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!
**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

\textbf{Max-Sat (‘Standard’ parameterization)}

\textit{Instance:} A CNF formula $F$ with $n$ variables, $m$ clauses.

\textit{Parameter:} $k$.

\textit{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 \texttt{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 $\text{YES}$; 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!
**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!
**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!
A better parameterization:

\textsc{Max-Sat} parameterized above $m/2$

\textit{Instance:} A CNF formula $F$ with $n$ variables, $m$ clauses.

\textit{Parameter:} $k$.

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

- In this case $k$ is smaller!
- And the problem becomes more interesting!
A better parameterization:

\[
\text{Max-Sat parameterized above } m/2
\]

\textbf{Instance:} A CNF formula } F \text{ with } n \text{ variables, } m \text{ clauses.}

\textbf{Parameter:} } k.

\textbf{Question:} Can we satisfy } \geq m/2 + k \text{ clauses?

• In this case } k \text{ is smaller!

• And the problem becomes more interesting!
A better parameterization:

\textsc{Max-Sat} parameterized above \( m/2 \)

\textbf{Instance}: A CNF formula \( F \) with \( n \) variables, \( m \) clauses.

\textbf{Parameter}: \( k \).

\textbf{Question}: Can we satisfy \( \geq m/2 + k \) clauses?

• In this case \( k \) is smaller!
• And the problem becomes more interesting!
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 \((\overline{x})\).
- 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.....
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 \((\overline{x})\).
- 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.....
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.....
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.....
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....
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 \((\overline{x})\).
- 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.....
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 $\phi m$ becomes a tight lower bound on the number of satisfied clauses, where $\phi = (\sqrt{5} - 1)/2 \approx 0.618$.
- Therefore there is a kernel.

**Proof:** If $k < (\phi - \frac{1}{2})m$ answer YES.
Otherwise $m \leq k/(\phi - \frac{1}{2})$.
This gives rise to a new problem.....
Parameterizing 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.
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.
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.
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.
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 \( I \) 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.

\[
\begin{align*}
z_1 &= 1 \\
z_1 + z_2 &= 0 \\
z_2 + z_3 &= 1 \\
z_1 + z_2 + z_3 &= 1
\end{align*}
\]

- **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.

\[
\begin{align*}
  z_1 &= 1 \\
  z_1 + z_2 &= 0 \\
  z_2 + z_3 &= 1 \\
  z_1 + z_2 + z_3 &= 1
\end{align*}
\]

**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.

\[
\begin{align*}
z_1 &= 1 \\
z_1 + z_2 &= 0 \\
z_2 + z_3 &= 1 \\
z_1 + z_2 + z_3 &= 1
\end{align*}
\]

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

\[
\begin{align*}
    z_1 &= 1 \\
    z_1 + z_2 &= 0 \\
    z_2 + z_3 &= 1 \\
    z_1 + z_2 + z_3 &= 1
\end{align*}
\]

- **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.

\[
\begin{align*}
    z_1 &= 1 \\
    z_1 + z_2 &= 0 \\
    z_2 + z_3 &= 1 \\
    z_1 + z_2 + z_3 &= 1
\end{align*}
\]

**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 $\max(I)$.
- e.g.

  \[
  \begin{align*}
  z_1 &= 0 \\
  z_1 &= 1 \\
  z_2 + z_3 &= 0 \\
  z_2 + z_3 &= 1 \\
  \ldots
  \end{align*}
  \]

**Theorem (Håstad, 2001)**

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

- $W/2$ is a **tight** lower bound on $\max(I)$.
- 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 $\text{Max-Lin}$ in which $\max(I) \leq (1/2 + \epsilon)m$, and instances in which $\max(I) \geq (1 - \epsilon)m$, unless $P = NP$. 
Let $\max(I)$ denote the maximum possible weight of satisfied equations in $I$.

**Max-Lin Above Average (Max-Lin-AA)**

**Instance:** A system $I$ of $m$ linear equations in $n$ variables over $F_2$, with total weight $W$.

**Parameter:** $k$.

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

Mahajan, Raman & Sikdar (2006) asked if Max-Lin-AA is FPT.
Let $\max(I)$ denote the maximum possible weight of satisfied equations in $I$.

**Max-Lin Above Average (Max-Lin-AA)**

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

*Parameter*: $k$.

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

- Mahajan, Raman & Sikdar (2006) asked if Max-Lin-AA is FPT.
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**

\(\text{Max-}r\text{-Lin}\) is equivalent to \(\text{Max-Lin}\) except all equations have at most \(r\) variables. \(\text{Max-Lin}\) and \(\text{Max-}r\text{-Lin}\) are important problems, for many reasons....

- Håstad said they were as basic as satisfiability.
- They are important tools for constraint satisfaction problems (such as \(\text{MaxSat}\) or \(\text{Max-}r\text{-Sat}\)).
- So \(\text{Max-Lin}\) and \(\text{Max-}r\text{-Lin}\) have attracted significant interest in algorithmics.
- A number of papers made progress on \(\text{Max-}r\text{-Lin-AA}\) before \(\text{Max-Lin-AA}\) 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:

$$z_1 = 1$$
$$z_2 = 1$$
$$z_1 + z_2 = 1$$
For a given assignment, the \textbf{excess} = total weight of satisfied equations – total weight of falsified equations.

\textbf{Max-Lin-AA} is equivalent to asking if the max excess is at least $2k$.

\textbf{Example:}

\begin{align*}
z_1 &= 1 \\
z_2 &= 1 \\
z_1 + z_2 &= 1
\end{align*}
### 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 \(w_1 - w_2\)).

<table>
<thead>
<tr>
<th>Original Equation</th>
<th>Weight</th>
<th>New Equation</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>(z_1 + z_2 = 1)</td>
<td>(w = 1)</td>
<td>(z_1 + z_2 = 1)</td>
<td>(w = 3)</td>
</tr>
<tr>
<td>(z_1 + z_2 = 1)</td>
<td>(w = 2)</td>
<td>(z_1 + z_2 = 1)</td>
<td>(w = 3)</td>
</tr>
<tr>
<td>(z_2 + z_3 + z_4 = 0)</td>
<td>(w = 3)</td>
<td>(z_2 + z_3 + z_4 = 0)</td>
<td>(w = 1)</td>
</tr>
<tr>
<td>(z_2 + z_3 + z_4 = 1)</td>
<td>(w = 2)</td>
<td>(z_2 + z_3 + z_4 = 1)</td>
<td>(w = 2)</td>
</tr>
</tbody>
</table>

- 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 $w_1 - w_2$).

\[
\begin{align*}
  z_1 + z_2 &= 1 \quad (w = 1) \quad \Rightarrow \quad z_1 + z_2 &= 1 \quad (w = 3) \\
  z_1 + z_2 &= 1 \quad (w = 2)
\end{align*}
\]

\[
\begin{align*}
  z_2 + z_3 + z_4 &= 0 \quad (w = 3) \quad \Rightarrow \quad z_2 + z_3 + z_4 &= 0 \quad (w = 1) \\
  z_2 + z_3 + z_4 &= 1 \quad (w = 2)
\end{align*}
\]

- 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 \( w_1 - w_2 \)).

\[
\begin{align*}
z_1 + z_2 &= 1 \\
z_1 + z_2 &= 1
\end{align*}
\]

\( (w = 1) \Rightarrow z_1 + z_2 = 1 \quad (w = 3) \)

\[
\begin{align*}
z_1 + z_2 &= 1 \\
z_1 + z_2 &= 1
\end{align*}
\]

\( (w = 2) \)

\[
\begin{align*}
z_2 + z_3 + z_4 &= 0 \\
z_2 + z_3 + z_4 &= 1
\end{align*}
\]

\( (w = 3) \Rightarrow z_2 + z_3 + z_4 = 0 \quad (w = 1) \)

\[
\begin{align*}
z_2 + z_3 + z_4 &= 0 \\
z_2 + z_3 + z_4 &= 1
\end{align*}
\]

\( (w = 2) \)

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 $w_1 - w_2$).

\[
\begin{align*}
  z_1 + z_2 &= 1 \quad (w = 1) \quad \Rightarrow \quad z_1 + z_2 = 1 \quad (w = 3) \\
  z_1 + z_2 &= 1 \quad (w = 2) \\
  z_2 + z_3 + z_4 &= 0 \quad (w = 3) \quad \Rightarrow \quad z_2 + z_3 + z_4 = 0 \quad (w = 1) \\
  z_2 + z_3 + z_4 &= 1 \quad (w = 2)
\end{align*}
\]

- 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 $a^{i_1}, \ldots, a^{i_t}$ of $A$ are linearly independent. Then delete all variables not in $\{z_{i_1}, \ldots, z_{i_t}\}$ from the equations of $S$.

\[
\begin{align*}
  z_1 + z_3 + z_4 &= 1 \\
  z_2 + z_3 + z_4 &= 0 \\
  z_2 + z_3 &= 0 \\
  z_1 + z_2 &= 1
\end{align*}
\]

\[
\begin{pmatrix}
  1 & 0 & 1 & 1 \\
  0 & 1 & 1 & 1 \\
  0 & 1 & 1 & 0 \\
  1 & 1 & 0 & 0
\end{pmatrix}
\Rightarrow
\begin{align*}
  z_1 + z_4 &= 1 \\
  z_2 + z_4 &= 0 \\
  z_2 &= 0 \\
  z_1 + z_2 &= 1
\end{align*}
\]
Reduction Rules

Reduction Rule (Rank rule)

Let $A$ be the matrix over $\mathbb{F}_2$ corresponding to the set of equations in $I$, such that $a_{ij} = 1$ if $z_i$ appears in equation $j$, and 0 otherwise. Let $t = \text{rank} A$ and suppose columns $a_{i1}, \ldots, a_{it}$ of $A$ are linearly independent. Then delete all variables not in $\{z_{i1}, \ldots, z_{it}\}$ from the equations of $S$.

\[
\begin{align*}
z_1 + z_3 + z_4 &= 1 \\
z_2 + z_3 + z_4 &= 0 \quad \Rightarrow \\
z_2 + z_3 &= 0 \\
z_1 + z_2 &= 1
\end{align*}
\]

\[
\begin{pmatrix}
1 & 0 & 1 & 1 \\
0 & 1 & 1 & 1 \\
0 & 1 & 1 & 0 \\
1 & 1 & 0 & 0
\end{pmatrix}
\]

\[
\begin{align*}
z_1 + z_4 &= 1 \\
z_2 + z_4 &= 0 \\
z_2 &= 0 \\
z_1 + z_2 &= 1
\end{align*}
\]
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 $a_{i_1}^{t}, \ldots, a_{i_t}^{t}$ of $A$ are linearly independent. Then delete all variables not in $\{z_{i_1}, \ldots, z_{i_t}\}$ from the equations of $S$.

\[
\begin{align*}
z_1 + z_3 + z_4 &= 1 \\
z_2 + z_3 + z_4 &= 0 \\z_2 + z_3 &= 0 \\
z_1 + z_2 &= 1
\end{align*}
\Rightarrow
\begin{bmatrix}
1 & 0 & 1 & 1 \\
0 & 1 & 1 & 1 \\
0 & 1 & 1 & 0 \\
1 & 1 & 0 & 0
\end{bmatrix}
\Rightarrow
\begin{align*}
z_1 + z_4 &= 1 \\
z_2 + z_4 &= 0 \\
z_2 &= 0 \\
z_1 + z_2 &= 1
\end{align*}
\]
**Reduction Rules**

Why does the **Rank Rule** work?

\[ \mathcal{I} \]

\[
\begin{align*}
  z_1 + z_3 + z_4 &= 1 \\
  z_2 + z_3 + z_4 &= 0 \\
  z_2 + z_3 &= 0 \\
  z_1 + z_2 &= 1
\end{align*}
\]

\[ \mathcal{I}' \]

\[
\begin{pmatrix}
  1 & 0 & 1 & 1 \\
  0 & 1 & 1 & 1 \\
  0 & 1 & 1 & 0 \\
  1 & 1 & 0 & 0
\end{pmatrix}
\]

\[
\begin{align*}
  z_1 + z_4 &= 1 \\
  z_2 + z_4 &= 0 \\
  z_2 &= 0 \\
  z_1 + z_2 &= 1
\end{align*}
\]

- 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.
Why does the Rank Rule work?

\[
\begin{align*}
\mathcal{I} & \quad \mathcal{I}' \\
\text{z}_1 + \text{z}_3 + \text{z}_4 &= 1 & \text{z}_1 + \text{z}_4 &= 1 \\
\text{z}_2 + \text{z}_3 + \text{z}_4 &= 0 & \Rightarrow & \quad \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix} & \Rightarrow & \quad \text{z}_2 + \text{z}_4 &= 0 \\
\text{z}_2 + \text{z}_3 &= 0 & \Rightarrow & \quad \text{z}_2 &= 0 \\
\text{z}_1 + \text{z}_2 &= 1 & \Rightarrow & \quad \text{z}_1 + \text{z}_2 &= 1
\end{align*}
\]

- 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.
Why does the Rank Rule work?

\[ I \]
\[
\begin{align*}
z_1 + z_3 + z_4 &= 1 \\
z_2 + z_3 + z_4 &= 0 \\
z_2 + z_3 &= 0 \\
z_1 + z_2 &= 1
\end{align*}
\]

\[ I' \]
\[
\begin{align*}
z_1 + z_4 &= 1 \\
z_2 + z_4 &= 0 \\
z_2 &= 0 \\
z_1 + z_2 &= 1
\end{align*}
\]

- Set \( z_3 = 0 \) and add a solution for \( I' \) to get a solution of equal weight for \( I \).
- Consider a solution for \( I \).
  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 \( I' \) of equal weight.
Reduction Rules

Why does the Rank Rule work?

\[ I \]

\[
\begin{align*}
z_1 + z_3 + z_4 &= 1 \\
z_2 + z_3 + z_4 &= 0 \\
z_2 + z_3 &= 0 \\
z_1 + z_2 &= 1
\end{align*}
\]

\[ I' \]

\[
\begin{pmatrix}
1 & 0 & 1 & 1 \\
0 & 1 & 1 & 1 \\
0 & 1 & 1 & 0 \\
1 & 1 & 0 & 0
\end{pmatrix}
\]

\[
\begin{align*}
z_1 + z_4 &= 1 \\
z_2 + z_4 &= 0 \\
z_2 &= 0 \\
z_1 + z_2 &= 1
\end{align*}
\]

- Set \( z_3 = 0 \) and add a solution for \( I' \) to get a solution of equal weight for \( I \).
- Consider a solution for \( I \).
  - 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 \( I' \) of equal weight.

Anders Yeo
Max-Lin Parameterized Above Average
Reduction Rules

Why does the Rank Rule work?

\[ I \]
\[ z_1 + z_3 + z_4 = 1 \]
\[ z_2 + z_3 + z_4 = 0 \quad \Rightarrow \quad \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix} \quad \Rightarrow \quad z_2 + z_4 = 0 \]
\[ z_2 + z_3 = 0 \]
\[ z_1 + z_2 = 1 \]
\[ I' \]
\[ z_1 + z_4 = 1 \]
\[ z_2 + z_4 = 0 \]
\[ z_2 = 0 \]
\[ z_1 + z_2 = 1 \]

- Set \( z_3 = 0 \) and add a solution for \( I' \) to get a solution of equal weight for \( I \).
- Consider a solution for \( I \).
  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 \( I' \) 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.

\[
\begin{align*}
    x_1 &= 1 \\
    x_2 &= 1 \\
    x_1 + x_2 &= 1 \\
    x_3 &= 1 \\
    x_1 + x_3 &= 1 \\
    x_2 + x_3 &= 1 \\
    x_1 + x_2 + x_3 &= 1
\end{align*}
\]
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.

\[
\begin{align*}
  x_1 &= 1 \\
  x_2 &= 1 \\
  x_1 + x_2 &= 1 \\
  x_3 &= 1 \\
  x_1 + x_3 &= 1 \\
  x_2 + x_3 &= 1 \\
  x_1 + x_2 + x_3 &= 1
\end{align*}
\]
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.

\[
\begin{align*}
  x_1 &= 1 \\
  x_2 &= 1 \\
  x_1 + x_2 &= 1 \\
  x_3 &= 1 \\
  x_1 + x_3 &= 1 \\
  x_2 + x_3 &= 1 \\
  x_1 + x_2 + x_3 &= 1
\end{align*}
\]
What we would like to show: For reduced instances, if $m$ is large enough the answer is \textit{YES}.

Sadly this is not true...

Consider a 'complete' system on $n$ variables with all RHS = 1.

\begin{align*}
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
\end{align*}
Reduction rule

- What we would like to show: For reduced instances, if $m$ is large enough the answer is \text{YES}.
- Sadly this is not true...
- Consider a 'complete' system on $n$ variables with all RHS $= 1$.
  \[
  \begin{align*}
  x_1 &= 1 \\
  x_2 &= 1 \\
  x_2 &= 0 \\
  x_3 &= 1 \\
  x_3 &= 0 \\
  x_2 + x_3 &= 1 \\
  x_2 + x_3 &= 0
  \end{align*}
  \]
- The maximum excess is 1 but $m = 2^n - 1$. 
Theorem A: [Crowston, Fellows, Gutin, Jones, Rosamond, Thomasse, Yeo, 2011] \( \text{MAX-LIN-AA} \) can be solved in time \( O^*(n^{2k}) \).

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

The above results can be combined to show the following

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

\( \text{MAX-LIN-AA} \) 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)

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

\[ z_1 + z_3 + z_5 = 1 \quad \Rightarrow \quad z_1 = z_3 + z_5 + 1 \]

\[ z_2 + z_3 = 1 \quad \Rightarrow \quad z_2 + z_3 = 1 \]
\[ z_1 + z_2 = 0 \quad \Rightarrow \quad z_3 + z_5 + 1 + z_2 = 0 \quad \Rightarrow \quad z_2 + z_3 + z_5 = 1 \]
\[ z_3 + z_4 + z_5 = 1 \quad \Rightarrow \quad z_3 + z_4 + z_5 = 1 \]
\[ z_1 + z_4 = 0 \quad \Rightarrow \quad z_3 + z_5 + 1 + z_4 = 0 \quad \Rightarrow \quad z_3 + z_4 + z_5 = 1 \]
\[ z_1 + z_2 + z_5 = 1 \quad \Rightarrow \quad z_3 + z_5 + 1 + z_2 + z_5 = 1 \quad \Rightarrow \quad z_2 + z_3 = 0 \]

Now we simplify......
Example

\[ z_1 + z_3 + z_5 = 1 \quad \Rightarrow \quad z_1 = z_3 + z_5 + 1 \]

\[ z_2 + z_3 = 1 \]
\[ z_1 + z_2 = 0 \quad \Rightarrow \quad z_3 + z_5 + 1 + z_2 = 0 \quad \Rightarrow \quad z_2 + z_3 + z_5 = 1 \]
\[ z_3 + z_4 + z_5 = 1 \]
\[ z_1 + z_4 = 0 \quad \Rightarrow \quad z_3 + z_5 + 1 + z_4 = 0 \quad \Rightarrow \quad z_3 + z_4 + z_5 = 1 \]
\[ z_1 + z_2 + z_5 = 1 \quad \Rightarrow \quad z_3 + z_5 + 1 + z_2 + z_5 = 1 \quad \Rightarrow \quad z_2 + z_3 = 0 \]

Now we simplify......
Example

\[ z_1 + z_3 + z_5 = 1 \quad \Rightarrow \quad z_1 = z_3 + z_5 + 1 \]

\[ z_2 + z_3 = 1 \]
\[ z_1 + z_2 = 0 \quad \Rightarrow \quad z_3 + z_5 + 1 + z_2 = 0 \]
\[ z_3 + z_4 + z_5 = 1 \]
\[ z_1 + z_4 = 0 \quad \Rightarrow \quad z_3 + z_5 + 1 + z_4 = 0 \]
\[ z_1 + z_2 + z_5 = 1 \quad \Rightarrow \quad z_3 + z_5 + 1 + z_2 + z_5 = 1 \]

Now we simplify......
Example

\[ z_1 + z_3 + z_5 = 1 \quad \Rightarrow \quad z_1 = z_3 + z_5 + 1 \]

\[ z_2 + z_3 = 1 \quad \Rightarrow \quad z_2 + z_3 = 1 \]
\[ z_1 + z_2 = 0 \quad \Rightarrow \quad z_3 + z_5 + 1 + z_2 = 0 \quad \Rightarrow \quad z_2 + z_3 + z_5 = 1 \]
\[ z_3 + z_4 + z_5 = 1 \quad \Rightarrow \quad z_3 + z_4 + z_5 = 1 \]
\[ z_1 + z_4 = 0 \quad \Rightarrow \quad z_3 + z_5 + 1 + z_4 = 0 \quad \Rightarrow \quad z_3 + z_4 + z_5 = 1 \]
\[ z_1 + z_2 + z_5 = 1 \quad \Rightarrow \quad z_3 + z_5 + 1 + z_2 + z_5 = 1 \quad \Rightarrow \quad z_2 + z_3 = 0 \]

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 \(z_1 = z_3 + z_5 + 1\).....
Parameterizing above tight bounds: Example Max-Sat
Max-Lin-AA
FPT Results
Related Results

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 $e_1, \ldots, e_n$ be a set of equations in $\mathcal{I}$ which are 'independent'.
  (LHSs correspond to independent rows in matrix $A$.)
  - Check unique assignment in which $e_1, \ldots, e_n$ all false. If this assignment achieves excess $2k$, return **YES**.
  - Otherwise, one of $e_1, \ldots, e_k$ 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 $e_1, \ldots, e_n$ be a set of equations in $I$ which are 'independent'.
  (LHSs correspond to independent rows in matrix $A$.)
- Check unique assignment in which $e_1, \ldots, e_n$ all false. If this assignment achieves excess $2k$, return $\text{YES}$.
- Otherwise, one of $e_1, \ldots, e_k$ must be true.
- Branch $n$ ways. In branch $i$ mark equation $e_i$ in Algorithm $H$ and solve resulting system.
- Since we can stop after $2k$ iterations of $H$, 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 $e_1, \ldots, e_n$ be a set of equations in $\mathcal{I}$ which are 'independent'. (LHSs correspond to independent rows in matrix $A$.)
- Check unique assignment in which $e_1, \ldots, e_n$ all false. If this assignment achieves excess $2k$, return YES.
- Otherwise, one of $e_1, \ldots, e_k$ 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 $e_1, \ldots, e_n$ be a set of equations in $\mathcal{I}$ which are 'independent'. (LHSs correspond to independent rows in matrix $A$.)
- Check unique assignment in which $e_1, \ldots, e_n$ all false. If this assignment achieves excess $2k$, return YES.
- Otherwise, one of $e_1, \ldots, e_k$ 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 $e_1, \ldots, e_n$ be a set of equations in $\mathcal{I}$ which are 'independent'. (LHSs correspond to independent rows in matrix $A$.)
  - Check unique assignment in which $e_1, \ldots, e_n$ all false. If this assignment achieves excess $2k$, return **YES**.
  - Otherwise, one of $e_1, \ldots, e_k$ 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 \textbf{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 \( M \).
Theorem B: [Crowston, Gutin, Jones, Kim, Ruzsa, 2010] If $I$ is reduced and $2k \leq m < 2^{n/2k}$, then $I$ is a $\text{YES}$-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 $\text{YES}$-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 $M$. 

Anders Yeo Max-Lin Parameterized Above Average
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 $e_1, \ldots, e_t$ 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 $e_1, \ldots, e_t$ 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 $e_1, \ldots, e_{i-1}$ 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: $e_i = f_i = e_{j_1} + e_{j_2} + \cdots + e_{j_a} + e'$ for some $\{j_1, \ldots, j_a\} \subseteq \{1, \ldots, i-1\}$ and $e'$ is any equation.
- However this implies that $e' = e_{j_1} + e_{j_2} + \cdots + e_{j_a} + e_i$, a contradiction.
Lemma View the LHSs of equations in $\mathcal{I}$ as a set $M$ of vectors in $\mathbb{F}_2^n$. Let $e_1, \ldots, e_t$ 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 $e_1, \ldots, e_t$ 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 $e_1, \ldots, e_{i-1}$ 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: $e_i = f_i = e_{j_1} + e_{j_2} + \cdots + e_{j_a} + e'$ for some $\{j_1, \ldots, j_a\} \subseteq \{1, \ldots, i-1\}$ and $e'$ is any equation.
- However this implies that $e' = e_{j_1} + e_{j_2} + \cdots + e_{j_a} + e_i$, a contradiction.
Proof of Theorem B

Lemma View the LHSs of equations in $I$ as a set $M$ of vectors in $\mathbb{F}_2^n$. Let $e_1, \ldots, e_t$ be a set of equations in $I$ that correspond to an $M$-sum-free set of vectors. Then we can run algorithm $\mathcal{H}$ for $t$ iterations, choosing equations $e_1, \ldots, e_t$ 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 $e_1, \ldots, e_{i-1}$ 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: $e_i = f_i = e_{j_1} + e_{j_2} + \cdots + e_{j_a} + e'$ for some $\{j_1, \ldots, j_a\} \subseteq \{1, \ldots, i-1\}$ and $e'$ is any equation.
- However this implies that $e' = e_{j_1} + e_{j_2} + \cdots + e_{j_a} + e_i$, 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 $e_1, \ldots, e_t$ 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 $e_1, \ldots, e_t$ 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 $e_1, \ldots, e_{i-1}$ 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: $e_i = f_i = e_{j_1} + e_{j_2} + \cdots + e_{j_a} + e'$ for some $\{j_1, \ldots, j_a\} \subseteq \{1, \ldots, i-1\}$ and $e'$ is any equation.
- However this implies that $e' = e_{j_1} + e_{j_2} + \cdots + e_{j_a} + e_i$, a contradiction.
Proof of Theorem B

Lemma View the LHSs of equations in \( I \) as a set \( M \) of vectors in \( \mathbb{F}_2^n \). Let \( e_1, \ldots e_t \) be a set of equations in \( I \) that correspond to an \( M \)-sum-free set of vectors. Then we can run algorithm \( \mathcal{H} \) for \( t \) iterations, choosing equations \( e_1, \ldots e_t \) 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 \( e_1, \ldots, e_{i-1} \) 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: \( e_i = f_i = e_{j_1} + e_{j_2} + \cdots + e_{j_a} + e' \) for some \( \{j_1, \ldots, j_a\} \subseteq \{1, \ldots, i-1\} \) and \( e' \) is any equation.
- However this implies that \( e' = e_{j_1} + e_{j_2} + \cdots + e_{j_a} + e_i \), 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 \( \text{YES} \)-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 \( e_1, \ldots, e_{2k} \) be the equations corresponding to \( K \), and run algorithm \( \mathcal{H} \) marking \( e_1, \ldots, e_{2k} \) in turn.
- Then we get excess \( 2k \), so the answer is \( \text{YES} \).
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 $I$ is a YES-instance.

- Suppose $I$ 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 $I$.
- Find an $M$-sum-free subset $K$ of $M$ s.t. $|K| = 2k$.
- Let $e_1, \ldots, e_{2k}$ be the equations corresponding to $K$, and run algorithm $H$ marking $e_1, \ldots, e_{2k}$ 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 $\mathbb{F}^n_2$ such that $\text{span}(M) = \mathbb{F}^n_2$. 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 $I$ is a \textsc{Yes}-instance.

- Suppose $I$ is reduced and $2k \leq m \leq 2^{n/2k}$.
- Let $M$ be the set of vectors in $\mathbb{F}^n_2$ corresponding to LHSs of equations in $I$.
  - Find an $M$-sum-free subset $K$ of $M$ s.t. $|K| = 2k$.
  - Let $e_1, \ldots, e_{2k}$ be the equations corresponding to $K$, and run algorithm $\mathcal{H}$ marking $e_1, \ldots, e_{2k}$ in turn.
  - Then we get excess $2k$, so the answer is \textsc{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 $I$ is a \text{YES}-instance.

- Suppose $I$ 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 $I$.
- Find an $M$-sum-free subset $K$ of $M$ s.t. $|K| = 2k$.
- Let $e_1, \ldots, e_{2k}$ be the equations corresponding to $K$, and run algorithm $\mathcal{H}$ marking $e_1, \ldots, e_{2k}$ in turn.
- Then we get excess $2k$, so the answer is \text{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 $\text{YES}$-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 $e_1, \ldots, e_{2k}$ be the equations corresponding to $K$, and run algorithm $H$ marking $e_1, \ldots, e_{2k}$ in turn.
- Then we get excess $2k$, so the answer is $\text{YES}$. 
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 $e_1, \ldots, e_{2k}$ be the equations corresponding to $K$, and run algorithm $\mathcal{H}$ marking $e_1, \ldots, e_{2k}$ 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] $\text{MAX-LIN-AA}$ 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 $\text{YES}$-instance.
Proof of our main result

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

\textbf{Max-Lin-AA has a kernel with at most } O(k^2 \log k) \textbf{ variables.}

\textbf{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 \textbf{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:** $m \geq n^{2^k}$. Then using $O^*(n^{2^k})$ algorithm, can solve in polynomial time.
- **Case 2:** $2k \leq m \leq 2^{n/2^k}$. 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 $2^{n/2^k} < m < n^{2^k}$. 

Anders Yeo

Max-Lin Parameterized Above Average
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 $I$ be a reduced system.

Case 1: $m \geq n^{2k}$. Then using $O^*(n^{2k})$ algorithm, can solve in polynomial time.

Case 2: $2k \leq m \leq 2^{n/2^k}$. By earlier Theorem return YES.

Case 3: $m < 2k$. Since $I$ reduced by Rank Rule, $n \leq m$ so $n = O(k^2 \log k)$.

Only remaining case is $2^{n/2^k} < m < n^{2k}$. 
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 \(I\) 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 \(I\) 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 $n = O(k^2 \log k)$. 

Anders Yeo
Max-Lin Parameterized Above Average
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 $n = O(k^2 \log k)$. 

Anders Yeo
Max-Lin Parameterized Above Average
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 $n = O(k^2 \log k)$. 
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?
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)**

\textbf{Max-Lin-AA has a kernel with at most }\(O(k^2 \log k)\)\textbf{ 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 \textbf{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 $I$ 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
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 $n \leq (2k - 1)r$. 

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 $n \leq (2k - 1)r$. 

Anders Yeo 
Max-Lin Parameterized Above Average
I will not say much about \textsc{Max-\textit{r}-Lin-AA} (where equations have at most \textit{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 \textbf{(Max-}r\text{-Lin-AA)}

I will not say much about \texttt{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 \(m < (2k - 1)^264^r\).
- 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 \(n \leq (2k - 1)r\).
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 $n \leq (2k - 1)r$. 
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 $f : \{-1, +1\}^n \to \mathbb{R}$
- 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 $\max(f(x)) = \max$ 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 $I$ of \textsc{Max-Lin-AA} such that $\max(f(x)) = \max$ excess of $I$.***
Lemma

For any pseudo-boolean function $f$ with integer coefficients and $c_\emptyset = 0$, there exists an instance $\mathcal{I}$ of $\text{Max-Lin-AA}$ 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.

Let $z_i = 0$ if $x_i = 1$ and $z_i = 1$ if $x_i = -1$.

$$z_1 = 0 \quad (w = 5)$$

$$5x_1 - 3x_2x_3 + x_1x_2x_3 \quad \Rightarrow \quad z_2 + z_3 = 1 \quad (w = 3)$$

$$z_1 + z_2 + z_3 = 0 \quad (w = 1)$$
Related Results

Lemma

For any pseudo-boolean function $f$ with integer coefficients and $c_{\emptyset} = 0$, there exists an instance $\mathcal{I}$ of $\text{Max-Lin-AA}$ 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.

Let $z_i = 0$ if $x_i = 1$ and $z_i = 1$ if $x_i = -1$.

$5x_1 - 3x_2x_3 + x_1x_2x_3 \Rightarrow z_2 + z_3 = 1 \quad (w = 3)$

$z_1 + z_2 + z_3 = 0 \quad (w = 1)$

Anders Yeo

Max-Lin Parameterized Above Average
**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.

\[
5x_1 - 3x_2x_3 + x_1x_2x_3 \Rightarrow z_2 + z_3 = 1 \quad (w = 3)
\]

\[
z_1 + z_2 + z_3 = 0 \quad (w = 1)
\]

Let $z_i = 0$ if $x_i = 1$ and $z_i = 1$ if $x_i = -1$.

\[f(x) = \text{weight of positive terms} - \text{weight of negative terms} = \text{weight of satisfied equations} - \text{weight of falsified equations}\]
Lemma

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

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.

$$5x_1 - 3x_2x_3 + x_1x_2x_3 \Rightarrow z_2 + z_3 = 1 \quad (w = 3)$$
$$z_1 = 0 \quad (w = 5)$$
$$z_1 + z_2 + z_3 = 0 \quad (w = 1)$$

Let $z_i = 0$ if $x_i = 1$ and $z_i = 1$ if $x_i = -1$.

$f(x) = \text{weight of positive terms} - \text{weight of negative terms} = \text{weight of satisfied equations} - \text{weight of falsified equations}$
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.
Can represent $\text{Max-} r\text{-SAT-AA}$ as a pseudo-boolean function, $f$.

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

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

We can then transform $f$ into an equivalent instance $\mathcal{I}$ of $\text{Max-Lin-AA}$ 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 $\text{Max-}r\text{-Lin-AA}$.

$\text{Max-}r\text{-Lin-AA}$ has a kernel with $(k' - 1)r$ variables

$\Rightarrow$ we can solve $\text{Max-}r\text{-SAT-AA}$ in time $O^*(2^{(2^r k - 1)r})$
Can represent $\text{Max}-r$-$\text{SAT}$-AA as a pseudo-boolean function, $f$.

We can then transform $f$ into an equivalent instance $I$ of $\text{Max-Lin}$-AA in time $O^*(2^r)$ with required excess $k' = 2^r k$.

$f(x)$ is of degree $r$.

Therefore $I$ is an instance of $\text{Max}-r$-$\text{Lin}$-AA.

$\text{Max}-r$-$\text{Lin}$-AA has a kernel with $(k' - 1)r$ variables

$\implies$ we can solve $\text{Max}-r$-$\text{SAT}$-AA in time $O^*(2^{(2^r k - 1)r})$
Can represent $\text{MAX-}r\text{-SAT-AA}$ as a pseudo-boolean function, $f$.

We can then transform $f$ into an equivalent instance $\mathcal{I}$ of $\text{MAX-LIN-AA}$ 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 $\text{MAX-}r\text{-LIN-AA}$.

$\text{MAX-}r\text{-LIN-AA}$ has a kernel with $(k' - 1)r$ variables

$\Rightarrow$ we can solve $\text{MAX-}r\text{-SAT-AA}$ 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 $C$ 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 $r$. 
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 $\text{MAX-LIN-AA}$ have kernel with polynomial number of equations?
Thank you!

The End