

Simultaneously Satisfying Linear Equations Over \mathbb{F}_2 : Parameterized Above Average

Anders Yeo

anders@cs.rhul.ac.uk

Department of Computer Science
Royal Holloway, University of London

Co-authors: Robert Crowston, Mike Fellows, Gregory Gutin,
Mark Jones, Frances Rosamond and Stéphan Thomassé

Outline

- 1 Parameterizing above tight bounds: Example Max-Sat
- 2 Max-Lin-AA
- 3 FPT Results
- 4 Related Results

Parameterized Above Tight Bounds: Max-Sat

MAX-SAT ('Standard' parameterization)

Instance: A CNF formula F with n variables, m clauses.

Parameter: k .

Question: Can we satisfy $\geq k$ clauses?

- **Known bound:** can satisfy at least $m/2$ clauses. Why?

This is a lower bound on the average number of satisfied clauses in a random assignment.

- So it is trivially FPT. Why?

If $k \leq m/2$ return YES; otherwise $m < 2k$ which is a kernel.

- So what does this mean?

Such a kernel is not very useful: There is no reductions and k ($> m/2$) is large for all non trivial cases!

Parameterized Above Tight Bounds: Max-Sat

MAX-SAT ('Standard' parameterization)

Instance: A CNF formula F with n variables, m clauses.

Parameter: k .

Question: Can we satisfy $\geq k$ clauses?

- **Known bound:** can satisfy at least $m/2$ clauses. Why?

This is a lower bound on the average number of satisfied clauses in a random assignment.

- So it is trivially FPT. Why?

If $k \leq m/2$ return YES; otherwise $m < 2k$ which is a kernel.

- So what does this mean?

Such a kernel is not very useful: There is no reductions and k ($> m/2$) is large for all non trivial cases!

Parameterized Above Tight Bounds: Max-Sat

MAX-SAT ('Standard' parameterization)

Instance: A CNF formula F with n variables, m clauses.

Parameter: k .

Question: Can we satisfy $\geq k$ clauses?

- **Known bound:** can satisfy at least $m/2$ clauses. Why?

This is a lower bound on the average number of satisfied clauses in a random assignment.

- So it is trivially FPT. Why?

If $k \leq m/2$ return YES; otherwise $m < 2k$ which is a kernel.

- So what does this mean?

Such a kernel is not very useful: There is no reductions and k ($> m/2$) is large for all non trivial cases!

Parameterized Above Tight Bounds: Max-Sat

MAX-SAT ('Standard' parameterization)

Instance: A CNF formula F with n variables, m clauses.

Parameter: k .

Question: Can we satisfy $\geq k$ clauses?

- **Known bound:** can satisfy at least $m/2$ clauses. Why?

This is a lower bound on the average number of satisfied clauses in a random assignment.

- So it is trivially FPT. Why?

If $k \leq m/2$ return YES; otherwise $m < 2k$ which is a kernel.

- So what does this mean?

Such a kernel is not very useful: There is no reductions and k ($> m/2$) is large for all non trivial cases!

Parameterized Above Tight Bounds: Max-Sat

MAX-SAT ('Standard' parameterization)

Instance: A CNF formula F with n variables, m clauses.

Parameter: k .

Question: Can we satisfy $\geq k$ clauses?

- **Known bound:** can satisfy at least $m/2$ clauses. Why?

This is a lower bound on the average number of satisfied clauses in a random assignment.

- So it is trivially FPT. Why?

If $k \leq m/2$ return YES; otherwise $m < 2k$ which is a kernel.

- So what does this mean?

Such a kernel is not very useful: There is no reductions and k ($> m/2$) is large for all non trivial cases!

Parameterized Above Tight Bounds: Max-Sat

MAX-SAT ('Standard' parameterization)

Instance: A CNF formula F with n variables, m clauses.

Parameter: k .

Question: Can we satisfy $\geq k$ clauses?

- **Known bound:** can satisfy at least $m/2$ clauses. Why?

This is a lower bound on the average number of satisfied clauses in a random assignment.

- So it is trivially FPT. Why?

If $k \leq m/2$ return YES; otherwise $m < 2k$ which is a kernel.

- So what does this mean?

Such a kernel is not very useful: There is no reductions and k ($> m/2$) is large for all non trivial cases!

Parameterized Above Tight Bounds: Max-Sat

MAX-SAT ('Standard' parameterization)

Instance: A CNF formula F with n variables, m clauses.

Parameter: k .

Question: Can we satisfy $\geq k$ clauses?

- **Known bound:** can satisfy at least $m/2$ clauses. Why?

This is a lower bound on the average number of satisfied clauses in a random assignment.

- So it is trivially FPT. Why?

If $k \leq m/2$ return YES; otherwise $m < 2k$ which is a kernel.

- So what does this mean?

Such a kernel is not very useful: There is no reductions and k ($> m/2$) is large for all non trivial cases!

Parameterized Above Tight Bounds: Max-Sat

- A better parameterization:

MAX-SAT parameterized above $m/2$

Instance: A CNF formula F with n variables, m clauses.

Parameter: k .

Question: Can we satisfy $\geq m/2 + k$ clauses?

- In this case k is smaller!
- And the problem becomes more interesting!

Parameterized Above Tight Bounds: Max-Sat

- A better parameterization:

MAX-SAT parameterized above $m/2$

Instance: A CNF formula F with n variables, m clauses.

Parameter: k .

Question: Can we satisfy $\geq m/2 + k$ clauses?

- In this case k is smaller!
- And the problem becomes more interesting!

Parameterized Above Tight Bounds: Max-Sat

- A better parameterization:

MAX-SAT parameterized above $m/2$

Instance: A CNF formula F with n variables, m clauses.

Parameter: k .

Question: Can we satisfy $\geq m/2 + k$ clauses?

- In this case k is smaller!
- And the problem becomes more interesting!

Parameterized Above Tight Bounds: Max-Sat

- The above problem was solved by Mahajan and Raman, who gave a linear kernel.
- It is still relatively easy due to the following:
 - Reduce an instance by removing any two clauses of the form (x) and (\bar{x}) .
 - Repeatedly doing this creates an instance of 2-satisfiable-SAT and does not change the problem.
 - However $\hat{\phi}m$ becomes a tight lower bound on the number of satisfied clauses, where $\hat{\phi} = (\sqrt{5} - 1)/2 \approx 0.618$.
 - Therefore there is a kernel.
Proof: If $k < (\hat{\phi} - \frac{1}{2})m$ answer YES.
Otherwise $m \leq k/(\hat{\phi} - \frac{1}{2})$.
 - This gives rise to a new problem.....

Parameterized Above Tight Bounds: Max-Sat

- The above problem was solved by Mahajan and Raman, who gave a linear kernel.
- It is still relatively easy due to the following:
 - Reduce an instance by removing any two clauses of the form (x) and (\bar{x}) .
 - Repeatedly doing this creates an instance of 2-satisfiable-SAT and does not change the problem.
 - However $\hat{\phi}m$ becomes a tight lower bound on the number of satisfied clauses, where $\hat{\phi} = (\sqrt{5} - 1)/2 \approx 0.618$.
 - Therefore there is a kernel.
Proof: If $k < (\hat{\phi} - \frac{1}{2})m$ answer YES.
Otherwise $m \leq k/(\hat{\phi} - \frac{1}{2})$.
 - This gives rise to a new problem.....

Parameterized Above Tight Bounds: Max-Sat

- The above problem was solved by Mahajan and Raman, who gave a linear kernel.
- It is still relatively easy due to the following:
 - Reduce an instance by removing any two clauses of the form (x) and (\bar{x}) .
 - Repeatedly doing this creates an instance of 2-satisfiable-SAT and does not change the problem.
 - However $\hat{\phi}m$ becomes a tight lower bound on the number of satisfied clauses, where $\hat{\phi} = (\sqrt{5} - 1)/2 \approx 0.618$.
 - Therefore there is a kernel.
Proof: If $k < (\hat{\phi} - \frac{1}{2})m$ answer YES.
Otherwise $m \leq k/(\hat{\phi} - \frac{1}{2})$.
 - This gives rise to a new problem.....

Parameterized Above Tight Bounds: Max-Sat

- The above problem was solved by Mahajan and Raman, who gave a linear kernel.
 - It is still relatively easy due to the following:
 - Reduce an instance by removing any two clauses of the form (x) and (\bar{x}) .
 - Repeatedly doing this creates an instance of 2-satisfiable-SAT and does not change the problem.
 - However $\hat{\phi}m$ becomes a tight lower bound on the number of satisfied clauses, where $\hat{\phi} = (\sqrt{5} - 1)/2 \approx 0.618$.
 - Therefore there is a kernel.
- Proof:** If $k < (\hat{\phi} - \frac{1}{2})m$ answer YES.
Otherwise $m \leq k/(\hat{\phi} - \frac{1}{2})$.
- This gives rise to a new problem.....

Parameterized Above Tight Bounds: Max-Sat

- The above problem was solved by Mahajan and Raman, who gave a linear kernel.
- It is still relatively easy due to the following:
 - Reduce an instance by removing any two clauses of the form (x) and (\bar{x}) .
 - Repeatedly doing this creates an instance of 2-satisfiable-SAT and does not change the problem.
 - However $\hat{\phi}m$ becomes a tight lower bound on the number of satisfied clauses, where $\hat{\phi} = (\sqrt{5} - 1)/2 \approx 0.618$.
 - Therefore there is a kernel.
Proof: If $k < (\hat{\phi} - \frac{1}{2})m$ answer YES.
Otherwise $m \leq k/(\hat{\phi} - \frac{1}{2})$.
- This gives rise to a new problem.....

Parameterized Above Tight Bounds: Max-Sat

- The above problem was solved by Mahajan and Raman, who gave a linear kernel.
- It is still relatively easy due to the following:
 - Reduce an instance by removing any two clauses of the form (x) and (\bar{x}) .
 - Repeatedly doing this creates an instance of 2-satisfiable-SAT and does not change the problem.
 - However $\hat{\phi}m$ becomes a tight lower bound on the number of satisfied clauses, where $\hat{\phi} = (\sqrt{5} - 1)/2 \approx 0.618$.
 - Therefore there is a kernel.
Proof: If $k < (\hat{\phi} - \frac{1}{2})m$ answer YES.
Otherwise $m \leq k/(\hat{\phi} - \frac{1}{2})$.
- This gives rise to a new problem.....

Parameterized Above Tight Bounds: Max-Sat

- The above problem was solved by Mahajan and Raman, who gave a linear kernel.
- It is still relatively easy due to the following:
 - Reduce an instance by removing any two clauses of the form (x) and (\bar{x}) .
 - Repeatedly doing this creates an instance of 2-satisfiable-SAT and does not change the problem.
 - However $\hat{\phi}m$ becomes a tight lower bound on the number of satisfied clauses, where $\hat{\phi} = (\sqrt{5} - 1)/2 \approx 0.618$.
 - Therefore there is a kernel.
Proof: If $k < (\hat{\phi} - \frac{1}{2})m$ answer YES.
Otherwise $m \leq k/(\hat{\phi} - \frac{1}{2})$.
 - This gives rise to a new problem.....

Parameterized Above Tight Bounds: Max-Sat

MAX-2-SATISFIABLE-SAT parameterized above $\hat{\phi}m$

Instance: A CNF formula F with n variables, m clauses and any two clauses can be simultaneously satisfied.

Parameter: k .

Question: Can we satisfy $\geq \hat{\phi}m + k$ clauses?

- The above problem was shown to have a kernel with at most $O(k)$ variables, by Crowston, Gutin, Jones and AY.
- This approach does not seem to be easily extendable.

Parameterized Above Tight Bounds: Max-Sat

MAX-2-SATISFIABLE-SAT parameterized above $\hat{\phi}m$

Instance: A CNF formula F with n variables, m clauses and any two clauses can be simultaneously satisfied.

Parameter: k .

Question: Can we satisfy $\geq \hat{\phi}m + k$ clauses?

- The above problem was shown to have a kernel with at most $O(k)$ variables, by Crowston, Gutin, Jones and AY.
- This approach does not seem to be easily extendable.

Parameterized Above Tight Bounds: Max-Sat

MAX-2-SATISFIABLE-SAT parameterized above $\hat{\phi}m$

Instance: A CNF formula F with n variables, m clauses and any two clauses can be simultaneously satisfied.

Parameter: k .

Question: Can we satisfy $\geq \hat{\phi}m + k$ clauses?

- The above problem was shown to have a kernel with at most $O(k)$ variables, by Crowston, Gutin, Jones and AY.
- This approach does not seem to be easily extendable.

Parameterized Above Tight Bounds: In general

- In problems **parameterized above (below) tight bounds**, we take a maximization (minimization) problem with a tight lower (upper) bound, and ask if we can get k above (below) this bound.
- Ensures the parameter is small in interesting cases.
- First introduced in a paper by Mahajan and Raman published in 1999.
- We say “above average” when the tight lower bound is the expectation of a random assignment.

Parameterized Above Tight Bounds: In general

- In problems **parameterized above (below) tight bounds**, we take a maximization (minimization) problem with a tight lower (upper) bound, and ask if we can get k above (below) this bound.
- Ensures the parameter is small in interesting cases.
- First introduced in a paper by Mahajan and Raman published in 1999.
- We say “above average” when the tight lower bound is the expectation of a random assignment.

Parameterized Above Tight Bounds: In general

- In problems **parameterized above (below) tight bounds**, we take a maximization (minimization) problem with a tight lower (upper) bound, and ask if we can get k above (below) this bound.
- Ensures the parameter is small in interesting cases.
- First introduced in a paper by Mahajan and Raman published in 1999.
- We say “above average” when the tight lower bound is the expectation of a random assignment.

Parameterized Above Tight Bounds: In general

- In problems **parameterized above (below) tight bounds**, we take a maximization (minimization) problem with a tight lower (upper) bound, and ask if we can get k above (below) this bound.
- Ensures the parameter is small in interesting cases.
- First introduced in a paper by Mahajan and Raman published in 1999.
- We say “above average” when the tight lower bound is the expectation of a random assignment.

Outline

- 1 Parameterizing above tight bounds: Example Max-Sat
- 2 Max-Lin-AA**
- 3 FPT Results
- 4 Related Results

Max-Lin Above Average

- MAX-LIN problem: given a system \mathcal{I} of m linear equations in n variables over \mathbb{F}_2 .
- \mathbb{F}_2 is the Galois field with 2 elements ($1 + 1 = 0$).
- Each equation is assigned a positive integer weight.
- We wish to find an assignment of values to the variables in order to maximize the total weight of satisfied equations.

$$z_1 = 1$$

$$z_1 + z_2 = 0$$

$$z_2 + z_3 = 1$$

$$z_1 + z_2 + z_3 = 1$$

- **Known bound:** can satisfy at least $W/2$, where W = total weight of equations.

Max-Lin Above Average

- MAX-LIN problem: given a system \mathcal{I} of m linear equations in n variables over \mathbb{F}_2 .
- \mathbb{F}_2 is the Galois field with 2 elements ($1 + 1 = 0$).
- Each equation is assigned a positive integer weight.
- We wish to find an assignment of values to the variables in order to maximize the total weight of satisfied equations.

$$z_1 = 1$$

$$z_1 + z_2 = 0$$

$$z_2 + z_3 = 1$$

$$z_1 + z_2 + z_3 = 1$$

- **Known bound:** can satisfy at least $W/2$, where W = total weight of equations.

Max-Lin Above Average

- MAX-LIN problem: given a system \mathcal{I} of m linear equations in n variables over \mathbb{F}_2 .
- \mathbb{F}_2 is the Galois field with 2 elements ($1 + 1 = 0$).
- Each equation is assigned a positive integer weight.
- We wish to find an assignment of values to the variables in order to maximize the total weight of satisfied equations.

$$z_1 = 1$$

$$z_1 + z_2 = 0$$

$$z_2 + z_3 = 1$$

$$z_1 + z_2 + z_3 = 1$$

- **Known bound:** can satisfy at least $W/2$, where W = total weight of equations.

Max-Lin Above Average

- MAX-LIN problem: given a system \mathcal{I} of m linear equations in n variables over \mathbb{F}_2 .
- \mathbb{F}_2 is the Galois field with 2 elements ($1 + 1 = 0$).
- Each equation is assigned a positive integer weight.
- We wish to find an assignment of values to the variables in order to maximize the total weight of satisfied equations.

$$z_1 = 1$$

$$z_1 + z_2 = 0$$

$$z_2 + z_3 = 1$$

$$z_1 + z_2 + z_3 = 1$$

- **Known bound:** can satisfy at least $W/2$, where W = total weight of equations.

Max-Lin Above Average

- MAX-LIN problem: given a system \mathcal{I} of m linear equations in n variables over \mathbb{F}_2 .
- \mathbb{F}_2 is the Galois field with 2 elements ($1 + 1 = 0$).
- Each equation is assigned a positive integer weight.
- We wish to find an assignment of values to the variables in order to maximize the total weight of satisfied equations.

$$z_1 = 1$$

$$z_1 + z_2 = 0$$

$$z_2 + z_3 = 1$$

$$z_1 + z_2 + z_3 = 1$$

- **Known bound:** can satisfy at least $W/2$, where W = total weight of equations.

Tightness of $W/2$ bound

- $W/2$ is a **tight** lower bound on $\max(\mathcal{I})$.
- e.g.

$$z_1 = 0$$

$$z_1 = 1$$

$$z_2 + z_3 = 0$$

$$z_2 + z_3 = 1$$

...

Theorem (Håstad, 2001)

For any $\epsilon > 0$, it is impossible to decide in polynomial time between instances of MAX-LIN in which $\max(\mathcal{I}) \leq (1/2 + \epsilon)m$, and instances in which $\max(\mathcal{I}) \geq (1 - \epsilon)m$, unless $P = NP$.

Tightness of $W/2$ bound

- $W/2$ is a **tight** lower bound on $\max(\mathcal{I})$.
- e.g.

$$z_1 = 0$$

$$z_1 = 1$$

$$z_2 + z_3 = 0$$

$$z_2 + z_3 = 1$$

...

Theorem (Håstad, 2001)

For any $\epsilon > 0$, it is impossible to decide in polynomial time between instances of MAX-LIN in which $\max(\mathcal{I}) \leq (1/2 + \epsilon)m$, and instances in which $\max(\mathcal{I}) \geq (1 - \epsilon)m$, unless $P = NP$.

Max-Lin Above Average

- Let $\max(\mathcal{I})$ denote the maximum possible weight of satisfied equations in \mathcal{I} .

MAX-LIN ABOVE AVERAGE (MAX-LIN-AA)

Instance: A system \mathcal{I} of m linear equations in n variables over \mathbb{F}_2 , with total weight W .

Parameter: k .

Question: Is $\max(\mathcal{I}) \geq W/2 + k$?

- Mahajan, Raman & Sikdar (2006) asked if MAX-LIN-AA is FPT.

Max-Lin Above Average

- Let $\max(\mathcal{I})$ denote the maximum possible weight of satisfied equations in \mathcal{I} .

MAX-LIN ABOVE AVERAGE (MAX-LIN-AA)

Instance: A system \mathcal{I} of m linear equations in n variables over \mathbb{F}_2 , with total weight W .

Parameter: k .

Question: Is $\max(\mathcal{I}) \geq W/2 + k$?

- Mahajan, Raman & Sikdar (2006) asked if MAX-LIN-AA is FPT.

Motivation

MAX- r -LIN is equivalent to MAX-LIN except all equations have at most r variables.

MAX-LIN and MAX- r -LIN are important problems, for many reasons....

- Håstad said they were as basic as satisfiability.
- They are important tools for constraint satisfaction problems (such as MAXSAT or MAX- r -SAT).
- So MAX-LIN and MAX- r -LIN have attracted significant interest in algorithmics.
- A number of papers made progress on MAX- r -LIN-AA before MAX-LIN-AA was shown to be FPT.

Motivation

MAX- r -LIN is equivalent to MAX-LIN except all equations have at most r variables.

MAX-LIN and MAX- r -LIN are important problems, for many reasons....

- Håstad said they were as basic as satisfiability.
- They are important tools for constraint satisfaction problems (such as MAXSAT or MAX- r -SAT).
- So MAX-LIN and MAX- r -LIN have attracted significant interest in algorithmics.
- A number of papers made progress on MAX- r -LIN-AA before MAX-LIN-AA was shown to be FPT.

Motivation

MAX- r -LIN is equivalent to MAX-LIN except all equations have at most r variables.

MAX-LIN and MAX- r -LIN are important problems, for many reasons....

- Håstad said they were as basic as satisfiability.
- They are important tools for constraint satisfaction problems (such as MAXSAT or MAX- r -SAT).
- So MAX-LIN and MAX- r -LIN have attracted significant interest in algorithmics.
- A number of papers made progress on MAX- r -LIN-AA before MAX-LIN-AA was shown to be FPT.

Motivation

MAX- r -LIN is equivalent to MAX-LIN except all equations have at most r variables.

MAX-LIN and MAX- r -LIN are important problems, for many reasons....

- Håstad said they were as basic as satisfiability.
- They are important tools for constraint satisfaction problems (such as MAXSAT or MAX- r -SAT).
- So MAX-LIN and MAX- r -LIN have attracted significant interest in algorithmics.
- A number of papers made progress on MAX- r -LIN-AA before MAX-LIN-AA was shown to be FPT.

Motivation

MAX- r -LIN is equivalent to MAX-LIN except all equations have at most r variables.

MAX-LIN and MAX- r -LIN are important problems, for many reasons....

- Håstad said they were as basic as satisfiability.
- They are important tools for constraint satisfaction problems (such as MAXSAT or MAX- r -SAT).
- So MAX-LIN and MAX- r -LIN have attracted significant interest in algorithmics.
- A number of papers made progress on MAX- r -LIN-AA before MAX-LIN-AA was shown to be FPT.

Outline

- 1 Parameterizing above tight bounds: Example Max-Sat
- 2 Max-Lin-AA
- 3 FPT Results**
- 4 Related Results

Overview

- Notation
- Reduction Rules
- Main Results
- Proof of the Main Results

Notation

- For a given assignment, the **excess** = total weight of satisfied equations – total weight of falsified equations.
- MAX-LIN-AA is equivalent to asking if the max excess is at least $2k$.

Example:

$$z_1 = 1$$

$$z_2 = 1$$

$$z_1 + z_2 = 1$$

Notation

- For a given assignment, the **excess** = total weight of satisfied equations – total weight of falsified equations.
- MAX-LIN-AA is equivalent to asking if the max excess is at least $2k$.

Example:

$$z_1 = 1$$

$$z_2 = 1$$

$$z_1 + z_2 = 1$$

Reduction Rules

Reduction Rule (LHS rule)

Suppose we have two equations, $\sum_{i \in S} z_i = b_1$ (weight w_1) and $\sum_{i \in S} z_i = b_2$ (weight w_2), where $w_1 \geq w_2$.

If $b_1 = b_2$, replace with one equation $\sum_{i \in S} z_i = b_1$ (weight $w_1 + w_2$).

If $b_1 \neq b_2$, replace with one equation $\sum_{i \in S} z_i = b_1$ (weight $w_1 - w_2$).

$$\begin{array}{ll} z_1 + z_2 = 1 & (w = 1) \\ z_1 + z_2 = 1 & (w = 2) \end{array} \Rightarrow z_1 + z_2 = 1 \quad (w = 3)$$

$$\begin{array}{ll} z_2 + z_3 + z_4 = 0 & (w = 3) \\ z_2 + z_3 + z_4 = 1 & (w = 2) \end{array} \Rightarrow z_2 + z_3 + z_4 = 0 \quad (w = 1)$$

- Allows us to assume no two equations have the same left-hand side.

Reduction Rules

Reduction Rule (LHS rule)

Suppose we have two equations, $\sum_{i \in S} z_i = b_1$ (weight w_1) and $\sum_{i \in S} z_i = b_2$ (weight w_2), where $w_1 \geq w_2$.

If $b_1 = b_2$, replace with one equation $\sum_{i \in S} z_i = b_1$ (weight $w_1 + w_2$).

If $b_1 \neq b_2$, replace with one equation $\sum_{i \in S} z_i = b_1$ (weight $w_1 - w_2$).

$$\begin{array}{ll} z_1 + z_2 = 1 & (w = 1) \\ z_1 + z_2 = 1 & (w = 2) \end{array} \Rightarrow z_1 + z_2 = 1 \quad (w = 3)$$

$$\begin{array}{ll} z_2 + z_3 + z_4 = 0 & (w = 3) \\ z_2 + z_3 + z_4 = 1 & (w = 2) \end{array} \Rightarrow z_2 + z_3 + z_4 = 0 \quad (w = 1)$$

- Allows us to assume no two equations have the same left-hand side.

Reduction Rules

Reduction Rule (LHS rule)

Suppose we have two equations, $\sum_{i \in S} z_i = b_1$ (weight w_1) and $\sum_{i \in S} z_i = b_2$ (weight w_2), where $w_1 \geq w_2$.

If $b_1 = b_2$, replace with one equation $\sum_{i \in S} z_i = b_1$ (weight $w_1 + w_2$).

If $b_1 \neq b_2$, replace with one equation $\sum_{i \in S} z_i = b_1$ (weight $w_1 - w_2$).

$$\begin{array}{ll} z_1 + z_2 = 1 & (w = 1) \\ z_1 + z_2 = 1 & (w = 2) \end{array} \Rightarrow z_1 + z_2 = 1 \quad (w = 3)$$

$$\begin{array}{ll} z_2 + z_3 + z_4 = 0 & (w = 3) \\ z_2 + z_3 + z_4 = 1 & (w = 2) \end{array} \Rightarrow z_2 + z_3 + z_4 = 0 \quad (w = 1)$$

- Allows us to assume no two equations have the same left-hand side.

Reduction Rules

Reduction Rule (LHS rule)

Suppose we have two equations, $\sum_{i \in S} z_i = b_1$ (weight w_1) and $\sum_{i \in S} z_i = b_2$ (weight w_2), where $w_1 \geq w_2$.

If $b_1 = b_2$, replace with one equation $\sum_{i \in S} z_i = b_1$ (weight $w_1 + w_2$).

If $b_1 \neq b_2$, replace with one equation $\sum_{i \in S} z_i = b_1$ (weight $w_1 - w_2$).

$$\begin{array}{ll} z_1 + z_2 = 1 & (w = 1) \\ z_1 + z_2 = 1 & (w = 2) \end{array} \Rightarrow z_1 + z_2 = 1 \quad (w = 3)$$

$$\begin{array}{ll} z_2 + z_3 + z_4 = 0 & (w = 3) \\ z_2 + z_3 + z_4 = 1 & (w = 2) \end{array} \Rightarrow z_2 + z_3 + z_4 = 0 \quad (w = 1)$$

- Allows us to assume no two equations have the same left-hand side.

Reduction Rules

Reduction Rule (Rank rule)

Let A be the matrix over \mathbb{F}_2 corresponding to the set of equations in \mathcal{I} , such that $a_{ji} = 1$ if z_i appears in equation j , and 0 otherwise. Let $t = \text{rank} A$ and suppose columns a^{i_1}, \dots, a^{i_t} of A are linearly independent. Then delete all variables not in $\{z_{i_1}, \dots, z_{i_t}\}$ from the equations of S .

$$\begin{array}{lcl}
 z_1 + z_3 + z_4 = 1 \\
 z_2 + z_3 + z_4 = 0 \\
 z_2 + z_3 = 0 \\
 z_1 + z_2 = 1
 \end{array}
 \Rightarrow
 \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix}
 \Rightarrow
 \begin{array}{l}
 z_1 + z_4 = 1 \\
 z_2 + z_4 = 0 \\
 z_2 = 0 \\
 z_1 + z_2 = 1
 \end{array}$$

Reduction Rules

Reduction Rule (Rank rule)

Let A be the matrix over \mathbb{F}_2 corresponding to the set of equations in \mathcal{I} , such that $a_{ji} = 1$ if z_i appears in equation j , and 0 otherwise. Let $t = \text{rank} A$ and suppose columns a^{i_1}, \dots, a^{i_t} of A are linearly independent. Then delete all variables not in $\{z_{i_1}, \dots, z_{i_t}\}$ from the equations of S .

$$\begin{array}{l}
 z_1 + z_3 + z_4 = 1 \\
 z_2 + z_3 + z_4 = 0 \\
 z_2 + z_3 = 0 \\
 z_1 + z_2 = 1
 \end{array}
 \Rightarrow
 \begin{pmatrix}
 1 & 0 & 1 & 1 \\
 0 & 1 & 1 & 1 \\
 0 & 1 & 1 & 0 \\
 1 & 1 & 0 & 0
 \end{pmatrix}
 \Rightarrow
 \begin{array}{l}
 z_1 + z_4 = 1 \\
 z_2 + z_4 = 0 \\
 z_2 = 0 \\
 z_1 + z_2 = 1
 \end{array}$$

Reduction Rules

Reduction Rule (Rank rule)

Let A be the matrix over \mathbb{F}_2 corresponding to the set of equations in \mathcal{I} , such that $a_{ji} = 1$ if z_i appears in equation j , and 0 otherwise. Let $t = \text{rank} A$ and suppose columns a^{i_1}, \dots, a^{i_t} of A are linearly independent. Then delete all variables not in $\{z_{i_1}, \dots, z_{i_t}\}$ from the equations of S .

$$\begin{array}{l}
 z_1 + z_3 + z_4 = 1 \\
 z_2 + z_3 + z_4 = 0 \\
 z_2 + z_3 = 0 \\
 z_1 + z_2 = 1
 \end{array}
 \Rightarrow
 \begin{pmatrix}
 \mathbf{1} & \mathbf{0} & 1 & \mathbf{1} \\
 \mathbf{0} & \mathbf{1} & 1 & \mathbf{1} \\
 \mathbf{0} & \mathbf{1} & 1 & \mathbf{0} \\
 \mathbf{1} & \mathbf{1} & 0 & \mathbf{0}
 \end{pmatrix}
 \Rightarrow
 \begin{array}{l}
 z_1 + z_4 = 1 \\
 z_2 + z_4 = 0 \\
 z_2 = 0 \\
 z_1 + z_2 = 1
 \end{array}$$

Reduction Rules

Why does the **Rank Rule** work?

$$\begin{array}{l}
 \mathcal{I} \\
 z_1 + z_3 + z_4 = 1 \\
 z_2 + z_3 + z_4 = 0 \\
 z_2 + z_3 = 0 \\
 z_1 + z_2 = 1
 \end{array}
 \Rightarrow
 \begin{pmatrix}
 1 & 0 & 1 & 1 \\
 0 & 1 & 1 & 1 \\
 0 & 1 & 1 & 0 \\
 1 & 1 & 0 & 0
 \end{pmatrix}
 \Rightarrow
 \begin{array}{l}
 \mathcal{I}' \\
 z_1 + z_4 = 1 \\
 z_2 + z_4 = 0 \\
 z_2 = 0 \\
 z_1 + z_2 = 1
 \end{array}$$

- Set $z_3 = 0$ and add a solution for \mathcal{I}' to get a solution of equal weight for \mathcal{I} .
- Consider a solution for \mathcal{I} .
 If $z_3 = 1$, then change the values of z_1, z_2, z_3 to get an equivalent solution with $z_3 = 0$. **Why does this work?**
 So $z_3 = 0$, and we have a solution for \mathcal{I}' of equal weight.

Reduction Rules

Why does the **Rank Rule** work?

$$\begin{array}{l}
 \mathcal{I} \\
 z_1 + z_3 + z_4 = 1 \\
 z_2 + z_3 + z_4 = 0 \\
 z_2 + z_3 = 0 \\
 z_1 + z_2 = 1
 \end{array}
 \Rightarrow
 \begin{pmatrix}
 1 & 0 & 1 & 1 \\
 0 & 1 & 1 & 1 \\
 0 & 1 & 1 & 0 \\
 1 & 1 & 0 & 0
 \end{pmatrix}
 \Rightarrow
 \begin{array}{l}
 \mathcal{I}' \\
 z_1 + z_4 = 1 \\
 z_2 + z_4 = 0 \\
 z_2 = 0 \\
 z_1 + z_2 = 1
 \end{array}$$

- Set $z_3 = 0$ and add a solution for \mathcal{I}' to get a solution of equal weight for \mathcal{I} .
- Consider a solution for \mathcal{I} .

If $z_3 = 1$, then change the values of z_1, z_2, z_3 to get an equivalent solution with $z_3 = 0$. **Why does this work?**

So $z_3 = 0$, and we have a solution for \mathcal{I}' of equal weight.

Reduction Rules

Why does the **Rank Rule** work?

$$\begin{array}{l}
 \mathcal{I} \\
 z_1 + z_3 + z_4 = 1 \\
 z_2 + z_3 + z_4 = 0 \\
 z_2 + z_3 = 0 \\
 z_1 + z_2 = 1
 \end{array}
 \Rightarrow
 \begin{pmatrix}
 1 & 0 & 1 & 1 \\
 0 & 1 & 1 & 1 \\
 0 & 1 & 1 & 0 \\
 1 & 1 & 0 & 0
 \end{pmatrix}
 \Rightarrow
 \begin{array}{l}
 \mathcal{I}' \\
 z_1 + z_4 = 1 \\
 z_2 + z_4 = 0 \\
 z_2 = 0 \\
 z_1 + z_2 = 1
 \end{array}$$

- Set $z_3 = 0$ and add a solution for \mathcal{I}' to get a solution of equal weight for \mathcal{I} .
- Consider a solution for \mathcal{I} .

If $z_3 = 1$, then change the values of z_1, z_2, z_3 to get an equivalent solution with $z_3 = 0$. **Why does this work?**

So $z_3 = 0$, and we have a solution for \mathcal{I}' of equal weight.

Reduction Rules

Why does the **Rank Rule** work?

$$\begin{array}{l}
 \mathcal{I} \\
 z_1 + z_3 + z_4 = 1 \\
 z_2 + z_3 + z_4 = 0 \\
 z_2 + z_3 = 0 \\
 z_1 + z_2 = 1
 \end{array}
 \Rightarrow
 \begin{pmatrix}
 1 & 0 & 1 & 1 \\
 0 & 1 & 1 & 1 \\
 0 & 1 & 1 & 0 \\
 1 & 1 & 0 & 0
 \end{pmatrix}
 \Rightarrow
 \begin{array}{l}
 \mathcal{I}' \\
 z_1 + z_4 = 1 \\
 z_2 + z_4 = 0 \\
 z_2 = 0 \\
 z_1 + z_2 = 1
 \end{array}$$

- Set $z_3 = 0$ and add a solution for \mathcal{I}' to get a solution of equal weight for \mathcal{I} .
- Consider a solution for \mathcal{I} .
 If $z_3 = 1$, then change the values of z_1, z_2, z_3 to get an equivalent solution with $z_3 = 0$. **Why does this work?**
 So $z_3 = 0$, and we have a solution for \mathcal{I}' of equal weight.

Reduction Rules

Why does the **Rank Rule** work?

$$\begin{array}{l}
 \mathcal{I} \\
 z_1 + z_3 + z_4 = 1 \\
 z_2 + z_3 + z_4 = 0 \\
 z_2 + z_3 = 0 \\
 z_1 + z_2 = 1
 \end{array}
 \Rightarrow
 \begin{pmatrix}
 1 & 0 & 1 & 1 \\
 0 & 1 & 1 & 1 \\
 0 & 1 & 1 & 0 \\
 1 & 1 & 0 & 0
 \end{pmatrix}
 \Rightarrow
 \begin{array}{l}
 \mathcal{I}' \\
 z_1 + z_4 = 1 \\
 z_2 + z_4 = 0 \\
 z_2 = 0 \\
 z_1 + z_2 = 1
 \end{array}$$

- Set $z_3 = 0$ and add a solution for \mathcal{I}' to get a solution of equal weight for \mathcal{I} .
- Consider a solution for \mathcal{I} .
 If $z_3 = 1$, then change the values of z_1, z_2, z_3 to get an equivalent solution with $z_3 = 0$. **Why does this work?**
 So $z_3 = 0$, and we have a solution for \mathcal{I}' of equal weight.

Reduction rule

- What we would like to show: For reduced instances, if m is large enough the answer is YES.
- Sadly this is not true...
- Consider a 'complete' system on n variables with all RHS = 1.

$$x_1 = 1$$

$$x_2 = 1$$

$$x_1 + x_2 = 1$$

$$x_3 = 1$$

$$x_1 + x_3 = 1$$

$$x_2 + x_3 = 1$$

$$x_1 + x_2 + x_3 = 1$$

Reduction rule

- What we would like to show: For reduced instances, if m is large enough the answer is YES.
- Sadly this is not true...
- Consider a 'complete' system on n variables with all RHS = 1.

$$x_1 = 1$$

$$x_2 = 1$$

$$x_1 + x_2 = 1$$

$$x_3 = 1$$

$$x_1 + x_3 = 1$$

$$x_2 + x_3 = 1$$

$$x_1 + x_2 + x_3 = 1$$

Reduction rule

- What we would like to show: For reduced instances, if m is large enough the answer is YES.
- Sadly this is not true...
- Consider a 'complete' system on n variables with all RHS = 1.

$$x_1 = 1$$

$$x_2 = 1$$

$$x_1 + x_2 = 1$$

$$x_3 = 1$$

$$x_1 + x_3 = 1$$

$$x_2 + x_3 = 1$$

$$x_1 + x_2 + x_3 = 1$$

Reduction rule

- What we would like to show: For reduced instances, if m is large enough the answer is YES.
- Sadly this is not true...
- Consider a 'complete' system on n variables with all RHS = 1.

$$x_1 = 1$$

$$x_2 = 1$$

$$x_2 = 0$$

$$x_3 = 1$$

$$x_3 = 0$$

$$x_2 + x_3 = 1$$

$$x_2 + x_3 = 0$$

Reduction rule

- What we would like to show: For reduced instances, if m is large enough the answer is YES.
- Sadly this is not true...
- Consider a 'complete' system on n variables with all RHS = 1.

$$x_1 = 1$$

$$x_2 = 1$$

$$x_2 = 0$$

$$x_3 = 1$$

$$x_3 = 0$$

$$x_2 + x_3 = 1$$

$$x_2 + x_3 = 0$$

- The maximum excess is 1 but $m = 2^n - 1$.

Main Results

- **Theorem A:** [Crowston, Fellows, Gutin, Jones, Rosamond, Thomasse, Yeo, 2011] MAX-LIN-AA can be solved in time $O^*(n^{2k})$.
- **Theorem B:** [Crowston, Gutin, Jones, Kim, Ruzsa, 2010] If \mathcal{I} is reduced and $2k \leq m < 2^{n/2k}$, then \mathcal{I} is a YES-instance.

The above results can be combined to show the following

Theorem (Crowston, Fellows, Gutin, Jones, Rosamond, Thomasse, Yeo, 2011)

MAX-LIN-AA is fixed-parameter tractable, and has a kernel with $O(k^2 \log k)$ variables.

Proof of Theorem A (Algorithm \mathcal{H})

Algorithm \mathcal{H} (More detail)

- ➊ Choose an equation e , which can be written as $\sum_{i \in S} z_i = b$, with weight $w(e)$.
- ➋ Choose some $j \in S$.
- ➌ Simplify the system under the assumption that e is true:
 - ➊ Remove equation e .
 - ➋ Perform the substitution $z_j = \sum_{(i \in S \setminus j)} z_i + b$ for all equations containing z_j .
 - ➌ Reduce the system by LHS Rule.
- ➍ Reduce k by $w(e)/2$.

Example

$$z_1 + z_3 + z_5 = 1 \Rightarrow z_1 = z_3 + z_5 + 1$$

$z_2 + z_3 = 1$		$\Rightarrow z_2 + z_3 = 1$
$z_1 + z_2 = 0$	$\Rightarrow z_3 + z_5 + 1 + z_2 = 0$	$\Rightarrow z_2 + z_3 + z_5 = 1$
$z_3 + z_4 + z_5 = 1$		$\Rightarrow z_3 + z_4 + z_5 = 1$
$z_1 + z_4 = 0$	$\Rightarrow z_3 + z_5 + 1 + z_4 = 0$	$\Rightarrow z_3 + z_4 + z_5 = 1$
$z_1 + z_2 + z_5 = 1$	$\Rightarrow z_3 + z_5 + 1 + z_2 + z_5 = 1$	$\Rightarrow z_2 + z_3 = 0$

Now we simplify.....

Example

$$z_1 + z_3 + z_5 = 1 \Rightarrow z_1 = z_3 + z_5 + 1$$

$z_2 + z_3 = 1$		$\Rightarrow z_2 + z_3 = 1$
$z_1 + z_2 = 0$	$\Rightarrow z_3 + z_5 + 1 + z_2 = 0$	$\Rightarrow z_2 + z_3 + z_5 = 1$
$z_3 + z_4 + z_5 = 1$		$\Rightarrow z_3 + z_4 + z_5 = 1$
$z_1 + z_4 = 0$	$\Rightarrow z_3 + z_5 + 1 + z_4 = 0$	$\Rightarrow z_3 + z_4 + z_5 = 1$
$z_1 + z_2 + z_5 = 1$	$\Rightarrow z_3 + z_5 + 1 + z_2 + z_5 = 1$	$\Rightarrow z_2 + z_3 = 0$

Now we simplify.....

Example

$$z_1 + z_3 + z_5 = 1 \Rightarrow z_1 = z_3 + z_5 + 1$$

$$z_2 + z_3 = 1$$

$$z_1 + z_2 = 0$$

$$z_3 + z_4 + z_5 = 1$$

$$z_1 + z_4 = 0$$

$$z_1 + z_2 + z_5 = 1$$

$$\Rightarrow z_3 + z_5 + 1 + z_2 = 0$$

$$\Rightarrow z_3 + z_5 + 1 + z_4 = 0$$

$$\Rightarrow z_3 + z_5 + 1 + z_2 + z_5 = 1$$

$$\Rightarrow z_2 + z_3 = 1$$

$$\Rightarrow z_2 + z_3 + z_5 = 1$$

$$\Rightarrow z_3 + z_4 + z_5 = 1$$

$$\Rightarrow z_3 + z_4 + z_5 = 1$$

$$\Rightarrow z_2 + z_3 = 0$$

Now we simplify.....

Example

$$z_1 + z_3 + z_5 = 1 \Rightarrow z_1 = z_3 + z_5 + 1$$

$$\begin{array}{llll} z_2 + z_3 = 1 & & \Rightarrow & z_2 + z_3 = 1 \\ z_1 + z_2 = 0 & \Rightarrow & z_3 + z_5 + 1 + z_2 = 0 & \Rightarrow z_2 + z_3 + z_5 = 1 \\ z_3 + z_4 + z_5 = 1 & & \Rightarrow & z_3 + z_4 + z_5 = 1 \\ z_1 + z_4 = 0 & \Rightarrow & z_3 + z_5 + 1 + z_4 = 0 & \Rightarrow z_3 + z_4 + z_5 = 1 \\ z_1 + z_2 + z_5 = 1 & \Rightarrow & z_3 + z_5 + 1 + z_2 + z_5 = 1 & \Rightarrow z_2 + z_3 = 0 \end{array}$$

Now we simplify.....

Example

$$(z_1 + z_3 + z_5 = 1)$$

$$z_2 + z_3 + z_5 = 1$$

$$z_3 + z_4 + z_5 = 1 \quad (w = 2)$$

So under the assumption that $e = "z_1 + z_3 + z_5 = 1"$ is true we have reduced \mathcal{I} to a smaller problem \mathcal{I}' such that we can do $w(e)/2$ more above average in \mathcal{I} than in \mathcal{I}' . Why?

Answer: For any solution of \mathcal{I}' , set $z_1 = z_3 + z_5 + 1$

Example

$$(z_1 + z_3 + z_5 = 1)$$

$$z_2 + z_3 + z_5 = 1$$

$$z_3 + z_4 + z_5 = 1 \quad (w = 2)$$

So under the assumption that $e = "z_1 + z_3 + z_5 = 1"$ is true we have reduced \mathcal{I} to a smaller problem \mathcal{I}' such that we can do $w(e)/2$ more above average in \mathcal{I} than in \mathcal{I}' . Why?

Answer: For any solution of \mathcal{I}' , set $z_1 = z_3 + z_5 + 1$

Example

$$(z_1 + z_3 + z_5 = 1)$$

$$z_2 + z_3 + z_5 = 1$$

$$z_3 + z_4 + z_5 = 1 \quad (w = 2)$$

So under the assumption that $e = "z_1 + z_3 + z_5 = 1"$ is true we have reduced \mathcal{I} to a smaller problem \mathcal{I}' such that we can do $w(e)/2$ more above average in \mathcal{I} than in \mathcal{I}' . Why?

Answer: For any solution of \mathcal{I}' , set $z_1 = z_3 + z_5 + 1$

Example

$$(z_1 + z_3 + z_5 = 1)$$

$$z_2 + z_3 + z_5 = 1$$

$$z_3 + z_4 + z_5 = 1 \quad (w = 2)$$

So under the assumption that $e = "z_1 + z_3 + z_5 = 1"$ is true we have reduced \mathcal{I} to a smaller problem \mathcal{I}' such that we can do $w(e)/2$ more above average in \mathcal{I} than in \mathcal{I}' . Why?

Answer: For any solution of \mathcal{I}' , set $z_1 = z_3 + z_5 + 1$

So what does Algorithm \mathcal{H} give us

Assume our instance is reduced.

- If we can mark equations of total weight R then the maximum excess is at least R (we can get at least $R/2$ above the average).
- If the maximum excess is R then if we keep choosing equations which are true in a given optimal solution, we will mark equations of total weight R .

How can this be used to prove Theorem A.....

So what does Algorithm \mathcal{H} give us

Assume our instance is reduced.

- If we can mark equations of total weight R then the maximum excess is at least R (we can get at least $R/2$ above the average).
- If the maximum excess is R then if we keep choosing equations which are true in a given optimal solution, we will mark equations of total weight R .

How can this be used to prove Theorem A.....

So what does Algorithm \mathcal{H} give us

Assume our instance is reduced.

- If we can mark equations of total weight R then the maximum excess is at least R (we can get at least $R/2$ above the average).
- If the maximum excess is R then if we keep choosing equations which are true in a given optimal solution, we will mark equations of total weight R .

How can this be used to prove Theorem A.....

So what does Algorithm \mathcal{H} give us

Assume our instance is reduced.

- If we can mark equations of total weight R then the maximum excess is at least R (we can get at least $R/2$ above the average).
- If the maximum excess is R then if we keep choosing equations which are true in a given optimal solution, we will mark equations of total weight R .

How can this be used to prove Theorem A.....

Proof of Theorem A

Theorem A [Crowston, Fellows, Gutin, Jones, Rosamond, Thomasse, Yeo, 2011] There exists an $O^*(n^{2k})$ -time algorithm for MAX-LIN-AA.

- **Proof (sketch):** Let e_1, \dots, e_n be a set of equations in \mathcal{I} which are 'independent'.
(LHSs correspond to independent rows in matrix A .)
- Check unique assignment in which e_1, \dots, e_n all false. If this assignment achieves excess $2k$, return YES.
- Otherwise, one of e_1, \dots, e_k must be true.
- Branch n ways. In branch i mark equation e_i in Algorithm \mathcal{H} and solve resulting system.
- Since we can stop after $2k$ iterations of \mathcal{H} , search tree has n^{2k} leaves.

Proof of Theorem A

Theorem A [Crowston, Fellows, Gutin, Jones, Rosamond, Thomasse, Yeo, 2011] There exists an $O^*(n^{2k})$ -time algorithm for MAX-LIN-AA.

- **Proof (sketch):** Let e_1, \dots, e_n be a set of equations in \mathcal{I} which are 'independent'.
(LHSs correspond to independent rows in matrix A .)
- Check unique assignment in which e_1, \dots, e_n all false. If this assignment achieves excess $2k$, return YES.
- Otherwise, one of e_1, \dots, e_k must be true.
- Branch n ways. In branch i mark equation e_i in Algorithm \mathcal{H} and solve resulting system.
- Since we can stop after $2k$ iterations of \mathcal{H} , search tree has n^{2k} leaves.

Proof of Theorem A

Theorem A [Crowston, Fellows, Gutin, Jones, Rosamond, Thomasse, Yeo, 2011] There exists an $O^*(n^{2k})$ -time algorithm for MAX-LIN-AA.

- **Proof (sketch):** Let e_1, \dots, e_n be a set of equations in \mathcal{I} which are 'independent'.
(LHSs correspond to independent rows in matrix A .)
- Check unique assignment in which e_1, \dots, e_n all false. If this assignment achieves excess $2k$, return YES.
- Otherwise, one of e_1, \dots, e_k must be true.
- Branch n ways. In branch i mark equation e_i in Algorithm \mathcal{H} and solve resulting system.
- Since we can stop after $2k$ iterations of \mathcal{H} , search tree has n^{2k} leaves.

Proof of Theorem A

Theorem A [Crowston, Fellows, Gutin, Jones, Rosamond, Thomasse, Yeo, 2011] There exists an $O^*(n^{2k})$ -time algorithm for MAX-LIN-AA.

- **Proof (sketch):** Let e_1, \dots, e_n be a set of equations in \mathcal{I} which are 'independent'.
(LHSs correspond to independent rows in matrix A .)
- Check unique assignment in which e_1, \dots, e_n all false. If this assignment achieves excess $2k$, return YES.
- Otherwise, one of e_1, \dots, e_k must be true.
- Branch n ways. In branch i mark equation e_i in Algorithm \mathcal{H} and solve resulting system.
- Since we can stop after $2k$ iterations of \mathcal{H} , search tree has n^{2k} leaves.

Proof of Theorem A

Theorem A [Crowston, Fellows, Gutin, Jones, Rosamond, Thomasse, Yeo, 2011] There exists an $O^*(n^{2k})$ -time algorithm for MAX-LIN-AA.

- **Proof (sketch):** Let e_1, \dots, e_n be a set of equations in \mathcal{I} which are 'independent'.
(LHSs correspond to independent rows in matrix A .)
- Check unique assignment in which e_1, \dots, e_n all false. If this assignment achieves excess $2k$, return YES.
- Otherwise, one of e_1, \dots, e_k must be true.
- Branch n ways. In branch i mark equation e_i in Algorithm \mathcal{H} and solve resulting system.
- Since we can stop after $2k$ iterations of \mathcal{H} , search tree has n^{2k} leaves.

Proof of Theorem B

Theorem B: [Crowston, Gutin, Jones, Kim, Ruzsa, 2010] If \mathcal{I} is reduced and $2k \leq m < 2^{n/2k}$, then \mathcal{I} is a YES-instance.

- If we can run algorithm \mathcal{H} for $2k$ iterations, we can get an excess of at least $2k$.
- Problem: After running \mathcal{H} a few times all the remaining equations may 'cancel out' under LHS Rule.
- One solution: M -sum-free vectors.
- Let K and M be sets of vectors in \mathbb{F}_2^n such that $K \subseteq M$.
- K is **M -sum-free** if no sum of two or more vectors in K is equal to a vector in M .

Proof of Theorem B

Theorem B: [Crowston, Gutin, Jones, Kim, Ruzsa, 2010] If \mathcal{I} is reduced and $2k \leq m < 2^{n/2k}$, then \mathcal{I} is a YES-instance.

- If we can run algorithm \mathcal{H} for $2k$ iterations, we can get an excess of at least $2k$.
- Problem: After running \mathcal{H} a few times all the remaining equations may 'cancel out' under LHS Rule.
- One solution: M -sum-free vectors.
- Let K and M be sets of vectors in \mathbb{F}_2^n such that $K \subseteq M$.
- K is **M -sum-free** if no sum of two or more vectors in K is equal to a vector in M .

Proof of Theorem B

Theorem B: [Crowston, Gutin, Jones, Kim, Ruzsa, 2010] If \mathcal{I} is reduced and $2k \leq m < 2^{n/2k}$, then \mathcal{I} is a YES-instance.

- If we can run algorithm \mathcal{H} for $2k$ iterations, we can get an excess of at least $2k$.
- Problem: After running \mathcal{H} a few times all the remaining equations may 'cancel out' under LHS Rule.
- One solution: M -sum-free vectors.
- Let K and M be sets of vectors in \mathbb{F}_2^n such that $K \subseteq M$.
- K is **M -sum-free** if no sum of two or more vectors in K is equal to a vector in M .

Proof of Theorem B

Theorem B: [Crowston, Gutin, Jones, Kim, Ruzsa, 2010] If \mathcal{I} is reduced and $2k \leq m < 2^{n/2k}$, then \mathcal{I} is a YES-instance.

- If we can run algorithm \mathcal{H} for $2k$ iterations, we can get an excess of at least $2k$.
- Problem: After running \mathcal{H} a few times all the remaining equations may 'cancel out' under LHS Rule.
- One solution: M -sum-free vectors.
- Let K and M be sets of vectors in \mathbb{F}_2^n such that $K \subseteq M$.
- K is **M -sum-free** if no sum of two or more vectors in K is equal to a vector in M .

Proof of Theorem B

Theorem B: [Crowston, Gutin, Jones, Kim, Ruzsa, 2010] If \mathcal{I} is reduced and $2k \leq m < 2^{n/2k}$, then \mathcal{I} is a YES-instance.

- If we can run algorithm \mathcal{H} for $2k$ iterations, we can get an excess of at least $2k$.
- Problem: After running \mathcal{H} a few times all the remaining equations may 'cancel out' under LHS Rule.
- One solution: M -sum-free vectors.
- Let K and M be sets of vectors in \mathbb{F}_2^n such that $K \subseteq M$.
- K is **M -sum-free** if no sum of two or more vectors in K is equal to a vector in M .

Proof of Theorem B

Lemma *View the LHSs of equations in \mathcal{I} as a set M of vectors in \mathbb{F}_2^n . Let e_1, \dots, e_t be a set of equations in \mathcal{I} that correspond to an M -sum-free set of vectors. Then we can run algorithm \mathcal{H} for t iterations, choosing equations e_1, \dots, e_t in turn, and get an excess of at least t .*

Why? Assume for the sake of contradiction e_i gets cancelled out.

- Then by picking e_1, \dots, e_{i-1} in Algorithm \mathcal{H} we have created a different equation, say f_i , with the same LHS as e_i .
- So considering LHSs we get: $e_i = f_i = e_{j_1} + e_{j_2} + \dots + e_{j_a} + e'$ for some $\{j_1, \dots, j_a\} \subseteq \{1, \dots, i-1\}$ and e' is any equation.
- However this implies that $e' = e_{j_1} + e_{j_2} + \dots + e_{j_a} + e_i$, a contradiction.

Proof of Theorem B

Lemma *View the LHSs of equations in \mathcal{I} as a set M of vectors in \mathbb{F}_2^n . Let e_1, \dots, e_t be a set of equations in \mathcal{I} that correspond to an M -sum-free set of vectors. Then we can run algorithm \mathcal{H} for t iterations, choosing equations e_1, \dots, e_t in turn, and get an excess of at least t .*

Why? Assume for the sake of contradiction e_i gets cancelled out.

- Then by picking e_1, \dots, e_{i-1} in Algorithm \mathcal{H} we have created a different equation, say f_i , with the same LHS as e_i .
- So considering LHSs we get: $e_i = f_i = e_{j_1} + e_{j_2} + \dots + e_{j_a} + e'$ for some $\{j_1, \dots, j_a\} \subseteq \{1, \dots, i-1\}$ and e' is any equation.
- However this implies that $e' = e_{j_1} + e_{j_2} + \dots + e_{j_a} + e_i$, a contradiction.

Proof of Theorem B

Lemma *View the LHSs of equations in \mathcal{I} as a set M of vectors in \mathbb{F}_2^n . Let e_1, \dots, e_t be a set of equations in \mathcal{I} that correspond to an M -sum-free set of vectors. Then we can run algorithm \mathcal{H} for t iterations, choosing equations e_1, \dots, e_t in turn, and get an excess of at least t .*

Why? Assume for the sake of contradiction e_i gets cancelled out.

- Then by picking e_1, \dots, e_{i-1} in Algorithm \mathcal{H} we have created a different equation, say f_i , with the same LHS as e_i .
- So considering LHSs we get: $e_i = f_i = e_{j_1} + e_{j_2} + \dots + e_{j_a} + e'$ for some $\{j_1, \dots, j_a\} \subseteq \{1, \dots, i-1\}$ and e' is any equation.
- However this implies that $e' = e_{j_1} + e_{j_2} + \dots + e_{j_a} + e_i$, a contradiction.

Proof of Theorem B

Lemma *View the LHSs of equations in \mathcal{I} as a set M of vectors in \mathbb{F}_2^n . Let e_1, \dots, e_t be a set of equations in \mathcal{I} that correspond to an M -sum-free set of vectors. Then we can run algorithm \mathcal{H} for t iterations, choosing equations e_1, \dots, e_t in turn, and get an excess of at least t .*

Why? Assume for the sake of contradiction e_i gets cancelled out.

- Then by picking e_1, \dots, e_{i-1} in Algorithm \mathcal{H} we have created a different equation, say f_i , with the same LHS as e_i .
- So considering LHSs we get: $e_i = f_i = e_{j_1} + e_{j_2} + \dots + e_{j_a} + e'$ for some $\{j_1, \dots, j_a\} \subseteq \{1, \dots, i-1\}$ and e' is any equation.
- However this implies that $e' = e_{j_1} + e_{j_2} + \dots + e_{j_a} + e_i$, a contradiction.

Proof of Theorem B

Lemma *View the LHSs of equations in \mathcal{I} as a set M of vectors in \mathbb{F}_2^n . Let e_1, \dots, e_t be a set of equations in \mathcal{I} that correspond to an M -sum-free set of vectors. Then we can run algorithm \mathcal{H} for t iterations, choosing equations e_1, \dots, e_t in turn, and get an excess of at least t .*

Why? Assume for the sake of contradiction e_i gets cancelled out.

- Then by picking e_1, \dots, e_{i-1} in Algorithm \mathcal{H} we have created a different equation, say f_i , with the same LHS as e_i .
- So considering LHSs we get: $e_i = f_i = e_{j_1} + e_{j_2} + \dots + e_{j_a} + e'$ for some $\{j_1, \dots, j_a\} \subseteq \{1, \dots, i-1\}$ and e' is any equation.
- However this implies that $e' = e_{j_1} + e_{j_2} + \dots + e_{j_a} + e_i$, a contradiction.

Proof of Theorem B

Lemma C [Crowston, Gutin, Jones, Kim and Ruzsa (2010)] Let M be a proper subset in \mathbb{F}_2^n such that $\text{span}(M) = \mathbb{F}_2^n$. If k is a positive integer and $t \leq |M| \leq 2^{n/t}$ then, in time $|M|^{O(1)}$, we can find an M -sum-free subset K of M s.t. $|K| = t$.

Theorem B: If $2k \leq m < 2^{n/2k}$, then \mathcal{I} is a YES-instance.

- Suppose \mathcal{I} is reduced and $2k \leq m \leq 2^{n/2k}$.
- Let M be the set of vectors in \mathbb{F}_2^n corresponding to LHSs of equations in \mathcal{I}
- Find an M -sum-free subset K of M s.t. $|K| = 2k$.
- Let e_1, \dots, e_{2k} be the equations corresponding to K , and run algorithm \mathcal{H} marking e_1, \dots, e_{2k} in turn.
- Then we get excess $2k$, so the answer is YES.

Proof of Theorem B

Lemma C [Crowston, Gutin, Jones, Kim and Ruzsa (2010)] Let M be a proper subset in \mathbb{F}_2^n such that $\text{span}(M) = \mathbb{F}_2^n$. If k is a positive integer and $t \leq |M| \leq 2^{n/t}$ then, in time $|M|^{O(1)}$, we can find an M -sum-free subset K of M s.t. $|K| = t$.

Theorem B: If $2k \leq m < 2^{n/2k}$, then \mathcal{I} is a YES-instance.

- Suppose \mathcal{I} is reduced and $2k \leq m \leq 2^{n/2k}$.
- Let M be the set of vectors in \mathbb{F}_2^n corresponding to LHSs of equations in \mathcal{I}
- Find an M -sum-free subset K of M s.t. $|K| = 2k$.
- Let e_1, \dots, e_{2k} be the equations corresponding to K , and run algorithm \mathcal{H} marking e_1, \dots, e_{2k} in turn.
- Then we get excess $2k$, so the answer is YES.

Proof of Theorem B

Lemma C [Crowston, Gutin, Jones, Kim and Ruzsa (2010)] Let M be a proper subset in \mathbb{F}_2^n such that $\text{span}(M) = \mathbb{F}_2^n$. If k is a positive integer and $t \leq |M| \leq 2^{n/t}$ then, in time $|M|^{O(1)}$, we can find an M -sum-free subset K of M s.t. $|K| = t$.

Theorem B: If $2k \leq m < 2^{n/2k}$, then \mathcal{I} is a YES-instance.

- Suppose \mathcal{I} is reduced and $2k \leq m \leq 2^{n/2k}$.
- Let M be the set of vectors in \mathbb{F}_2^n corresponding to LHSs of equations in \mathcal{I}
- Find an M -sum-free subset K of M s.t. $|K| = 2k$.
- Let e_1, \dots, e_{2k} be the equations corresponding to K , and run algorithm \mathcal{H} marking e_1, \dots, e_{2k} in turn.
- Then we get excess $2k$, so the answer is YES.

Proof of Theorem B

Lemma C [Crowston, Gutin, Jones, Kim and Ruzsa (2010)] Let M be a proper subset in \mathbb{F}_2^n such that $\text{span}(M) = \mathbb{F}_2^n$. If k is a positive integer and $t \leq |M| \leq 2^{n/t}$ then, in time $|M|^{O(1)}$, we can find an M -sum-free subset K of M s.t. $|K| = t$.

Theorem B: If $2k \leq m < 2^{n/2k}$, then \mathcal{I} is a YES-instance.

- Suppose \mathcal{I} is reduced and $2k \leq m \leq 2^{n/2k}$.
- Let M be the set of vectors in \mathbb{F}_2^n corresponding to LHSs of equations in \mathcal{I}
- Find an M -sum-free subset K of M s.t. $|K| = 2k$.
- Let e_1, \dots, e_{2k} be the equations corresponding to K , and run algorithm \mathcal{H} marking e_1, \dots, e_{2k} in turn.
- Then we get excess $2k$, so the answer is YES.

Proof of Theorem B

Lemma C [Crowston, Gutin, Jones, Kim and Ruzsa (2010)] Let M be a proper subset in \mathbb{F}_2^n such that $\text{span}(M) = \mathbb{F}_2^n$. If k is a positive integer and $t \leq |M| \leq 2^{n/t}$ then, in time $|M|^{O(1)}$, we can find an M -sum-free subset K of M s.t. $|K| = t$.

Theorem B: If $2k \leq m < 2^{n/2k}$, then \mathcal{I} is a YES-instance.

- Suppose \mathcal{I} is reduced and $2k \leq m \leq 2^{n/2k}$.
- Let M be the set of vectors in \mathbb{F}_2^n corresponding to LHSs of equations in \mathcal{I}
- Find an M -sum-free subset K of M s.t. $|K| = 2k$.
- Let e_1, \dots, e_{2k} be the equations corresponding to K , and run algorithm \mathcal{H} marking e_1, \dots, e_{2k} in turn.
- Then we get excess $2k$, so the answer is YES.

Proof of Theorem B

Lemma C [Crowston, Gutin, Jones, Kim and Ruzsa (2010)] Let M be a proper subset in \mathbb{F}_2^n such that $\text{span}(M) = \mathbb{F}_2^n$. If k is a positive integer and $t \leq |M| \leq 2^{n/t}$ then, in time $|M|^{O(1)}$, we can find an M -sum-free subset K of M s.t. $|K| = t$.

Theorem B: If $2k \leq m < 2^{n/2k}$, then \mathcal{I} is a YES-instance.

- Suppose \mathcal{I} is reduced and $2k \leq m \leq 2^{n/2k}$.
- Let M be the set of vectors in \mathbb{F}_2^n corresponding to LHSs of equations in \mathcal{I}
- Find an M -sum-free subset K of M s.t. $|K| = 2k$.
- Let e_1, \dots, e_{2k} be the equations corresponding to K , and run algorithm \mathcal{H} marking e_1, \dots, e_{2k} in turn.
- Then we get excess $2k$, so the answer is YES.

Recall Theorem A and Theorem B

Theorem A: [Crowston, Fellows, Gutin, Jones, Rosamond, Thomasse, Yeo, 2011] MAX-LIN-AA can be solved in time $O^*(n^{2k})$.

Theorem B: [Crowston, Gutin, Jones, Kim, Ruzsa, 2010] If \mathcal{I} is reduced and $2k \leq m < 2^{n/2k}$, then \mathcal{I} is a YES-instance.

Proof of our main result

Theorem (Crowston, Fellows, Gutin, Jones, Rosamond, Thomasse, Yeo, 2011)

MAX-LIN-AA *has a kernel with at most $O(k^2 \log k)$ variables.*

- **Proof:** Let \mathcal{I} be a reduced system.
- Case 1: $m \geq n^{2k}$. Then using $O^*(n^{2k})$ algorithm, can solve in polynomial time.
- Case 2: $2k \leq m \leq 2^{n/2k}$. By earlier Theorem return YES.
- Case 3: $m < 2k$. Since \mathcal{I} reduced by Rank Rule, $n \leq m$ so $n = O(k^2 \log k)$.
- Only remaining case is $2^{n/2k} < m < n^{2k}$.

Proof of our main result

Theorem (Crowston, Fellows, Gutin, Jones, Rosamond, Thomasse, Yeo, 2011)

MAX-LIN-AA has a kernel with at most $O(k^2 \log k)$ variables.

- **Proof:** Let \mathcal{I} be a reduced system.
- Case 1: $m \geq n^{2k}$. Then using $O^*(n^{2k})$ algorithm, can solve in polynomial time.
- Case 2: $2k \leq m \leq 2^{n/2k}$. By earlier Theorem return YES.
- Case 3: $m < 2k$. Since \mathcal{I} reduced by Rank Rule, $n \leq m$ so $n = O(k^2 \log k)$.
- Only remaining case is $2^{n/2k} < m < n^{2k}$.

Proof of our main result

Theorem (Crowston, Fellows, Gutin, Jones, Rosamond, Thomasse, Yeo, 2011)

MAX-LIN-AA has a kernel with at most $O(k^2 \log k)$ variables.

- **Proof:** Let \mathcal{I} be a reduced system.
- Case 1: $m \geq n^{2k}$. Then using $O^*(n^{2k})$ algorithm, can solve in polynomial time.
- Case 2: $2k \leq m \leq 2^{n/2k}$. By earlier Theorem return YES.
- Case 3: $m < 2k$. Since \mathcal{I} reduced by Rank Rule, $n \leq m$ so $n = O(k^2 \log k)$.
- Only remaining case is $2^{n/2k} < m < n^{2k}$.

Proof of our main result

Theorem (Crowston, Fellows, Gutin, Jones, Rosamond, Thomasse, Yeo, 2011)

MAX-LIN-AA has a kernel with at most $O(k^2 \log k)$ variables.

- **Proof:** Let \mathcal{I} be a reduced system.
- Case 1: $m \geq n^{2k}$. Then using $O^*(n^{2k})$ algorithm, can solve in polynomial time.
- Case 2: $2k \leq m \leq 2^{n/2k}$. By earlier Theorem return YES.
- Case 3: $m < 2k$. Since \mathcal{I} reduced by Rank Rule, $n \leq m$ so $n = O(k^2 \log k)$.
- Only remaining case is $2^{n/2k} < m < n^{2k}$.

Proof of our main result

Theorem (Crowston, Fellows, Gutin, Jones, Rosamond, Thomasse, Yeo, 2011)

MAX-LIN-AA has a kernel with at most $O(k^2 \log k)$ variables.

- **Proof:** Let \mathcal{I} be a reduced system.
- Case 1: $m \geq n^{2k}$. Then using $O^*(n^{2k})$ algorithm, can solve in polynomial time.
- Case 2: $2k \leq m \leq 2^{n/2k}$. By earlier Theorem return YES.
- Case 3: $m < 2k$. Since \mathcal{I} reduced by Rank Rule, $n \leq m$ so $n = O(k^2 \log k)$.
- Only remaining case is $2^{n/2k} < m < n^{2k}$.

Proof of our main result

Theorem (Crowston, Fellows, Gutin, Jones, Rosamond, Thomasse, Yeo, 2011)

MAX-LIN-AA has a kernel with at most $O(k^2 \log k)$ variables.

- **Proof:** Let \mathcal{I} be a reduced system.
- Case 1: $m \geq n^{2k}$. Then using $O^*(n^{2k})$ algorithm, can solve in polynomial time.
- Case 2: $2k \leq m \leq 2^{n/2k}$. By earlier Theorem return YES.
- Case 3: $m < 2k$. Since \mathcal{I} reduced by Rank Rule, $n \leq m$ so $n = O(k^2 \log k)$.
- Only remaining case is $2^{n/2k} < m < n^{2k}$.

Proof of our main result (continued)

- Suppose $2^{n/2k} < m < n^{2k}$. Then $n/2k < 2k \log n$.
- So $n < 4k^2 \log n$.
- In order to bound $\log n$ we note that $\sqrt{n} < n/\log n < 4k^2$.
- Therefore $n < (2k)^4$ and $\log n < 4 \log(2k)$
- So $n < 4k^2 \log n < 16k^2(\log k + 1)$
- So $n = O(k^2 \log k)$.

Proof of our main result (continued)

- Suppose $2^{n/2k} < m < n^{2k}$. Then $n/2k < 2k \log n$.
- So $n < 4k^2 \log n$.
- In order to bound $\log n$ we note that $\sqrt{n} < n/\log n < 4k^2$.
- Therefore $n < (2k)^4$ and $\log n < 4 \log(2k)$
- So $n < 4k^2 \log n < 16k^2(\log k + 1)$
- So $n = O(k^2 \log k)$.

Proof of our main result (continued)

- Suppose $2^{n/2k} < m < n^{2k}$. Then $n/2k < 2k \log n$.
- So $n < 4k^2 \log n$.
- In order to bound $\log n$ we note that $\sqrt{n} < n/\log n < 4k^2$.
- Therefore $n < (2k)^4$ and $\log n < 4 \log(2k)$
- So $n < 4k^2 \log n < 16k^2(\log k + 1)$
- So $n = O(k^2 \log k)$.

Proof of our main result (continued)

- Suppose $2^{n/2k} < m < n^{2k}$. Then $n/2k < 2k \log n$.
- So $n < 4k^2 \log n$.
- In order to bound $\log n$ we note that $\sqrt{n} < n/\log n < 4k^2$.
- Therefore $n < (2k)^4$ and $\log n < 4 \log(2k)$
- So $n < 4k^2 \log n < 16k^2(\log k + 1)$
- So $n = O(k^2 \log k)$.

Proof of our main result (continued)

- Suppose $2^{n/2k} < m < n^{2k}$. Then $n/2k < 2k \log n$.
- So $n < 4k^2 \log n$.
- In order to bound $\log n$ we note that $\sqrt{n} < n/\log n < 4k^2$.
- Therefore $n < (2k)^4$ and $\log n < 4 \log(2k)$
- So $n < 4k^2 \log n < 16k^2(\log k + 1)$
- So $n = O(k^2 \log k)$.

Proof of our main result (continued)

- Suppose $2^{n/2k} < m < n^{2k}$. Then $n/2k < 2k \log n$.
- So $n < 4k^2 \log n$.
- In order to bound $\log n$ we note that $\sqrt{n} < n/\log n < 4k^2$.
- Therefore $n < (2k)^4$ and $\log n < 4 \log(2k)$
- So $n < 4k^2 \log n < 16k^2(\log k + 1)$
- So $n = O(k^2 \log k)$.

Our Main Result!

Recall our main result.

Theorem (Crowston, Fellows, Gutin, Jones, Rosamond, Thomasse, Yeo, 2011)

MAX-LIN-AA has a kernel with at most $O(k^2 \log k)$ variables.

- This kernel has a polynomial number of variables, but it is not a polynomial kernel!
- Number of equations may be $O(2^n)$.
- **Open question:** Does MAX-LIN-AA have a polynomial kernel?

Our Main Result!

Recall our main result.

Theorem (Crowston, Fellows, Gutin, Jones, Rosamond, Thomasse, Yeo, 2011)

MAX-LIN-AA has a kernel with at most $O(k^2 \log k)$ variables.

- This kernel has a polynomial number of variables, but it is not a polynomial kernel!
- Number of equations may be $O(2^n)$.
- **Open question:** Does MAX-LIN-AA have a polynomial kernel?

Our Main Result!

Recall our main result.

Theorem (Crowston, Fellows, Gutin, Jones, Rosamond, Thomasse, Yeo, 2011)

MAX-LIN-AA has a kernel with at most $O(k^2 \log k)$ variables.

- This kernel has a polynomial number of variables, but it is not a polynomial kernel!
- Number of equations may be $O(2^n)$.
- **Open question:** Does MAX-LIN-AA have a polynomial kernel?

Application of our Main Result

Theorem (Crowston, Fellows, Gutin, Jones, Rosamond, Thomasse, Yeo, 2011)

MAX-LIN-AA *can be solved in time* $O^*(2^{O(k \log k)})$.

- **Proof:** Assume \mathcal{I} is an irreducible system with m equations and n variables.
- In polynomial time, we either solve MAX-LIN-AA or get a kernel with $O(k^2 \log k)$ variables.
- If we have a kernel, apply the $O^*(n^{2k})$ -time algorithm.
- Since $n = O(k^2 \log k)$, we have running time $O^*((O(k^2 \log k))^{2k}) = O^*(2^{O(2k \log(k^2 \log k))}) = O^*(2^{O(k \log k)})$.

Outline

- 1 Parameterizing above tight bounds: Example Max-Sat
- 2 Max-Lin-AA
- 3 FPT Results
- 4 Related Results**

Related Results (Max- r -Lin-AA)

I will not say much about MAX- r -LIN-AA (where equations have at most r variables) as this will be covered in the next talk!

- Gutin, Kim, Szeider, Yeo (2009) - kernel with $m < (2k - 1)^2 64^r$.
- Crowston, Gutin, Kim, Jones, Rusza (2010) - kernel with $n = O(k \log k)$.
- Kim, Williams (2011) - kernel with $n < kr(r + 1)$
- Crowston et.al (2011) - kernel with $n \leq (2k - 1)r$.

Related Results (Max- r -Lin-AA)

I will not say much about MAX- r -LIN-AA (where equations have at most r variables) as this will be covered in the next talk!

- Gutin, Kim, Szeider, Yeo (2009) - kernel with $m < (2k - 1)^2 64^r$.
- Crowston, Gutin, Kim, Jones, Rusza (2010) - kernel with $n = O(k \log k)$.
- Kim, Williams (2011) - kernel with $n < kr(r + 1)$
- Crowston et.al (2011) - kernel with $n \leq (2k - 1)r$.

Related Results (Max- r -Lin-AA)

I will not say much about MAX- r -LIN-AA (where equations have at most r variables) as this will be covered in the next talk!

- Gutin, Kim, Szeider, Yeo (2009) - kernel with $m < (2k - 1)^2 64^r$.
- Crowston, Gutin, Kim, Jones, Rusza (2010) - kernel with $n = O(k \log k)$.
- Kim, Williams (2011) - kernel with $n < kr(r + 1)$
- Crowston et.al (2011) - kernel with $n \leq (2k - 1)r$.

Related Results (Max- r -Lin-AA)

I will not say much about MAX- r -LIN-AA (where equations have at most r variables) as this will be covered in the next talk!

- Gutin, Kim, Szeider, Yeo (2009) - kernel with $m < (2k - 1)^2 64^r$.
- Crowston, Gutin, Kim, Jones, Rusza (2010) - kernel with $n = O(k \log k)$.
- Kim, Williams (2011) - kernel with $n < kr(r + 1)$
- Crowston et.al (2011) - kernel with $n \leq (2k - 1)r$.

Related Results (Max- r -Lin-AA)

I will not say much about MAX- r -LIN-AA (where equations have at most r variables) as this will be covered in the next talk!

- Gutin, Kim, Szeider, Yeo (2009) - kernel with $m < (2k - 1)^2 64^r$.
- Crowston, Gutin, Kim, Jones, Rusza (2010) - kernel with $n = O(k \log k)$.
- Kim, Williams (2011) - kernel with $n < kr(r + 1)$
- Crowston et.al (2011) - kernel with $n \leq (2k - 1)r$.

Related Results

- **Pseudo-boolean function:** a function $f : \{-1, +1\}^n \rightarrow \mathbb{R}$
- Suppose we know the Fourier expansion of $f(x)$

$$f(x) = \sum_{S \subseteq [n]} c_S \prod_{i \in S} x_i$$

Lemma

For any pseudo-boolean function f with integer coefficients and $c_\emptyset = 0$, there exists an instance \mathcal{I} of MAX-LIN-AA such that $\max(f(x)) = \max \text{ excess of } \mathcal{I}$.

Related Results

- **Pseudo-boolean function:** a function $f : \{-1, +1\}^n \rightarrow \mathbb{R}$
- Suppose we know the Fourier expansion of $f(x)$

$$f(x) = \sum_{S \subseteq [n]} c_S \prod_{i \in S} x_i$$

Lemma

For any pseudo-boolean function f with integer coefficients and $c_\emptyset = 0$, there exists an instance \mathcal{I} of MAX-LIN-AA such that $\max(f(x)) = \max \text{ excess of } \mathcal{I}$.

Related Results

Lemma

For any pseudo-boolean function f with integer coefficients and $c_\emptyset = 0$, there exists an instance \mathcal{I} of MAX-LIN-AA such that $\max(f(x)) = \max \text{ excess of } \mathcal{I}$.

- **Proof:** For every $\emptyset \neq S \subseteq [n]$ with $c_S \neq 0$, construct equation $\sum_{i \in S} z_i = b_S$ with weight $|c_S|$, where $b_S = 0$ if c_S is positive and $b_S = 1$ if c_S is negative.

$$\begin{array}{rcl}
 & & z_1 = 0 \quad (w = 5) \\
 5x_1 - 3x_2x_3 + x_1x_2x_3 & \Rightarrow & z_2 + z_3 = 1 \quad (w = 3) \\
 & & z_1 + z_2 + z_3 = 0 \quad (w = 1)
 \end{array}$$

- Let $z_i = 0$ if $x_i = 1$ and $z_i = 1$ if $x_i = -1$.
- $f(x) = \text{weight of positive terms} - \text{weight of negative terms} = \text{weight of satisfied equations} - \text{weight of falsified equations}$

Related Results

Lemma

For any pseudo-boolean function f with integer coefficients and $c_\emptyset = 0$, there exists an instance \mathcal{I} of MAX-LIN-AA such that $\max(f(x)) = \max \text{ excess of } \mathcal{I}$.

- **Proof:** For every $\emptyset \neq S \subseteq [n]$ with $c_S \neq 0$, construct equation $\sum_{i \in S} z_i = b_S$ with weight $|c_S|$, where $b_S = 0$ if c_S is positive and $b_S = 1$ if c_S is negative.

$$\begin{array}{rcl}
 & z_1 = 0 & (w = 5) \\
 5x_1 - 3x_2x_3 + x_1x_2x_3 & \Rightarrow & z_2 + z_3 = 1 \quad (w = 3) \\
 & z_1 + z_2 + z_3 = 0 & (w = 1)
 \end{array}$$

- Let $z_i = 0$ if $x_i = 1$ and $z_i = 1$ if $x_i = -1$.
- $f(x) = \text{weight of positive terms} - \text{weight of negative terms} = \text{weight of satisfied equations} - \text{weight of falsified equations}$

Related Results

Lemma

For any pseudo-boolean function f with integer coefficients and $c_\emptyset = 0$, there exists an instance \mathcal{I} of MAX-LIN-AA such that $\max(f(x)) = \max \text{ excess of } \mathcal{I}$.

- **Proof:** For every $\emptyset \neq S \subseteq [n]$ with $c_S \neq 0$, construct equation $\sum_{i \in S} z_i = b_S$ with weight $|c_S|$, where $b_S = 0$ if c_S is positive and $b_S = 1$ if c_S is negative.

$$\begin{array}{rcl}
 & z_1 = 0 & (w = 5) \\
 5x_1 - 3x_2x_3 + x_1x_2x_3 & \Rightarrow & z_2 + z_3 = 1 \quad (w = 3) \\
 & z_1 + z_2 + z_3 = 0 & (w = 1)
 \end{array}$$

- Let $z_i = 0$ if $x_i = 1$ and $z_i = 1$ if $x_i = -1$.
- $f(x) = \text{weight of positive terms} - \text{weight of negative terms} = \text{weight of satisfied equations} - \text{weight of falsified equations}$

Related Results

Lemma

For any pseudo-boolean function f with integer coefficients and $c_\emptyset = 0$, there exists an instance \mathcal{I} of MAX-LIN-AA such that $\max(f(x)) = \max$ excess of \mathcal{I} .

- **Proof:** For every $\emptyset \neq S \subseteq [n]$ with $c_S \neq 0$, construct equation $\sum_{i \in S} z_i = b_S$ with weight $|c_S|$, where $b_S = 0$ if c_S is positive and $b_S = 1$ if c_S is negative.

$$\begin{array}{rcl}
 & z_1 = 0 & (w = 5) \\
 5x_1 - 3x_2x_3 + x_1x_2x_3 & \Rightarrow & z_2 + z_3 = 1 \quad (w = 3) \\
 & z_1 + z_2 + z_3 = 0 & (w = 1)
 \end{array}$$

- Let $z_i = 0$ if $x_i = 1$ and $z_i = 1$ if $x_i = -1$.
- $f(x) = \text{weight of positive terms} - \text{weight of negative terms} = \text{weight of satisfied equations} - \text{weight of falsified equations}$

Related Results

Consider the following problem.

MAX- r -SAT parameterized above average (MAX- r -SAT-AA)

Instance: A CNF formula F with n variables, m clauses, such that each clause has r variables.

Parameter: k .

Question: Can we satisfy $\geq (1 - 1/2^r)m + k$ clauses?

- $(1 - 1/2^r)m$ is the expected number of clauses satisfied by a random assignment.

Related Results

- Can represent MAX- r -SAT-AA as a pseudo-boolean function, f .
- We can then transform f into an equivalent instance \mathcal{I} of MAX-LIN-AA in time $O^*(2^r)$ with required excess $k' = 2^r k$.
- $f(x)$ is of degree r .
- Therefore \mathcal{I} is an instance of MAX- r -LIN-AA.
- MAX- r -LIN-AA has a kernel with $(k' - 1)r$ variables
 \Rightarrow we can solve MAX- r -SAT-AA in time $O^*(2^{(2^r k - 1)r})$

Related Results

- Can represent MAX- r -SAT-AA as a pseudo-boolean function, f .
- We can then transform f into an equivalent instance \mathcal{I} of MAX-LIN-AA in time $O^*(2^r)$ with required excess $k' = 2^r k$.
- $f(x)$ is of degree r .
- Therefore \mathcal{I} is an instance of MAX- r -LIN-AA.
- MAX- r -LIN-AA has a kernel with $(k' - 1)r$ variables
 \Rightarrow we can solve MAX- r -SAT-AA in time $O^*(2^{(2^r k - 1)r})$

Related Results

- Can represent MAX- r -SAT-AA as a pseudo-boolean function, f .
- We can then transform f into an equivalent instance \mathcal{I} of MAX-LIN-AA in time $O^*(2^r)$ with required excess $k' = 2^r k$.
- $f(x)$ is of degree r .
- Therefore \mathcal{I} is an instance of MAX- r -LIN-AA.
- MAX- r -LIN-AA has a kernel with $(k' - 1)r$ variables
 \Rightarrow we can solve MAX- r -SAT-AA in time $O^*(2^{(2^r k - 1)r})$

Related Results

- Can represent MAX- r -SAT-AA as a pseudo-boolean function, f .
- We can then transform f into an equivalent instance \mathcal{I} of MAX-LIN-AA in time $O^*(2^r)$ with required excess $k' = 2^r k$.
- $f(x)$ is of degree r .
- Therefore \mathcal{I} is an instance of MAX- r -LIN-AA.
- MAX- r -LIN-AA has a kernel with $(k' - 1)r$ variables
 \Rightarrow we can solve MAX- r -SAT-AA in time $O^*(2^{(2^r k - 1)r})$

Related Results

- Can represent MAX- r -SAT-AA as a pseudo-boolean function, f .
- We can then transform f into an equivalent instance \mathcal{I} of MAX-LIN-AA in time $O^*(2^r)$ with required excess $k' = 2^r k$.
- $f(x)$ is of degree r .
- Therefore \mathcal{I} is an instance of MAX- r -LIN-AA.
- MAX- r -LIN-AA has a kernel with $(k' - 1)r$ variables
 \Rightarrow we can solve MAX- r -SAT-AA in time $O^*(2^{(2^r k - 1)r})$

- This approach can be extended to any boolean CSP where each constraint is on at most r variables.

MAX- r -CSP parameterized above average (MAX- r -CSP-AA)

Instance: A set V of n boolean variables, and a set \mathcal{C} of m constraints, where each constraint C is a boolean function acting on at most r variables of V .

Parameter: k .

Question: Can we satisfy $E + k$ constraints, where E is the expected number of constraints satisfied by a random assignment?

Theorem (Alon, Gutin, Kim, Szeider, Yeo (2010))

MAX- r -CSP-AA is FPT for fixed r .

More applications...

- In PERMUTATION-MAX- c -CSP, we are to find an *ordering* on a set of elements, and each constraint is a set of acceptable orderings for some subset of size $\leq r$.
- Gutin, van Iersel, Mnich, Yeo (2010) showed PERMUTATION-MAX-3-CSP-AA is FPT; Kim, Williams (2011) improve this to a linear kernel.

Theorem (Kim, Williams, 2011)

PERMUTATION-MAX-3-CSP-AA *has a kernel with less than $15k$ variables.*

Open Problem

- **Open questions:** Does MAX-LIN-AA have kernel with polynomial number of equations?

Thank you!

The End