Approximation algorithms: Linear and Integer Programming

Maria Serna

Spring 2025-2026

1. LP and IP
2. Relax and round
3. LP Duality

Linear programming

- In a linear programming problem, we are given a set of variables, an objective linear function a set of linear constrains and want to assign real values to the variables as to:

Linear programming

- In a linear programming problem, we are given a set of variables, an objective linear function a set of linear constrains and want to assign real values to the variables as to:
    - satisfy the set of linear inequalities (equations or constraints),

Linear programming

- In a linear programming problem, we are given a set of variables, an objective linear function a set of linear constrains and want to assign real values to the variables as to:
  - satisfy the set of linear inequalities (equations or constraints),
  - maximize or minimize the objective function.

Linear programming

- In a linear programming problem, we are given a set of variables, an objective linear function a set of linear constrains and want to assign real values to the variables as to:
  - satisfy the set of linear inequalities (equations or constraints),
  - maximize or minimize the objective function.
- LP is a pure algebraic problem.

Linear programming: An example

$$\max\ x_1 + 6x_2$$

subject to

$$x_1 \leq 200$$

$$x_2 \leq 300$$

$$x_1 + x_2 \leq 400$$

$$x_1, x_2 \geq 0$$

Linear programming: feasible region

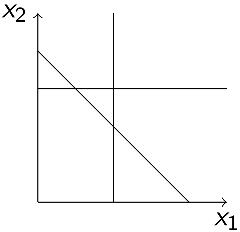
- A linear equality defines a hyperplane.
- A linear inequality defines a half-space.

Linear programming: feasible region

- A linear equality defines a hyperplane.
- A linear inequality defines a half-space.
- The solutions to the linear constraints lie inside a feasible region limited by the polytope (convex polygon in $\mathbb{R}^2$) defined by the linear constraints.

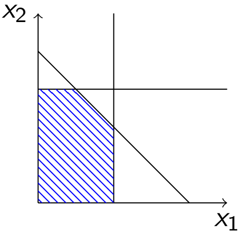
Linear programming: feasible region

- A linear equality defines a hyperplane.
- A linear inequality defines a half-space.
- The solutions to the linear constraints lie inside a feasible region limited by the polytope (convex polygon in $\mathbb{R}^2$) defined by the linear constraints.



Linear programming: feasible region

- A linear equality defines a hyperplane.
- A linear inequality defines a half-space.
- The solutions to the linear constraints lie inside a feasible region limited by the polytope (convex polygon in $\mathbb{R}^2$) defined by the linear constraints.



Linear programming: infeasibility

- A linear programming is infeasible if

Linear programming: infeasibility

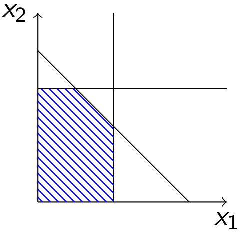
- A linear programming is infeasible if
  - The constrains are so tight that it is impossible to satisfy all of them. For ex. $x \geq 2$ and $x \leq 1$

Linear programming: infeasibility

- A linear programming is infeasible if
    - The constrains are so tight that it is impossible to satisfy all of them. For ex. $x \geq 2$ and $x \leq 1$
    - The constrains are so loose that the feasible region is unbounded allowing the objective function to go to ∞ .
    For ex. max $x_1 + x_2$ subject to $x_1, x_2 \geq 0$

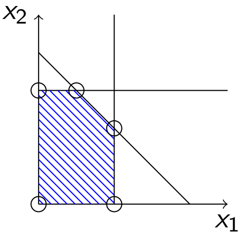
Linear programming: optimum

- In a feasible linear programming the optimum is achieved at a vertex of the feasible region.



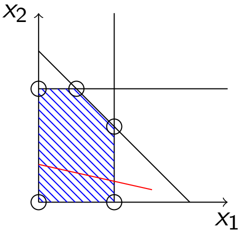
Linear programming: optimum

- In a feasible linear programming the optimum is achieved at a vertex of the feasible region.



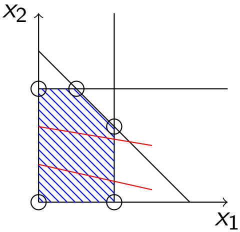
Linear programming: optimum

- In a feasible linear programming the optimum is achieved at a vertex of the feasible region.



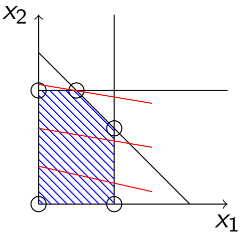
Linear programming: optimum

- In a feasible linear programming the optimum is achieved at a vertex of the feasible region.



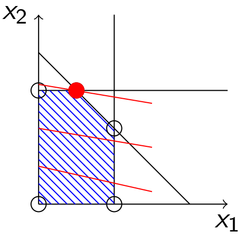
Linear programming: optimum

- In a feasible linear programming the optimum is achieved at a vertex of the feasible region.



Linear programming: optimum

- In a feasible linear programming the optimum is achieved at a vertex of the feasible region.



Linear programming: standard formulation

A LP has many degrees of freedom.

Linear programming: standard formulation

A LP has many degrees of freedom.

- maximization or minimization.

Linear programming: standard formulation

A LP has many degrees of freedom.

- maximization or minimization.
- constrains could be $=$, $\geq$, $\leq$, $<$ or $>$.

Linear programming: standard formulation

A LP has many degrees of freedom.

- maximization or minimization.
- constrains could be $=$, $\geq$, $\leq$, $<$ or $>$.
- variables are often restricted to be non-negative, but they also could be unrestricted.

Linear programming: standard formulation

A LP has many degrees of freedom.

- maximization or minimization.
- constrains could be $=$, $\geq$, $\leq$, $<$ or $>$.
- variables are often restricted to be non-negative, but they also could be unrestricted.

- standard form?

Linear programming: standard formulation

- From max to min (or min to max)

Linear programming: standard formulation

- From max to min (or min to max)
  multiply by -1 the coefficients of the objective function.

Linear programming: standard formulation

- From max to min (or min to max)
  multiply by -1 the coefficients of the objective function.
- To reverse an inequality (for ex. $\geq$ to $\leq$)

Linear programming: standard formulation

- From max to min (or min to max)
  multiply by -1 the coefficients of the objective function.
- To reverse an inequality (for ex. $\geq$ to $\leq$)
  multiply all coefficients and the independent term by -1.

Linear programming: standard formulation

- From max to min (or min to max)
  multiply by -1 the coefficients of the objective function.
- To reverse an inequality (for ex. $\geq$ to $\leq$)
  multiply all coefficients and the independent term by -1.
- From $<$ to $\leq$ (or to $=$)

Linear programming: standard formulation

- From max to min (or min to max)
  multiply by -1 the coefficients of the objective function.
- To reverse an inequality (for ex. $\geq$ to $\leq$)
  multiply all coefficients and the independent term by -1.
- From $<$ to $\leq$ (or to $=$)
  create a new positive variable and add it with coefficient 1 to the left par of the inequality.

Linear programming: standard formulation

- From max to min (or min to max)
  multiply by -1 the coefficients of the objective function.
- To reverse an inequality (for ex. $\geq$ to $\leq$)
  multiply all coefficients and the independent term by -1.
- From $<$ to $\leq$ (or to $=$)
  create a new positive variable and add it with coefficient 1 to the left par of the inequality.
- From $=$ to $\leq$ (or to $\geq$)

Linear programming: standard formulation

- From max to min (or min to max)
  multiply by -1 the coefficients of the objective function.
- To reverse an inequality (for ex. $\geq$ to $\leq$)
  multiply all coefficients and the independent term by -1.
- From $<$ to $\leq$ (or to $=$)
  create a new positive variable and add it with coefficient 1 to the left par of the inequality.
- From $=$ to $\leq$ (or to $\geq$)
  put two versions one with $\leq$ and the other with $\geq$, multiply the last one by -1.

Linear programming: standard formulation

- From max to min (or min to max)
  multiply by -1 the coefficients of the objective function.
- To reverse an inequality (for ex. $\geq$ to $\leq$)
  multiply all coefficients and the independent term by -1.
- From $<$ to $\leq$ (or to $=$)
  create a new positive variable and add it with coefficient 1 to the left par of the inequality.
- From $=$ to $\leq$ (or to $\geq$)
  put two versions one with $\leq$ and the other with $\geq$, multiply the last one by -1.
- From x unrestricted to non-negative variables,

Linear programming: standard formulation

- From max to min (or min to max)
  multiply by -1 the coefficients of the objective function.
- To reverse an inequality (for ex. $\geq$ to $\leq$)
  multiply all coefficients and the independent term by -1.
- From $<$ to $\leq$ (or to $=$)
  create a new positive variable and add it with coefficient 1 to the left par of the inequality.
- From $=$ to $\leq$ (or to $\geq$)
  put two versions one with $\leq$ and the other with $\geq$, multiply the last one by -1.
- From x unrestricted to non-negative variables,
  create two new variables x^+ and x^-, both non negative, replace x by $x^+ - x^-$.

Linear programming: standard formulation

LP standard form

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & Ax \geq b \\ & x \geq 0 \end{aligned}$$

Where

- $x = (x_1, \dots, x_n)$, $c = (c_1, \dots, c_n)$.
- $b^T = (b_1, \dots, b_m)$
- A is a $n \times m$ matrix.

Linear programming: problem

Given

- $c = (c_1, \dots, c_n)$,
- $b^T = (b_1, \dots, b_m)$,
- and a $n \times m$ matrix A.

find $x = (x_1, \dots, x_n) \geq 0$, so that

- $Ax \geq b$ and $c^T x$ is minimized.

Linear programming: algorithms

Linear programming: algorithms

We can solve Linear Programming in polynomial time

Linear programming: algorithms

We can solve Linear Programming in polynomial time

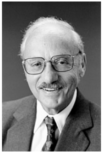
Linear programming: algorithms

We can solve Linear Programming in polynomial time

- Simplex method: Dantzig in 1947
  (exponential time Klee and Minty 1972)

Linear programming: algorithms

We can solve Linear Programming in polynomial time



- Simplex method: Dantzig in 1947
  (exponential time Klee and Minty 1972)
- Ellipsoid method: Khachiyan 1979 ($O(n^6)$)

Linear programming: algorithms

We can solve Linear Programming in polynomial time

- Simplex method: Dantzig in 1947
  (exponential time Klee and Minty 1972)
- Ellipsoid method: Khachiyan 1979 ($O(n^6)$)
- Interior-point method: Karmarkar 1984 ($O(n^3)$)

Linear programming: algorithms

We can solve Linear Programming in polynomial time

- Simplex method: Dantzig in 1947
  (exponential time Klee and Minty 1972)
- Ellipsoid method: Khachiyan 1979 ($O(n^6)$)
- Interior-point method: Karmarkar 1984 ($O(n^3)$)

- Most used algorithm is still Simplex (fast on average).
- Many commercial LP solvers CPLEX and open source Gurobi

Integer programming

Integer programming

- An integer programming (IP) problem is a linear programming problem with the additional restriction that the values of the variables must be integers.

Integer programming

- An integer programming (IP) problem is a linear programming problem with the additional restriction that the values of the variables must be integers.
- A mixed integer programming (MIP) problem is a linear programming problem with the additional restriction that, the values of some variables must be integers.

Integer programming

- An integer programming (IP) problem is a linear programming problem with the additional restriction that the values of the variables must be integers.
- A mixed integer programming (MIP) problem is a linear programming problem with the additional restriction that, the values of some variables must be integers.

- Many NPO problems can be easily expressed as IP or MIP problems

Integer programming

- An integer programming (IP) problem is a linear programming problem with the additional restriction that the values of the variables must be integers.
- A mixed integer programming (MIP) problem is a linear programming problem with the additional restriction that, the values of some variables must be integers.

- Many NPO problems can be easily expressed as IP or MIP problems
- IP is NP-hard

Max SAT as integer program

- Max Sat: Input a set of m clauses on n variables, find an assignment that maximizes the number of satisfied clauses.

Max SAT as integer program

- Max Sat: Input a set of m clauses on n variables, find an assignment that maximizes the number of satisfied clauses.
- For a clause j, the set of variables that appear in C_j
  - positive is $P(j)$
  - negative is $N(j)$

Max SAT as integer program

- Max Sat: Input a set of m clauses on n variables, find an assignment that maximizes the number of satisfied clauses.
- For a clause j, the set of variables that appear in C_j
  - positive is $P(j)$
  - negative is $N(j)$
- We consider $n + m$ integer variables,

Max SAT as integer program

- Max Sat: Input a set of m clauses on n variables, find an assignment that maximizes the number of satisfied clauses.
- For a clause j, the set of variables that appear in C_j
  - positive is $P(j)$
  - negative is $N(j)$
- We consider $n + m$ integer variables,
  - $x_1, \ldots, x_n$, one per each variable

Max SAT as integer program

- Max Sat: Input a set of m clauses on n variables, find an assignment that maximizes the number of satisfied clauses.
- For a clause j, the set of variables that appear in C_j
  - positive is $P(j)$
  - negative is $N(j)$
- We consider $n + m$ integer variables,
  - $x_1, \dots, x_n$, one per each variable
  - $y_1, \dots, y_m$, one per each clause

Max SAT as integer program

- Max Sat: Input a set of m clauses on n variables, find an assignment that maximizes the number of satisfied clauses.
- For a clause j, the set of variables that appear in C_j
  - positive is $P(j)$
  - negative is $N(j)$
- We consider $n + m$ integer variables,
  - $x_1, \ldots, x_n$, one per each variable
  - $y_1, \ldots, y_m$, one per each clause

  The variables will be restricted to have values in $\{0, 1\}$
  This is a simplification of saying that they must hold integer values and that all of them are ≤ 1.

Max SAT as integer program

Max SAT-IP

$$
\begin{array}{ll}
\max & \displaystyle\sum_{j=1}^{m} y_j \\
\text{s.t.} & \displaystyle\sum_{i\in P(j)} x_i + \sum_{i\in N(j)} (1-x_i) \geq y_j \qquad 1 \leq j \leq m \\
& y_j \in \{0,1\} \qquad 1 \leq j \leq m \\
& x_i \in \{0,1\} \qquad 1 \leq i \leq n
\end{array}
$$

The size of the IP is polynomial in the size of the Max SAT,

Max SAT as integer program

Max SAT-IP

$$\begin{array}{ll} \max & \displaystyle\sum_{j=1}^{m} y_j \\ \text{s.t.} & \displaystyle\sum_{i\in P(j)} x_i + \sum_{i\in N(j)} (1-x_i) \geq y_j \qquad 1 \leq j \leq m \\ & y_j \in \{0,1\} \qquad 1 \leq j \leq m \\ & x_i \in \{0,1\} \qquad 1 \leq i \leq n \end{array}$$

The size of the IP is polynomial in the size of the Max SAT, so the transformation is a polynomial Turing reduction from Max SAT to IP.

Vertex cover as integer program

VC

Given a graph $G = (V, E)$ we want to find a set $S \subset V$ with minimum cardinality, so that every edge in G has at least one end point in S.

Vertex cover as integer program

VC

Given a graph $G = (V, E)$ we want to find a set $S \subset V$ with minimum cardinality, so that every edge in G has at least one end point in S.

VC-IP

$$
\begin{array}{lll}
\text{min} & \displaystyle\sum_{i=1}^{n} x_i & \\
\text{s.t.} & x_i + x_j \geq 1 & \text{for all } (i,j) \in E \\
 & x_i \in \{0,1\} & \text{for all } i \in V
\end{array}
$$

Weighted Vertex cover as integer program

WVC

Given a graph $G = (V, A)$ with weights w associated to the vertices, we want to find a set $S \subset V$ with minimum weight, so that every edge in G has at least one end point in S.

Weighted Vertex cover as integer program

WVC

Given a graph $G = (V, A)$ with weights w associated to the vertices, we want to find a set $S \subset V$ with minimum weight, so that every edge in G has at least one end point in S.

VC-IP

$$
\begin{array}{lll}
\text{min} & \displaystyle\sum_{i=1}^{n} w_i x_i & \\
\text{s.t.} & x_i + x_j \geq 1 & \text{for all } (i,j) \in E \\
 & x_i \in \{0,1\} & \text{for all } i \in V
\end{array}
$$

Exercise

Try to write a LP or IP formulation for the problems

- Min Weighted Matching
- Set cover
- Max Flow

1 LP and IP

2 Relax and round

3 LP Duality

Relaxation and rounding

- Many real-life problems can be modeled as Integer Linear Programs (IP).
- The IP can be relaxed to a linear program (LP) by eliminating the integrity constraints.

Relaxation and rounding

- Many real-life problems can be modeled as Integer Linear Programs (IP).
- The IP can be relaxed to a linear program (LP) by eliminating the integrity constraints.
- By doing so the optimum cost can only improve, i.e.,
  opt of LP is better than opt of IP.

Relaxation and rounding

- Many real-life problems can be modeled as Integer Linear Programs (IP).
- The IP can be relaxed to a linear program (LP) by eliminating the integrity constraints.
- By doing so the optimum cost can only improve, i.e.,
  opt of LP is better than opt of IP.
- We can solve the LP in polynomial time.

Relaxation and rounding

- Many real-life problems can be modeled as Integer Linear Programs (IP).
- The IP can be relaxed to a linear program (LP) by eliminating the integrity constraints.
- By doing so the optimum cost can only improve, i.e.,
  opt of LP is better than opt of IP.
- We can solve the LP in polynomial time.
- The LP optimal solution might not be integral, when possible, transform it to get a feasible integer solution not far from opt of IP.

Vertex cover

VC

Given a graph $G = (V, A)$ we want to find a set $S \subset V$ with minimum cardinality, so that every edge in G has at least one end point in S.

VC-IP

$$\begin{array}{ll} \text{min} & \sum_{i=1}^{n} x_i \\ \text{s.t.} & x_i + x_j \geq 1 \quad \text{for all } (i,j) \in E \\ & x_i \in \{0,1\} \quad \text{for all } i \in V \end{array}$$

Vertex cover

VC

Given a graph $G = (V, A)$ we want to find a set $S \subset V$ with minimum cardinality, so that every edge in G has at least one end point in S.

VC-IP	VC-LP
$\begin{array}{lll} \text{min} & \sum_{i=1}^{n} x_i & \\ \text{s.t.} & x_i + x_j \geq 1 & \text{for all } (i,j) \in E \\ & x_i \in \{0,1\} & \text{for all } i \in V \end{array}$	$\begin{array}{lll} \text{min} & \sum_{i=1}^{n} x_i & \\ \text{s.t.} & x_i + x_j \geq 1 & \text{for all}(i,j) \in E \\ & x_i \geq 0 & \text{for all } i \in V \end{array}$

Vertex cover: another approximation algorithm

Vertex cover: another approximation algorithm

Lemma

VC-LP *has an optimal solution* x^* *such that* $x_i \in \{0, 1, 1/2\}$. *Furthermore, such a solution can be computed in polynomial time.*

Proof.

Vertex cover: another approximation algorithm

Lemma

VC-LP *has an optimal solution* x^* *such that* $x_i \in \{0, 1, 1/2\}$.
Furthermore, such a solution can be computed in polynomial time.

Proof.

Let y be an optimal solution s.t. not all its coordinates are in $\{0, 1, 1/2\}$.

Vertex cover: another approximation algorithm

Lemma

VC-LP *has an optimal solution* x^* *such that* $x_i \in \{0, 1, 1/2\}$.
Furthermore, such a solution can be computed in polynomial time.

Proof.

Let y be an optimal solution s.t. not all its coordinates are in $\{0, 1, 1/2\}$.
Set $\epsilon = \min_{y_i \notin \{0,1,1/2\}} \{y_i, |y_i - 1/2|, 1 - y_i\}$. Consider

Vertex cover: another approximation algorithm

Lemma

VC-LP *has an optimal solution* x^* *such that* $x_i \in \{0, 1, 1/2\}$.
Furthermore, such a solution can be computed in polynomial time.

Proof.

Let y be an optimal solution s.t. not all its coordinates are in $\{0, 1, 1/2\}$.
Set $\epsilon = \min_{y_i \notin \{0,1,1/2\}} \{y_i, |y_i - 1/2|, 1 - y_i\}$. Consider

$$y'_i = \begin{cases} y_i - \epsilon & 0 < y_i < 1/2 \\ y_i + \epsilon & 1/2 < y_i < 1 \\ y_i & \text{otherwise} \end{cases} \qquad y''_i = \begin{cases} y_i + \epsilon & 0 < y_i < 1/2 \\ y_i - \epsilon & 1/2 < y_i < 1 \\ y_i & \text{otherwise} \end{cases}$$

Vertex cover: another approximation algorithm

Lemma

VC-LP *has an optimal solution* x^* *such that* $x_i \in \{0, 1, 1/2\}$. *Furthermore, such a solution can be computed in polynomial time.*

Proof.

Let y be an optimal solution s.t. not all its coordinates are in $\{0, 1, 1/2\}$. Set $\epsilon = \min_{y_i \notin \{0,1,1/2\}} \{y_i, |y_i - 1/2|, 1 - y_i\}$. Consider

$$y'_i = \begin{cases} y_i - \epsilon & 0 < y_i < 1/2 \\ y_i + \epsilon & 1/2 < y_i < 1 \\ y_i & \text{otherwise} \end{cases} \qquad y''_i = \begin{cases} y_i + \epsilon & 0 < y_i < 1/2 \\ y_i - \epsilon & 1/2 < y_i < 1 \\ y_i & \text{otherwise} \end{cases}$$

$\sum y_i = (\sum y'_i + \sum y''_i)/2$, so both are optimal solutions.

Vertex cover: another approximation algorithm

Lemma

VC-LP *has an optimal solution* x^* *such that* $x_i \in \{0, 1, 1/2\}$.
Furthermore, such a solution can be computed in polynomial time.

Proof.

Let y be an optimal solution s.t. not all its coordinates are in $\{0, 1, 1/2\}$.
Set $\epsilon = \min_{y_i \notin \{0,1,1/2\}} \{y_i, |y_i - 1/2|, 1 - y_i\}$. Consider

$$y'_i = \begin{cases} y_i - \epsilon & 0 < y_i < 1/2 \\ y_i + \epsilon & 1/2 < y_i < 1 \\ y_i & \text{otherwise} \end{cases} \qquad y''_i = \begin{cases} y_i + \epsilon & 0 < y_i < 1/2 \\ y_i - \epsilon & 1/2 < y_i < 1 \\ y_i & \text{otherwise} \end{cases}$$

$\sum y_i = (\sum y'_i + \sum y''_i)/2$, so both are optimal solutions. One of them has more $\{0, 1, 1/2\}$ coordinates than y.

Vertex cover: another approximation algorithm

Lemma

VC-LP *has an optimal solution* x^* *such that* $x_i \in \{0, 1, 1/2\}$. *Furthermore, such a solution can be computed in polynomial time.*

Proof.

Let y be an optimal solution s.t. not all its coordinates are in $\{0, 1, 1/2\}$. Set $\epsilon = \min_{y_i \notin \{0,1,1/2\}} \{y_i, |y_i - 1/2|, 1 - y_i\}$. Consider

$$y'_i = \begin{cases} y_i - \epsilon & 0 < y_i < 1/2 \\ y_i + \epsilon & 1/2 < y_i < 1 \\ y_i & \text{otherwise} \end{cases} \qquad y''_i = \begin{cases} y_i + \epsilon & 0 < y_i < 1/2 \\ y_i - \epsilon & 1/2 < y_i < 1 \\ y_i & \text{otherwise} \end{cases}$$

$\sum y_i = (\sum y'_i + \sum y''_i)/2$, so both are optimal solutions. One of them has more $\{0, 1, 1/2\}$ coordinates than y. ☺

Vertex cover

Vertex cover

```
function RELAX+ROUND VC($G$)
    Construct the LP-VC associated $G$
    Let $y$ be an optimal relaxed solution (of the LP instance)
```

Vertex cover

```
function RELAX+ROUND VC($G$)
    Construct the LP-VC associated $G$
    Let $y$ be an optimal relaxed solution (of the LP instance)
    Using the previous lemma, construct an optimal relaxed
        solution $y'$ such that $y'_i \in \{0, 1, 1/2\}$
```

Vertex cover

```
function RELAX+ROUND VC($G$)
    Construct the LP-VC associated $G$
    Let $y$ be an optimal relaxed solution (of the LP instance)
    Using the previous lemma, construct an optimal relaxed
        solution $y'$ such that $y'_i \in \{0, 1, 1/2\}$
    Let $x$ defined as $x_i = 0$ if $y'_i = 0$, $x_i = 1$ otherwise.
    return ($x$)
```

Vertex cover

```
function RELAX+ROUND VC($G$)
    Construct the LP-VC associated $G$
    Let $y$ be an optimal relaxed solution (of the LP instance)
    Using the previous lemma, construct an optimal relaxed
        solution $y'$ such that $y'_i \in \{0, 1, 1/2\}$
    Let $x$ defined as $x_i = 0$ if $y'_i = 0$, $x_i = 1$ otherwise.
    return ($x$)
```

RELAX+ROUND VC

Vertex cover

```
function RELAX+ROUND VC($G$)
    Construct the LP-VC associated $G$
    Let $y$ be an optimal relaxed solution (of the LP instance)
    Using the previous lemma, construct an optimal relaxed
        solution $y'$ such that $y'_i \in \{0, 1, 1/2\}$
    Let $x$ defined as $x_i = 0$ if $y'_i = 0$, $x_i = 1$ otherwise.
    return ($x$)
```

RELAX+ROUND VC

- runs in polynomial time

Vertex cover

```
function Relax+Round VC($G$)
    Construct the LP-VC associated $G$
    Let $y$ be an optimal relaxed solution (of the LP instance)
    Using the previous lemma, construct an optimal relaxed
        solution $y'$ such that $y'_i \in \{0, 1, 1/2\}$
    Let $x$ defined as $x_i = 0$ if $y'_i = 0$, $x_i = 1$ otherwise.
    return ($x$)
```

Relax+Round VC

- runs in polynomial time
- x defines a vertex cover

Vertex cover

```
function RELAX+ROUND VC($G$)
    Construct the LP-VC associated $G$
    Let $y$ be an optimal relaxed solution (of the LP instance)
    Using the previous lemma, construct an optimal relaxed
        solution $y'$ such that $y'_i \in \{0, 1, 1/2\}$
    Let $x$ defined as $x_i = 0$ if $y'_i = 0$, $x_i = 1$ otherwise.
    return ($x$)
```

RELAX+ROUND VC

- runs in polynomial time
- x defines a vertex cover
- $\sum_{i=1}^{n} x_i \leq 2 \sum_{i=1}^{n} y'_i \leq 2\text{opt}$

Vertex cover

```
function RELAX+ROUND VC($G$)
    Construct the LP-VC associated $G$
    Let $y$ be an optimal relaxed solution (of the LP instance)
    Using the previous lemma, construct an optimal relaxed
        solution $y'$ such that $y'_i \in \{0, 1, 1/2\}$
    Let $x$ defined as $x_i = 0$ if $y'_i = 0$, $x_i = 1$ otherwise.
    return ($x$)
```

RELAX+ROUND VC

- runs in polynomial time
- x defines a vertex cover
- $\sum_{i=1}^{n} x_i \leq 2 \sum_{i=1}^{n} y'_i \leq 2\text{opt}$
- is a 2-approximation for VC.

Weighted vertex cover: Relax+Round approximation

LP WVC

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} w_i x_i \\ \text{s.t.} \quad & x_i + x_j \geq 1 \quad \text{for all} (i,j) \in E \\ & x_i \geq 0 \quad \text{for all } i \in V \end{aligned}$$

Weighted vertex cover: Relax+Round approximation

LP WVC

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} w_i x_i \\ \text{s.t.} \quad & x_i + x_j \geq 1 \quad \text{for all} (i,j) \in E \\ & x_i \geq 0 \quad \text{for all } i \in V \end{aligned}$$

```
function WVC$(G, c)$
    Construct the LP WVC, $I$
    $y = LP.solve(I)$
    for $i = 1, \dots, n$ do
        if $y_i < 1/2$ then
            $x_i = 0$
        else
            $x_i = 1$
    return $(x)$
```

Weighted vertex cover: Relax+Round approximation

LP WVC

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} w_i x_i \\ \text{s.t.} \quad & x_i + x_j \geq 1 \quad \text{for all} (i,j) \in E \\ & x_i \geq 0 \quad \text{for all } i \in V \end{aligned}$$

```
function WVC(G, c)
    Construct the LP WVC, $I$
    $y = LP.solve(I)$
    for $i = 1, \dots, n$ do
        if $y_i < 1/2$ then
            $x_i = 0$
        else
            $x_i = 1$
    return ($x$)
```

Relax+Round WVC

Weighted vertex cover: Relax+Round approximation

LP WVC

$$\begin{array}{lll} \min & \sum_{i=1}^{n} w_i x_i & \\ \text{s.t.} & x_i + x_j \geq 1 & \text{for all}(i,j) \in E \\ & x_i \geq 0 & \text{for all } i \in V \end{array}$$

```
function WVC(G, c)
    Construct the LP WVC, $I$
    $y = LP.solve(I)$
    for $i = 1, \ldots, n$ do
        if $y_i < 1/2$ then
            $x_i = 0$
        else
            $x_i = 1$
    return $(x)$
```

RELAX+ROUND WVC

- runs in polynomial time

Weighted vertex cover: Relax+Round approximation

LP WVC

$$\begin{array}{lll} \min & \displaystyle\sum_{i=1}^{n} w_i x_i & \\ \text{s.t.} & x_i + x_j \geq 1 & \text{for all}(i,j) \in E \\ & x_i \geq 0 & \text{for all } i \in V \end{array}$$

```
function WVC($(G, c)$)
    Construct the LP WVC, $I$
    $y = LP.solve(I)$
    for $i = 1, \ldots, n$ do
        if $y_i < 1/2$ then
            $x_i = 0$
        else
            $x_i = 1$
    return $(x)$
```

RELAX+ROUND WVC

- runs in polynomial time
- x defines a vertex cover

Weighted vertex cover: Relax+Round approximation

LP WVC

$$\begin{array}{lll} \min & \sum_{i=1}^{n} w_i x_i & \\ \text{s.t.} & x_i + x_j \geq 1 & \text{for all}(i,j) \in E \\ & x_i \geq 0 & \text{for all } i \in V \end{array}$$

```
function WVC(G, c)
    Construct the LP WVC, $I$
    $y = LP.solve(I)$
    for $i = 1, \dots, n$ do
        if $y_i < 1/2$ then
            $x_i = 0$
        else
            $x_i = 1$
    return $(x)$
```

RELAX+ROUND WVC

- runs in polynomial time
- x defines a vertex cover
- $\sum_{i=1}^{n} w_i x_i \leq 2 \sum_{i=1}^{n} w_i y_i \leq 2\text{opt}$

Weighted vertex cover: Relax+Round approximation

LP WVC

$$\begin{array}{lll} \min & \sum_{i=1}^{n} w_i x_i & \\ \text{s.t.} & x_i + x_j \geq 1 & \text{for all} (i,j) \in E \\ & x_i \geq 0 & \text{for all } i \in V \end{array}$$

```
function WVC(G, c)
    Construct the LP WVC, I
    y = LP.solve(I)
    for i = 1, ..., n do
        if y_i < 1/2 then
            x_i = 0
        else
            x_i = 1
    return (x)
```

RELAX+ROUND WVC

- runs in polynomial time
- x defines a vertex cover
- $\sum_{i=1}^{n} w_i x_i \leq 2 \sum_{i=1}^{n} w_i y_i \leq 2\text{opt}$
- is a 2-approximation for WVC.

Minimum 2-Satisfiability

MIN 2-SAT

Given a Boolean formula in 2-CNF, determine whether it is satisfiable and, in such a case, find a satisfying assignment with minimum number of true variables.

Minimum 2-Satisfiability

MIN 2-SAT
Given a Boolean formula in 2-CNF, determine whether it is satisfiable and, in such a case, find a satisfying assignment with minimum number of true variables.

- 2-SAT

Minimum 2-Satisfiability

MIN 2-SAT

Given a Boolean formula in 2-CNF, determine whether it is satisfiable and, in such a case, find a satisfying assignment with minimum number of true variables.

- 2-SAT can be solved in polynomial time.

Minimum 2-Satisfiability

MIN 2-SAT

Given a Boolean formula in 2-CNF, determine whether it is satisfiable and, in such a case, find a satisfying assignment with minimum number of true variables.

- 2-SAT can be solved in polynomial time.
- MIN 2-SAT is NP-hard.

Minimum 2-Satisfiability

MIN 2-SAT

Given a Boolean formula in 2-CNF, determine whether it is satisfiable and, in such a case, find a satisfying assignment with minimum number of true variables.

- 2-SAT can be solved in polynomial time.
- Min 2-SAT is NP-hard.
- Min 2-SAT IP formulation?

Minimum 2-Satisfiability: IP formulation

Suppose that F has n variables $x_1, \dots x_n$ and m clauses with 2 literals per clause

Minimum 2-Satisfiability: IP formulation

Suppose that F has n variables $x_1, \dots x_n$ and m clauses with 2 literals per clause

IP Min 2-SAT

$$\begin{array}{lrl} \min & \displaystyle\sum_{i=1}^{n} x_i & \\ \text{s.t.} & x_i + x_j \geq 1 & \text{for all clauses } (x_i \vee x_j) \in F \\ & (1 - x_i) + x_j \geq 1 & \text{for all clauses } (\overline{x}_i \vee x_j) \in F \\ & (1 - x_i) + (1 - x_j) \geq 1 & \text{for all clauses } (\overline{x}_i \vee \overline{x}_j) \in F \\ & x_i \in \{0, 1\} & 1 \leq i \leq n \end{array}$$

Minimum 2-Satisfiability: IP formulation

Suppose that F has n variables $x_1, \dots x_n$ and m clauses with 2 literals per clause

IP Min 2-SAT

$$\begin{array}{lrl} \min & \sum_{i=1}^{n} x_i & \\ \text{s.t.} & x_i + x_j \geq 1 & \text{for all clauses } (x_i \vee x_j) \in F \\ & (1 - x_i) + x_j \geq 1 & \text{for all clauses } (\overline{x}_i \vee x_j) \in F \\ & (1 - x_i) + (1 - x_j) \geq 1 & \text{for all clauses } (\overline{x}_i \vee \overline{x}_j) \in F \\ & x_i \in \{0, 1\} & 1 \leq i \leq n \end{array}$$

LP Min 2-SAT is obtaining replacing $x_i \in \{0, 1\}$ by $x_i \geq 0$.

Minimum 2-Satisfiability: LP relaxation

LP Min 2-SAT

$$
\begin{array}{lrl}
\min & \displaystyle\sum_{i=1}^{n} x_i & \\
\text{s.t.} & x_i + x_j \geq 1 & \text{for all clauses } (x_i \vee x_j) \in F \\
& (1 - x_i) + x_j \geq 1 & \text{for all clauses } (\overline{x}_i \vee x_j) \in F \\
& (1 - x_i) + (1 - x_j) \geq 1 & \text{for all clauses } (\overline{x}_i \vee \overline{x}_j) \in F \\
& x_i \geq 0 & 1 \leq i \leq n
\end{array}
$$

Minimum 2-Satisfiability: LP relaxation

LP Min 2-SAT

$$\begin{array}{lrl} \min & \displaystyle\sum_{i=1}^{n} x_i & \\ \text{s.t.} & x_i + x_j \geq 1 & \text{for all clauses } (x_i \vee x_j) \in F \\ & (1 - x_i) + x_j \geq 1 & \text{for all clauses } (\overline{x}_i \vee x_j) \in F \\ & (1 - x_i) + (1 - x_j) \geq 1 & \text{for all clauses } (\overline{x}_i \vee \overline{x}_j) \in F \\ & x_i \geq 0 & 1 \leq i \leq n \end{array}$$

Minimum 2-Satisfiability: LP relaxation

LP Min 2-SAT

$$\begin{array}{llr}
\min & & \displaystyle\sum_{i=1}^{n} x_i \\
\text{s.t.} & x_i + x_j \geq 1 & \text{for all clauses } (x_i \vee x_j) \in F \\
& (1 - x_i) + x_j \geq 1 & \text{for all clauses } (\overline{x}_i \vee x_j) \in F \\
& (1 - x_i) + (1 - x_j) \geq 1 & \text{for all clauses } (\overline{x}_i \vee \overline{x}_j) \in F \\
& x_i \geq 0 & 1 \leq i \leq n
\end{array}$$

- Let y be an optimal solution to LP Min 2-SAT.

Minimum 2-Satisfiability: LP relaxation

LP Min 2-SAT

$$
\begin{array}{llr}
\text{min} & \displaystyle\sum_{i=1}^{n} x_i & \\
\text{s.t.} & x_i + x_j \geq 1 & \text{for all clauses } (x_i \vee x_j) \in F \\
 & (1 - x_i) + x_j \geq 1 & \text{for all clauses } (\overline{x}_i \vee x_j) \in F \\
 & (1 - x_i) + (1 - x_j) \geq 1 & \text{for all clauses } (\overline{x}_i \vee \overline{x}_j) \in F \\
 & x_i \geq 0 & 1 \leq i \leq n
\end{array}
$$

- Let y be an optimal solution to LP Min 2-SAT.
- Can we use the same rounding scheme as for WVC?

Minimum 2-Satisfiability: LP relaxation

LP Min 2-SAT

$$\begin{array}{lrl} & \min & \sum_{i=1}^{n} x_i \\ \text{s.t.} & x_i + x_j \geq 1 & \text{for all clauses } (x_i \vee x_j) \in F \\ & (1 - x_i) + x_j \geq 1 & \text{for all clauses } (\overline{x}_i \vee x_j) \in F \\ & (1 - x_i) + (1 - x_j) \geq 1 & \text{for all clauses } (\overline{x}_i \vee \overline{x}_j) \in F \\ & x_i \geq 0 & 1 \leq i \leq n \end{array}$$

- Let y be an optimal solution to LP Min 2-SAT.
- Can we use the same rounding scheme as for WVC?
- Setting $x_i = 1$ if $y_i > 1/2$ and $x_i = 0$ if $y_i < 1/2$ is safe, all clauses with at least one literal with value $> 1/2$ will be satisfied.

Minimum 2-Satisfiability: LP relaxation

LP Min 2-SAT

$$\begin{array}{llll} & \min & \displaystyle\sum_{i=1}^{n} x_i & \\ \text{s.t.} & & x_i + x_j \geq 1 & \text{for all clauses } (x_i \vee x_j) \in F \\ & & (1 - x_i) + x_j \geq 1 & \text{for all clauses } (\overline{x}_i \vee x_j) \in F \\ & & (1 - x_i) + (1 - x_j) \geq 1 & \text{for all clauses } (\overline{x}_i \vee \overline{x}_j) \in F \\ & & x_i \geq 0 & 1 \leq i \leq n \end{array}$$

- Let y be an optimal solution to LP Min 2-SAT.
- Can we use the same rounding scheme as for WVC?
- Setting $x_i = 1$ if $y_i > 1/2$ and $x_i = 0$ if $y_i < 1/2$ is safe, all clauses with at least one literal with value $> 1/2$ will be satisfied.
- When $y_i = 1/2$?

Minimum 2-Satisfiability: LP relaxation

- Let y be an optimal solution to IP Min 2-SAT.
- What to do when $y_i = 1/2$? 1? 0?

Minimum 2-Satisfiability: LP relaxation

- Let y be an optimal solution to IP Min 2-SAT.
- What to do when $y_i = 1/2$? 1? 0?
- If F contains the clauses $(x_i \vee x_j)$ and $(\overline{x}_i \vee \overline{x}_j)$ and $y_i = y_j = 1/2$, neither $x_i = x_j = 1$ nor $x_i = x_j = 0$ satisfy the formula.

Minimum 2-Satisfiability: LP relaxation

- Let y be an optimal solution to IP Min 2-SAT.
- What to do when $y_i = 1/2$? 1? 0?
- If F contains the clauses $(x_i \vee x_j)$ and $(\overline{x}_i \vee \overline{x}_j)$ and $y_i = y_j = 1/2$, neither $x_i = x_j = 1$ nor $x_i = x_j = 0$ satisfy the formula.
- F_1 = clauses whose two variables have y value $= 1/2$.

Minimum 2-Satisfiability: LP relaxation

- Let y be an optimal solution to IP Min 2-SAT.
- What to do when $y_i = 1/2$? 1? 0?
- If F contains the clauses $(x_i \vee x_j)$ and $(\overline{x}_i \vee \overline{x}_j)$ and $y_i = y_j = 1/2$, neither $x_i = x_j = 1$ nor $x_i = x_j = 0$ satisfy the formula.
- $F_1 =$ clauses whose two variables have y value $= 1/2$.
- Rounding those values to 1 or 0 would keep the approximation ratio to 2, provided the constructed solution x to MIN 2-SAT is still a satisfying assignment.

Minimum 2-Satisfiability: LP relaxation

- Let y be an optimal solution to IP Min 2-SAT.
- What to do when $y_i = 1/2$? 1? 0?
- If F contains the clauses $(x_i \vee x_j)$ and $(\overline{x}_i \vee \overline{x}_j)$ and $y_i = y_j = 1/2$, neither $x_i = x_j = 1$ nor $x_i = x_j = 0$ satisfy the formula.
- F_1 = clauses whose two variables have y value $= 1/2$.
- Rounding those values to 1 or 0 would keep the approximation ratio to 2, provided the constructed solution x to MIN 2-SAT is still a satisfying assignment.
- Any satisfying assignment for the clauses in F_1 and get a 2-approximation ☺

Minimum 2-Satisfiability: Relax+Round approximation

```
function RELAX+ROUND MIN 2-SAT($F$)
    if $F$ is not satisfiable then return false
    Construct the LP Min 2-SAT, $I$
    $y = LP.solve(I)$
    for $i = 1, \dots, n$ do
        if $y'_i < 1/2$ then $x_i = 0$
        if $y'_i > 1/2$ then $x_i = 1$
    $F_1$ = clauses with both $y$ values $= 1/2$.
    Let $J = \{j \mid x_j \in F_1\}$
    for i=1,. . . , n do
        if $y_i = 1/2$ and $i \notin J$ then $x_i = 1$
    Complete $x$ with a satisfying assignment for $F_1$
    return $(x)$
```

Minimum 2-Satisfiability: Relax+Round approximation

Theorem

RELAX+ROUND MIN 2-SAT *is a 2-approximation for* MIN 2-SAT.

Max Satisfiability

MAX SAT

Given a Boolean formula in CNF and weights for each clause, find a Boolean assignment to maximize the weight of the satisfied clauses.

Max Satisfiability

MAX SAT

Given a Boolean formula in CNF and weights for each clause, find a Boolean assignment to maximize the weight of the satisfied clauses.

Suppose that F has n variables $x_1, \dots x_n$ and m clauses $C_1, \dots, C_m$.

Max Satisfiability

MAX SAT

Given a Boolean formula in CNF and weights for each clause, find a Boolean assignment to maximize the weight of the satisfied clauses.

Suppose that F has n variables $x_1, \dots x_n$ and m clauses $C_1, \dots, C_m$.

IP Max SAT

$$\begin{aligned} \max \quad & \sum_{j=1}^{m} w_j z_j \\ \text{s.t.} \quad & \sum_{x_i \in C_j} y_i + \sum_{\overline{x}_i \in C_j} (1 - y_i) \geq z_j \quad \jmath = 1, \dots, m \\ & y_i \in \{0,1\} \qquad 1 \leq i \leq n \\ & z_j \in \{0,1\} \qquad 1 \leq j \leq m \end{aligned}$$

Max Satisfiability

MAX SAT

Given a Boolean formula in CNF and weights for each clause, find a Boolean assignment to maximize the weight of the satisfied clauses.

Suppose that F has n variables $x_1, \dots x_n$ and m clauses $C_1, \dots, C_m$.

IP Max SAT

$$\begin{aligned} \max \quad & \sum_{j=1}^{m} w_j z_j \\ \text{s.t.} \quad & \sum_{x_i \in C_j} y_i + \sum_{\overline{x}_i \in C_j} (1 - y_i) \geq z_j \quad \jmath = 1, \dots, m \\ & y_i \in \{0,1\} \qquad 1 \leq i \leq n \\ & z_j \in \{0,1\} \qquad 1 \leq j \leq m \end{aligned}$$

LP Max SAT is obtaining replacing $a \in \{0,1\}$ by $0 \leq a \leq 1$.

Max Satisfiability: Relax+RRound

Max Satisfiability: Relax+RRound

```
function RELAX+RROUND($F$)
    Construct the LP Max SAT, $I$
    $(y, z) = LP.solve(I)$
    for i=1,. . . , n do
        Set $x_i = 1$ with probability $y_i$
    return $(x)$
```

Max Satisfiability: Relax+RRound

```
function RELAX+RROUND($F$)
    Construct the LP Max SAT, $I$
    $(y, z) = LP.solve(I)$
    for i=1,. . . , n do
        Set $x_i = 1$ with probability $y_i$
    return $(x)$
```

- The optimal LP solution is used as an indicator of the probability that the variable has to been set to 1.

Max Satisfiability: Relax+RRound

```
function RELAX+RROUND($F$)
    Construct the LP Max SAT, $I$
    $(y, z) = LP.solve(I)$
    for i=1,. . . , n do
        Set $x_i = 1$ with probability $y_i$
    return $(x)$
```

- The optimal LP solution is used as an indicator of the probability that the variable has to been set to 1.
- The performance of a randomized algorithm is the expected number of satisfiable clause.

Max Satisfiability: Relax+RRound

```
function Relax+RRound($F$)
    Construct the LP Max SAT, $I$
    $(y, z) = LP.solve(I)$
    for i=1,. . . , n do
        Set $x_i = 1$ with probability $y_i$
    return $(x)$
```

- The optimal LP solution is used as an indicator of the probability that the variable has to been set to 1.
- The performance of a randomized algorithm is the expected number of satisfiable clause.
- This expectation has to be compared with opt.

Max Satisfiability: Relax+RRound

Max Satisfiability: Relax+RRound

- Let (y^*, z^*) be an optimal solution of LP Max SAT
- Let Z_j be the indicator random variable for the event that clause C_j is satisfied.
- Assume that C_j has k-literals and that ℓ of them are negated variables.

Max Satisfiability: Relax+RRound

- Let (y^*, z^*) be an optimal solution of LP Max SAT
- Let Z_j be the indicator random variable for the event that clause C_j is satisfied.
- Assume that C_j has k-literals and that ℓ of them are negated variables.

Lemma

For any $1 \leq j \leq m$, $E[Z_j] \geq z_j^*(1 - 1/e)$.

Max Satisfiability: Relax+RRound

- Let (y^*, z^*) be an optimal solution of LP Max SAT
- Let Z_j be the indicator random variable for the event that clause C_j is satisfied.
- Assume that C_j has k-literals and that ℓ of them are negated variables.

Lemma

For any $1 \leq j \leq m$, $E[Z_j] \geq z_j^*(1 - 1/e)$.

Recall $(a_1 \dots a_k)^{1/k} \leq (a_1 + \cdots + a_k)/k$

Max Satisfiability: Relax+RRound

- Let (y^*, z^*) be an optimal solution of LP Max SAT
- Let Z_j be the indicator random variable for the event that clause C_j is satisfied.
- Assume that C_j has k-literals and that ℓ of them are negated variables.

Lemma

For any $1 \leq j \leq m$, $E[Z_j] \geq z_j^*(1 - 1/e)$.

Recall $(a_1 \ldots a_k)^{1/k} \leq (a_1 + \cdots + a_k)/k$ or equivalently
$(a_1 \ldots a_k) \leq ((a_1 + \cdots + a_k)/k)^k$

Max Satisfiability: Relax+RRound

Proof.

☺

Max Satisfiability: Relax+RRound

Proof.

Z_j is an indicator random variable, and so
$E[Z_j] = Pr[Z_j = 1] = 1 - Pr[Z_j = 0]$

☺

Max Satisfiability: Relax+RRound

Proof.

Z_j is an indicator random variable, and so
$E[Z_j] = Pr[Z_j = 1] = 1 - Pr[Z_j = 0]$

$$Pr[Z_j = 0] = \prod_{x_i \in C_j} (1 - y_i^*) \cdot \prod_{\overline{x}_i \in C_j} y_i^* \leq \left(\frac{(k-\ell) - \sum_{x_i \in C_j} y_i^* + \sum_{\overline{x}_i \in C_j} y_i^*}{k} \right)^k$$

$$\leq \left(\frac{(k - \sum_{x_i \in C_j} y_i^* - \sum_{\overline{x}_i \in C_j} (1 - y_i^*))}{k} \right)^k \leq \left(\frac{(k - z_j^*)}{k} \right)^k \leq \left(1 - \frac{z_j^*}{k} \right)^k$$

$$E[Z_j] \geq 1 - \left(1 - \frac{z_j^*}{k} \right)^k \geq z_j^* \left(1 - \frac{1}{k} \right)^k \geq z_j^* (1 - 1/e)$$

Max Satisfiability: Relax+RRound

Proof.

Z_j is an indicator random variable, and so
$E[Z_j] = Pr[Z_j = 1] = 1 - Pr[Z_j = 0]$

$$Pr[Z_j = 0] = \prod_{x_i \in C_j} (1 - y_i^*) \cdot \prod_{\overline{x}_i \in C_j} y_i^* \le \left(\frac{(k - \ell) - \sum_{x_i \in C_j} y_i^* + \sum_{\overline{x}_i \in C_j} y_i^*}{k} \right)^k$$

$$\le \left(\frac{(k - \sum_{x_i \in C_j} y_i^* - \sum_{\overline{x}_i \in C_j} (1 - y_i^*)}{k} \right)^k \le \left(\frac{(k - z_j^*)}{k} \right)^k \le \left(1 - \frac{z_j^*}{k} \right)^k$$

$$E[Z_j] \ge 1 - \left(1 - \frac{z_j^*}{k} \right)^k \ge z_j^* \left(1 - \frac{1}{k} \right)^k \ge z_j^* (1 - 1/e)$$

☺

Max Satisfiability: Relax+RRound approximation

Theorem

RELAX+RROUND *is a* $e/(e-1)$*-approximation for* MAX SAT.

Max Satisfiability: Relax+RRound approximation

Theorem

Relax+RRound *is a* $e/(e-1)$*-approximation for* Max SAT.

Proof.

Max Satisfiability: Relax+RRound approximation

Theorem

Relax+RRound *is a* $e/(e-1)$*-approximation for* Max SAT.

Proof.

- Let (y^*, z^*) be an optimal solution of LP Max SAT
- Let Z_j be the indicator r.v.a for clause C_j is satisfied.
- Let W be the r.v. weight of satisfied clauses: $W = \sum_{j=1}^{m} w_j Z_j$.

Max Satisfiability: Relax+RRound approximation

Theorem

RELAX+RROUND *is a* $e/(e-1)$*-approximation for* MAX SAT.

Proof.

- Let (y^*, z^*) be an optimal solution of LP Max SAT
- Let Z_j be the indicator r.v.a for clause C_j is satisfied.
- Let W be the r.v. weight of satisfied clauses: $W = \sum_{j=1}^{m} w_j Z_j$.
- $E[W] = \sum_{j=1}^{m} w_j E[Z_j] \geq (1 - 1/e) \sum_{j=1}^{m} w_j z_j^* \geq (1 - 1/e)\text{opt}$

☺

Max Satisfiability:RandAssign

Max Satisfiability:RandAssign

```
function RandAssign($F$)
    for i=1,..., n do
        Set $x_i = 1$ with probability $1/2$
    return ($x$)
```

Max Satisfiability:RandAssign

```
function RandAssign($F$)
    for i=1,..., n do
        Set $x_i = 1$ with probability 1/2
    return ($x$)
```

Theorem

RANDASSIGN *is a 2-approximation for* MAX SAT*.*

Max Satisfiability:RandAssign

```
function RandAssign(F)
    for i=1,. . . , n do
        Set $x_i = 1$ with probability 1/2
    return (x)
```

Theorem

RandAssign *is a 2-approximation for* Max SAT.

Proof.

$E[W] = \sum_{j=1}^{m} w_j E[Z_j] = \sum_{j=1}^{m} w_j \left(1 - \left(\frac{1}{2}\right)^{k_j}\right) \geq \frac{1}{2} \sum_{j=1}^{m} w_j \geq \frac{1}{2}\text{opt}.$ ☺

Max Satisfiability:RandAssign

```
function RandAssign($F$)
    for i=1,. . . , n do
        Set $x_i = 1$ with probability 1/2
    return ($x$)
```

Theorem

RandAssign *is a 2-approximation for* Max SAT.

Proof.

$E[W] = \sum_{j=1}^{m} w_j E[Z_j] = \sum_{j=1}^{m} w_j \left(1 - (\frac{1}{2})^{k_j}\right) \geq \frac{1}{2} \sum_{j=1}^{m} w_j \geq \frac{1}{2}\text{opt}.$ ☺

We move from $r = 2$ (RandAssign) to $r = 1.581977$ (Relax+RRound).

Max Satisfiability:Best2

```
function Best2($F$)
    $x_1, W_1 = $ RandAssign($F$)
    $x_2, W_2 = $ Relax+RRound($F$)
    if $W_1 \geq W_2$ then
        return ($x_1$)
    else
        return ($x_2$)
```

Max Satisfiability:Best2

```
function Best2($F$)
    $x_1, W_1 =$ RandAssign($F$)
    $x_2, W_2 =$ Relax+RRound($F$)
    if $W_1 \geq W_2$ then
        return ($x_1$)
    else
        return ($x_2$)
```

Theorem

Best2 *is a* 4/3 *(1.33333)-approximation for* Max SAT*.*

Max Satisfiability:Best2

Max Satisfiability:Best2

Proof.

Max Satisfiability:Best2

Proof.

- $E[W] = E[\max\{W_1, W_2\}] \geq E[(W_1 + W_2)/2\}]$.

Max Satisfiability:Best2

Proof.

- $E[W] = E[\max\{W_1, W_2\}] \geq E[(W_1 + W_2)/2\}]$.

$$E[W] \geq \sum_{j=1}^{m} w_j \left[\frac{1}{2}\left(1 - \left(\frac{1}{2}\right)^{k_j}\right) + \frac{1}{2} z_j^* \left(1 - \left(\frac{1}{k_j}\right)^{k_j}\right)\right]$$

$$\geq \sum_{j=1}^{m} w_j \frac{3}{4} z_j^* \geq \frac{3}{4} \sum_{j=1}^{m} w_j z_j^* \geq \frac{3}{4}\text{opt}.$$

Max Satisfiability:Best2

Proof.

- Is $\left[\frac{1}{2}\left(1-\left(\frac{1}{2}\right)^{k_j}\right)+\frac{1}{2}z_j^*\left(1-\left(\frac{1}{k_j}\right)^{k_j}\right)\right] \geq \frac{3}{4}z_j^*$?

Max Satisfiability:Best2

Proof.

- Is $\left[\frac{1}{2}\left(1-\left(\frac{1}{2}\right)^{k_j}\right)+\frac{1}{2}z_j^*\left(1-\left(\frac{1}{k_j}\right)^{k_j}\right)\right] \geq \frac{3}{4}z_j^*$?
- $k_j = 1$: $\frac{1}{2}\frac{1}{2}+\frac{1}{2}z_j^* \geq \frac{3}{4}z_j^*$.

Max Satisfiability:Best2

Proof.

- Is $\left[\frac{1}{2}\left(1-\left(\frac{1}{2}\right)^{k_j}\right)+\frac{1}{2}z_j^*\left(1-\left(\frac{1}{k_j}\right)^{k_j}\right)\right]\geq\frac{3}{4}z_j^*$?
- $k_j=1$: $\frac{1}{2}\frac{1}{2}+\frac{1}{2}z_j^*\geq\frac{3}{4}z_j^*$.
- $k_j=2$: $\frac{1}{2}\frac{3}{4}+\frac{1}{2}\frac{3}{4}z_j^*\geq\frac{3}{4}z_j^*$.

Max Satisfiability:Best2

Proof.

- Is $\left[\frac{1}{2}\left(1-\left(\frac{1}{2}\right)^{k_j}\right)+\frac{1}{2}z_j^*\left(1-\left(\frac{1}{k_j}\right)^{k_j}\right)\right]\geq\frac{3}{4}z_j^*$?
- $k_j=1$: $\frac{1}{2}\frac{1}{2}+\frac{1}{2}z_j^*\geq\frac{3}{4}z_j^*$.
- $k_j=2$: $\frac{1}{2}\frac{3}{4}+\frac{1}{2}\frac{3}{4}z_j^*\geq\frac{3}{4}z_j^*$.
- $k_j\geq 3$: the minimum possible of each term is

$$\frac{1}{2}\frac{7}{8}+\frac{1}{2}\left(1-\frac{1}{e}\right)z_j^*\geq\frac{3}{4}z_j^*$$

Max Satisfiability:Best2

Proof.

- Is $\left[\frac{1}{2}\left(1-\left(\frac{1}{2}\right)^{k_j}\right)+\frac{1}{2}z_j^*\left(1-\left(\frac{1}{k_j}\right)^{k_j}\right)\right]\geq\frac{3}{4}z_j^*$?
- $k_j=1$: $\frac{1}{2}\frac{1}{2}+\frac{1}{2}z_j^*\geq\frac{3}{4}z_j^*$.
- $k_j=2$: $\frac{1}{2}\frac{3}{4}+\frac{1}{2}\frac{3}{4}z_j^*\geq\frac{3}{4}z_j^*$.
- $k_j\geq 3$: the minimum possible of each term is

$$\frac{1}{2}\frac{7}{8}+\frac{1}{2}\left(1-\frac{1}{e}\right)z_j^*\geq\frac{3}{4}z_j^*$$

☺

1. LP and IP

2. Relax and round

3. LP Duality