rounding

The LP-based approach

1. Let C_j^+ denote the set of literals that appear in uncomplemented form in clause C_j. Likewise, let C_j^- denote the set of literals that appear in complemented form in clause C_j.
2. The MaxSAT problem can then be modeled through the following integer program:

Randomized Rounding

The LP-based approach

1. Let C_j^+ denote the set of literals that appear in uncomplemented form in clause C_j. Likewise, let C_j^- denote the set of literals that appear in complemented form in clause C_j.
2. The MaxSAT problem can then be modeled through the following integer program:

$$\max \sum_{j=1}^{m} z_i$$

Randomized Rounding

The LP-based approach

1. Let C_j^+ denote the set of literals that appear in uncomplemented form in clause C_j. Likewise, let C_j^- denote the set of literals that appear in complemented form in clause C_j.
2. The MaxSAT problem can then be modeled through the following integer program:

$$\max \sum_{j=1}^{m} z_i$$

subject to

Randomized Rounding

The LP-based approach

1. Let C_j^+ denote the set of literals that appear in uncomplemented form in clause C_j. Likewise, let C_j^- denote the set of literals that appear in complemented form in clause C_j.
2. The MaxSAT problem can then be modeled through the following integer program:

$$\max \sum_{j=1}^{m} z_i$$
$$\text{subject to} \quad \sum_{i \in C_j^+} y_i + \sum_{i \in C_j^-} (1 - y_i) \geq z_j$$

Randomized Rounding

The LP-based approach

1. Let C_j^+ denote the set of literals that appear in uncomplemented form in clause C_j. Likewise, let C_j^- denote the set of literals that appear in complemented form in clause C_j.
2. The MaxSAT problem can then be modeled through the following integer program:

$$\begin{aligned} & \max \sum_{j=1}^{m} z_i \\ \text{subject to} \quad & \sum_{i \in C_j^+} y_i + \sum_{i \in C_j^-} (1 - y_i) \geq z_j \\ & y_i, z_j \in \{0, 1\} \qquad \forall i, j \end{aligned}$$

Randomized Rounding

The LP-based approach

1. Let C_j^+ denote the set of literals that appear in uncomplemented form in clause C_j. Likewise, let C_j^- denote the set of literals that appear in complemented form in clause C_j.
2. The MaxSAT problem can then be modeled through the following integer program:

$$
\begin{array}{lll}
 & \max \sum_{j=1}^{m} z_i & \\
\text{subject to} & \sum_{i \in C_j^+} y_i + \sum_{i \in C_j^-} (1 - y_i) \geq z_j & \\
 & y_i, z_j \in \{0, 1\} & \forall i, j
\end{array}
$$

3. Relax the above integer program to a linear program and solve it.

Randomized Rounding

The LP-based approach

1. Let C_j^+ denote the set of literals that appear in uncomplemented form in clause C_j. Likewise, let C_j^- denote the set of literals that appear in complemented form in clause C_j.
2. The MaxSAT problem can then be modeled through the following integer program:

$$\begin{array}{lll} & \max \sum_{j=1}^{m} z_i & \\ \text{subject to} & \sum_{i \in C_j^+} y_i + \sum_{i \in C_j^-} (1 - y_i) \geq z_j & \\ & y_i, z_j \in \{0, 1\} & \forall i, j \end{array}$$

3. Relax the above integer program to a linear program and solve it.
4. Let $\hat{\mathbf{y}}$ and $\hat{\mathbf{z}}$ denote the values of the variables at the optimum solution.

Randomized Rounding

The LP-based approach

1. Let C_j^+ denote the set of literals that appear in uncomplemented form in clause C_j. Likewise, let C_j^- denote the set of literals that appear in complemented form in clause C_j.
2. The MaxSAT problem can then be modeled through the following integer program:

$$\begin{array}{lll} & \max \sum_{j=1}^{m} z_i & \\ \text{subject to} & \sum_{i \in C_j^+} y_i + \sum_{i \in C_j^-} (1 - y_i) \geq z_j & \\ & y_i, z_j \in \{0, 1\} & \forall i, j \end{array}$$

3. Relax the above integer program to a linear program and solve it.
4. Let $\hat{\mathbf{y}}$ and $\hat{\mathbf{z}}$ denote the values of the variables at the optimum solution.
5. Independently set each y_i to 1, with probability $\hat{y}_i$.

Randomized Rounding

The LP-based approach

1. Let C_j^+ denote the set of literals that appear in uncomplemented form in clause C_j. Likewise, let C_j^- denote the set of literals that appear in complemented form in clause C_j.
2. The MaxSAT problem can then be modeled through the following integer program:

$$
\begin{array}{lll}
 & \max \sum_{j=1}^{m} z_i & \\
\text{subject to} & \sum_{i \in C_j^+} y_i + \sum_{i \in C_j^-} (1 - y_i) \geq z_j & \\
 & y_i, z_j \in \{0, 1\} & \forall i, j
\end{array}
$$

3. Relax the above integer program to a linear program and solve it.
4. Let $\hat{\mathbf{y}}$ and $\hat{\mathbf{z}}$ denote the values of the variables at the optimum solution.
5. Independently set each y_i to 1, with probability $\hat{y}_i$.
6. Output the number of satisfied clauses.

Analysis

Analysis

Lemma

Analysis

Lemma

Let $\beta_k = 1 - (1 - \frac{1}{k})^k$.

Analysis

Lemma

Let $\beta_k = 1 - (1 - \frac{1}{k})^k$. For any clause C_j with k literals, the probability that it is satisfied by the LP-rounding assignment is at least $\beta_k \hat{z}_j$.

Analysis

Lemma

Let $\beta_k = 1 - (1 - \frac{1}{k})^k$. For any clause C_j with k literals, the probability that it is satisfied by the LP-rounding assignment is at least $\beta_k \hat{z}_j$.

Proof.

Analysis

Lemma

Let $\beta_k = 1 - (1 - \frac{1}{k})^k$. For any clause C_j with k literals, the probability that it is satisfied by the LP-rounding assignment is at least $\beta_k \hat{z}_j$.

Proof.

1. Focus on a specific clause C_j with k literals.

Analysis

Lemma

Let $\beta_k = 1 - (1 - \frac{1}{k})^k$. For any clause C_j with k literals, the probability that it is satisfied by the LP-rounding assignment is at least $\beta_k \hat{z}_j$.

Proof.

1. Focus on a specific clause C_j with k literals.
2. Without loss of generality, we can assume that $C_j = (x_1, x_2, \ldots, x_k)$.

Analysis

Lemma

Let $\beta_k = 1 - (1 - \frac{1}{k})^k$. For any clause C_j with k literals, the probability that it is satisfied by the LP-rounding assignment is at least $\beta_k \hat{z}_j$.

Proof.

1. Focus on a specific clause C_j with k literals.
2. Without loss of generality, we can assume that $C_j = (x_1, x_2, \ldots, x_k)$.
3. Since the LP was solved optimally, we must have,

Analysis

Lemma

Let $\beta_k = 1 - (1 - \frac{1}{k})^k$. For any clause C_j with k literals, the probability that it is satisfied by the LP-rounding assignment is at least $\beta_k \hat{z}_j$.

Proof.

1. Focus on a specific clause C_j with k literals.
2. Without loss of generality, we can assume that $C_j = (x_1, x_2, \ldots, x_k)$.
3. Since the LP was solved optimally, we must have,

$$\hat{y}_1 + \hat{y}_2 + \ldots \hat{y}_k \geq \hat{z}_j.$$

Analysis

Lemma

Let $\beta_k = 1 - (1 - \frac{1}{k})^k$. For any clause C_j with k literals, the probability that it is satisfied by the LP-rounding assignment is at least $\beta_k \hat{z}_j$.

Proof.

1. Focus on a specific clause C_j with k literals.
2. Without loss of generality, we can assume that $C_j = (x_1, x_2, \ldots, x_k)$.
3. Since the LP was solved optimally, we must have,

$$\hat{y}_1 + \hat{y}_2 + \ldots \hat{y}_k \geq \hat{z}_j.$$

4. The probability that clause C_j remains unsatisfied after the rounding, is precisely the probability that each $y_i, i = 1, 2, \ldots k$ was set to 0.

Analysis

Lemma

Let $\beta_k = 1 - (1 - \frac{1}{k})^k$. For any clause C_j with k literals, the probability that it is satisfied by the LP-rounding assignment is at least $\beta_k \hat{z}_j$.

Proof.

1. Focus on a specific clause C_j with k literals.
2. Without loss of generality, we can assume that $C_j = (x_1, x_2, \ldots, x_k)$.
3. Since the LP was solved optimally, we must have,

$$\hat{y}_1 + \hat{y}_2 + \ldots \hat{y}_k \geq \hat{z}_j.$$

4. The probability that clause C_j remains unsatisfied after the rounding, is precisely the probability that each $y_i, i = 1, 2, \ldots k$ was set to 0. This probability is clearly

Analysis

Lemma

Let $\beta_k = 1 - (1 - \frac{1}{k})^k$. For any clause C_j with k literals, the probability that it is satisfied by the LP-rounding assignment is at least $\beta_k \hat{z}_j$.

Proof.

1. Focus on a specific clause C_j with k literals.
2. Without loss of generality, we can assume that $C_j = (x_1, x_2, \ldots, x_k)$.
3. Since the LP was solved optimally, we must have,

$$\hat{y}_1 + \hat{y}_2 + \ldots \hat{y}_k \geq \hat{z}_j.$$

4. The probability that clause C_j remains unsatisfied after the rounding, is precisely the probability that each $y_i, i = 1, 2, \ldots k$ was set to 0. This probability is clearly $\Pi_{i=1}^{k}(1 - \hat{y}_i)$.

Analysis

Lemma

Let $\beta_k = 1 - (1 - \frac{1}{k})^k$. For any clause C_j with k literals, the probability that it is satisfied by the LP-rounding assignment is at least $\beta_k \hat{z}_j$.

Proof.

1. Focus on a specific clause C_j with k literals.
2. Without loss of generality, we can assume that $C_j = (x_1, x_2, \ldots, x_k)$.
3. Since the LP was solved optimally, we must have,

$$\hat{y}_1 + \hat{y}_2 + \ldots \hat{y}_k \geq \hat{z}_j.$$

4. The probability that clause C_j remains unsatisfied after the rounding, is precisely the probability that each $y_i, i = 1, 2, \ldots k$ was set to 0. This probability is clearly $\Pi_{i=1}^{k}(1 - \hat{y}_i)$.
5. We thus need to show that

Analysis

Lemma

Let $\beta_k = 1 - (1 - \frac{1}{k})^k$. For any clause C_j with k literals, the probability that it is satisfied by the LP-rounding assignment is at least $\beta_k \hat{z}_j$.

Proof.

1. Focus on a specific clause C_j with k literals.
2. Without loss of generality, we can assume that $C_j = (x_1, x_2, \ldots, x_k)$.
3. Since the LP was solved optimally, we must have,

$$\hat{y}_1 + \hat{y}_2 + \ldots \hat{y}_k \geq \hat{z}_j.$$

4. The probability that clause C_j remains unsatisfied after the rounding, is precisely the probability that each $y_i, i = 1, 2, \ldots k$ was set to 0. This probability is clearly $\Pi_{i=1}^{k}(1 - \hat{y}_i)$.
5. We thus need to show that

$$1 - \Pi_{i=1}^{k}(1 - \hat{y}_i) \geq$$

Analysis

Lemma

Let $\beta_k = 1 - (1 - \frac{1}{k})^k$. For any clause C_j with k literals, the probability that it is satisfied by the LP-rounding assignment is at least $\beta_k \hat{z}_j$.

Proof.

1. Focus on a specific clause C_j with k literals.
2. Without loss of generality, we can assume that $C_j = (x_1, x_2, \ldots, x_k)$.
3. Since the LP was solved optimally, we must have,

$$\hat{y}_1 + \hat{y}_2 + \ldots \hat{y}_k \geq \hat{z}_j.$$

4. The probability that clause C_j remains unsatisfied after the rounding, is precisely the probability that each $y_i, i = 1, 2, \ldots k$ was set to 0. This probability is clearly $\Pi_{i=1}^{k}(1 - \hat{y}_i)$.
5. We thus need to show that

$$1 - \Pi_{i=1}^{k}(1 - \hat{y}_i) \geq \beta_k \cdot \hat{z}_j.$$

□

Analysis (contd.)

Analysis (contd.)

Proof.

Analysis (contd.)

Proof.

1. The expression $1 - \Pi_{i=1}^{k}(1 - \hat{y}_i)$ is minimized at $\hat{y}_i = \frac{\hat{z}_j}{k}$.

Analysis (contd.)

Proof.

1. The expression $1 - \Pi_{i=1}^{k}(1 - \hat{y}_i)$ is minimized at $\hat{y}_i = \frac{\hat{z}_j}{k}$.
2. It therefore suffices to show that

Analysis (contd.)

Proof.

1. The expression $1 - \Pi_{i=1}^{k}(1 - \hat{y}_i)$ is minimized at $\hat{y}_i = \frac{\hat{z}_j}{k}$.
2. It therefore suffices to show that $(1 - (1 - \frac{z}{k})^k) \geq \beta_k \cdot z$, for all integers k and all $z \in [0, 1]$.

Analysis (contd.)

Proof.

1. The expression $1 - \Pi_{i=1}^{k}(1 - \hat{y}_i)$ is minimized at $\hat{y}_i = \frac{\hat{z}_j}{k}$.
2. It therefore suffices to show that $(1 - (1 - \frac{z}{k})^k) \geq \beta_k \cdot z$, for all integers k and all $z \in [0, 1]$.
3. Observe that $f(x) = 1 - (1 - \frac{x}{k})^k$ is a concave function.

Analysis (contd.)

Proof.

1. The expression $1 - \Pi_{i=1}^{k}(1 - \hat{y}_i)$ is minimized at $\hat{y}_i = \frac{\hat{z}_j}{k}$.
2. It therefore suffices to show that $(1 - (1 - \frac{z}{k})^k) \geq \beta_k \cdot z$, for all integers k and all $z \in [0, 1]$.
3. Observe that $f(x) = 1 - (1 - \frac{x}{k})^k$ is a concave function.
4. It therefore suffices to verify the inequality $f(x) \geq \beta_k \cdot x$ at $x = 0$ and $x = 1$.

Analysis (contd.)

Proof.

1. The expression $1 - \Pi_{i=1}^{k}(1 - \hat{y}_i)$ is minimized at $\hat{y}_i = \frac{\hat{z}_j}{k}$.
2. It therefore suffices to show that $(1 - (1 - \frac{z}{k})^k) \geq \beta_k \cdot z$, for all integers k and all $z \in [0, 1]$.
3. Observe that $f(x) = 1 - (1 - \frac{x}{k})^k$ is a concave function.
4. It therefore suffices to verify the inequality $f(x) \geq \beta_k \cdot x$ at $x = 0$ and $x = 1$.
5. Note that $f(0) = 0 \geq \beta_k \cdot 0$ and $f(1) = \beta_k \geq \beta_k \cdot 1$.

Analysis (contd.)

Proof.

1. The expression $1 - \Pi_{i=1}^{k}(1 - \hat{y}_i)$ is minimized at $\hat{y}_i = \frac{\hat{z}_j}{k}$.
2. It therefore suffices to show that $(1 - (1 - \frac{z}{k})^k) \geq \beta_k \cdot z$, for all integers k and all $z \in [0, 1]$.
3. Observe that $f(x) = 1 - (1 - \frac{x}{k})^k$ is a concave function.
4. It therefore suffices to verify the inequality $f(x) \geq \beta_k \cdot x$ at $x = 0$ and $x = 1$.
5. Note that $f(0) = 0 \geq \beta_k \cdot 0$ and $f(1) = \beta_k \geq \beta_k \cdot 1$.
6. We apply the same logic to the linear function $g(z) = \beta_k \cdot z$ and the lemma follows.

□

Approximation bound

Approximation bound

Theorem

Let expected number of clauses satisfied by the randomized rounding algorithm is at least $(1 - \frac{1}{e}) \cdot OPT$.

Approximation bound

Theorem

Let expected number of clauses satisfied by the randomized rounding algorithm is at least $(1 - \frac{1}{e}) \cdot OPT$.

Proof.

Approximation bound

Theorem

Let expected number of clauses satisfied by the randomized rounding algorithm is at least $(1 - \frac{1}{e}) \cdot OPT$.

Proof.

1. Let X_i be 1, if clause C_i is satisfied, and 0 otherwise.

Approximation bound

Theorem

Let expected number of clauses satisfied by the randomized rounding algorithm is at least $(1 - \frac{1}{e}) \cdot OPT$.

Proof.

1. Let X_i be 1, if clause C_i is satisfied, and 0 otherwise.
2. Let $X = \sum_{j=1}^{m} X_i$.

Approximation bound

Theorem

Let expected number of clauses satisfied by the randomized rounding algorithm is at least $(1 - \frac{1}{e}) \cdot OPT$.

Proof.

1. Let X_i be 1, if clause C_i is satisfied, and 0 otherwise.
2. Let $X = \sum_{j=1}^{m} X_i$.
3. Let k_i be the number of literals in clause C_i and let p_i denote the probability that clause C_i is satisfied.

□

Final Steps (contd.)

Final Steps (contd.)

Proof.

Final Steps (contd.)

Proof.

1. As discussed before,

Final Steps (contd.)

Proof.

1. As discussed before,

Final Steps (contd.)

Proof.

1. As discussed before,

$$\mathbf{E}[X] \quad =$$

Final Steps (contd.)

Proof.

1. As discussed before,

$$\mathbf{E}[X] = \mathbf{E}[\sum_{j=1}^{m} X_i]$$

Final Steps (contd.)

Proof.

1. As discussed before,

$$\begin{aligned} \mathbf{E}[X] &= \mathbf{E}[\sum_{j=1}^{m} X_i] \\ &= \sum_{j=1}^{m} \mathbf{E}[X_i] \end{aligned}$$

Final Steps (contd.)

Proof.

1. As discussed before,

$$\begin{aligned} \mathbf{E}[X] &= \mathbf{E}[\sum_{j=1}^{m} X_i] \\ &= \sum_{j=1}^{m} \mathbf{E}[X_i] \\ &= \sum_{j=1}^{m} p_i \end{aligned}$$

Final Steps (contd.)

Proof.

1. As discussed before,

$$\begin{aligned} \mathbf{E}[X] &= \mathbf{E}[\sum_{j=1}^{m} X_i] \\ &= \sum_{j=1}^{m} \mathbf{E}[X_i] \\ &= \sum_{j=1}^{m} p_i \\ &\geq \sum_{j=1}^{m} \beta_{k_j} \cdot \hat{z}_j \end{aligned}$$

Final Steps (contd.)

Proof.

1. As discussed before,

$$\begin{aligned} \mathbf{E}[X] &= \mathbf{E}[\sum_{j=1}^{m} X_i] \\ &= \sum_{j=1}^{m} \mathbf{E}[X_i] \\ &= \sum_{j=1}^{m} p_i \\ &\geq \sum_{j=1}^{m} \beta_{k_i} \cdot \hat{z}_j \end{aligned}$$

□

Final Steps (contd.)

Proof.

Final Steps (contd.)

Proof.

$$\mathbf{E}[X] \geq \sum_{j=1}^{m} (1 - \frac{1}{e}) \cdot \hat{z}_j,$$

Final Steps (contd.)

Proof.

$$\mathbf{E}[X] \geq \sum_{j=1}^{m}(1-\frac{1}{e})\cdot\hat{z}_j, \text{ since } \beta_k = 1-(1-\frac{1}{k})^k \geq (1-\frac{1}{e}), \text{ for all positive integers } k$$

Final Steps (contd.)

Proof.

$$\begin{aligned} \mathbf{E}[X] &\geq \sum_{j=1}^{m}(1-\frac{1}{e})\cdot\hat{z}_j, \text{ since } \beta_k = 1-(1-\frac{1}{k})^k \geq (1-\frac{1}{e}), \text{ for all positive integers } k \\ &= (1-\frac{1}{e})\cdot\sum_{j=1}^{m}\hat{z}_j \end{aligned}$$

Final Steps (contd.)

Proof.

$$\begin{aligned} \mathbf{E}[X] &\geq \sum_{j=1}^{m}(1-\frac{1}{e})\cdot \hat{z}_j, \text{ since } \beta_k = 1-(1-\frac{1}{k})^k \geq (1-\frac{1}{e}), \text{ for all positive integers } k \\ &= (1-\frac{1}{e})\cdot \sum_{j=1}^{m}\hat{z}_j \\ &\geq (1-\frac{1}{e})\cdot OPT. \end{aligned}$$

□

A $\frac{3}{4}$ factor algorithm

A $\frac{3}{4}$ factor algorithm

Combination Algorithm

A $\frac{3}{4}$ factor algorithm

Combination Algorithm

1. Run the variable flipping algorithm.

A $\frac{3}{4}$ factor algorithm

Combination Algorithm

1. Run the variable flipping algorithm. Let n_1 be the number of clauses satisfied by this algorithm.

A $\frac{3}{4}$ factor algorithm

Combination Algorithm

1. Run the variable flipping algorithm. Let n_1 be the number of clauses satisfied by this algorithm.
2. Run the LP-based randomized rounding algorithm.

A $\frac{3}{4}$ factor algorithm

Combination Algorithm

1. Run the variable flipping algorithm. Let n_1 be the number of clauses satisfied by this algorithm.
2. Run the LP-based randomized rounding algorithm. Let n_2 be the number of clauses satisfied by this algorithm.

A $\frac{3}{4}$ factor algorithm

Combination Algorithm

1. Run the variable flipping algorithm. Let n_1 be the number of clauses satisfied by this algorithm.
2. Run the LP-based randomized rounding algorithm. Let n_2 be the number of clauses satisfied by this algorithm.
3. **return**$(\max(n_1, n_2))$.

Analysis

Analysis

Algorithm performance and clause width

k	$(1 - \frac{1}{2^k})$	β_k

Analysis

Algorithm performance and clause width

k	$(1 - \frac{1}{2^k})$	β_k
1	0.5	1.0

Analysis

Algorithm performance and clause width

k	$(1 - \frac{1}{2^k})$	β_k
1	0.5	1.0
2	0.75	0.75

Analysis

Algorithm performance and clause width

k	$(1 - \frac{1}{2^k})$	β_k
1	0.5	1.0
2	0.75	0.75
3	0.875	0.704

Analysis

Algorithm performance and clause width

k	$(1 - \frac{1}{2^k})$	β_k
1	0.5	1.0
2	0.75	0.75
3	0.875	0.704
4	0.938	0.684

Analysis

Algorithm performance and clause width

k	$(1 - \frac{1}{2^k})$	β_k
1	0.5	1.0
2	0.75	0.75
3	0.875	0.704
4	0.938	0.684
5	0.969	0.672

Analysis

Algorithm performance and clause width

k	$(1 - \frac{1}{2^k})$	β_k
1	0.5	1.0
2	0.75	0.75
3	0.875	0.704
4	0.938	0.684
5	0.969	0.672

Lemma

Analysis

Algorithm performance and clause width

k	$(1 - \frac{1}{2^k})$	β_k
1	0.5	1.0
2	0.75	0.75
3	0.875	0.704
4	0.938	0.684
5	0.969	0.672

Lemma

Analysis

Algorithm performance and clause width

k	$(1 - \frac{1}{2^k})$	β_k
1	0.5	1.0
2	0.75	0.75
3	0.875	0.704
4	0.938	0.684
5	0.969	0.672

Lemma

$$\max(n_1, n_2) \geq \frac{3}{4} \cdot OPT.$$

Analysis (contd.)

Analysis (contd.)

Proof.

Analysis (contd.)

Proof.

1. We will show that $\frac{n_1+n_2}{2} \geq \frac{3}{4} \cdot \sum_j \hat{z}_j$.

Analysis (contd.)

Proof.

1. We will show that $\frac{n_1+n_2}{2} \geq \frac{3}{4} \cdot \sum_j \hat{z}_j$.
2. Let S^k denote the set of clauses that contain k literals.

Analysis (contd.)

Proof.

1. We will show that $\frac{n_1+n_2}{2} \geq \frac{3}{4} \cdot \sum_j \hat{z}_j$.
2. Let S^k denote the set of clauses that contain k literals.
3. As per the variable rounding algorithm,

Analysis (contd.)

Proof.

1. We will show that $\frac{n_1+n_2}{2} \geq \frac{3}{4} \cdot \sum_j \hat{z}_j$.
2. Let S^k denote the set of clauses that contain k literals.
3. As per the variable rounding algorithm,

$$n_1 =$$

Analysis (contd.)

Proof.

1. We will show that $\frac{n_1+n_2}{2} \geq \frac{3}{4} \cdot \sum_j \hat{z}_j$.
2. Let S^k denote the set of clauses that contain k literals.
3. As per the variable rounding algorithm,

$$n_1 = \sum_k \sum_{C_j \in S^k} (1 - 2^{-k})$$

Analysis (contd.)

Proof.

1. We will show that $\frac{n_1+n_2}{2} \geq \frac{3}{4} \cdot \sum_j \hat{z}_j$.
2. Let S^k denote the set of clauses that contain k literals.
3. As per the variable rounding algorithm,

$$\begin{aligned} n_1 &= \sum_k \sum_{C_j \in S^k} (1 - 2^{-k}) \\ &\geq \end{aligned}$$

Analysis (contd.)

Proof.

1. We will show that $\frac{n_1+n_2}{2} \geq \frac{3}{4} \cdot \sum_j \hat{z}_j$.
2. Let S^k denote the set of clauses that contain k literals.
3. As per the variable rounding algorithm,

$$\begin{aligned} n_1 &= \sum_k \sum_{C_j \in S^k} (1 - 2^{-k}) \\ &\geq \sum_k \sum_{C_j \in S^k} (1 - 2^{-k}) \cdot \hat{z}_j. \end{aligned}$$

Analysis (contd.)

Proof.

1. We will show that $\frac{n_1+n_2}{2} \geq \frac{3}{4} \cdot \sum_j \hat{z}_j$.
2. Let S^k denote the set of clauses that contain k literals.
3. As per the variable rounding algorithm,

$$\begin{aligned} n_1 &= \sum_k \sum_{C_j \in S^k} (1-2^{-k}) \\ &\geq \sum_k \sum_{C_j \in S^k} (1-2^{-k}) \cdot \hat{z}_j. \end{aligned}$$

4. As per the LP-based randomized rounding algorithm,

Analysis (contd.)

Proof.

1. We will show that $\frac{n_1+n_2}{2} \geq \frac{3}{4} \cdot \sum_j \hat{z}_j$.
2. Let S^k denote the set of clauses that contain k literals.
3. As per the variable rounding algorithm,

$$\begin{aligned} n_1 &= \sum_k \sum_{C_j \in S^k} (1 - 2^{-k}) \\ &\geq \sum_k \sum_{C_j \in S^k} (1 - 2^{-k}) \cdot \hat{z}_j. \end{aligned}$$

4. As per the LP-based randomized rounding algorithm,

$$n_2 \geq$$

Analysis (contd.)

Proof.

1. We will show that $\frac{n_1+n_2}{2} \geq \frac{3}{4} \cdot \sum_j \hat{z}_j$.
2. Let S^k denote the set of clauses that contain k literals.
3. As per the variable rounding algorithm,

$$\begin{aligned} n_1 &= \sum_k \sum_{C_j \in S^k} (1 - 2^{-k}) \\ &\geq \sum_k \sum_{C_j \in S^k} (1 - 2^{-k}) \cdot \hat{z}_j. \end{aligned}$$

4. As per the LP-based randomized rounding algorithm,

$$n_2 \geq \sum_k \sum_{C_j \in S^k} \beta_k \cdot \hat{z}_j.$$

□

Final Steps

Final Steps

Proof.

Final Steps

Proof.

1. It follows that

Final Steps

Proof.

1. It follows that

$$\frac{n_1 + n_2}{2} \geq$$

Final Steps

Proof.

1. It follows that

$$\frac{n_1 + n_2}{2} \geq \sum_k \sum_{C_j \in S^k} \frac{(1 - 2^{-k}) + \beta_k}{2} \cdot \hat{z}_j.$$

Final Steps

Proof.

1. It follows that

$$\frac{n_1 + n_2}{2} \geq \sum_k \sum_{C_j \in S^k} \frac{(1 - 2^{-k}) + \beta_k}{2} \cdot \hat{z}_j.$$

2. It is not hard to verify that

Final Steps

Proof.

1. It follows that

$$\frac{n_1 + n_2}{2} \geq \sum_k \sum_{C_j \in S^k} \frac{(1 - 2^{-k}) + \beta_k}{2} \cdot \hat{z}_j.$$

2. It is not hard to verify that $((1 - \frac{1}{2^k}) + \beta_k) \geq \frac{3}{2}$, for all k.

Final Steps

Proof.

1. It follows that

$$\frac{n_1 + n_2}{2} \geq \sum_k \sum_{C_j \in S^k} \frac{(1 - 2^{-k}) + \beta_k}{2} \cdot \hat{z}_j.$$

2. It is not hard to verify that $((1 - \frac{1}{2^k}) + \beta_k) \geq \frac{3}{2}$, for all k.
3. Therefore,

Final Steps

Proof.

1. It follows that

$$\frac{n_1+n_2}{2} \geq \sum_k \sum_{C_j \in S^k} \frac{(1-2^{-k})+\beta_k}{2} \cdot \hat{z}_j.$$

2. It is not hard to verify that $((1-\frac{1}{2^k})+\beta_k) \geq \frac{3}{2}$, for all k.
3. Therefore,

$$\frac{n_1+n_2}{2} \quad \geq$$

Final Steps

Proof.

1. It follows that

$$\frac{n_1 + n_2}{2} \geq \sum_k \sum_{C_j \in S^k} \frac{(1 - 2^{-k}) + \beta_k}{2} \cdot \hat{z}_j.$$

2. It is not hard to verify that $((1 - \frac{1}{2^k}) + \beta_k) \geq \frac{3}{2}$, for all k.
3. Therefore,

$$\frac{n_1 + n_2}{2} \geq \sum_k \sum_{C_j \in S^k} (\frac{\frac{3}{2}}{2}) \cdot \hat{z}_j$$

Final Steps

Proof.

1. It follows that

$$\frac{n_1+n_2}{2} \geq \sum_k \sum_{C_j \in S^k} \frac{(1-2^{-k})+\beta_k}{2} \cdot \hat{z}_j.$$

2. It is not hard to verify that $((1-\frac{1}{2^k})+\beta_k) \geq \frac{3}{2}$, for all k.
3. Therefore,

$$\begin{aligned} \frac{n_1+n_2}{2} &\geq \sum_k \sum_{C_j \in S^k} (\frac{\frac{3}{2}}{2}) \cdot \hat{z}_j \\ &= \end{aligned}$$

Final Steps

Proof.

1. It follows that

$$\frac{n_1+n_2}{2} \geq \sum_k \sum_{C_j \in S^k} \frac{(1-2^{-k})+\beta_k}{2} \cdot \hat{z}_j.$$

2. It is not hard to verify that $((1-\frac{1}{2^k})+\beta_k) \geq \frac{3}{2}$, for all k.
3. Therefore,

$$\begin{aligned} \frac{n_1+n_2}{2} &\geq \sum_k \sum_{C_j \in S^k} (\frac{\frac{3}{2}}{2}) \cdot \hat{z}_j \\ &= \frac{3}{4} \cdot \sum_k \sum_{C_j \in S^k} \hat{z}_j \end{aligned}$$

Final Steps

Proof.

1. It follows that

$$\frac{n_1 + n_2}{2} \geq \sum_k \sum_{C_j \in S^k} \frac{(1 - 2^{-k}) + \beta_k}{2} \cdot \hat{z}_j.$$

2. It is not hard to verify that $((1 - \frac{1}{2^k}) + \beta_k) \geq \frac{3}{2}$, for all k.
3. Therefore,

$$\begin{aligned} \frac{n_1 + n_2}{2} &\geq \sum_k \sum_{C_j \in S^k} (\frac{\frac{3}{2}}{2}) \cdot \hat{z}_j \\ &= \frac{3}{4} \cdot \sum_k \sum_{C_j \in S^k} \hat{z}_j \\ &= \end{aligned}$$

Final Steps

Proof.

1. It follows that

$$\frac{n_1+n_2}{2} \geq \sum_k \sum_{C_j \in S^k} \frac{(1-2^{-k})+\beta_k}{2} \cdot \hat{z}_j.$$

2. It is not hard to verify that $((1-\frac{1}{2^k})+\beta_k) \geq \frac{3}{2}$, for all k.
3. Therefore,

$$\begin{aligned}\frac{n_1+n_2}{2} &\geq \sum_k \sum_{C_j \in S^k} \left(\frac{\frac{3}{2}}{2}\right) \cdot \hat{z}_j \\ &= \frac{3}{4} \cdot \sum_k \sum_{C_j \in S^k} \hat{z}_j \\ &= \frac{3}{4} \cdot \sum_j \hat{z}_j\end{aligned}$$

Final Steps

Proof.

1. It follows that

$$\frac{n_1+n_2}{2} \geq \sum_k \sum_{C_j \in S^k} \frac{(1-2^{-k})+\beta_k}{2} \cdot \hat{z}_j.$$

2. It is not hard to verify that $((1-\frac{1}{2^k})+\beta_k) \geq \frac{3}{2}$, for all k.
3. Therefore,

$$\begin{aligned}\frac{n_1+n_2}{2} &\geq \sum_k \sum_{C_j \in S^k} (\frac{\frac{3}{2}}{2}) \cdot \hat{z}_j \\ &= \frac{3}{4} \cdot \sum_k \sum_{C_j \in S^k} \hat{z}_j \\ &= \frac{3}{4} \cdot \sum_j \hat{z}_j \\ &\geq \end{aligned}$$

Final Steps

Proof.

1. It follows that

$$\frac{n_1 + n_2}{2} \geq \sum_k \sum_{C_j \in S^k} \frac{(1 - 2^{-k}) + \beta_k}{2} \cdot \hat{z}_j.$$

2. It is not hard to verify that $((1 - \frac{1}{2^k}) + \beta_k) \geq \frac{3}{2}$, for all k.
3. Therefore,

$$\begin{aligned} \frac{n_1 + n_2}{2} &\geq \sum_k \sum_{C_j \in S^k} (\frac{\frac{3}{2}}{2}) \cdot \hat{z}_j \\ &= \frac{3}{4} \cdot \sum_k \sum_{C_j \in S^k} \hat{z}_j \\ &= \frac{3}{4} \cdot \sum_j \hat{z}_j \\ &\geq \frac{3}{4} \cdot OPT. \end{aligned}$$

□