Approximation Algorithms

Lecture 11:
MaxSat via Randomized Rounding

Part I:
Maximum Satisfiability (MaxSat)

Philipp Kindermann

Summer Semester 2020

Maximum Satisfiability (MaxSat)

Given: Boolean variables $x_1, \ldots, x_n$,

Maximum Satisfiability (MaxSat)

Given: Boolean variables $x_1, \ldots, x_n$,
clauses $C_1, \ldots, C_m$

Maximum Satisfiability (MaxSat)

Given: Boolean variables $x_1, \ldots, x_n$,
clauses $C_1, \ldots, C_m$ with weight $w_1, \ldots, w_m$.

Maximum Satisfiability (MaxSat)

Given: Boolean variables $x_1, \ldots, x_n$,
clauses $C_1, \ldots, C_m$ with weight $w_1, \ldots, w_m$.

Task: Find an assignment of the variables $x_1, \ldots, x_n$

Maximum Satisfiability (MaxSat)

Given: Boolean variables $x_1, \ldots, x_n$,
clauses $C_1, \ldots, C_m$ with weight $w_1, \ldots, w_m$.

Task: Find an assignment of the variables $x_1, \ldots, x_n$ such that the total weight of the *satisfied* clauses is **maximized**.

Maximum Satisfiability (MaxSat)

Given: Boolean variables $x_1, \ldots, x_n$,
clauses $C_1, \ldots, C_m$ with weight $w_1, \ldots, w_m$.

Task: Find an assignment of the variables $x_1, \ldots, x_n$ such that the total weight of the *satisfied* clauses is **maximized**.

Literal: Variable or negation of variable – e.g. x_1, $\overline{x_1}$

Maximum Satisfiability (MaxSat)

Given: Boolean variables $x_1, \ldots, x_n$,
clauses $C_1, \ldots, C_m$ with weight $w_1, \ldots, w_m$.

Task: Find an assignment of the variables $x_1, \ldots, x_n$ such that the total weight of the *satisfied* clauses is **maximized**.

Literal: Variable or negation of variable – e.g. x_1, $\overline{x_1}$

Clause: Disjunction of literals – e.g. $x_1 \vee \overline{x_2} \vee x_3$

Maximum Satisfiability (MaxSat)

Given: Boolean variables $x_1, \ldots, x_n$,
clauses $C_1, \ldots, C_m$ with weight $w_1, \ldots, w_m$.

Task: Find an assignment of the variables $x_1, \ldots, x_n$ such that the total weight of the *satisfied* clauses is **maximized**.

Literal: Variable or negation of variable – e.g. x_1, $\overline{x_1}$

Clause: Disjunction of literals – e.g. $x_1 \vee \overline{x_2} \vee x_3$

Length of a clause: Number of literals

Maximum Satisfiability (MaxSat)

Given: Boolean variables $x_1, \ldots, x_n$,
clauses $C_1, \ldots, C_m$ with weight $w_1, \ldots, w_m$.

Task: Find an assignment of the variables $x_1, \ldots, x_n$ such that the total weight of the *satisfied* clauses is **maximized**.

Literal: Variable or negation of variable – e.g. x_1, $\overline{x_1}$

Clause: Disjunction of literals – e.g. $x_1 \vee \overline{x_2} \vee x_3$

Length of a clause: Number of literals

Problem is NP-hard since Satisfiability (Sat) is NP-hard: Is a given propositional formula (in conjunctive normal form) satisfiable?

Maximum Satisfiability (MaxSat)

Given: Boolean variables $x_1, \dots, x_n$,
clauses $C_1, \dots, C_m$ with weight $w_1, \dots, w_m$.

Task: Find an assignment of the variables $x_1, \dots, x_n$ such that the total weight of the *satisfied* clauses is **maximized**.

Literal: Variable or negation of variable – e.g. x_1, $\overline{x_1}$

Clause: Disjunction of literals – e.g. $x_1 \vee \overline{x_2} \vee x_3$

Length of a clause: Number of literals

Problem is NP-hard since Satisfiability (Sat) is NP-hard: Is a given propositional formula (in conjunctive normal form) satisfiable? E.g. $(x_1 \vee \overline{x_2} \vee x_3) \wedge (x_2 \vee \overline{x_3} \vee x_4) \wedge (x_1 \vee \overline{x_4})$.

Approximation Algorithms

Lecture 11:
MaxSat via Randomized Rounding

Part II:
A Simple Randomized Algorithm

Philipp Kindermann

Summer Semester 2020

A Simple Randomized Algorithm

Theorem. Independently setting each variable to 1 (`true`) with probability 1/2 provides an expected -approximation for MaxSat.

A Simple Randomized Algorithm

Theorem. Independently setting each variable to 1 (`true`) with probability 1/2 provides an expected 1/2-approximation for MaxSat.

A Simple Randomized Algorithm

Theorem. Independently setting each variable to 1 (`true`) with probability 1/2 provides an expected 1/2-approximation for MaxSat.

Proof.

Let $Y_j \in \{0, 1\}$ be random variable for the truth value of clause C_j.

A Simple Randomized Algorithm

Theorem. Independently setting each variable to 1 (`true`) with probability 1/2 provides an expected 1/2-approximation for MaxSat.

Proof.

Let $Y_j \in \{0,1\}$ be random variable for the truth value of clause C_j.

Let W be random variable for the weight of satisfied clauses.

A Simple Randomized Algorithm

Theorem. Independently setting each variable to 1 (`true`) with probability 1/2 provides an expected 1/2-approximation for MaxSat.

Proof.

Let $Y_j \in \{0, 1\}$ be random variable for the truth value of clause C_j.

Let W be random variable for the weight of satisfied clauses.

$$E[W] =$$

A Simple Randomized Algorithm

Theorem. Independently setting each variable to 1 (`true`) with probability 1/2 provides an expected 1/2-approximation for MaxSat.

Proof.

Let $Y_j \in \{0, 1\}$ be random variable for the truth value of clause C_j.

Let W be random variable for the weight of satisfied clauses.

$$E[W] = E\left[\sum_{j=1}^{m} w_j Y_j\right] =$$

A Simple Randomized Algorithm

Theorem. Independently setting each variable to 1 (`true`) with probability 1/2 provides an expected 1/2-approximation for MaxSat.

Proof.

Let $Y_j \in \{0,1\}$ be random variable for the truth value of clause C_j.

Let W be random variable for the weight of satisfied clauses.

$$E[W] = E\left[\sum_{j=1}^{m} w_j Y_j\right] = \sum_{j=1}^{m} w_j E[Y_j] =$$

A Simple Randomized Algorithm

Theorem. Independently setting each variable to 1 (`true`) with probability 1/2 provides an expected 1/2-approximation for MaxSat.

Proof.

Let $Y_j \in \{0,1\}$ be random variable for the truth value of clause C_j.

Let W be random variable for the weight of satisfied clauses.

$$E[W] = E\left[\sum_{j=1}^{m} w_j Y_j\right] = \sum_{j=1}^{m} w_j E[Y_j] = \sum_{j=1}^{m} w_j \Pr[C_j \text{ satisfied}]$$

A Simple Randomized Algorithm

Theorem. Independently setting each variable to 1 (`true`) with probability 1/2 provides an expected 1/2-approximation for MaxSat.

Proof.

Let $Y_j \in \{0, 1\}$ be random variable for the truth value of clause C_j.

Let W be random variable for the weight of satisfied clauses.

$$E[W] = E\left[\sum_{j=1}^{m} w_j Y_j\right] = \sum_{j=1}^{m} w_j E[Y_j] = \sum_{j=1}^{m} w_j \Pr[C_j \text{ satisfied}]$$

Let l_j be length of C_j.

A Simple Randomized Algorithm

Theorem. Independently setting each variable to 1 (`true`) with probability 1/2 provides an expected 1/2-approximation for MaxSat.

Proof.

Let $Y_j \in \{0, 1\}$ be random variable for the truth value of clause C_j.

Let W be random variable for the weight of satisfied clauses.

$$E[W] = E\left[\sum_{j=1}^{m} w_j Y_j\right] = \sum_{j=1}^{m} w_j E[Y_j] = \sum_{j=1}^{m} w_j \Pr[C_j \text{ satisfied}]$$

Let l_j be length of C_j. $\Rightarrow \Pr[C_j \text{ satisfied}] =$

A Simple Randomized Algorithm

Theorem. Independently setting each variable to 1 (`true`) with probability 1/2 provides an expected 1/2-approximation for MaxSat.

Proof.

Let $Y_j \in \{0, 1\}$ be random variable for the truth value of clause C_j.

Let W be random variable for the weight of satisfied clauses.

$$E[W] = E\left[\sum_{j=1}^{m} w_j Y_j\right] = \sum_{j=1}^{m} w_j E[Y_j] = \sum_{j=1}^{m} w_j \Pr[C_j \text{ satisfied}]$$

Let l_j be length of C_j. $\Rightarrow \Pr[C_j \text{ satisfied}] = 1 - (1/2)^{l_j} \geq$

A Simple Randomized Algorithm

Theorem. Independently setting each variable to 1 (`true`) with probability 1/2 provides an expected 1/2-approximation for MaxSat.

Proof.

Let $Y_j \in \{0,1\}$ be random variable for the truth value of clause C_j.

Let W be random variable for the weight of satisfied clauses.

$$E[W] = E\left[\sum_{j=1}^{m} w_j Y_j\right] = \sum_{j=1}^{m} w_j E[Y_j] = \sum_{j=1}^{m} w_j \Pr[C_j \text{ satisfied}]$$

Let l_j be length of C_j. $\Rightarrow \Pr[C_j \text{ satisfied}] = 1 - (1/2)^{l_j} \geq 1/2.$

A Simple Randomized Algorithm

Theorem. Independently setting each variable to 1 (`true`) with probability 1/2 provides an expected 1/2-approximation for MaxSat.

Proof.

Let $Y_j \in \{0, 1\}$ be random variable for the truth value of clause C_j.

Let W be random variable for the weight of satisfied clauses.

$$E[W] = E\left[\sum_{j=1}^{m} w_j Y_j\right] = \sum_{j=1}^{m} w_j E[Y_j] = \sum_{j=1}^{m} w_j \Pr[C_j \text{ satisfied}]$$

Let l_j be length of C_j. $\Rightarrow \Pr[C_j \text{ satisfied}] = 1 - (1/2)^{l_j} \geq 1/2$.

Thus, $E[W] \geq$

A Simple Randomized Algorithm

Theorem. Independently setting each variable to 1 (`true`) with probability 1/2 provides an expected 1/2-approximation for MaxSat.

Proof.

Let $Y_j \in \{0, 1\}$ be random variable for the truth value of clause C_j.

Let W be random variable for the weight of satisfied clauses.

$$E[W] = E\left[\sum_{j=1}^{m} w_j Y_j\right] = \sum_{j=1}^{m} w_j E[Y_j] = \sum_{j=1}^{m} w_j \Pr[C_j \text{ satisfied}]$$

Let l_j be length of C_j. $\Rightarrow \Pr[C_j \text{ satisfied}] = 1 - (1/2)^{l_j} \geq 1/2$.

Thus, $E[W] \geq 1/2 \sum_{j=1}^{m} w_j \geq$

A Simple Randomized Algorithm

Theorem. Independently setting each variable to 1 (`true`) with probability $1/2$ provides an expected $1/2$-approximation for MaxSat.

Proof.

Let $Y_j \in \{0, 1\}$ be random variable for the truth value of clause C_j.

Let W be random variable for the weight of satisfied clauses.

$$E[W] = E\left[\sum_{j=1}^{m} w_j Y_j\right] = \sum_{j=1}^{m} w_j E[Y_j] = \sum_{j=1}^{m} w_j \Pr[C_j \text{ satisfied}]$$

Let l_j be length of C_j. $\Rightarrow \Pr[C_j \text{ satisfied}] = 1 - (1/2)^{l_j} \geq 1/2$.

Thus, $E[W] \geq 1/2 \sum_{j=1}^{m} w_j \geq \text{OPT}/2$. $\square$

Approximation Algorithms

Lecture 11:
MaxSat via Randomized Rounding

Part III:
Derandomization by Conditional Expectation

Philipp Kindermann

Summer Semester 2020

Derandomization by Conditional Expectation

Theorem. The previous algorithm can be derandomized, i.e., there is a deterministic $1/2$-approximation algorithm for MaxSat.

Derandomization by Conditional Expectation

Theorem. The previous algorithm can be derandomized, i.e., there is a deterministic $1/2$-approximation algorithm for MaxSat.

Proof.

We set x_1 deterministically, but $x_2, \ldots, x_n$ randomly.

Derandomization by Conditional Expectation

Theorem. The previous algorithm can be derandomized, i.e., there is a deterministic $1/2$-approximation algorithm for MaxSat.

Proof.

We set x_1 deterministically, but $x_2, \ldots, x_n$ randomly.

Namely: set $x_1 = 1 \Leftrightarrow E[W | x_1 = 1] \geq E[W | x_1 = 0]$.

Derandomization by Conditional Expectation

Theorem. The previous algorithm can be derandomized, i.e., there is a deterministic $1/2$-approximation algorithm for MaxSat.

Proof.

We set x_1 deterministically, but $x_2, \ldots, x_n$ randomly.

Namely: set $x_1 = 1 \Leftrightarrow E[W|x_1 = 1] \geq E[W|x_1 = 0]$.

$E[W] =$

Derandomization by Conditional Expectation

Theorem. The previous algorithm can be derandomized, i.e., there is a deterministic $1/2$-approximation algorithm for MaxSat.

Proof.

We set x_1 deterministically, but $x_2, \ldots, x_n$ randomly.

Namely: set $x_1 = 1 \Leftrightarrow E[W|x_1 = 1] \geq E[W|x_1 = 0]$.

$$E[W] = (E[W|x_1 = 0] + E[W|x_1 = 1])/2.$$

Derandomization by Conditional Expectation

Theorem. The previous algorithm can be derandomized, i.e., there is a deterministic $1/2$-approximation algorithm for MaxSat.

Proof.

We set x_1 deterministically, but $x_2, \ldots, x_n$ randomly.

Namely: set $x_1 = 1 \Leftrightarrow E[W|x_1 = 1] \geq E[W|x_1 = 0]$.

$E[W] = (E[W|x_1 = 0] + E[W|x_1 = 1])/2$. [because of original random choice of x_1]

Derandomization by Conditional Expectation

Theorem. The previous algorithm can be derandomized, i.e., there is a deterministic $1/2$-approximation algorithm for MaxSat.

Proof.

We set x_1 deterministically, but $x_2, \ldots, x_n$ randomly.

Namely: set $x_1 = 1 \Leftrightarrow E[W|x_1 = 1] \geq E[W|x_1 = 0]$.

$$E[W] = (E[W|x_1 = 0] + E[W|x_1 = 1])/2.$$

[because of original random choice of x_1]

If x_1 was set to $b_1 \in \{0, 1\}$,

Derandomization by Conditional Expectation

Theorem. The previous algorithm can be derandomized, i.e., there is a deterministic $1/2$-approximation algorithm for MaxSat.

Proof.

We set x_1 deterministically, but $x_2, \dots, x_n$ randomly.

Namely: set $x_1 = 1 \iff E[W|x_1 = 1] \geq E[W|x_1 = 0]$.

$$E[W] = (E[W|x_1 = 0] + E[W|x_1 = 1])/2.$$

[because of original random choice of x_1]

If x_1 was set to $b_1 \in \{0, 1\}$,
then $E[W|x_1 = b_1] \geq$

Derandomization by Conditional Expectation

Theorem. The previous algorithm can be derandomized, i.e., there is a deterministic $1/2$-approximation algorithm for MaxSat.

Proof.

We set x_1 deterministically, but $x_2, \dots, x_n$ randomly.

Namely: set $x_1 = 1 \Leftrightarrow E[W|x_1 = 1] \geq E[W|x_1 = 0]$.

$E[W] = (E[W|x_1 = 0] + E[W|x_1 = 1])/2.$ [because of original random choice of x_1]

If x_1 was set to $b_1 \in \{0, 1\}$,
then $E[W|x_1 = b_1] \geq E[W] \geq$

Derandomization by Conditional Expectation

Theorem. The previous algorithm can be derandomized, i.e., there is a deterministic $1/2$-approximation algorithm for MaxSat.

Proof.

We set x_1 deterministically, but $x_2, \ldots, x_n$ randomly.

Namely: set $x_1 = 1 \Leftrightarrow E[W|x_1 = 1] \geq E[W|x_1 = 0]$.

$E[W] = (E[W|x_1 = 0] + E[W|x_1 = 1])/2.$ [because of original random choice of x_1]

If x_1 was set to $b_1 \in \{0,1\}$,
then $E[W|x_1 = b_1] \geq E[W] \geq \text{OPT}/2$.

Derandomization by Conditional Expectation

Assume (by induction) that we have set $x_1, \ldots, x_i$ to $b_1, \ldots, b_i$ such that

Derandomization by Conditional Expectation

Assume (by induction) that we have set $x_1, \ldots, x_i$ to $b_1, \ldots, b_i$ such that

$$E[W | x_1 = b_1, \ldots, x_i = b_i] \geq \mathrm{OPT}/2$$

Derandomization by Conditional Expectation

Assume (by induction) that we have set $x_1, \ldots, x_i$ to $b_1, \ldots, b_i$ such that

$$E[W | x_1 = b_1, \ldots, x_i = b_i] \geq \text{OPT}/2$$

Then (similar to the base case):

Derandomization by Conditional Expectation

Assume (by induction) that we have set $x_1, \ldots, x_i$ to $b_1, \ldots, b_i$ such that

$$E[W | x_1 = b_1, \ldots, x_i = b_i] \geq \mathrm{OPT}/2$$

Then (similar to the base case):

$$\big(E[W | x_1 = b_1, \ldots, x_i = b_i, x_{i+1} = 0] \\ + E[W | x_1 = b_1, \ldots, x_i = b_i, x_{i+1} = 1]\big)/2$$

Derandomization by Conditional Expectation

Assume (by induction) that we have set $x_1, \ldots, x_i$ to $b_1, \ldots, b_i$ such that

$$E[W | x_1 = b_1, \ldots, x_i = b_i] \geq \text{OPT}/2$$

Then (similar to the base case):

$$\big(E[W | x_1 = b_1, \ldots, x_i = b_i, x_{i+1} = 0]$$
$$+ E[W | x_1 = b_1, \ldots, x_i = b_i, x_{i+1} = 1]\big)/2$$
$$= E[W | x_1 = b_1, \ldots, x_i = b_i] \geq \text{OPT}/2$$

Derandomization by Conditional Expectation

Assume (by induction) that we have set $x_1, \ldots, x_i$ to $b_1, \ldots, b_i$ such that

$$E[W | x_1 = b_1, \ldots, x_i = b_i] \geq \mathrm{OPT}/2$$

Then (similar to the base case):

$$\big(E[W | x_1 = b_1, \ldots, x_i = b_i, x_{i+1} = 0]$$
$$+ E[W | x_1 = b_1, \ldots, x_i = b_i, x_{i+1} = 1]\big)/2$$
$$= E[W | x_1 = b_1, \ldots, x_i = b_i] \geq \mathrm{OPT}/2$$

So we set $x_{i+1} = 1 \Leftrightarrow$

Derandomization by Conditional Expectation

Assume (by induction) that we have set $x_1, \ldots, x_i$ to $b_1, \ldots, b_i$ such that

$$E[W|x_1 = b_1, \ldots, x_i = b_i] \geq \text{OPT}/2$$

Then (similar to the base case):

$$\big(E[W|x_1 = b_1, \ldots, x_i = b_i, x_{i+1} = 0]$$

$$+E[W|x_1 = b_1, \ldots, x_i = b_i, x_{i+1} = 1]\big)/2$$

$$= E[W|x_1 = b_1, \ldots, x_i = b_i] \geq \text{OPT}/2$$

So we set $x_{i+1} = 1 \Leftrightarrow$

$$E[W|x_1 = b_1, \ldots, x_i = b_i, x_{i+1} = 1]$$

$$\geq E[W|x_1 = b_1, \ldots, x_i = b_i, x_{i+1} = 0]$$

Derandomization by Conditional Expectation

Thus, the algorithm can be derandomized if the conditional expectation can be computed efficiently!

Derandomization by Conditional Expectation

Thus, the algorithm can be derandomized if the conditional expectation can be computed efficiently!

Consider a partial assignment $x_1 = b_1, \ldots, x_i = b_i$ and a clause C_j.

Derandomization by Conditional Expectation

Thus, the algorithm can be derandomized if the conditional expectation can be computed efficiently!

Consider a partial assignment $x_1 = b_1, \ldots, x_i = b_i$ and a clause C_j.

If C_j is already satisfied, then it contributes exactly to $E[W | x_1 = b_1, \ldots, x_i = b_i]$.

Derandomization by Conditional Expectation

Thus, the algorithm can be derandomized if the conditional expectation can be computed efficiently!

Consider a partial assignment $x_1 = b_1, \ldots, x_i = b_i$ and a clause C_j.

If C_j is already satisfied, then it contributes exactly w_j to $E[W | x_1 = b_1, \ldots, x_i = b_i]$.

Derandomization by Conditional Expectation

Thus, the algorithm can be derandomized if the conditional expectation can be computed efficiently!

Consider a partial assignment $x_1 = b_1, \ldots, x_i = b_i$ and a clause C_j.

If C_j is already satisfied, then it contributes exactly w_j to $E[W|x_1 = b_1, \ldots, x_i = b_i]$.

If C_j is not yet satisfied and contains k unassigned variables, then it contributes exactly to $E[W|x_1 = b_1, \ldots, x_i = b_i]$.

Derandomization by Conditional Expectation

Thus, the algorithm can be derandomized if the conditional expectation can be computed efficiently!

Consider a partial assignment $x_1 = b_1, \ldots, x_i = b_i$ and a clause C_j.

If C_j is already satisfied, then it contributes exactly w_j to $E[W|x_1 = b_1, \ldots, x_i = b_i]$.

If C_j is not yet satisfied and contains k unassigned variables, then it contributes exactly $w_j(1 - (1/2)^k)$ to $E[W|x_1 = b_1, \ldots, x_i = b_i]$.

Derandomization by Conditional Expectation

Thus, the algorithm can be derandomized if the conditional expectation can be computed efficiently!

Consider a partial assignment $x_1 = b_1, \ldots, x_i = b_i$ and a clause C_j.

If C_j is already satisfied, then it contributes exactly w_j to $E[W | x_1 = b_1, \ldots, x_i = b_i]$.

If C_j is not yet satisfied and contains k unassigned variables, then it contributes exactly $w_j(1 - (1/2)^k)$ to $E[W | x_1 = b_1, \ldots, x_i = b_i]$.

The conditional expectation is simply the sum of the contributions from each clause. $\square$

Summary

Standard procedure with which many randomized algorithms can be derandomized.

Summary

Standard procedure with which many randomized algorithms can be derandomized.

Requirement: respective conditional probabilities can be appropriately estimated for each random decision.

Summary

Standard procedure with which many randomized algorithms can be derandomized.

Requirement: respective conditional probabilities can be appropriately estimated for each random decision.

The algorithm simply chooses the best option at each step.

Summary

Standard procedure with which many randomized algorithms can be derandomized.

Requirement: respective conditional probabilities can be appropriately estimated for each random decision.

The algorithm simply chooses the best option at each step.

Quality of the obtained solution is then at least as high as the expected value.

Summary

Standard procedure with which many randomized algorithms can be derandomized.

Requirement: respective conditional probabilities can be appropriately estimated for each random decision.

The algorithm simply chooses the best option at each step.

Quality of the obtained solution is then at least as high as the expected value.

The algorithm iteratively sets the variables and greedily decides for the locally best assignment.

Summary

Standard procedure with which many randomized algorithms can be derandomized.

Requirement: respective conditional probabilities can be appropriately estimated for each random decision.

The algorithm simply chooses the best option at each step.

Quality of the obtained solution is then at least as high as the expected value.

The algorithm iteratively sets the variables and greedily decides for the locally best assignment.

Global optimization?

Approximation Algorithms

Lecture 11:
MaxSat via Randomized Rounding

Part IV:
Randomized Rounding

Philipp Kindermann Summer Semester 2020

An ILP

maximize

subject to

where $C_j = \bigvee_{i \in P_j} x_i \vee \bigvee_{i \in N_j} \bar{x}_i$ for $j = 1, \ldots, m$.

An ILP

maximize

subject to

$$y_i \in \{0, 1\}, \qquad \text{for } i = 1, \ldots, n$$

where $C_j = \bigvee_{i \in P_j} x_i \vee \bigvee_{i \in N_j} \bar{x}_i$ for $j = 1, \ldots, m$.

An ILP

maximize

subject to

$$y_i \in \{0,1\}, \qquad \text{for } i = 1,\ldots,n$$
$$z_j \in \{0,1\}, \qquad \text{for } j = 1,\ldots,m$$

where $C_j = \bigvee_{i \in P_j} x_i \vee \bigvee_{i \in N_j} \bar{x}_i$ for $j = 1,\ldots,m$.

An ILP

maximize $\sum_{j=1}^{m} w_j z_j$

subject to

$$y_i \in \{0,1\}, \qquad \text{for } i = 1,\ldots,n$$

$$z_j \in \{0,1\}, \qquad \text{for } j = 1,\ldots,m$$

where $C_j = \bigvee_{i \in P_j} x_i \vee \bigvee_{i \in N_j} \bar{x}_i$ for $j = 1,\ldots,m$.

An ILP

$$\begin{aligned} \textbf{maximize} \quad & \sum_{j=1}^{m} w_j z_j \\ \textbf{subject to} \quad & \sum_{i \in P_j} \qquad + \sum_{i \in N_j} & \text{for } j = 1, \ldots, m \\ & y_i \in \{0, 1\}, & \text{for } i = 1, \ldots, n \\ & z_j \in \{0, 1\}, & \text{for } j = 1, \ldots, m \end{aligned}$$

where $C_j = \bigvee_{i \in P_j} x_i \vee \bigvee_{i \in N_j} \bar{x}_i \quad \text{for } j = 1, \ldots, m.$

An ILP

$$
\begin{aligned}
\textbf{maximize} \quad & \sum_{j=1}^{m} w_j z_j \\
\textbf{subject to} \quad & \sum_{i \in P_j} y_i + \sum_{i \in N_j} && \text{for } j = 1, \ldots, m \\
& y_i \in \{0, 1\}, && \text{for } i = 1, \ldots, n \\
& z_j \in \{0, 1\}, && \text{for } j = 1, \ldots, m
\end{aligned}
$$

where $C_j = \bigvee_{i \in P_j} x_i \vee \bigvee_{i \in N_j} \bar{x}_i \quad \text{for } j = 1, \ldots, m.$

An ILP

$$
\begin{aligned}
\textbf{maximize} \quad & \sum_{j=1}^{m} w_j z_j \\
\textbf{subject to} \quad & \sum_{i \in P_j} y_i + \sum_{i \in N_j} (1 - y_i) && \text{for } j = 1, \ldots, m \\
& y_i \in \{0, 1\}, && \text{for } i = 1, \ldots, n \\
& z_j \in \{0, 1\}, && \text{for } j = 1, \ldots, m
\end{aligned}
$$

where $C_j = \bigvee_{i \in P_j} x_i \vee \bigvee_{i \in N_j} \bar{x}_i \quad \text{for } j = 1, \ldots, m.$

An ILP

$$\begin{aligned}
&\textbf{maximize} \quad \sum_{j=1}^{m} w_j z_j \\
&\textbf{subject to} \quad \sum_{i \in P_j} y_i + \sum_{i \in N_j} (1 - y_i) \geq \qquad \text{for } j = 1, \ldots, m \\
&\qquad y_i \in \{0, 1\}, \qquad \text{for } i = 1, \ldots, n \\
&\qquad z_j \in \{0, 1\}, \qquad \text{for } j = 1, \ldots, m
\end{aligned}$$

where $C_j = \bigvee_{i \in P_j} x_i \vee \bigvee_{i \in N_j} \bar{x}_i$ for $j = 1, \ldots, m$.

An ILP

$$\begin{aligned} \textbf{maximize} \quad & \sum_{j=1}^{m} w_j z_j \\ \textbf{subject to} \quad & \sum_{i \in P_j} y_i + \sum_{i \in N_j} (1 - y_i) \geq z_j \quad && \text{for } j = 1, \ldots, m \\ & y_i \in \{0, 1\}, && \text{for } i = 1, \ldots, n \\ & z_j \in \{0, 1\}, && \text{for } j = 1, \ldots, m \end{aligned}$$

where $C_j = \bigvee_{i \in P_j} x_i \vee \bigvee_{i \in N_j} \bar{x}_i$ for $j = 1, \ldots, m$.

An ILP

$$\textbf{maximize} \quad \sum_{j=1}^{m} w_j z_j$$

$$\textbf{subject to} \quad \sum_{i \in P_j} y_i + \sum_{i \in N_j} (1 - y_i) \geq z_j \quad \text{for } j = 1, \ldots, m$$

$$y_i \in \{0, 1\}, \quad \text{for } i = 1, \ldots, n$$

~~$z_j \in \{0, 1\}$~~, $0 \leq z_j \leq 1$ for $j = 1, \ldots, m$

where $C_j = \bigvee_{i \in P_j} x_i \vee \bigvee_{i \in N_j} \bar{x}_i$ for $j = 1, \ldots, m$.

An ILP

$$
\begin{aligned}
\textbf{maximize} \quad & \sum_{j=1}^{m} w_j z_j \\
\textbf{subject to} \quad & \boxed{\sum_{i \in P_j} y_i + \sum_{i \in N_j} (1 - y_i)} \geq z_j && \text{for } j = 1, \ldots, m \\
& y_i \in \{0, 1\}, && \text{for } i = 1, \ldots, n \\
& \cancel{z_j \in \{0, 1\}}, \; 0 \leq z_j \leq 1 && \text{for } j = 1, \ldots, m
\end{aligned}
$$

That suffices because [boxed expression] $\in \mathbb{Z}$.

where $C_j = \bigvee_{i \in P_j} x_i \vee \bigvee_{i \in N_j} \bar{x}_i$ for $j = 1, \ldots, m$.

... and its Relaxation

$$\begin{aligned}
\textbf{maximize} \quad & \sum_{j=1}^{m} w_j z_j \\
\textbf{subject to} \quad & \sum_{i \in P_j} y_i + \sum_{i \in N_j} (1 - y_i) \geq z_j && \text{for } j = 1, \ldots, m \\
& 0 \leq y_i \leq 1, && \text{for } i = 1, \ldots, n \\
& 0 \leq z_j \leq 1, && \text{for } j = 1, \ldots, m
\end{aligned}$$

where $C_j = \bigvee_{i \in P_j} x_i \vee \bigvee_{i \in N_j} \bar{x}_i$ for $j = 1, \ldots, m$.

Randomized Rounding

Theorem. Let (y^*, z^*) be an optimal solution to the LP-relaxation.

Randomized Rounding

Theorem. Let (y^*, z^*) be an optimal solution to the LP-relaxation. Independently setting each variable x_i to 1

Randomized Rounding

Theorem. Let (y^*, z^*) be an optimal solution to the LP-relaxation. Independently setting each variable x_i to 1 with probability y_i^* provides a ( )-approximation for MaxSat.

Randomized Rounding

Theorem. Let (y^*, z^*) be an optimal solution to the LP-relaxation. Independently setting each variable x_i to 1 with probability y_i^* provides a $(1 - 1/e)$-approximation for MaxSat.

Randomized Rounding

Theorem. Let (y^*, z^*) be an optimal solution to the LP-relaxation. Independently setting each variable x_i to 1 with probability y_i^* provides a $(1 - 1/e)$-approximation for MaxSat.

$\underbrace{1 - 1/e}_{\approx 0.63}$

Approximation Algorithms

Lecture 11: MaxSat via Randomized Rounding

Part V: Randomized Rounding – Proof

Philipp Kindermann

Summer Semester 2020

Mathematical Toolkit

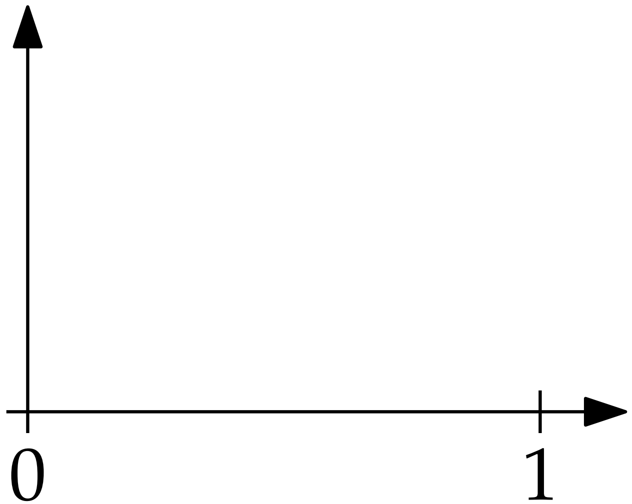
Let f be function that is concave on $[0, 1]$

Mathematical Toolkit

Let f be function that is concave on $[0, 1]$
(i.e. $f''(x) \leq 0$ on $[0, 1]$)

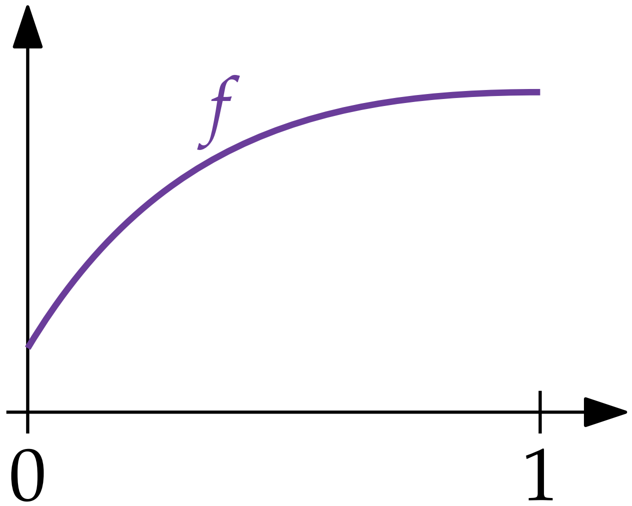
Mathematical Toolkit

Let f be function that is concave on $[0, 1]$
(i.e. $f''(x) \leq 0$ on $[0, 1]$)



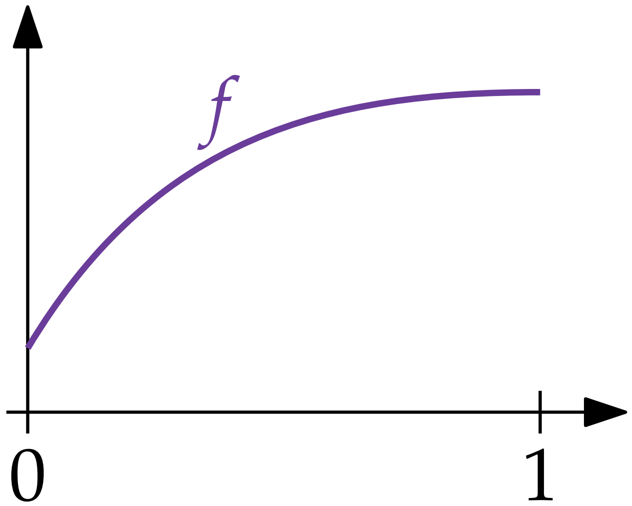
Mathematical Toolkit

Let f be function that is concave on $[0, 1]$
(i.e. $f''(x) \leq 0$ on $[0, 1]$)



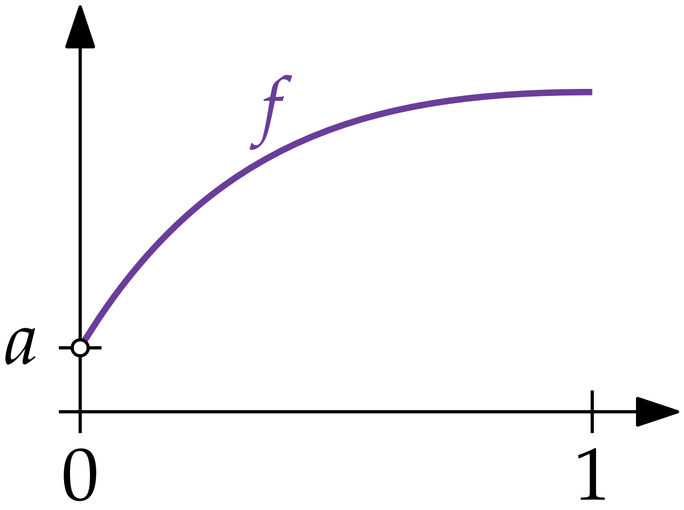
Mathematical Toolkit

Let f be function that is concave on $[0,1]$ (i.e. $f''(x) \leq 0$ on $[0,1]$) with $f(0) = a$



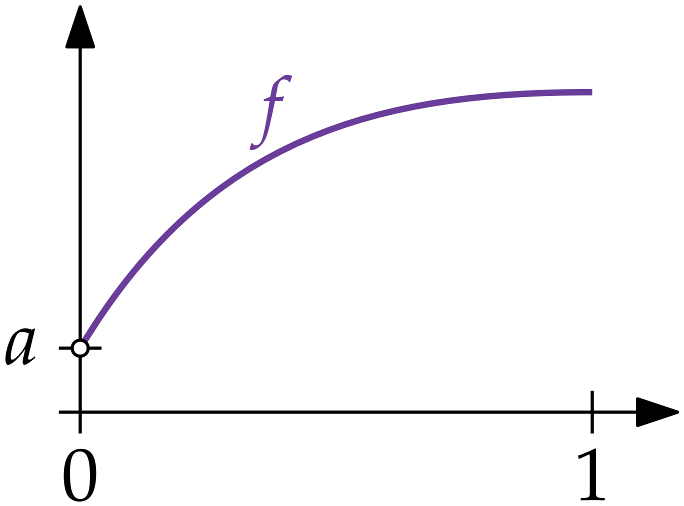
Mathematical Toolkit

Let f be function that is concave on $[0, 1]$ (i.e. $f''(x) \leq 0$ on $[0, 1]$) with $f(0) = a$



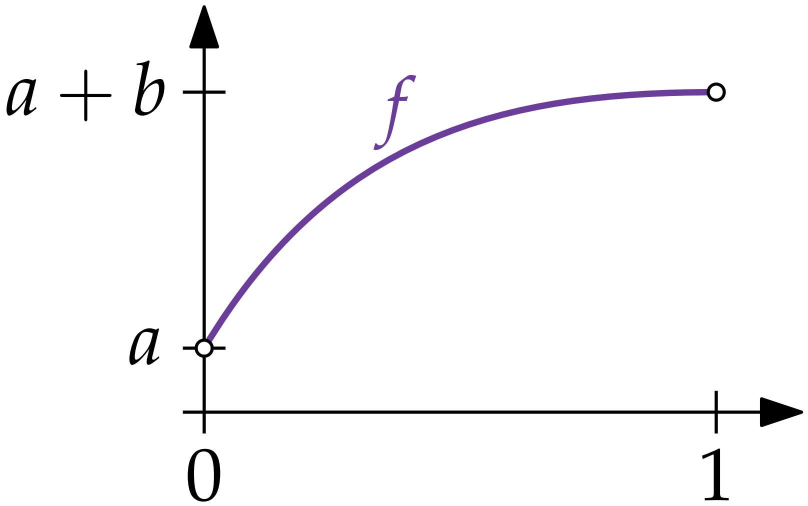
Mathematical Toolkit

Let f be function that is concave on $[0, 1]$
(i.e. $f''(x) \leq 0$ on $[0, 1]$) with $f(0) = a$ and $f(1) = a + b$



Mathematical Toolkit

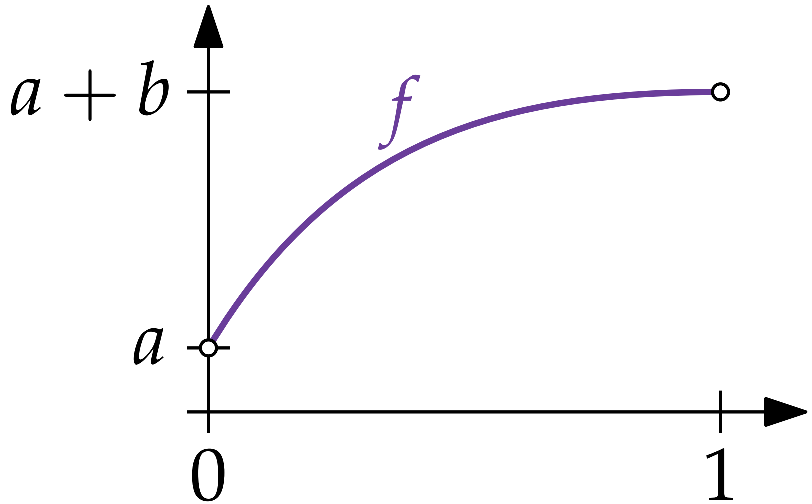
Let f be function that is concave on $[0, 1]$
(i.e. $f''(x) \leq 0$ on $[0, 1]$) with $f(0) = a$ and $f(1) = a + b$



Mathematical Toolkit

Let f be function that is concave on $[0, 1]$
(i.e. $f''(x) \leq 0$ on $[0, 1]$) with $f(0) = a$ and $f(1) = a + b$

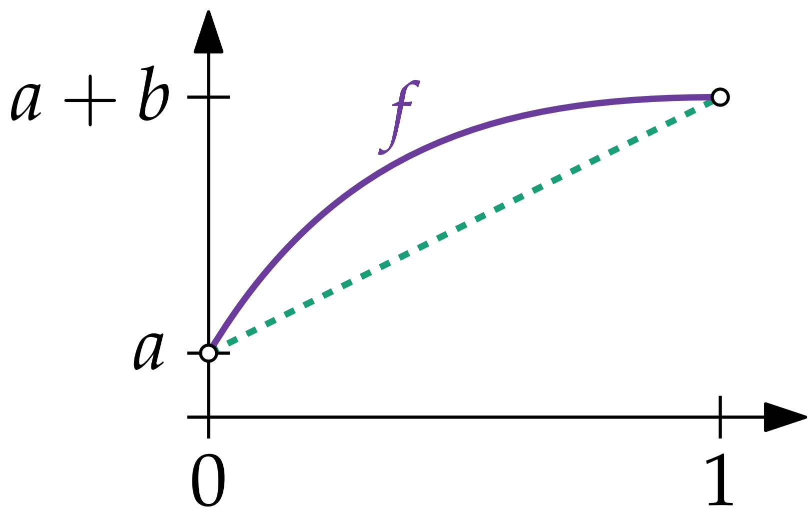
$\Rightarrow f(x) \geq bx + a$ for $x \in [0, 1]$.



Mathematical Toolkit

Let f be function that is concave on $[0,1]$
(i.e. $f''(x) \leq 0$ on $[0,1]$) with $f(0) = a$ and $f(1) = a + b$

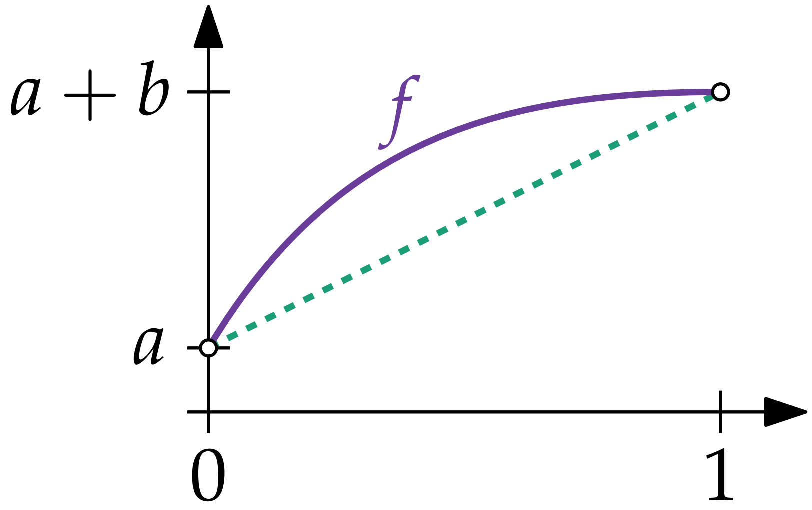
$$\Rightarrow f(x) \geq bx + a \text{ for } x \in [0,1].$$



Mathematical Toolkit

Let f be function that is concave on $[0, 1]$
(i.e. $f''(x) \leq 0$ on $[0, 1]$) with $f(0) = a$ and $f(1) = a + b$

$$\Rightarrow f(x) \geq bx + a \text{ for } x \in [0, 1].$$

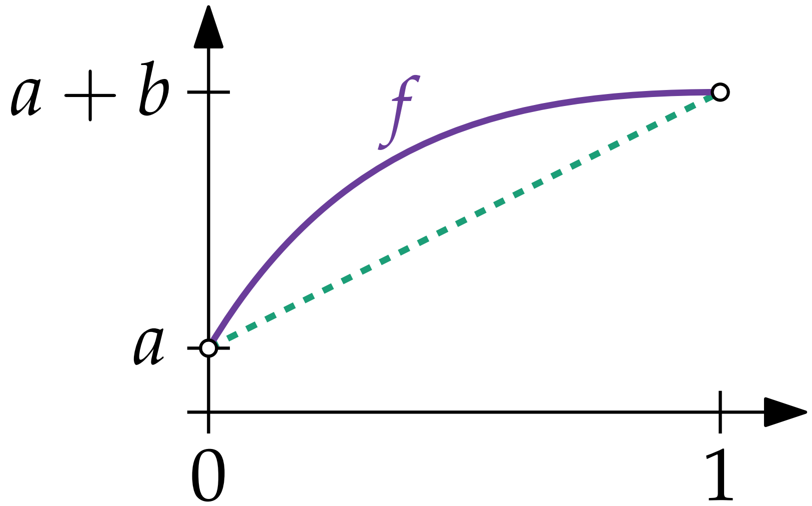


Arithmetic-Geometric Mean Inequality (AGMI):

Mathematical Toolkit

Let f be function that is concave on $[0,1]$
(i.e. $f''(x) \leq 0$ on $[0,1]$) with $f(0) = a$ and $f(1) = a + b$

$\Rightarrow f(x) \geq bx + a$ for $x \in [0,1]$.



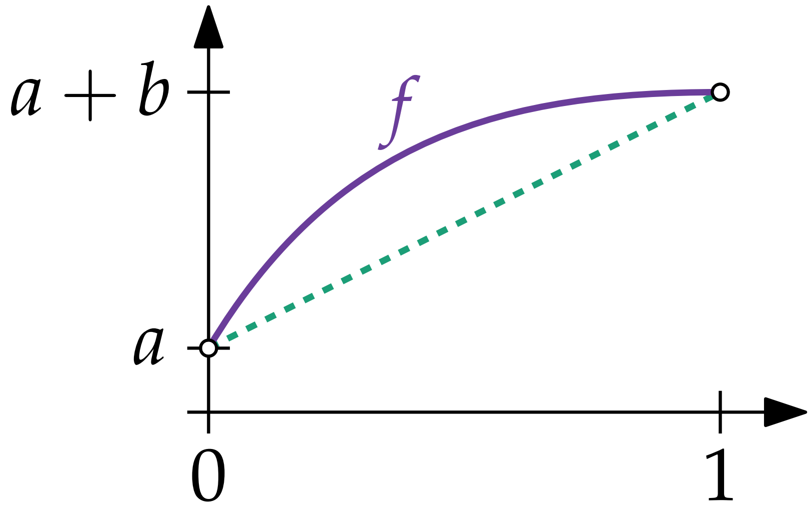
Arithmetic-Geometric Mean Inequality (AGMI):

For all non-negative numbers $a_1, \ldots, a_k$:

Mathematical Toolkit

Let f be function that is concave on $[0,1]$
(i.e. $f''(x) \leq 0$ on $[0,1]$) with $f(0) = a$ and $f(1) = a + b$

$$\Rightarrow f(x) \geq bx + a \text{ for } x \in [0,1].$$



Arithmetic-Geometric Mean Inequality (AGMI):

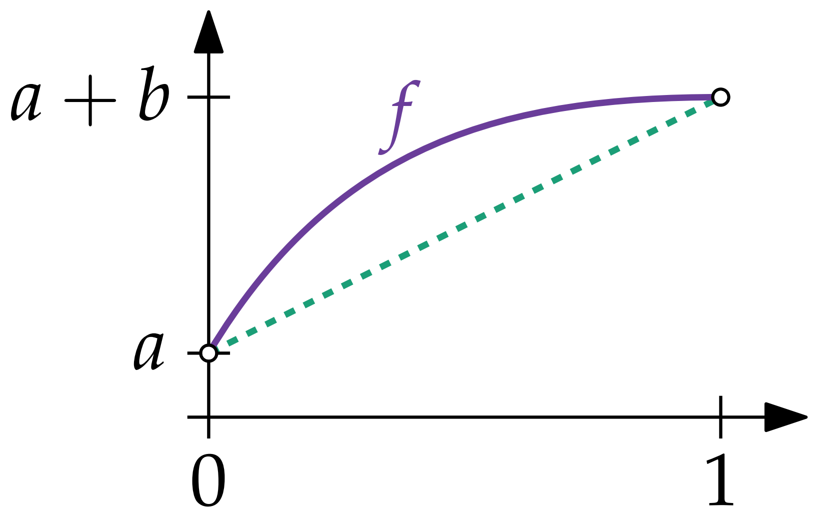
For all non-negative numbers $a_1, \ldots, a_k$:

$$\left(\prod_{i=1}^{k} a_i\right)^{1/k} \leq$$

Mathematical Toolkit

Let f be function that is concave on $[0,1]$
(i.e. $f''(x) \leq 0$ on $[0,1]$) with $f(0) = a$ and $f(1) = a + b$

$$\Rightarrow f(x) \geq bx + a \text{ for } x \in [0,1].$$



Arithmetic-Geometric Mean Inequality (AGMI):

For all non-negative numbers $a_1, \ldots, a_k$:

$$\left(\prod_{i=1}^{k} a_i\right)^{1/k} \leq \frac{1}{k}\left(\sum_{i=1}^{k} a_i\right)$$

Randomized Rounding (Proof)

Consider a fixed clause C_j of length l_j. Then we have:

Randomized Rounding (Proof)

Consider a fixed clause C_j of length l_j. Then we have:

$$\Pr[C_j \text{ not sat.}] =$$

Randomized Rounding (Proof)

Consider a fixed clause C_j of length l_j. Then we have:

$$\Pr[C_j \text{ not sat.}] = \prod_{i \in P_j} (1 - y_i^*)$$

Randomized Rounding (Proof)

Consider a fixed clause C_j of length l_j. Then we have:

$$\Pr[C_j \text{ not sat.}] = \prod_{i \in P_j} (1 - y_i^*) \prod_{i \in N_j} y_i^*$$

$$\leq$$

Randomized Rounding (Proof)

Consider a fixed clause C_j of length l_j. Then we have:

$$\Pr[C_j \text{ not sat.}] = \prod_{i \in P_j} (1 - y_i^*) \prod_{i \in N_j} y_i^*$$

$$\left(\prod_{i=1}^{k} a_i \right)^{1/k} \leq \frac{1}{k} \left(\sum_{i=1}^{k} a_i \right)$$

AGMI

$$\leq$$

Randomized Rounding (Proof)

Consider a fixed clause C_j of length l_j. Then we have:

$$\Pr[C_j \text{ not sat.}] = \prod_{i \in P_j} (1 - y_i^*) \prod_{i \in N_j} y_i^*$$

$$\left(\prod_{i=1}^{k} a_i\right)^{1/k} \leq \frac{1}{k}\left(\sum_{i=1}^{k} a_i\right)$$

AGMI

$$\leq \left(\sum_{i \in P_j} (1 - y_i^*) + \sum_{i \in N_j} y_i^*\right)$$

Randomized Rounding (Proof)

Consider a fixed clause C_j of length l_j. Then we have:

$$\Pr[C_j \text{ not sat.}] = \prod_{i \in P_j} (1 - y_i^*) \prod_{i \in N_j} y_i^*$$

$$\left(\prod_{i=1}^{k} a_i \right)^{1/k} \leq \frac{1}{k} \left(\sum_{i=1}^{k} a_i \right)$$

AGMI

$$\leq \left[\frac{1}{l_j} \left(\sum_{i \in P_j} (1 - y_i^*) + \sum_{i \in N_j} y_i^* \right) \right]^{l_j}$$

Randomized Rounding (Proof)

Consider a fixed clause C_j of length l_j. Then we have:

$$\Pr[C_j \text{ not sat.}] = \prod_{i \in P_j} (1 - y_i^*) \prod_{i \in N_j} y_i^*$$

$$\left(\prod_{i=1}^{k} a_i \right)^{1/k} \leq \frac{1}{k} \left(\sum_{i=1}^{k} a_i \right)$$

AGMI

$$\leq \left[\frac{1}{l_j} \left(\sum_{i \in P_j} (1 - y_i^*) + \sum_{i \in N_j} y_i^* \right) \right]^{l_j}$$

$$= \left[1 - \frac{1}{l_j} \left(\sum_{i \in P_j} y_i^* + \sum_{i \in N_j} (1 - y_i^*) \right) \right]^{l_j}$$

Randomized Rounding (Proof)

Consider a fixed clause C_j of length l_j. Then we have:

$$\Pr[C_j \text{ not sat.}] = \prod_{i\in P_j}(1-y_i^*)\prod_{i\in N_j} y_i^*$$

$$\left(\prod_{i=1}^{k} a_i\right)^{1/k} \leq \frac{1}{k}\left(\sum_{i=1}^{k} a_i\right)$$

AGMI

$$\leq \left[\frac{1}{l_j}\left(\sum_{i\in P_j}(1-y_i^*) + \sum_{i\in N_j} y_i^*\right)\right]^{l_j}$$

$$= \left[1 - \frac{1}{l_j}\underbrace{\left(\sum_{i\in P_j} y_i^* + \sum_{i\in N_j}(1-y_i^*)\right)}_{\geq}\right]^{l_j}$$

Randomized Rounding (Proof)

Consider a fixed clause C_j of length l_j. Then we have:

$$\Pr[C_j \text{ not sat.}] = \prod_{i\in P_j}(1-y_i^*)\prod_{i\in N_j} y_i^*$$

$$\left(\prod_{i=1}^{k} a_i\right)^{1/k} \le \frac{1}{k}\left(\sum_{i=1}^{k} a_i\right)$$

AGMI

$$\le \left[\frac{1}{l_j}\left(\sum_{i\in P_j}(1-y_i^*)+\sum_{i\in N_j} y_i^*\right)\right]^{l_j}$$

$$= \left[1-\frac{1}{l_j}\underbrace{\left(\sum_{i\in P_j} y_i^*+\sum_{i\in N_j}(1-y_i^*)\right)}_{\ge}\right]^{l_j}$$

by LP constraints

Randomized Rounding (Proof)

Consider a fixed clause C_j of length l_j. Then we have:

$$\Pr[C_j \text{ not sat.}] = \prod_{i \in P_j} (1 - y_i^*) \prod_{i \in N_j} y_i^*$$

$$\left(\prod_{i=1}^{k} a_i \right)^{1/k} \leq \frac{1}{k} \left(\sum_{i=1}^{k} a_i \right)$$

AGMI

$$\leq \left[\frac{1}{l_j} \left(\sum_{i \in P_j} (1 - y_i^*) + \sum_{i \in N_j} y_i^* \right) \right]^{l_j}$$

$$= \left[1 - \frac{1}{l_j} \underbrace{\left(\sum_{i \in P_j} y_i^* + \sum_{i \in N_j} (1 - y_i^*) \right)}_{\geq z_j^* \text{ by LP constraints}} \right]^{l_j}$$

Randomized Rounding (Proof)

Consider a fixed clause C_j of length l_j. Then we have:

$$\Pr[C_j \text{ not sat.}] = \prod_{i \in P_j} (1 - y_i^*) \prod_{i \in N_j} y_i^*$$

$$\left(\prod_{i=1}^{k} a_i \right)^{1/k} \leq \frac{1}{k} \left(\sum_{i=1}^{k} a_i \right)$$

AGMI

$$\leq \left[\frac{1}{l_j} \left(\sum_{i \in P_j} (1 - y_i^*) + \sum_{i \in N_j} y_i^* \right) \right]^{l_j}$$

$$= \left[1 - \frac{1}{l_j} \underbrace{\left(\sum_{i \in P_j} y_i^* + \sum_{i \in N_j} (1 - y_i^*) \right)}_{\geq z_j^* \text{ by LP constraints}} \right]^{l_j}$$

$$\leq \left(1 - \frac{z_j^*}{l_j} \right)^{l_j}$$

Randomized Rounding (Proof)

The function $f(z_j^*) = 1 - \left(1 - \frac{z_j^*}{l_j}\right)^{l_j}$ is concave on $[0, 1]$.

Randomized Rounding (Proof)

The function $f(z_j^*) = 1 - \left(1 - \frac{z_j^*}{l_j}\right)^{l_j}$ is concave on $[0, 1]$. Thus

$$\Pr[C_j \text{ satisfied}] \geq$$

Randomized Rounding (Proof)

The function $f(z_j^*) = 1 - \left(1 - \frac{z_j^*}{l_j}\right)^{l_j}$ is concave on $[0, 1]$.

Thus

$$\Pr[C_j \text{ satisfied}] \geq f(z_j^*) \geq$$

Randomized Rounding (Proof)

The function $f(z_j^*) = 1 - \left(1 - \frac{z_j^*}{l_j}\right)^{l_j}$ is concave on $[0,1]$.

Thus

$$\Pr[C_j \text{ satisfied}] \geq f(z_j^*) \geq f(1) \cdot z_j^* + f(0)$$

$$\geq$$

Randomized Rounding (Proof)

The function $f(z_j^*) = 1 - \left(1 - \frac{z_j^*}{l_j}\right)^{l_j}$ is concave on $[0, 1]$.

Thus

$$\begin{aligned} \Pr[C_j \text{ satisfied}] &\geq f(z_j^*) \geq f(1) \cdot z_j^* + f(0) \\ &\geq \left[1 - \left(1 - \frac{1}{l_j}\right)^{l_j}\right] z_j^* \\ &\geq \end{aligned}$$

Randomized Rounding (Proof)

The function $f(z_j^*) = 1 - \left(1 - \frac{z_j^*}{l_j}\right)^{l_j}$ is concave on $[0,1]$.

Thus

$$\Pr[C_j \text{ satisfied}] \geq f(z_j^*) \geq f(1) \cdot z_j^* + f(0)$$

$$\geq \left[1 - \left(1 - \frac{1}{l_j}\right)^{l_j}\right] z_j^*$$

$$\geq$$

$1 + x \leq e^x$

Randomized Rounding (Proof)

The function $f(z_j^*) = 1 - \left(1 - \frac{z_j^*}{l_j}\right)^{l_j}$ is concave on $[0, 1]$.

Thus

$$\Pr[C_j \text{ satisfied}] \geq f(z_j^*) \geq f(1) \cdot z_j^* + f(0)$$
$$\geq \left[1 - \left(1 - \frac{1}{l_j}\right)^{l_j}\right] z_j^*$$
$$\geq$$

$1 + x \leq e^x$

$x = -\frac{1}{l_j}$

Randomized Rounding (Proof)

The function $f(z_j^*) = 1 - \left(1 - \frac{z_j^*}{l_j}\right)^{l_j}$ is concave on $[0, 1]$.

Thus

$$\Pr[C_j \text{ satisfied}] \geq f(z_j^*) \geq f(1) \cdot z_j^* + f(0)$$
$$\geq \left[1 - \left(1 - \frac{1}{l_j}\right)^{l_j}\right] z_j^*$$
$$\geq$$

$1 + x \leq e^x$

$x = -\frac{1}{l_j} \Rightarrow 1 - \frac{1}{l_j} \geq e^{-1/l_j}$

Randomized Rounding (Proof)

The function $f(z_j^*) = 1 - \left(1 - \frac{z_j^*}{l_j}\right)^{l_j}$ is concave on $[0, 1]$.

Thus

$$\Pr[C_j \text{ satisfied}] \geq f(z_j^*) \geq f(1) \cdot z_j^* + f(0)$$
$$\geq \left[1 - \left(1 - \frac{1}{l_j}\right)^{l_j}\right] z_j^*$$
$$\geq \left(1 - \frac{1}{e}\right) z_j^*$$

$1 + x \leq e^x$

$x = -\frac{1}{l_j} \Rightarrow 1 - \frac{1}{l_j} \geq e^{-1/l_j}$

Randomized Rounding (Proof)

Therefore

$$E[W] = \sum_{j=1}^{m} \Pr[C_j \text{ satisfied}] \cdot w_j$$

$$\geq$$

Randomized Rounding (Proof)

Therefore

$$E[W] = \sum_{j=1}^{m} \Pr[C_j \text{ satisfied}] \cdot w_j$$

$$\geq \left(1 - \frac{1}{e}\right) \sum_{j=1}^{m} w_j z_j^*$$

$$=$$

Randomized Rounding (Proof)

Therefore

$$E[W] = \sum_{j=1}^{m} \Pr[C_j \text{ satisfied}] \cdot w_j$$

$$\geq \left(1 - \frac{1}{e}\right) \boxed{\sum_{j=1}^{m} w_j z_j^*} \quad \text{(LP target function)}$$

$$=$$

Randomized Rounding (Proof)

Therefore

$$E[W] = \sum_{j=1}^{m} \Pr[C_j \text{ satisfied}] \cdot w_j$$

$$\geq \left(1 - \frac{1}{e}\right) \boxed{\sum_{j=1}^{m} w_j z_j^*} \quad \text{(LP target function)}$$

$$= \left(1 - \frac{1}{e}\right) \mathrm{OPT}_{\mathrm{LP}}$$

$$\geq$$

Randomized Rounding (Proof)

Therefore

$$
\begin{aligned}
E[W] &= \sum_{j=1}^{m} \Pr[C_j \text{ satisfied}] \cdot w_j \\
&\geq \left(1 - \frac{1}{e}\right) \boxed{\sum_{j=1}^{m} w_j z_j^*} \quad \text{(LP target function)} \\
&= \left(1 - \frac{1}{e}\right) \mathrm{OPT}_{\mathrm{LP}} \\
&\geq \left(1 - \frac{1}{e}\right) \mathrm{OPT}
\end{aligned}
$$

$\square$

Randomized Rounding (Proof)

Therefore

$$E[W] = \sum_{j=1}^{m} \Pr[C_j \text{ satisfied}] \cdot w_j$$

$$\geq \left(1 - \frac{1}{e}\right) \boxed{\sum_{j=1}^{m} w_j z_j^*} \quad \text{(LP target function)}$$

$$= \left(1 - \frac{1}{e}\right) \mathrm{OPT}_{\mathrm{LP}}$$

$$\geq \left(1 - \frac{1}{e}\right) \mathrm{OPT} \qquad \square$$

Theorem. The previous algorithm can be derandomized by the method of conditional expectation.

Approximation Algorithms

Lecture 11:
MaxSat via Randomized Rounding

Part VI:
Combining the Algorithms

Philipp Kindermann

Summer Semester 2020

Take the better of the two solutions!

Theorem. The better solution among the randomized algorithm and the randomized LP-rounding algorithm provides a -approximation for MaxSat.

Take the better of the two solutions!

Theorem. The better solution among the randomized algorithm and the randomized LP-rounding algorithm provides a $3/4$-approximation for MaxSat.

Take the better of the two solutions!

Theorem. The better solution among the randomized algorithm and the randomized LP-rounding algorithm provides a $3/4$-approximation for MaxSat.

Proof.

We use another probabilistic argument. With probability $1/2$ choose the solution of the first algorithm, otherwise the solution of the second algorithm.

Take the better of the two solutions!

Theorem. The better solution among the randomized algorithm and the randomized LP-rounding algorithm provides a $3/4$-approximation for MaxSat.

Proof.

We use another probabilistic argument. With probability $1/2$ choose the solution of the first algorithm, otherwise the solution of the second algorithm.

The better solution is at least as good as the expectation of the above algorithm.

Take the better of the two solutions!

The probability that clause C_j is satisfied is at least:

Take the better of the two solutions!

The probability that clause C_j is satisfied is at least:

$$\frac{1}{2}\left[\qquad \qquad \qquad \qquad \qquad \qquad \right].$$

Take the better of the two solutions!

The probability that clause C_j is satisfied is at least:

$$\frac{1}{2}\left[\underbrace{\left(1-\left(1-\frac{1}{l_j}\right)^{l_j}\right) z_j^*}_{\text{LP-Rounding}} \qquad\qquad\qquad\right].$$

Take the better of the two solutions!

The probability that clause C_j is satisfied is at least:

$$\frac{1}{2}\left[\underbrace{\left(1-\left(1-\frac{1}{l_j}\right)^{l_j}\right)z_j^*}_{\text{LP-Rounding}}+\underbrace{\left(1-2^{-l_j}\right)}_{\text{rand. Alg.}}\right].$$

Take the better of the two solutions!

The probability that clause C_j is satisfied is at least:

$$\underbrace{\frac{1}{2}\left[\left(1-\left(1-\frac{1}{l_j}\right)^{l_j}\right)}_{\text{LP-Rounding}} + \underbrace{\left(1-2^{-l_j}\right)\right]}_{\text{rand. Alg.}} z_j^*.$$

Take the better of the two solutions!

The probability that clause C_j is satisfied is at least:

$$\frac{1}{2}\left[\underbrace{\left(1-\left(1-\frac{1}{l_j}\right)^{l_j}\right)}_{\text{LP-Rounding}}+\underbrace{\left(1-2^{-l_j}\right)}_{\text{rand. Alg.}}\right] z_j^*.$$

We claim that this is at least $3/4 \cdot z_j^*$.

Take the better of the two solutions!

The probability that clause C_j is satisfied is at least:

$$\frac{1}{2}\left[\underbrace{\left(1-\left(1-\frac{1}{l_j}\right)^{l_j}\right)}_{\text{LP-Rounding}}+\underbrace{\left(1-2^{-l_j}\right)}_{\text{rand. Alg.}}\right] z_j^*.$$

We claim that this is at least $3/4 \cdot z_j^*$.

(The rest follows similarly to the previous two Theorems by the linearity of expectation).

Take the better of the two solutions!

The probability that clause C_j is satisfied is at least:

$$\frac{1}{2}\left[\underbrace{\left(1-\left(1-\frac{1}{l_j}\right)^{l_j}\right)}_{\text{LP-Rounding}}+\underbrace{\left(1-2^{-l_j}\right)}_{\text{rand. Alg.}}\right]z_j^*.$$

We claim that this is at least $3/4 \cdot z_j^*$.

(The rest follows similarly to the previous two Theorems by the linearity of expectation).

For $l_j = 1, 2$, a simple calculation gives exactly $3/4 \cdot z_j^*$.

Take the better of the two solutions!

The probability that clause C_j is satisfied is at least:

$$\frac{1}{2}\left[\underbrace{\left(1-\left(1-\frac{1}{l_j}\right)^{l_j}\right)}_{\text{LP-Rounding}}+\underbrace{\left(1-2^{-l_j}\right)}_{\text{rand. Alg.}}\right] z_j^*.$$

We claim that this is at least $3/4 \cdot z_j^*$.

(The rest follows similarly to the previous two Theorems by the linearity of expectation).

For $l_j = 1, 2$, a simple calculation gives exactly $3/4 \cdot z_j^*$.

For $l_j \geq 3$, $1-(1-1/l_j)^{l_j} \geq (1-1/e)$ and $1-2^{-l_j} \geq \frac{7}{8}$.

Take the better of the two solutions!

The probability that clause C_j is satisfied is at least:

$$\frac{1}{2}\Bigg[\underbrace{\left(1-\left(1-\frac{1}{l_j}\right)^{l_j}\right)}_{\text{LP-Rounding}}+\underbrace{\left(1-2^{-l_j}\right)}_{\text{rand. Alg.}}\Bigg] z_j^*.$$

We claim that this is at least $3/4 \cdot z_j^*$.

(The rest follows similarly to the previous two Theorems by the linearity of expectation).

For $l_j = 1, 2$, a simple calculation gives exactly $3/4 \cdot z_j^*$.

For $l_j \geq 3$, $1-(1-1/l_j)^{l_j} \geq (1-1/e)$ and $1-2^{-l_j} \geq \frac{7}{8}$. Thus, we have at least:

$$\frac{1}{2}\left[\left(1-\frac{1}{e}\right)+\frac{7}{8}\right] z_j^* \approx$$

□

Take the better of the two solutions!

The probability that clause C_j is satisfied is at least:

$$\frac{1}{2}\left[\underbrace{\left(1-\left(1-\frac{1}{l_j}\right)^{l_j}\right)}_{\text{LP-Rounding}}+\underbrace{\left(1-2^{-l_j}\right)}_{\text{rand. Alg.}}\right] z_j^*.$$

We claim that this is at least $3/4 \cdot z_j^*$.

(The rest follows similarly to the previous two Theorems by the linearity of expectation).

For $l_j = 1, 2$, a simple calculation gives exactly $3/4 \cdot z_j^*$.

For $l_j \geq 3$, $1-(1-1/l_j)^{l_j} \geq (1-1/e)$ and $1-2^{-l_j} \geq \frac{7}{8}$.
Thus, we have at least:

$$\frac{1}{2}\left[\left(1-\frac{1}{e}\right)+\frac{7}{8}\right] z_j^* \approx 0.753 z_j^* \qquad \square$$

Take the better of the two solutions!

The probability that clause C_j is satisfied is at least:

$$\frac{1}{2}\left[\underbrace{\left(1-\left(1-\frac{1}{l_j}\right)^{l_j}\right)}_{\text{LP-Rounding}}+\underbrace{\left(1-2^{-l_j}\right)}_{\text{rand. Alg.}}\right] z_j^*.$$

We claim that this is at least $3/4 \cdot z_j^*$.

(The rest follows similarly to the previous two Theorems by the linearity of expectation).

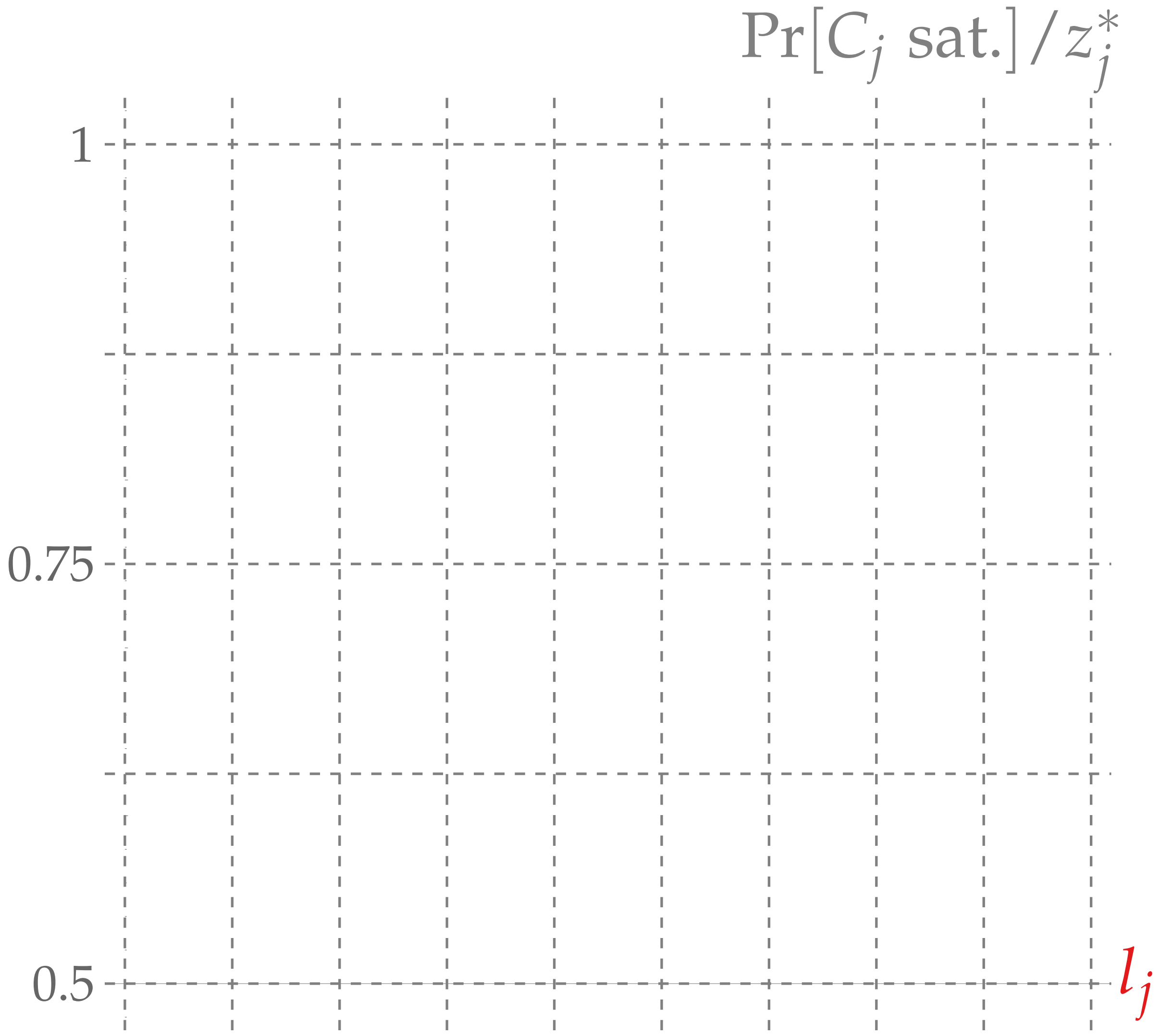
For $l_j = 1, 2$, a simple calculation gives exactly $3/4 \cdot z_j^*$.

For $l_j \geq 3$, $1-(1-1/l_j)^{l_j} \geq (1-1/e)$ and $1-2^{-l_j} \geq \frac{7}{8}$.
Thus, we have at least:

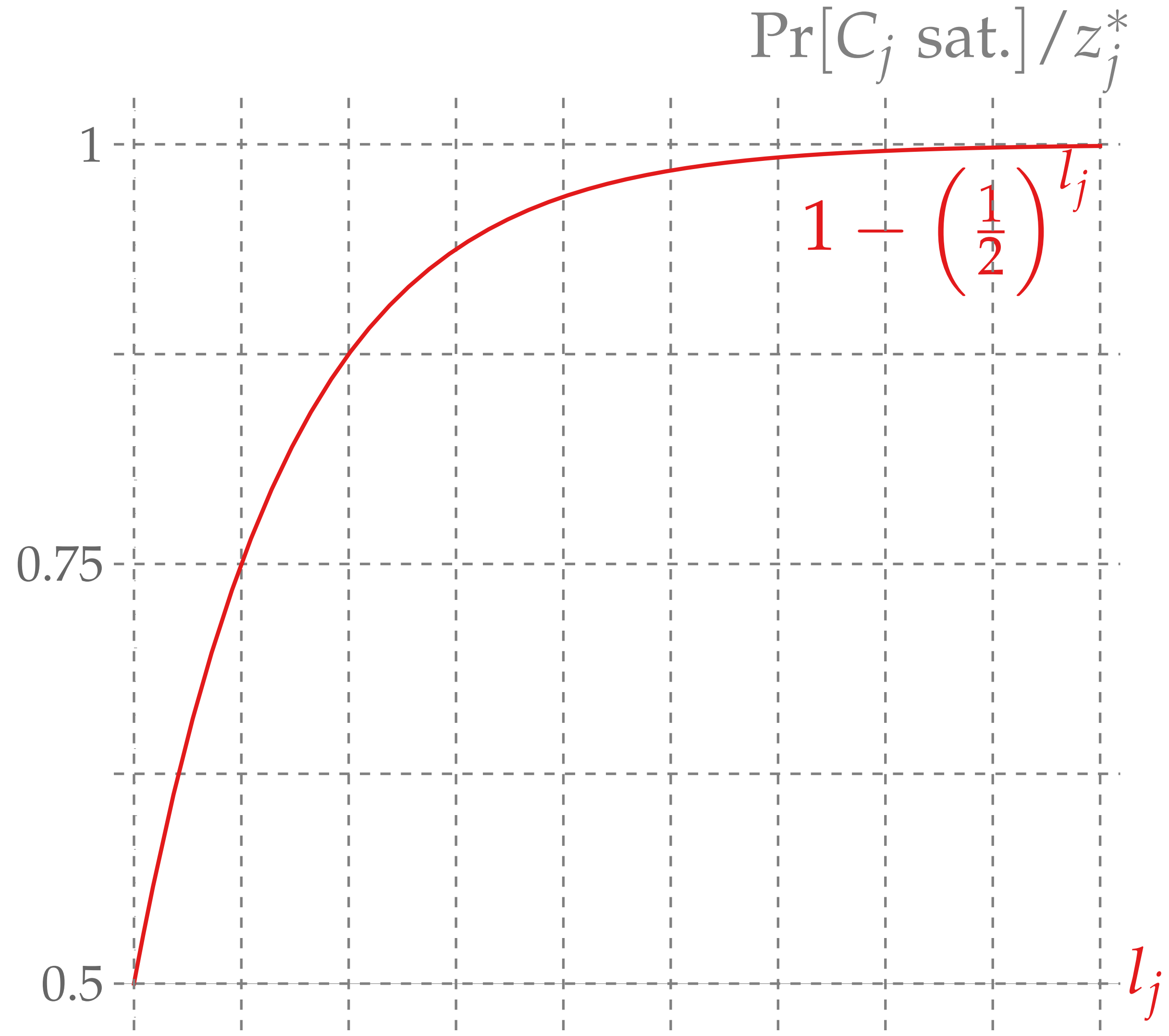
$$\frac{1}{2}\left[\left(1-\frac{1}{e}\right)+\frac{7}{8}\right] z_j^* \approx 0.753 z_j^* \geq \frac{3}{4} z_j^* \qquad \square$$

Visualization and Derandomization

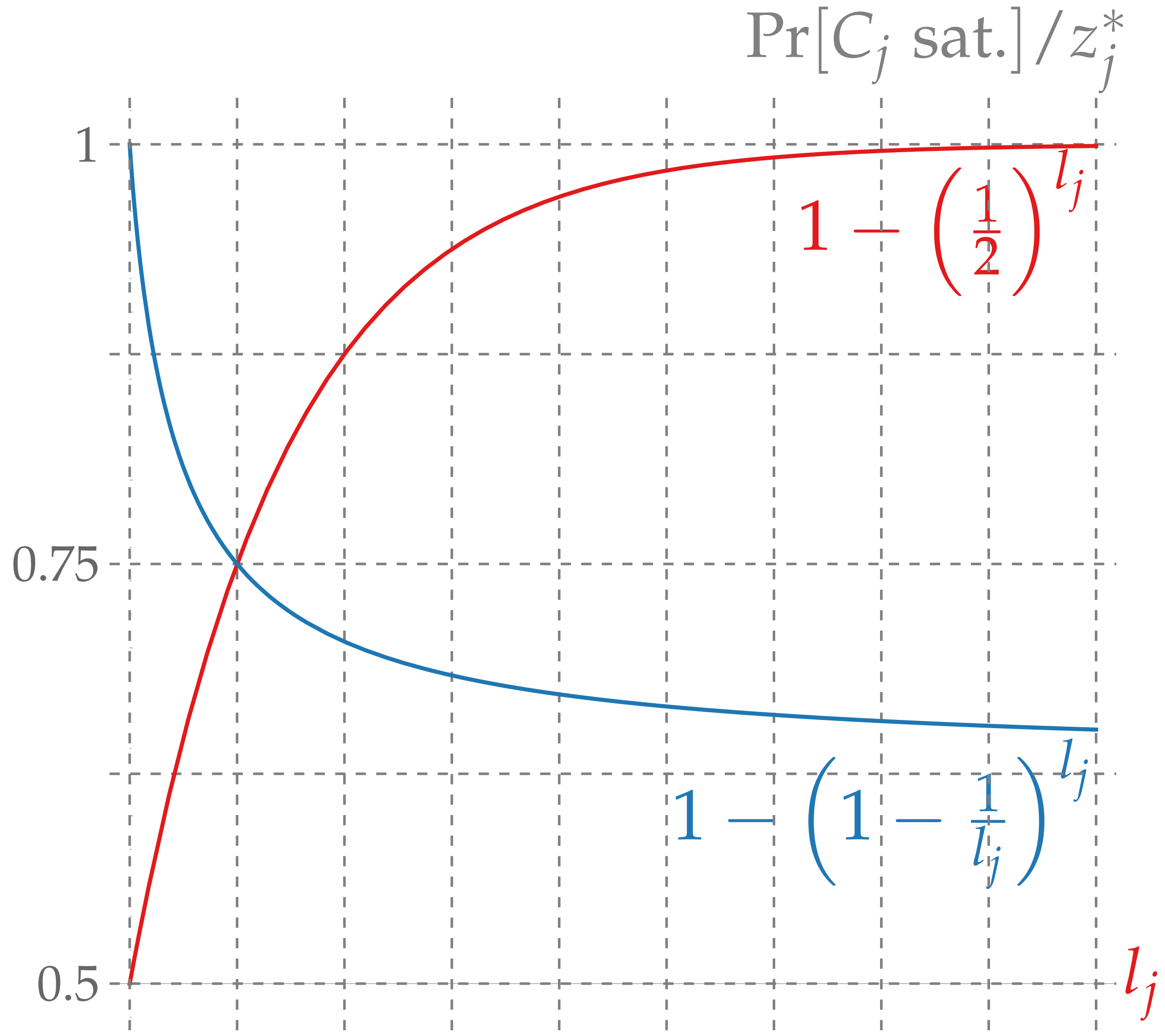
Visualization and Derandomization



Visualization and Derandomization

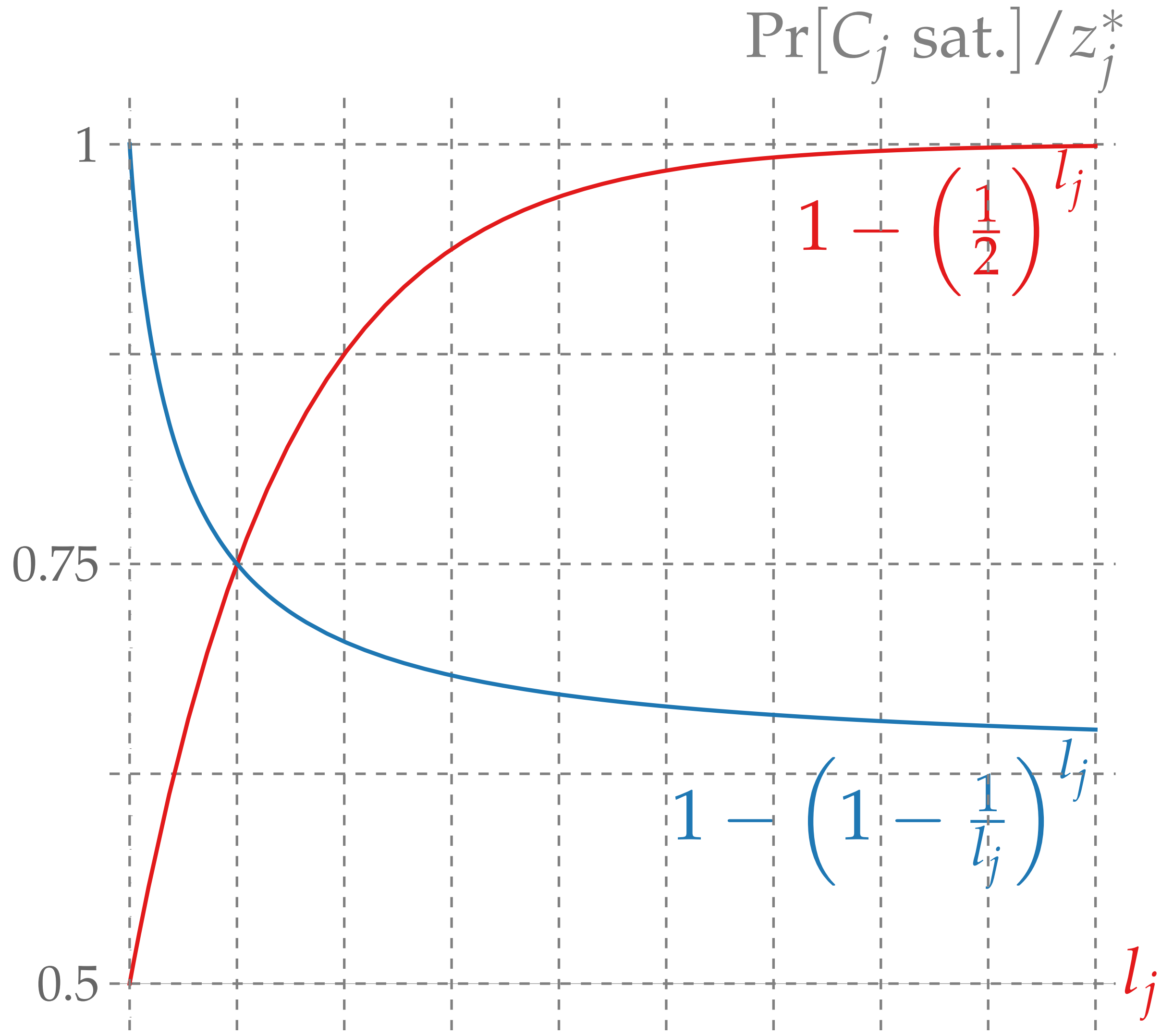


Visualization and Derandomization



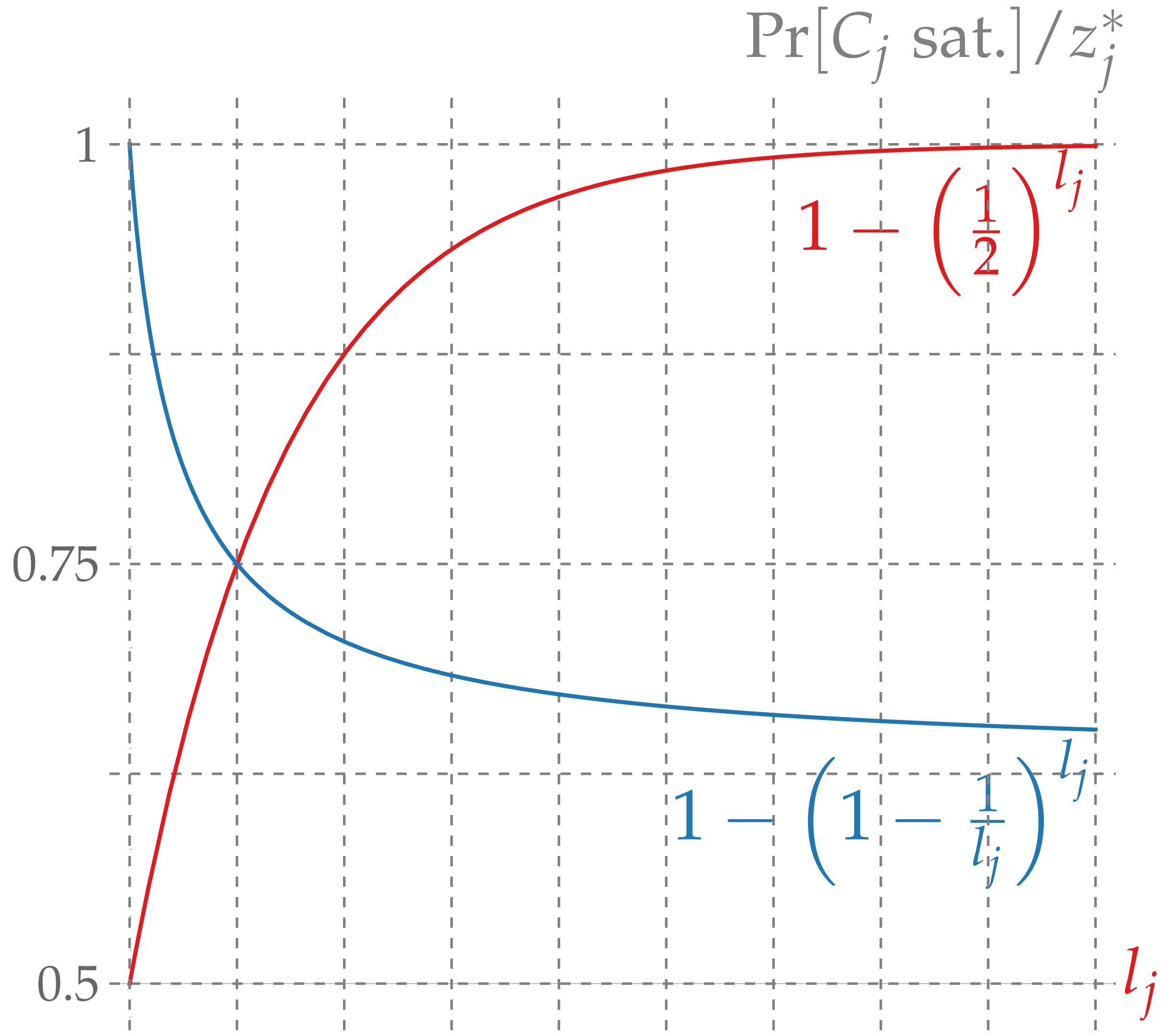
Visualization and Derandomization

– Randomized alg. is better for large values of l_j.



Visualization and Derandomization

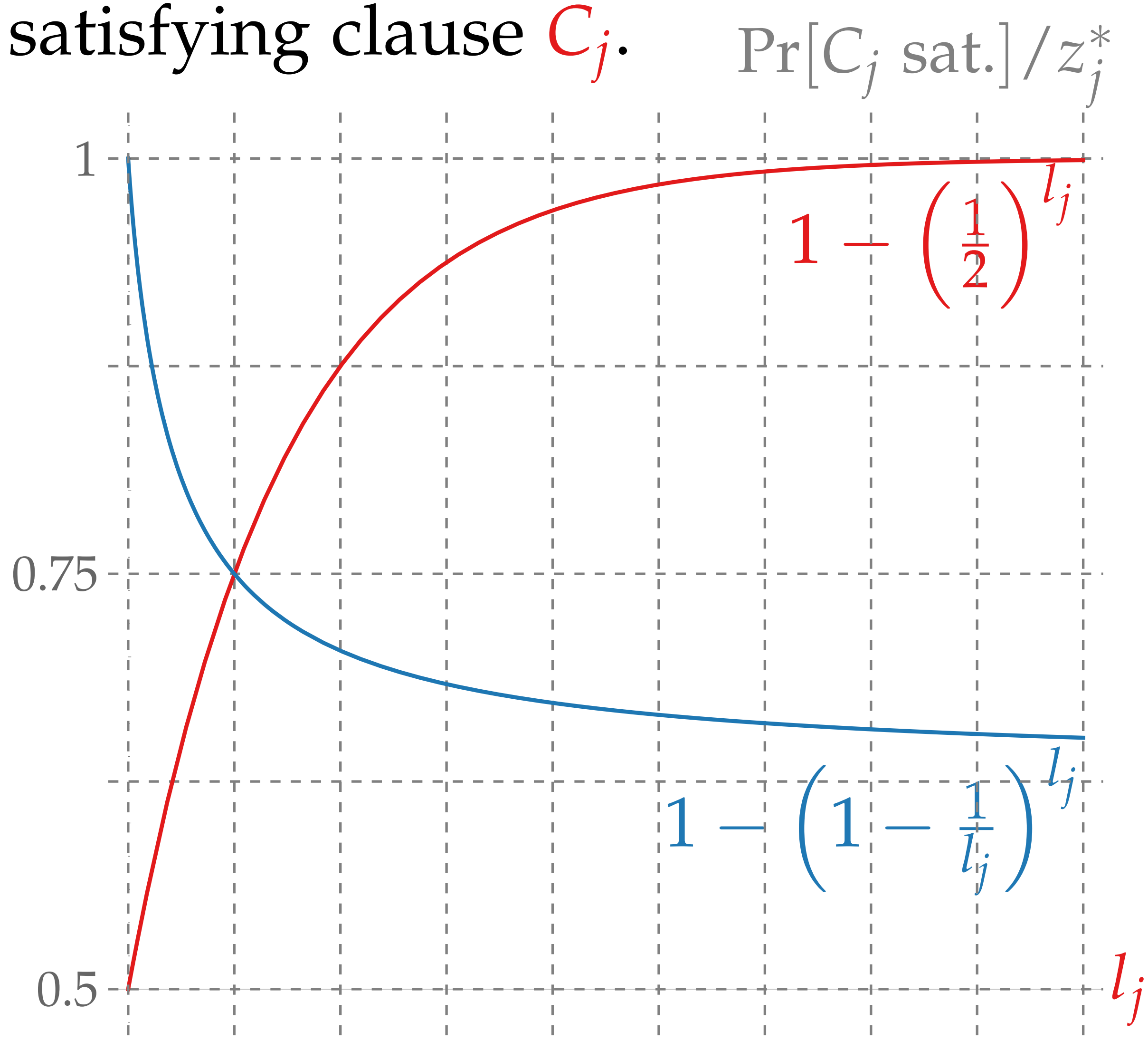
- Randomized alg. is better for large values of l_j.
- Randomized LP-rounding is better for small values of l_j



Visualization and Derandomization

- Randomized alg. is better for large values of l_j.
- Randomized LP-rounding is better for small values of l_j

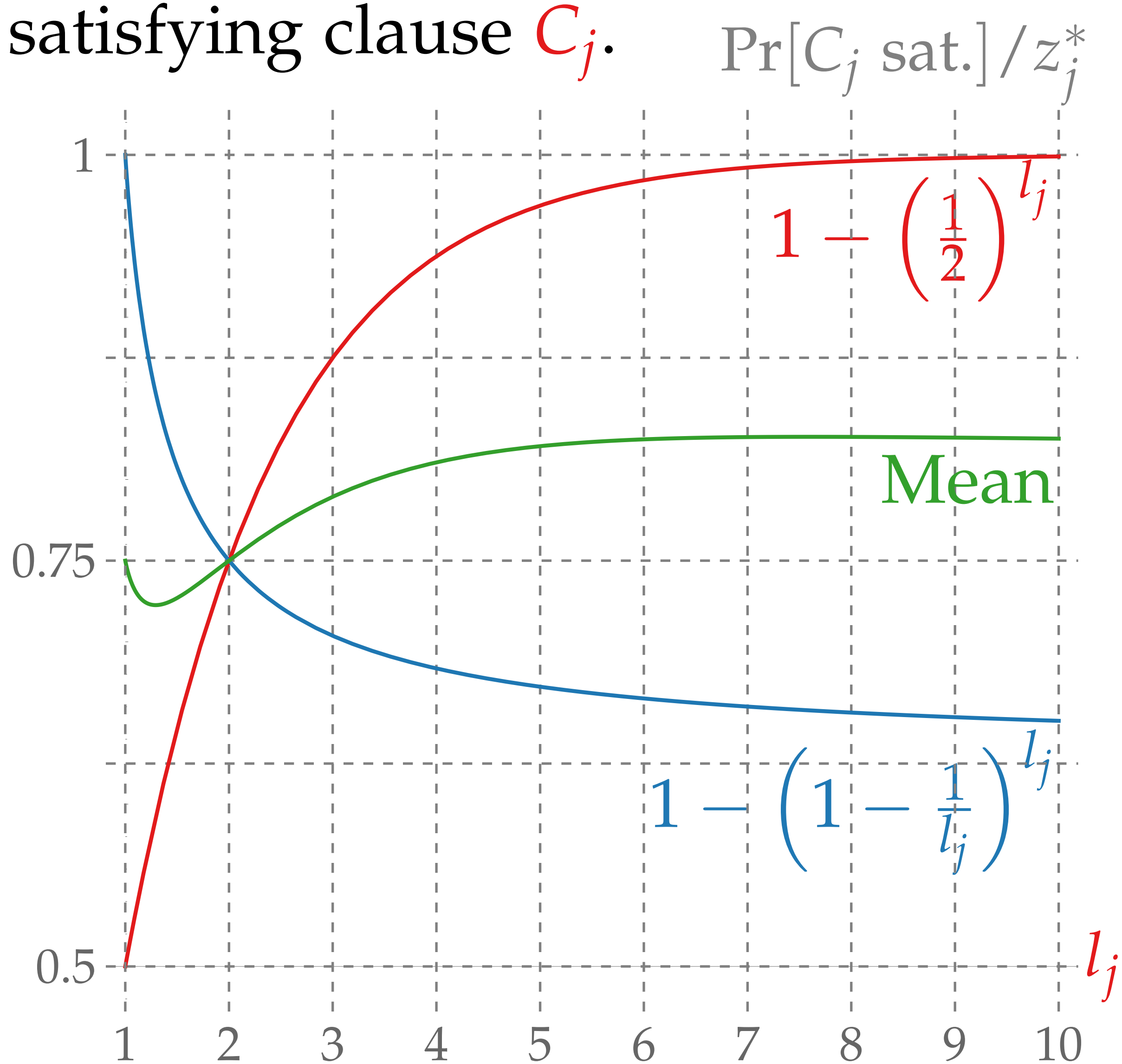
$\Rightarrow$ higher probability of satisfying clause C_j.



Visualization and Derandomization

- Randomized alg. is better for large values of l_j.
- Randomized LP-rounding is better for small values of l_j

$\Rightarrow$ higher probability of satisfying clause C_j.

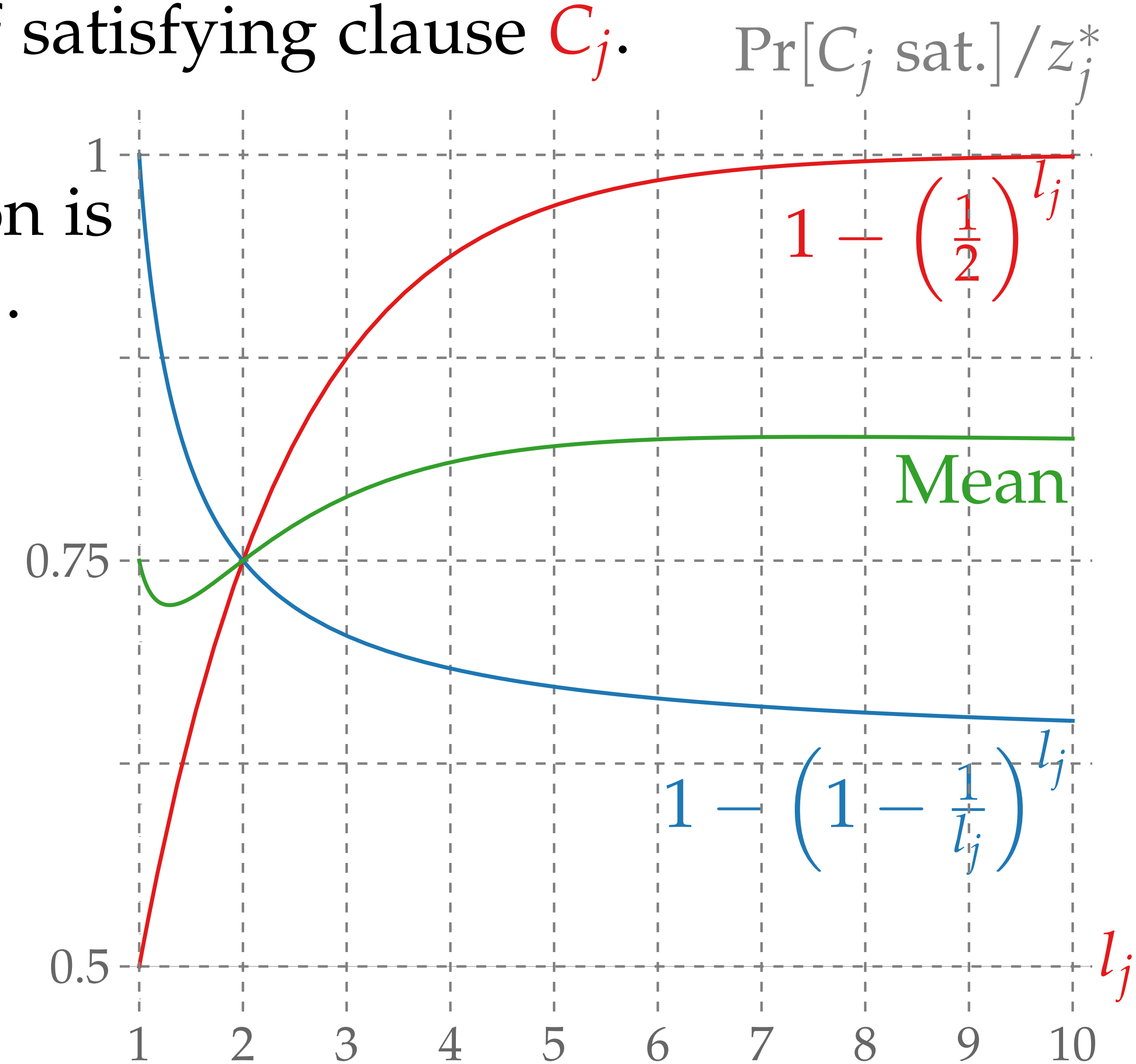


Visualization and Derandomization

- Randomized alg. is better for large values of l_j.
- Randomized LP-rounding is better for small values of l_j

$\Rightarrow$ higher probability of satisfying clause C_j.

Mean of the two solution is at least $3/4$ for *integer* l_j.



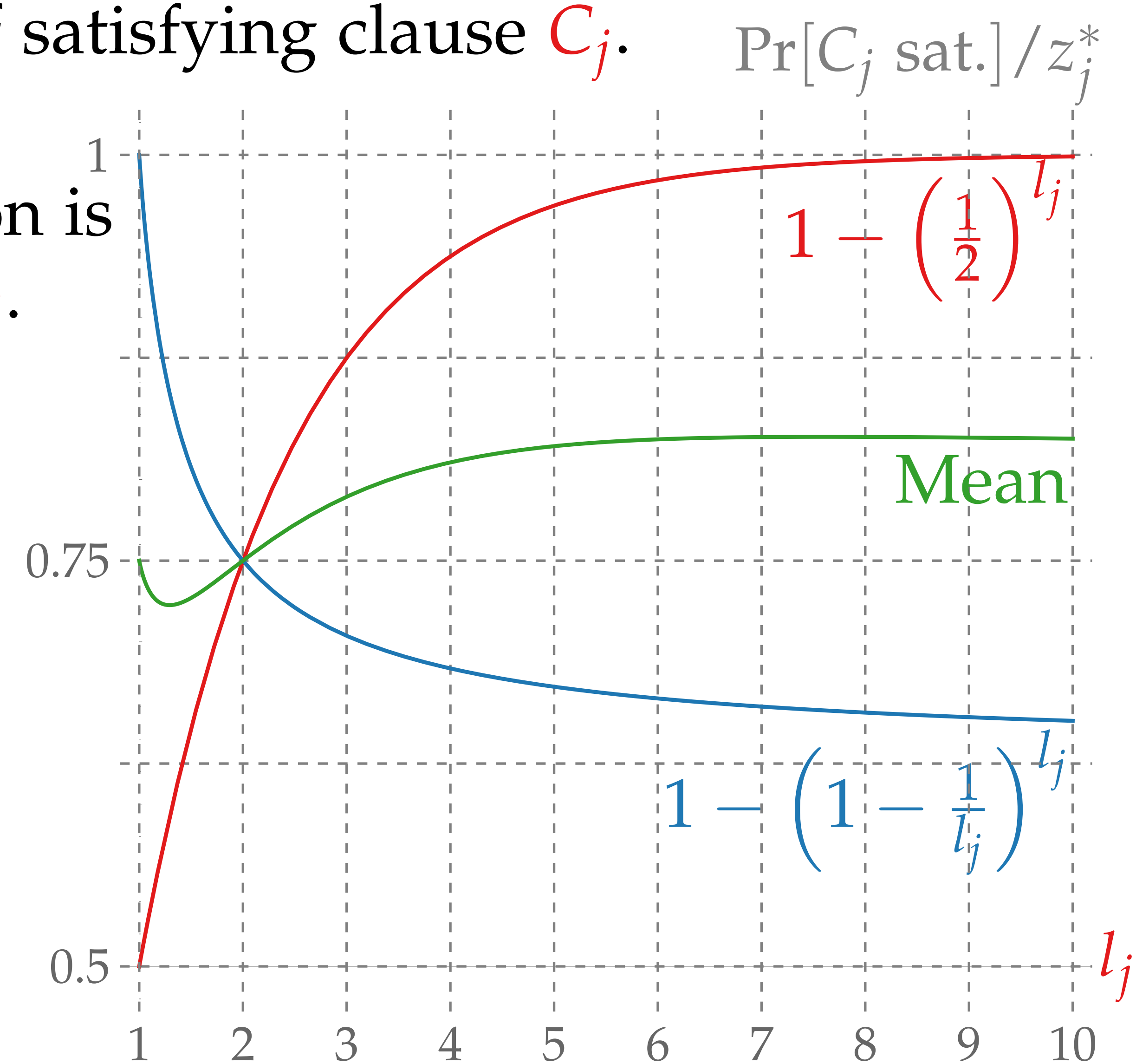
Visualization and Derandomization

- Randomized alg. is better for large values of l_j.
- Randomized LP-rounding is better for small values of l_j

$\Rightarrow$ higher probability of satisfying clause C_j.

Mean of the two solution is at least 3/4 for *integer* l_j.

Maximum is at least as large as the mean.



Visualization and Derandomization

– Randomized alg. is better for large values of l_j.
– Randomized LP-rounding is better for small values of l_j

$\Rightarrow$ higher probability of satisfying clause C_j.

Mean of the two solution is at least 3/4 for *integer* l_j.

Maximum is at least as large as the mean.

This algorithm can also be derandomized by conditional expectation.