

# 16.1 MAXSAT

## Problem definition:

- ▶  $n$  Boolean variables
- ▶  $m$  clauses  $C_1, \dots, C_m$ . For example

$$C_7 = x_3 \vee \bar{x}_5 \vee \bar{x}_9$$

- ▶ Non-negative weight  $w_j$  for each clause  $C_j$ .
- ▶ Find an assignment of true/false to the variables such that the total weight of clauses that are **satisfied** is maximum.

# 16.1 MAXSAT

## Problem definition:

- ▶  $n$  Boolean variables
- ▶  $m$  clauses  $C_1, \dots, C_m$ . For example

$$C_7 = x_3 \vee \bar{x}_5 \vee \bar{x}_9$$

- ▶ Non-negative weight  $w_j$  for each clause  $C_j$ .
- ▶ Find an assignment of true/false to the variables such that the total weight of clauses that are **satisfied** is maximum.

# 16.1 MAXSAT

## Problem definition:

- ▶  $n$  Boolean variables
- ▶  $m$  clauses  $C_1, \dots, C_m$ . For example

$$C_7 = x_3 \vee \bar{x}_5 \vee \bar{x}_9$$

- ▶ Non-negative weight  $w_j$  for each clause  $C_j$ .
- ▶ Find an assignment of true/false to the variables such that the total weight of clauses that are **satisfied** is maximum.

# 16.1 MAXSAT

## Problem definition:

- ▶  $n$  Boolean variables
- ▶  $m$  clauses  $C_1, \dots, C_m$ . For example

$$C_7 = x_3 \vee \bar{x}_5 \vee \bar{x}_9$$

- ▶ Non-negative weight  $w_j$  for each clause  $C_j$ .
- ▶ Find an assignment of true/false to the variables such that the total weight of clauses that are **satisfied** is maximum.

# 16.1 MAXSAT

## Terminology:

- ▶ A variable  $x_i$  and its negation  $\bar{x}_i$  are called **literals**.
- ▶ Hence, each clause consists of a set of literals (i.e., no duplications:  $x_i \vee x_i \vee \bar{x}_j$  is not a clause).
- ▶ We assume a clause does not contain  $x_i$  and  $\bar{x}_i$  for any  $i$ .
- ▶  $x_i$  is called a **positive literal** while the negation  $\bar{x}_i$  is called a **negative literal**.
- ▶ For a given clause  $C_j$  the number of its literals is called its **length** or **size** and denoted with  $\ell_j$ .
- ▶ Clauses of length one are called **unit clauses**.

# 16.1 MAXSAT

## Terminology:

- ▶ A variable  $x_i$  and its negation  $\bar{x}_i$  are called **literals**.
- ▶ Hence, each clause consists of a set of literals (i.e., no duplications:  $x_i \vee x_i \vee \bar{x}_j$  is **not** a clause).
- ▶ We assume a clause does not contain  $x_i$  and  $\bar{x}_i$  for any  $i$ .
- ▶  $x_i$  is called a **positive literal** while the negation  $\bar{x}_i$  is called a **negative literal**.
- ▶ For a given clause  $C_j$  the number of its literals is called its **length** or **size** and denoted with  $\ell_j$ .
- ▶ Clauses of length one are called **unit clauses**.

# 16.1 MAXSAT

## Terminology:

- ▶ A variable  $x_i$  and its negation  $\bar{x}_i$  are called **literals**.
- ▶ Hence, each clause consists of a set of literals (i.e., no duplications:  $x_i \vee x_i \vee \bar{x}_j$  is **not** a clause).
- ▶ We assume a clause does not contain  $x_i$  and  $\bar{x}_i$  for any  $i$ .
- ▶  $x_i$  is called a **positive literal** while the negation  $\bar{x}_i$  is called a **negative literal**.
- ▶ For a given clause  $C_j$  the number of its literals is called its **length** or **size** and denoted with  $\ell_j$ .
- ▶ Clauses of length one are called **unit clauses**.

# 16.1 MAXSAT

## Terminology:

- ▶ A variable  $x_i$  and its negation  $\bar{x}_i$  are called **literals**.
- ▶ Hence, each clause consists of a set of literals (i.e., no duplications:  $x_i \vee x_i \vee \bar{x}_j$  is **not** a clause).
- ▶ We assume a clause does not contain  $x_i$  and  $\bar{x}_i$  for any  $i$ .
- ▶  $x_i$  is called a **positive literal** while the negation  $\bar{x}_i$  is called a **negative literal**.
- ▶ For a given clause  $C_j$  the number of its literals is called its **length** or **size** and denoted with  $\ell_j$ .
- ▶ Clauses of length one are called **unit clauses**.



# 16.1 MAXSAT

## Terminology:

- ▶ A variable  $x_i$  and its negation  $\bar{x}_i$  are called **literals**.
- ▶ Hence, each clause consists of a set of literals (i.e., no duplications:  $x_i \vee x_i \vee \bar{x}_j$  is **not** a clause).
- ▶ We assume a clause does not contain  $x_i$  and  $\bar{x}_i$  for any  $i$ .
- ▶  $x_i$  is called a **positive literal** while the negation  $\bar{x}_i$  is called a **negative literal**.
- ▶ For a given clause  $C_j$  the number of its literals is called its **length** or **size** and denoted with  $\ell_j$ .
- ▶ Clauses of length one are called **unit clauses**.

# 16.1 MAXSAT

## Terminology:

- ▶ A variable  $x_i$  and its negation  $\bar{x}_i$  are called **literals**.
- ▶ Hence, each clause consists of a set of literals (i.e., no duplications:  $x_i \vee x_i \vee \bar{x}_j$  is **not** a clause).
- ▶ We assume a clause does not contain  $x_i$  and  $\bar{x}_i$  for any  $i$ .
- ▶  $x_i$  is called a **positive literal** while the negation  $\bar{x}_i$  is called a **negative literal**.
- ▶ For a given clause  $C_j$  the number of its literals is called its **length** or **size** and denoted with  $\ell_j$ .
- ▶ Clauses of length one are called **unit clauses**.

# MAXSAT: Flipping Coins

Set each  $x_i$  independently to **true** with probability  $\frac{1}{2}$  (and, hence, to **false** with probability  $\frac{1}{2}$ , as well).

Define random variable  $X_j$  with

$$X_j = \begin{cases} 1 & \text{if } C_j \text{ satisfied} \\ 0 & \text{otw.} \end{cases}$$

Then the total weight  $W$  of satisfied clauses is given by

$$W = \sum_j w_j X_j$$

Define random variable  $X_j$  with

$$X_j = \begin{cases} 1 & \text{if } C_j \text{ satisfied} \\ 0 & \text{otw.} \end{cases}$$

Then the total weight  $W$  of satisfied clauses is given by

$$W = \sum_j w_j X_j$$

$E[W]$

$$E[W] = \sum_j w_j E[X_j]$$

$$\begin{aligned} E[W] &= \sum_j w_j E[X_j] \\ &= \sum_j w_j \Pr[C_j \text{ is satisfied}] \end{aligned}$$



$$\begin{aligned} E[W] &= \sum_j w_j E[X_j] \\ &= \sum_j w_j \Pr[C_j \text{ is satisfied}] \\ &= \sum_j w_j \left(1 - \left(\frac{1}{2}\right)^{\ell_j}\right) \end{aligned}$$

$$\begin{aligned} E[W] &= \sum_j w_j E[X_j] \\ &= \sum_j w_j \Pr[C_j \text{ is satisfied}] \\ &= \sum_j w_j \left(1 - \left(\frac{1}{2}\right)^{\ell_j}\right) \\ &\geq \frac{1}{2} \sum_j w_j \end{aligned}$$

$$\begin{aligned} E[W] &= \sum_j w_j E[X_j] \\ &= \sum_j w_j \Pr[C_j \text{ is satisfied}] \\ &= \sum_j w_j \left(1 - \left(\frac{1}{2}\right)^{\ell_j}\right) \\ &\geq \frac{1}{2} \sum_j w_j \\ &\geq \frac{1}{2} \text{OPT} \end{aligned}$$

# MAXSAT: LP formulation

- ▶ Let for a clause  $C_j$ ,  $P_j$  be the set of positive literals and  $N_j$  the set of negative literals.

$$C_j = \bigvee_{i \in P_j} x_i \vee \bigvee_{i \in N_j} \bar{x}_i$$

$$\begin{array}{ll} \max & \sum_j w_j z_j \\ \text{s.t.} & \forall j \quad \sum_{i \in P_j} y_i + \sum_{i \in N_j} (1 - y_i) \geq z_j \\ & \forall i \quad y_i \in \{0, 1\} \\ & \forall j \quad z_j \leq 1 \end{array}$$

# MAXSAT: LP formulation

- ▶ Let for a clause  $C_j$ ,  $P_j$  be the set of positive literals and  $N_j$  the set of negative literals.

$$C_j = \bigvee_{i \in P_j} x_i \vee \bigvee_{i \in N_j} \bar{x}_i$$

$$\begin{array}{ll} \max & \sum_j w_j z_j \\ \text{s.t.} & \forall j \quad \sum_{i \in P_j} y_i + \sum_{i \in N_j} (1 - y_i) \geq z_j \\ & \forall i \quad y_i \in \{0, 1\} \\ & \forall j \quad z_j \leq 1 \end{array}$$

# MAXSAT: Randomized Rounding

Set each  $x_i$  independently to **true** with probability  $y_i$  (and, hence, to **false** with probability  $(1 - y_i)$ ).

## Lemma 84 (Geometric Mean $\leq$ Arithmetic Mean)

For any nonnegative  $a_1, \dots, a_k$

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

## Definition 85

A function  $f$  on an interval  $I$  is **concave** if for any two points  $s$  and  $r$  from  $I$  and any  $\lambda \in [0, 1]$  we have

$$f(\lambda s + (1 - \lambda)r) \geq \lambda f(s) + (1 - \lambda)f(r)$$

## Lemma 86

Let  $f$  be a concave function on the interval  $[0, 1]$ , with  $f(0) = a$  and  $f(1) = a + b$ . Then

$$f(\lambda)$$

for  $\lambda \in [0, 1]$ .



## Definition 85

A function  $f$  on an interval  $I$  is **concave** if for any two points  $s$  and  $r$  from  $I$  and any  $\lambda \in [0, 1]$  we have

$$f(\lambda s + (1 - \lambda)r) \geq \lambda f(s) + (1 - \lambda)f(r)$$

## Lemma 86

Let  $f$  be a concave function on the interval  $[0, 1]$ , with  $f(0) = a$  and  $f(1) = a + b$ . Then

$$\begin{aligned} f(\lambda) &= f((1 - \lambda)0 + \lambda 1) \\ &\geq (1 - \lambda)f(0) + \lambda f(1) \\ &= a + \lambda b \end{aligned}$$

for  $\lambda \in [0, 1]$ .

## Definition 85

A function  $f$  on an interval  $I$  is **concave** if for any two points  $s$  and  $r$  from  $I$  and any  $\lambda \in [0, 1]$  we have

$$f(\lambda s + (1 - \lambda)r) \geq \lambda f(s) + (1 - \lambda)f(r)$$

## Lemma 86

Let  $f$  be a concave function on the interval  $[0, 1]$ , with  $f(0) = a$  and  $f(1) = a + b$ . Then

$$\begin{aligned} f(\lambda) &= f((1 - \lambda)0 + \lambda 1) \\ &\geq (1 - \lambda)f(0) + \lambda f(1) \\ &= a + \lambda b \end{aligned}$$

for  $\lambda \in [0, 1]$ .

## Definition 85

A function  $f$  on an interval  $I$  is **concave** if for any two points  $s$  and  $r$  from  $I$  and any  $\lambda \in [0, 1]$  we have

$$f(\lambda s + (1 - \lambda)r) \geq \lambda f(s) + (1 - \lambda)f(r)$$

## Lemma 86

Let  $f$  be a concave function on the interval  $[0, 1]$ , with  $f(0) = a$  and  $f(1) = a + b$ . Then

$$\begin{aligned} f(\lambda) &= f((1 - \lambda)0 + \lambda 1) \\ &\geq (1 - \lambda)f(0) + \lambda f(1) \\ &= a + \lambda b \end{aligned}$$

for  $\lambda \in [0, 1]$ .

$\Pr[C_j \text{ not satisfied}]$

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

$$\begin{aligned}\Pr[C_j \text{ not satisfied}] &= \prod_{i \in P_j} (1 - y_i) \prod_{i \in N_j} y_i \\ &\leq \left[ \frac{1}{\ell_j} \left( \sum_{i \in P_j} (1 - y_i) + \sum_{i \in N_j} y_i \right) \right]^{\ell_j}\end{aligned}$$

$$\begin{aligned}\Pr[C_j \text{ not satisfied}] &= \prod_{i \in P_j} (1 - y_i) \prod_{i \in N_j} y_i \\ &\leq \left[ \frac{1}{\ell_j} \left( \sum_{i \in P_j} (1 - y_i) + \sum_{i \in N_j} y_i \right) \right]^{\ell_j} \\ &= \left[ 1 - \frac{1}{\ell_j} \left( \sum_{i \in P_j} y_i + \sum_{i \in N_j} (1 - y_i) \right) \right]^{\ell_j}\end{aligned}$$

$$\begin{aligned}
\Pr[C_j \text{ not satisfied}] &= \prod_{i \in P_j} (1 - y_i) \prod_{i \in N_j} y_i \\
&\leq \left[ \frac{1}{\ell_j} \left( \sum_{i \in P_j} (1 - y_i) + \sum_{i \in N_j} y_i \right) \right]^{\ell_j} \\
&= \left[ 1 - \frac{1}{\ell_j} \left( \sum_{i \in P_j} y_i + \sum_{i \in N_j} (1 - y_i) \right) \right]^{\ell_j} \\
&\leq \left( 1 - \frac{z_j}{\ell_j} \right)^{\ell_j} .
\end{aligned}$$



The function  $f(z) = 1 - (1 - \frac{z}{\ell})^\ell$  is concave. Hence,

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

The function  $f(z) = 1 - (1 - \frac{z}{\ell})^\ell$  is concave. Hence,

$$\Pr[C_j \text{ satisfied}] \geq 1 - \left(1 - \frac{z_j}{\ell_j}\right)^{\ell_j}$$

The function  $f(z) = 1 - (1 - \frac{z}{\ell})^\ell$  is concave. Hence,

$$\begin{aligned}\Pr[C_j \text{ satisfied}] &\geq 1 - \left(1 - \frac{z_j}{\ell_j}\right)^{\ell_j} \\ &\geq \left[1 - \left(1 - \frac{1}{\ell_j}\right)^{\ell_j}\right] \cdot z_j .\end{aligned}$$

The function  $f(z) = 1 - (1 - \frac{z}{\ell})^\ell$  is concave. Hence,

$$\begin{aligned}\Pr[C_j \text{ satisfied}] &\geq 1 - \left(1 - \frac{z_j}{\ell_j}\right)^{\ell_j} \\ &\geq \left[1 - \left(1 - \frac{1}{\ell_j}\right)^{\ell_j}\right] \cdot z_j .\end{aligned}$$

$f''(z) = -\frac{\ell-1}{\ell} \left[1 - \frac{z}{\ell}\right]^{\ell-2} \leq 0$  for  $z \in [0, 1]$ . Therefore,  $f$  is concave.

$E[W]$

$$E[W] = \sum_j w_j \Pr[C_j \text{ is satisfied}]$$

$$\begin{aligned} E[W] &= \sum_j w_j \Pr[C_j \text{ is satisfied}] \\ &\geq \sum_j w_j z_j \left[ 1 - \left( 1 - \frac{1}{\ell_j} \right)^{\ell_j} \right] \end{aligned}$$

$$\begin{aligned} E[W] &= \sum_j w_j \Pr[C_j \text{ is satisfied}] \\ &\geq \sum_j w_j z_j \left[ 1 - \left( 1 - \frac{1}{\ell_j} \right)^{\ell_j} \right] \\ &\geq \left( 1 - \frac{1}{e} \right) \text{OPT} . \end{aligned}$$



# MAXSAT: The better of two

## Theorem 87

*Choosing the better of the two solutions given by randomized rounding and coin flipping yields a  $\frac{3}{4}$ -approximation.*

Let  $W_1$  be the value of randomized rounding and  $W_2$  the value obtained by coin flipping.

$$E[\max\{W_1, W_2\}]$$

Let  $W_1$  be the value of randomized rounding and  $W_2$  the value obtained by coin flipping.

$$\begin{aligned} E[\max\{W_1, W_2\}] \\ \geq E[\frac{1}{2}W_1 + \frac{1}{2}W_2] \end{aligned}$$

Let  $W_1$  be the value of randomized rounding and  $W_2$  the value obtained by coin flipping.

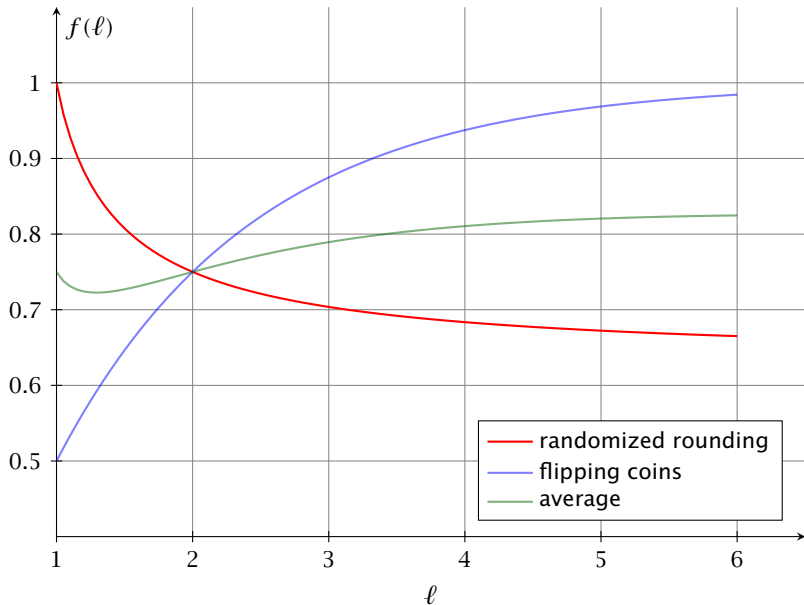
$$\begin{aligned} E[\max\{W_1, W_2\}] &\geq E[\frac{1}{2}W_1 + \frac{1}{2}W_2] \\ &\geq \frac{1}{2} \sum_j w_j z_j \left[ 1 - \left(1 - \frac{1}{\ell_j}\right)^{\ell_j} \right] + \frac{1}{2} \sum_j w_j \left(1 - \left(\frac{1}{2}\right)^{\ell_j}\right) \end{aligned}$$

Let  $W_1$  be the value of randomized rounding and  $W_2$  the value obtained by coin flipping.

$$\begin{aligned} E[\max\{W_1, W_2\}] &\geq E\left[\frac{1}{2}W_1 + \frac{1}{2}W_2\right] \\ &\geq \frac{1}{2} \sum_j w_j z_j \left[ 1 - \left(1 - \frac{1}{\ell_j}\right)^{\ell_j} \right] + \frac{1}{2} \sum_j w_j \left(1 - \left(\frac{1}{2}\right)^{\ell_j}\right) \\ &\geq \sum_j w_j z_j \underbrace{\left[ \frac{1}{2} \left(1 - \left(1 - \frac{1}{\ell_j}\right)^{\ell_j}\right) + \frac{1}{2} \left(1 - \left(\frac{1}{2}\right)^{\ell_j}\right) \right]}_{\geq \frac{3}{4} \text{ for all integers}} \end{aligned}$$

Let  $W_1$  be the value of randomized rounding and  $W_2$  the value obtained by coin flipping.

$$\begin{aligned} & E[\max\{W_1, W_2\}] \\ & \geq E\left[\frac{1}{2}W_1 + \frac{1}{2}W_2\right] \\ & \geq \frac{1}{2} \sum_j w_j z_j \left[1 - \left(1 - \frac{1}{\ell_j}\right)^{\ell_j}\right] + \frac{1}{2} \sum_j w_j \left(1 - \left(\frac{1}{2}\right)^{\ell_j}\right) \\ & \geq \sum_j w_j z_j \underbrace{\left[\frac{1}{2} \left(1 - \left(1 - \frac{1}{\ell_j}\right)^{\ell_j}\right) + \frac{1}{2} \left(1 - \left(\frac{1}{2}\right)^{\ell_j}\right)\right]}_{\geq \frac{3}{4} \text{ for all integers}} \\ & \geq \frac{3}{4} \text{OPT} \end{aligned}$$



# MAXSAT: Nonlinear Randomized Rounding

So far we used **linear** randomized rounding, i.e., the probability that a variable is set to 1/true was exactly the value of the corresponding variable in the linear program.

We could define a function  $f : [0, 1] \rightarrow [0, 1]$  and set  $x_i$  to true with probability  $f(y_i)$ .



# MAXSAT: Nonlinear Randomized Rounding

So far we used **linear** randomized rounding, i.e., the probability that a variable is set to 1/true was exactly the value of the corresponding variable in the linear program.

We could define a function  $f : [0, 1] \rightarrow [0, 1]$  and set  $x_i$  to true with probability  $f(y_i)$ .

# MAXSAT: Nonlinear Randomized Rounding

Let  $f : [0, 1] \rightarrow [0, 1]$  be a function with

$$1 - 4^{-x} \leq f(x) \leq 4^{x-1}$$

## Theorem 88

*Rounding the LP-solution with a function  $f$  of the above form gives a  $\frac{3}{4}$ -approximation.*

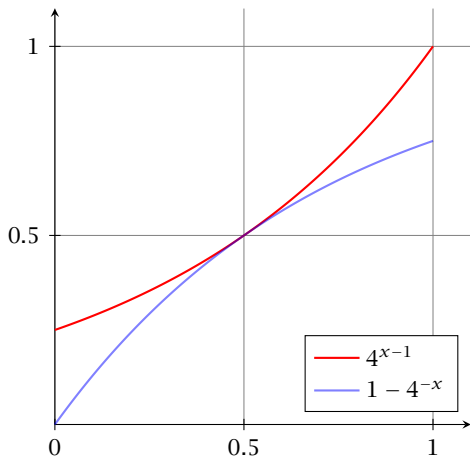
# MAXSAT: Nonlinear Randomized Rounding

Let  $f : [0, 1] \rightarrow [0, 1]$  be a function with

$$1 - 4^{-x} \leq f(x) \leq 4^{x-1}$$

## Theorem 88

*Rounding the LP-solution with a function  $f$  of the above form gives a  $\frac{3}{4}$ -approximation.*



$\Pr[C_j \text{ not satisfied}]$

$$\Pr[C_j \text{ not satisfied}] = \prod_{i \in P_j} (1 - f(y_i)) \prod_{i \in N_j} f(y_i)$$

$$\begin{aligned}\Pr[C_j \text{ not satisfied}] &= \prod_{i \in P_j} (1 - f(y_i)) \prod_{i \in N_j} f(y_i) \\ &\leq \prod_{i \in P_j} 4^{-y_i} \prod_{i \in N_j} 4^{y_i - 1}\end{aligned}$$

$$\begin{aligned}\Pr[C_j \text{ not satisfied}] &= \prod_{i \in P_j} (1 - f(y_i)) \prod_{i \in N_j} f(y_i) \\ &\leq \prod_{i \in P_j} 4^{-y_i} \prod_{i \in N_j} 4^{y_i - 1} \\ &= 4^{-(\sum_{i \in P_j} y_i + \sum_{i \in N_j} (1 - y_i))}\end{aligned}$$



$$\begin{aligned}\Pr[C_j \text{ not satisfied}] &= \prod_{i \in P_j} (1 - f(y_i)) \prod_{i \in N_j} f(y_i) \\ &\leq \prod_{i \in P_j} 4^{-y_i} \prod_{i \in N_j} 4^{y_i - 1} \\ &= 4^{-(\sum_{i \in P_j} y_i + \sum_{i \in N_j} (1 - y_i))} \\ &\leq 4^{-z_j}\end{aligned}$$

The function  $g(z) = 1 - 4^{-z}$  is concave on  $[0, 1]$ . Hence,

The function  $g(z) = 1 - 4^{-z}$  is concave on  $[0, 1]$ . Hence,

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

The function  $g(z) = 1 - 4^{-z}$  is concave on  $[0, 1]$ . Hence,

$$\Pr[C_j \text{ satisfied}] \geq 1 - 4^{-z_j}$$

The function  $g(z) = 1 - 4^{-z}$  is concave on  $[0, 1]$ . Hence,

$$\Pr[C_j \text{ satisfied}] \geq 1 - 4^{-z_j} \geq \frac{3}{4}z_j .$$

The function  $g(z) = 1 - 4^{-z}$  is concave on  $[0, 1]$ . Hence,

$$\Pr[C_j \text{ satisfied}] \geq 1 - 4^{-z_j} \geq \frac{3}{4}z_j .$$

The function  $g(z) = 1 - 4^{-z}$  is concave on  $[0, 1]$ . Hence,

$$\Pr[C_j \text{ satisfied}] \geq 1 - 4^{-z_j} \geq \frac{3}{4}z_j .$$

Therefore,

$$E[W]$$

The function  $g(z) = 1 - 4^{-z}$  is concave on  $[0, 1]$ . Hence,

$$\Pr[C_j \text{ satisfied}] \geq 1 - 4^{-z_j} \geq \frac{3}{4}z_j .$$

Therefore,

$$E[W] = \sum_j w_j \Pr[C_j \text{ satisfied}]$$



The function  $g(z) = 1 - 4^{-z}$  is concave on  $[0, 1]$ . Hence,

$$\Pr[C_j \text{ satisfied}] \geq 1 - 4^{-z_j} \geq \frac{3}{4}z_j .$$

Therefore,

$$E[W] = \sum_j w_j \Pr[C_j \text{ satisfied}] \geq \frac{3}{4} \sum_j w_j z_j$$

The function  $g(z) = 1 - 4^{-z}$  is concave on  $[0, 1]$ . Hence,

$$\Pr[C_j \text{ satisfied}] \geq 1 - 4^{-z_j} \geq \frac{3}{4}z_j .$$

Therefore,

$$E[W] = \sum_j w_j \Pr[C_j \text{ satisfied}] \geq \frac{3}{4} \sum_j w_j z_j \geq \frac{3}{4} \text{OPT}$$

## Can we do better?

Not if we compare ourselves to the value of an optimum LP-solution.

### Definition 89 (Integrality Gap)

The integrality gap for an ILP is the worst-case ratio over all instances of the problem of the value of an optimal IP-solution to the value of an optimal solution to its linear programming relaxation.

Note that the integrality is less than one for maximization problems and larger than one for minimization problems (of course, equality is possible).

Note that an integrality gap only holds for one specific ILP formulation.

## Can we do better?

Not if we compare ourselves to the value of an optimum LP-solution.

### Definition 89 (Integrality Gap)

The integrality gap for an ILP is the worst-case ratio over all instances of the problem of the value of an optimal IP-solution to the value of an optimal solution to its linear programming relaxation.

Note that the integrality is less than one for maximization problems and larger than one for minimization problems (of course, equality is possible).

Note that an integrality gap only holds for one specific ILP formulation.

## Can we do better?

Not if we compare ourselves to the value of an optimum LP-solution.

### Definition 89 (Integrality Gap)

The integrality gap for an ILP is the worst-case ratio over all instances of the problem of the value of an optimal IP-solution to the value of an optimal solution to its linear programming relaxation.

Note that the integrality is less than one for maximization problems and larger than one for minimization problems (of course, equality is possible).

Note that an integrality gap only holds for one specific ILP formulation.

## Can we do better?

Not if we compare ourselves to the value of an optimum LP-solution.

### Definition 89 (Integrality Gap)

The integrality gap for an ILP is the worst-case ratio over all instances of the problem of the value of an optimal IP-solution to the value of an optimal solution to its linear programming relaxation.

Note that the integrality is less than one for maximization problems and larger than one for minimization problems (of course, equality is possible).

Note that an integrality gap only holds for one specific ILP formulation.

## Can we do better?

Not if we compare ourselves to the value of an optimum LP-solution.

### Definition 89 (Integrality Gap)

The integrality gap for an ILP is the worst-case ratio over all instances of the problem of the value of an optimal IP-solution to the value of an optimal solution to its linear programming relaxation.

Note that the integrality is less than one for maximization problems and larger than one for minimization problems (of course, equality is possible).

Note that an integrality gap only holds for one specific ILP formulation.

## Lemma 90

Our ILP-formulation for the MAXSAT problem has integrality gap at most  $\frac{3}{4}$ .

$$\begin{array}{ll} \max & \sum_j w_j z_j \\ \text{s.t.} & \forall j \quad \sum_{i \in P_j} y_i + \sum_{i \in N_j} (1 - y_i) \geq z_j \\ & \forall i \quad y_i \in \{0, 1\} \\ & \forall j \quad z_j \leq 1 \end{array}$$

Consider:  $(x_1 \vee x_2) \wedge (\bar{x}_1 \vee x_2) \wedge (x_1 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee \bar{x}_2)$

- ▶ any solution can satisfy at most 3 clauses
- ▶ we can set  $y_1 = y_2 = 1/2$  in the LP; this allows to set  $z_1 = z_2 = z_3 = z_4 = 1$
- ▶ hence, the LP has value 4.



## Lemma 90

Our ILP-formulation for the MAXSAT problem has integrality gap at most  $\frac{3}{4}$ .

$$\begin{array}{ll} \max & \sum_j w_j z_j \\ \text{s.t.} & \forall j \quad \sum_{i \in P_j} y_i + \sum_{i \in N_j} (1 - y_i) \geq z_j \\ & \forall i \quad y_i \in \{0, 1\} \\ & \forall j \quad z_j \leq 1 \end{array}$$

Consider:  $(x_1 \vee x_2) \wedge (\bar{x}_1 \vee x_2) \wedge (x_1 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee \bar{x}_2)$

- ▶ any solution can satisfy at most 3 clauses
- ▶ we can set  $y_1 = y_2 = 1/2$  in the LP; this allows to set  $z_1 = z_2 = z_3 = z_4 = 1$
- ▶ hence, the LP has value 4.

## MaxCut

Given a weighted graph  $G = (V, E, w)$ ,  $w(v) \geq 0$ , partition the vertices into two parts. Maximize the weight of edges between the parts.

## Trivial 2-approximation

# Semidefinite Programming

$$\begin{array}{ll} \max / \min & \sum_{i,j} c_{ij} x_{ij} \\ \text{s.t.} & \forall k \quad \sum_{i,j} a_{ijk} x_{ij} = b_k \\ & \forall i, j \quad x_{ij} = x_{ji} \\ & X = (x_{ij}) \text{ is psd.} \end{array}$$

- ▶ linear objective, linear constraints
- ▶ we can constrain a square matrix of variables to be symmetric positive definite

# Vector Programming

$$\begin{array}{ll} \max / \min & \sum_{i,j} c_{ij} (v_i^t v_j) \\ \text{s.t. } \forall k & \sum_{i,j,k} a_{ijk} (v_i^t v_j) = b_k \\ & v_i \in \mathbb{R}^n \end{array}$$

- ▶ variables are vectors in  $n$ -dimensional space
- ▶ objective functions and constraints are linear in inner products of the vectors

This is equivalent!

## **Fact [without proof]**

We (essentially) can solve Semidefinite Programs in polynomial time...

# Quadratic Programs

**Quadratic Program for MaxCut:**

$$\begin{array}{ll} \max & \frac{1}{2} \sum_{i,j} w_{ij} (1 - y_i y_j) \\ \forall i & y_i \in \{-1, 1\} \end{array}$$

This is exactly MaxCut!

# Semidefinite Relaxation

$$\begin{array}{ll} \max & \frac{1}{2} \sum_{i,j} w_{ij} (1 - v_i^t v_j) \\ & \forall i \quad v_i^t v_i = 1 \\ & \forall i \quad v_i \in \mathbb{R}^n \end{array}$$

- ▶ this is clearly a relaxation
- ▶ the solution will be vectors on the unit sphere

# Rounding the SDP-Solution

- ▶ Choose a random vector  $r$  such that  $r/\|r\|$  is uniformly distributed on the unit sphere.
- ▶ If  $r^t v_i > 0$  set  $y_i = 1$  else set  $y_i = -1$



## Rounding the SDP-Solution

Choose the  $i$ -th coordinate  $r_i$  as a Gaussian with mean 0 and variance 1, i.e.,  $r_i \sim \mathcal{N}(0, 1)$ .

Density function:

$$\varphi(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$$

Then

$$\begin{aligned} \Pr[r = (x_1, \dots, x_n)] &= \frac{1}{(\sqrt{2\pi})^n} e^{-x_1^2/2} \cdot e^{-x_2^2/2} \cdot \dots \cdot e^{-x_n^2/2} dx_1 \cdot \dots \cdot dx_n \\ &= \frac{1}{(\sqrt{2\pi})^n} e^{-\frac{1}{2}(x_1^2 + \dots + x_n^2)} dx_1 \cdot \dots \cdot dx_n \end{aligned}$$

Hence the probability for a point only depends on its distance to the origin.

## Rounding the SDP-Solution

Choose the  $i$ -th coordinate  $r_i$  as a Gaussian with mean 0 and variance 1, i.e.,  $r_i \sim \mathcal{N}(0, 1)$ .

Density function:

$$\varphi(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$$

Then

$$\begin{aligned} \Pr[r = (x_1, \dots, x_n)] &= \frac{1}{(\sqrt{2\pi})^n} e^{-x_1^2/2} \cdot e^{-x_2^2/2} \cdot \dots \cdot e^{-x_n^2/2} dx_1 \cdot \dots \cdot dx_n \\ &= \frac{1}{(\sqrt{2\pi})^n} e^{-\frac{1}{2}(x_1^2 + \dots + x_n^2)} dx_1 \cdot \dots \cdot dx_n \end{aligned}$$

Hence the probability for a point only depends on its distance to the origin.

# Rounding the SDP-Solution

## Fact

The projection of  $r$  onto two unit vectors  $e_1$  and  $e_2$  are independent and are normally distributed with mean 0 and variance 1 iff  $e_1$  and  $e_2$  are orthogonal.

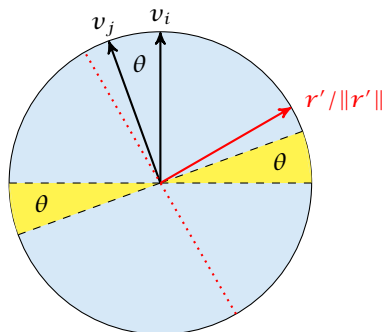
Note that this is clear if  $e_1$  and  $e_2$  are standard basis vectors.

# Rounding the SDP-Solution

## Corollary

If we project  $r$  onto a hyperplane its normalized projection  $(r' / \|r'\|)$  is uniformly distributed on the unit circle within the hyperplane.

# Rounding the SDP-Solution



- ▶ if the normalized projection falls into the shaded region,  $v_i$  and  $v_j$  are rounded to different values
- ▶ this happens with probability  $\theta/\pi$

# Rounding the SDP-Solution

- ▶ contribution of edge  $(i, j)$  to the SDP-relaxation:

$$\frac{1}{2}w_{ij}(1 - v_i^t v_j)$$

- ▶ (expected) contribution of edge  $(i, j)$  to the rounded instance  $w_{ij} \arccos(v_i^t v_j) / \pi$
- ▶ ratio is at most

$$\min_{x \in [-1, 1]} \frac{2 \arccos(x)}{\pi(1-x)} \geq 0.878$$

# Rounding the SDP-Solution

- ▶ contribution of edge  $(i, j)$  to the SDP-relaxation:

$$\frac{1}{2}w_{ij}(1 - v_i^t v_j)$$

- ▶ (expected) contribution of edge  $(i, j)$  to the rounded instance  $w_{ij} \arccos(v_i^t v_j) / \pi$
- ▶ ratio is at most

$$\min_{x \in [-1, 1]} \frac{2 \arccos(x)}{\pi(1-x)} \geq 0.878$$

# Rounding the SDP-Solution

- ▶ contribution of edge  $(i, j)$  to the SDP-relaxation:

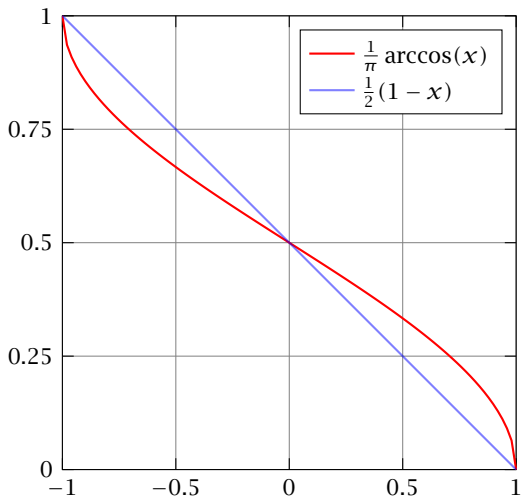
$$\frac{1}{2}w_{ij}(1 - v_i^t v_j)$$

- ▶ (expected) contribution of edge  $(i, j)$  to the rounded instance  $w_{ij} \arccos(v_i^t v_j) / \pi$
- ▶ ratio is at most

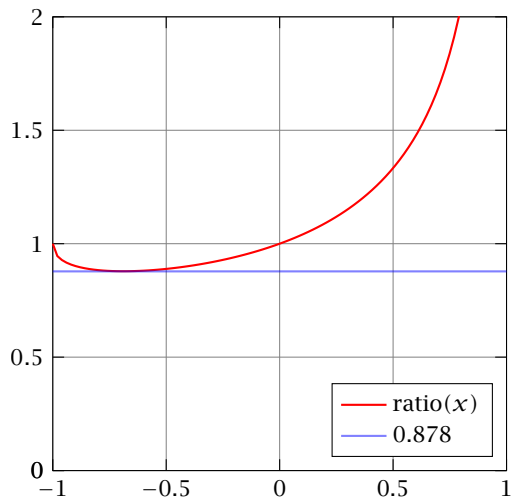
$$\min_{x \in [-1, 1]} \frac{2 \arccos(x)}{\pi(1 - x)} \geq 0.878$$



# Rounding the SDP-Solution



## Rounding the SDP-Solution



# Rounding the SDP-Solution

## Theorem 91

*Given the unique games conjecture, there is no  $\alpha$ -approximation for the maximum cut problem with constant*

$$\alpha > \min_{x \in [-1,1]} \frac{2 \arccos(x)}{\pi(1-x)}$$

*unless  $P = NP$ .*