A \((1 + \epsilon)\)-Approximation Algorithm for Partitioning Hypergraphs Using A New Algorithmic Version of the Lovász Local Lemma

Mohammad R. Salavatipour

Department of Computer Science
University of Toronto
mreza@cs.toronto.edu
What is the Local Lemma?
What is the Local Lemma?

Lovász Local Lemma: $\mathcal{A} = \{A_1, \ldots, A_n\}$ a set of random events, $A_i$ has probability at most $p$ and is mutually independent of all but at most $d$ other events. If $ep(d + 1) \leq 1$, where $e = 2.7182 \ldots$, then

$$\Pr(\overline{A}_1 \land \ldots \land \overline{A}_n) > 0.$$
What is the Local Lemma?

**Lovász Local Lemma:** $A = \{A_1, \ldots, A_n\}$ a set of random events, $A_i$ has probability at most $p$ and is mutually independent of all but at most $d$ other events. If $ep(d + 1) \leq 1$, where $e = 2.7182\ldots$, then

$$\Pr(\overline{A_1} \land \ldots \land \overline{A_n}) > 0.$$ 

Extremely powerful; has several applications in:

- Combinatorics and Graph Theory
**What is the Local Lemma?**

**Lovász Local Lemma:** \( \mathcal{A} = \{A_1, \ldots, A_n\} \) a set of random events, \( A_i \) has probability at most \( p \) and is mutually independent of all but at most \( d \) other events. If \( ep(d + 1) \leq 1 \), where \( e = 2.7182 \ldots \), then

\[
\Pr(\overline{A}_1 \land \ldots \land \overline{A}_n) > 0.
\]

Extremely powerful; has several applications in:

- Combinatorics and Graph Theory
- Packet routing problems
What is the Local Lemma?

**Lovász Local Lemma:** $\mathcal{A} = \{A_1, \ldots, A_n\}$ a set of random events, $A_i$ has probability at most $p$ and is mutually independent of all but at most $d$ other events. If $ep(d + 1) \leq 1$, where $e = 2.7182\ldots$, then

$$\Pr(\overline{A}_1 \land \ldots \land \overline{A}_n) > 0.$$ 

Extremely powerful; has several applications in:

- Combinatorics and Graph Theory
- Packet routing problems
- Job shop scheduling
What is the Local Lemma?

**Lovász Local Lemma**: \( \mathcal{A} = \{A_1, \ldots, A_n\} \) a set of random events, \( A_i \) has probability at most \( p \) and is mutually independent of all but at most \( d \) other events. If \( ep(d + 1) \leq 1 \), where \( e = 2.7182 \ldots \), then

\[
\Pr(\overline{A_1} \land \ldots \land \overline{A_n}) > 0.
\]

Extremely powerful; has several applications in:

- Combinatorics and Graph Theory
- Packet routing problems
- Job shop scheduling
- Finding disjoint paths in expander graphs
The most typical example of applications of the LLL is:

**Theorem 1:** If $H(V, E)$ is a $k$-uniform hypergraph and each edge intersects less than $\frac{2^{k-1}}{e}$ other edges, then $H$ is 2-colorable.
The most typical example of applications of the LLL is:

**Theorem 1:** If $H(V, E)$ is a $k$-uniform hypergraph and each edge intersects less than $\frac{2^{k-1}}{e}$ other edges, then $H$ is 2-colorable.

**Proof:**

- Color $V$ with $\{\text{Red}, \text{Blue}\}$ uniformly randomly.
The most typical example of applications of the LLL is:

**Theorem 1:** If $H(V, E)$ is a $k$-uniform hypergraph and each edge intersects less than $\frac{2^{k-1}}{e}$ other edges, then $H$ is 2-colorable.

**Proof:**

- Color $V$ with $\{Red, Blue\}$ uniformly randomly.
- Let $A_i$ be the event that $E_i$ is monochromatic.
The most typical example of applications of the LLL is:

**Theorem 1:** If $H(V, E)$ is a $k$-uniform hypergraph and each edge intersects less than $\frac{2^{k-1}}{e}$ other edges, then $H$ is 2-colorable.

**Proof:**

- Color $V$ with $\{\text{Red, Blue}\}$ uniformly randomly.

- Let $A_i$ be the event that $E_i$ is monochromatic.

- $\Pr(A_i) = \frac{2}{2^k} = 2^{(1-k)}$
The most typical example of applications of the LLL is:

**Theorem 1:** If $H(V, E)$ is a $k$-uniform hypergraph and each edge intersects less than $\frac{2^{k-1}}{e}$ other edges, then $H$ is 2-colorable.

**Proof:**

- Color $V$ with $\{Red, Blue\}$ uniformly randomly.
- Let $A_i$ be the event that $E_i$ is monochromatic.
- $\Pr(A_i) = \frac{2}{2^k} = 2^{(1-k)}$
- $ep(d + 1) \leq e2^{(1-k)}(\frac{2^{k-1}}{e}) = 1$
The most typical example of applications of the LLL is:

**Theorem 1:** If $H(V, E)$ is a $k$-uniform hypergraph and each edge intersects less than $\frac{2^{k-1}}{e}$ other edges, then $H$ is 2-colorable.

**Proof:**

- Color $V$ with $\{Red, Blue\}$ uniformly randomly.
- Let $A_i$ be the event that $E_i$ is monochromatic.
- $\Pr(A_i) = \frac{2}{2^k} = 2^{(1-k)}$
- $ep(d + 1) \leq e2^{(1-k)}\left(\frac{2^{k-1}}{e}\right) = 1$

There is a more general form of the LLL, by which we can show:
The most typical example of applications of the LLL is:

**Theorem 1:** If $H(V, E)$ is a $k$-uniform hypergraph and each edge intersects less than $\frac{2^{k-1}}{e}$ other edges, then $H$ is 2-colorable.

**Proof:**

- Color $V$ with $\{\text{Red, Blue}\}$ uniformly randomly.
- Let $A_i$ be the event that $E_i$ is monochromatic.
- $\Pr(A_i) = \frac{2}{2^k} = 2^{(1-k)}$
- $e^p(d + 1) \leq e^{2^{(1-k)}(\frac{2^{k-1}}{e})} = 1$

There is a more general form of the LLL, by which we can show:

**Theorem 2:** If $H(V, E)$ is non-uniform and each edge $E_i$ has size at least $3$ and intersects at most $2^{O(k)}$ other edges of size $k$, then $H$ is 2-colorable.
The only drawback of the LLL: *it is non-constructive*
• The only drawback of the LLL: *it is non-constructive*

• Furthermore, the probability guaranteed in LLL is exponentially small.
• The only drawback of the LLL: *it is non-constructive*

• Furthermore, the probability guaranteed in LLL is exponentially small.

• The method of conditional probability (by Erdős) doesn’t give polytime algorithm.
• The only drawback of the LLL: it is non-constructive

• Furthermore, the probability guaranteed in LLL is exponentially small.

• The method of conditional probability (by Erdős) doesn’t give polytime algorithm.

The first algorithmic version of the LLL:

**Theorem 3 [Beck 1991]:** If $H(V, E)$ is $k$-uniform and each edge intersects at most $d = 2^{ck}$ other edges, $c \leq \frac{1}{48}$, there is an algorithm which runs in $O(\text{Poly}(n, m))$ that finds a 2-coloring of $H$. 
The only drawback of the LLL: \textit{it is non-constructive}

Furthermore, the probability guaranteed in LLL is exponentially small.

The method of conditional probability (by Erdős) doesn’t give polytime algorithm.

The first algorithmic version of the LLL:

**Theorem 3 [Beck 1991]:** If $H(V, E)$ is $k$-uniform and each edge intersects at most $d = 2^{ck}$ other edges, $c \leq \frac{1}{48}$, there is an algorithm which runs in $O(Poly(n, m))$ that finds a 2-coloring of $H$.

- Alon [FOCS’91] gave parallel version of this theorem with $c \approx \frac{1}{500}$. 
• The only drawback of the LLL: it is non-constructive

• Furthermore, the probability guaranteed in LLL is exponentially small.

• The method of conditional probability (by Erdős) doesn’t give polytime algorithm.

The first algorithmic version of the LLL:

**Theorem 3 [Beck 1991]:** If $H(V, E)$ is $k$-uniform and each edge intersects at most $d = 2^{ck}$ other edges, $c \leq \frac{1}{48}$, there is an algorithm which runs in $O(\text{Poly}(n, m))$ that finds a 2-coloring of $H$.

• Alon [FOCS’91] gave parallel version of this theorem with $c \approx \frac{1}{500}$.

• Molloy & Reed [STOC’98] gave more general algorithmic version of the LLL, which applies to a wider range of applications.
• The only drawback of the LLL: *it is non-constructive*

• Furthermore, the probability guaranteed in LLL is exponentially small.

• The method of conditional probability (by Erdős) doesn’t give polytime algorithm.

The first algorithmic version of the LLL:

**Theorem 3 [Beck 1991]:** If $H(V, E)$ is $k$-uniform and each edge intersects at most $d = 2^{ck}$ other edges, $c \leq \frac{1}{48}$, there is an algorithm which runs in $O(Poly(n, m))$ that finds a 2-coloring of $H$.

• Alon [FOCS’91] gave parallel version of this theorem with $c \approx \frac{1}{500}$.

• Molloy & Reed [STOC’98] gave more general algorithmic version of the LLL, which applies to a wider range of applications.

• None of these algorithms work for the case that $H$ is non-uniform.
The first algorithmic version for *non-uniform* hypergraphs:

**Theorem 4 [Czumaj & Scheideler SODA’00]:** We can find a 2-coloring of a non-uniform $H(V, E)$, as long as no edge $e \in E$ intersects more than $O(|e|2^{O(k)})$ edges of size at most $k$. 
The first algorithmic version for non-uniform hypergraphs:

**Theorem 4 [Czumaj & Scheideler SODA’00]:** We can find a 2-coloring of a non-uniform $H(V, E)$, as long as no edge $e \in E$ intersects more than $O(|e|2^{O(k)})$ edges of size at most $k$.

**Problem:** Try to find a 2-coloring s.t. the number of Red and Blue vertices are almost equal. (*hypergraph partitioning* problem).
The first algorithmic version for non-uniform hypergraphs:

**Theorem 4 [Czumaj & Scheideler SODA’00]:** We can find a 2-coloring of a non-uniform $H(V, E)$, as long as no edge $e \in E$ intersects more than $O(|e|2^{O(k)})$ edges of size at most $k$.

**Problem:** Try to find a 2-coloring s.t. the number of Red and Blue vertices are almost equal. (*hypergraph partitioning* problem).

That is, if $R(E_i)$ ($B(E_i)$) is the number of Red (Blue) vertices in $E_i$:

$$\forall i : (1 - \epsilon)\frac{|E_i|}{2} \leq R(E_i) \leq (1 + \epsilon)\frac{|E_i|}{2}$$
The first algorithmic version for non-uniform hypergraphs:

**Theorem 4 [Czumaj & Scheideler SODA’00]:** We can find a 2-coloring of a non-uniform $H(V, E)$, as long as no edge $e \in E$ intersects more than $O(|e|2^{O(k)})$ edges of size at most $k$.

**Problem:** Try to find a 2-coloring s.t. the number of Red and Blue vertices are almost equal. (hypergraph partitioning problem).

That is, if $R(E_i)$ ($B(E_i)$) is the number of Red (Blue) vertices in $E_i$:

$$\forall i : (1 - \epsilon)\frac{|E_i|}{2} \leq R(E_i) \leq (1 + \epsilon)\frac{|E_i|}{2}$$

- One of the applications of this problem is in splitting expander graphs [Frieze & Molloy’00]
The first algorithmic version for *non-uniform* hypergraphs:

**Theorem 4 [Czumaj & Scheideler SODA’00]:** We can find a 2-coloring of a non-uniform \( H(V, E) \), as long as no edge \( e \in E \) intersects more than \( O(|e|2^{O(k)}) \) edges of size at most \( k \).

**Problem:** Try to find a 2-coloring s.t. the number of Red and Blue vertices are almost equal. (*hypergraph partitioning* problem).

That is, if \( R(E_i) \) (\( B(E_i) \)) is the number of Red (Blue) vertices in \( E_i \):

\[
\forall i : (1 - \epsilon)\frac{|E_i|}{2} \leq R(E_i) \leq (1 + \epsilon)\frac{|E_i|}{2}
\]

- One of the applications of this problem is in splitting expander graphs [Frieze & Molloy’00]
- Theorem 4 does not extend to hypergraph partitioning
For uniform hypergraphs:

**Theorem 5 [Beck 1991]**: If $H(V, E)$ is $k$-uniform and each edge intersects at most $O(2^{O(k)})$ other edges, then there is a polytime algorithm that finds a “partitioning” of $H$. 
For uniform hypergraphs:

**Theorem 5 [Beck 1991]:** If $H(V, E)$ is $k$-uniform and each edge intersects at most $O(2^{O(k)})$ other edges, then there is a polytime algorithm that finds a "partitioning" of $H$.

**Theorem 6 [Czumaj & Scheideler STOC’00]:** We can find a partitioning of a "non-uniform" hypergraph, as long as no edge $e \in E$ intersects more than $O(2^{o(k)})$ edges of size at most $k$. 
For uniform hypergraphs:

**Theorem 5 [Beck 1991]:** If $H(V, E)$ is $k$-uniform and each edge intersects at most $O(2^{O(k)})$ other edges, then there is a polytime algorithm that finds a “partitioning” of $H$.

**Theorem 6 [Czumaj & Scheideler STOC’00]:** We can find a partitioning of a “non-uniform” hypergraph, as long as no edge $e \in E$ intersects more than $O(2^{o(k)})$ edges of size at most $k$.

We extend Theorem 6 to match Theorem 5 for non-uniform hypergraphs:
For uniform hypergraphs:

**Theorem 5 [Beck 1991]:** If $H(V, E)$ is $k$-uniform and each edge intersects at most $O(2^{O(k)})$ other edges, then there is a polytime algorithm that finds a “partitioning” of $H$.

**Theorem 6 [Czumaj & Scheideler STOC’00]:** We can find a partitioning of a “non-uniform” hypergraph, as long as no edge $e \in E$ intersects more than $O(2^{o(k)})$ edges of size at most $k$.

We extend Theorem 6 to match Theorem 5 for non-uniform hypergraphs:

**Theorem 7 [This talk]:** We can find a partitioning of a “non-uniform” hypergraph, as long as no edge $e \in E$ intersects more than $O(2^{O(k)})$ edges of size at most $k$. 
For uniform hypergraphs:

**Theorem 5 [Beck 1991]:** If $H(V, E)$ is $k$-uniform and each edge intersects at most $O(2^{O(k)})$ other edges, then there is a polytime algorithm that finds a “partitioning” of $H$.

**Theorem 6 [Czumaj & Scheideler STOC’00]:** We can find a partitioning of a “non-uniform” hypergraph, as long as no edge $e \in E$ intersects more than $O(2^{o(k)})$ edges of size at most $k$.

We extend Theorem 6 to match Theorem 5 for non-uniform hypergraphs:

**Theorem 7 [This talk]:** We can find a partitioning of a “non-uniform” hypergraph, as long as no edge $e \in E$ intersects more than $O(2^{O(k)})$ edges of size at most $k$.

- Both Theorems 6 and 7 are proved in more general settings.
For uniform hypergraphs:

**Theorem 5 [Beck 1991]:** If $H(V, E)$ is $k$-uniform and each edge intersects at most $O(2^{O(k)})$ other edges, then there is a polytime algorithm that finds a “partitioning” of $H$.

**Theorem 6 [Czumaj & Scheideler STOC’00]:** We can find a partitioning of a “non-uniform” hypergraph, as long as no edge $e \in E$ intersects more than $O(2^{o(k)})$ edges of size at most $k$.

We extend Theorem 6 to match Theorem 5 for non-uniform hypergraphs:

**Theorem 7 [This talk]:** We can find a partitioning of a “non-uniform” hypergraph, as long as no edge $e \in E$ intersects more than $O(2^{O(k)})$ edges of size at most $k$.

- Both Theorems 6 and 7 are proved in more general settings.
- Algorithm is Randomized; Expected Running time is linear in size of $H$. 
For uniform hypergraphs:

**Theorem 5 [Beck 1991]:** If $H(V, E)$ is $k$-uniform and each edge intersects at most $O(2^{O(k)})$ other edges, then there is a polytime algorithm that finds a “partitioning” of $H$.

**Theorem 6 [Czumaj & Scheideler STOC’00]:** We can find a partitioning of a “non-uniform” hypergraph, as long as no edge $e \in E$ intersects more than $O(2^{o(k)})$ edges of size at most $k$.

We extend Theorem 6 to match Theorem 5 for non-uniform hypergraphs:

**Theorem 7 [This talk]:** We can find a partitioning of a “non-uniform” hypergraph, as long as no edge $e \in E$ intersects more than $O(2^{O(k)})$ edges of size at most $k$.

- Both Theorems 6 and 7 are proved in more general settings.
- Algorithm is Randomized; Expected Running time is linear in size of $H$.
- Algorithm is simple; proof of correctness is too complicated to present here.
Goal: given a non-uniform hypergraph $H(V, E)$, find a 2-coloring of $V$ with \{R, B\}, s.t.

$$\forall E_i : (1 - 6\epsilon) \frac{|E_i|}{2} \leq R(E_i) \leq (1 + 6\epsilon) \frac{|E_i|}{2}$$
Goal: given a non-uniform hypergraph $H(V, E)$, find a 2-coloring of $V$ with $\{R, B\}$, s.t.

$$\forall E_i : (1 - 6\epsilon)\frac{|E_i|}{2} \leq R(E_i) \leq (1 + 6\epsilon)\frac{|E_i|}{2}$$

**High Level Algorithm**

- Color each vertex uniformly at random with Red/Blue.
Goal: given a non-uniform hypergraph \( H(V, E) \), find a 2-coloring of \( V \) with \( \{R, B\} \), s.t.

\[
\forall E_i: (1 - 6\epsilon)\frac{|E_i|}{2} \leq R(E_i) \leq (1 + 6\epsilon)\frac{|E_i|}{2}
\]

**High Level Algorithm**

- Color each vertex uniformly at random with Red/Blue.
- We may break each edge \( E_i \) into 3 smaller edges: \( E_i^1, E_i^2, E_i^3 \).
Goal: given a non-uniform hypergraph $H(V, E)$, find a 2-coloring of $V$ with \{R, B\}, s.t.

\[ \forall E_i : (1 - 6\epsilon)\frac{|E_i|}{2} \leq R(E_i) \leq (1 + 6\epsilon)\frac{|E_i|}{2} \]

**High Level Algorithm**

- Color each vertex uniformly at random with Red/Blue.
- We may break each edge $E_i$ into 3 smaller edges: $E_i^1$, $E_i^2$, $E_i^3$.
- Edge $E_i^j$ is *bad* if the difference of Red/Blue vertices in $E_i^j$ is $\geq \epsilon\frac{|E_i|}{2}$.
Goal: given a non-uniform hypergraph $H(V, E)$, find a 2-coloring of $V$ with \{R, B\}, s.t.

$$\forall E_i : (1 - 6\epsilon) \frac{|E_i|}{2} \leq R(E_i) \leq (1 + 6\epsilon) \frac{|E_i|}{2}$$

**High Level Algorithm**

- Color each vertex uniformly at random with Red/Blue.
- We may break each edge $E_i$ into 3 smaller edges: $E_i^1, E_i^2, E_i^3$.
- Edge $E_i^j$ is *bad* if the difference of Red/Blue vertices in $E_i^j$ is $\geq \epsilon \frac{|E_i|}{2}$.
- So, if $|E_i^j| \leq \epsilon \frac{|E_i|}{2}$, it cannot be bad, even if $E_i^j$ is monochromatic.
A \((1 + \epsilon)\)-Approximation Algorithm for Partitioning Hypergraphs

Goal: given a non-uniform hypergraph \(H(V, E)\), find a 2-coloring of \(V\) with \(\{R, B\}\), s.t.

\[
\forall E_i : (1 - 6\epsilon) \frac{|E_i|}{2} \leq R(E_i) \leq (1 + 6\epsilon) \frac{|E_i|}{2}
\]

High Level Algorithm

- Color each vertex uniformly at random with Red/Blue.
- We may break each edge \(E_i\) into 3 smaller edges: \(E^1_i, E^2_i, E^3_i\).
- Edge \(E^j_i\) is \textit{bad} if the difference of Red/Blue vertices in \(E^j_i\) is \(\geq \epsilon \frac{|E_i|}{2}\).
- So, if \(|E^j_i| \leq \epsilon \frac{|E_i|}{2}\), it cannot be bad, even if \(E^j_i\) is monochromatic.
- Find connected components of \textit{bad} edges.
Goal: given a non-uniform hypergraph $H(V, E)$, find a 2-coloring of $V$ with \{$R, B$\}, s.t.

$$\forall E_i : (1 - 6\epsilon)\frac{|E_i|}{2} \leq R(E_i) \leq (1 + 6\epsilon)\frac{|E_i|}{2}$$

**High Level Algorithm**

- Color each vertex uniformly at random with Red/Blue.
- We may break each edge $E_i$ into 3 smaller edges: $E^1_i, E^2_i, E^3_i$.
- Edge $E^j_i$ is **bad** if the difference of Red/Blue vertices in $E^j_i$ is $\geq \epsilon \frac{|E_i|}{2}$.
- So, if $|E^j_i| \leq \epsilon \frac{|E_i|}{2}$, it cannot be bad, even if $E^j_i$ is monochromatic.
- Find connected components of **bad** edges.
- Recolor the vertices of these components by exhaustive search, s.t. no bad remains. Such a coloring exists by the LLL.
Goal: given a non-uniform hypergraph $H(V, E)$, find a 2-coloring of $V$ with \{R, B\}, s.t.

$$\forall E_i : (1 - 6\epsilon)\frac{|E_i|}{2} \leq R(E_i) \leq (1 + 6\epsilon)\frac{|E_i|}{2}$$

**High Level Algorithm**

- Color each vertex uniformly at random with Red/Blue.
- We may break each edge $E_i$ into 3 smaller edges: $E_i^1$, $E_i^2$, $E_i^3$.
- Edge $E_i^j$ is **bad** if the difference of Red/Blue vertices in $E_i^j$ is $\geq \epsilon \frac{|E_i|}{2}$.
- So, if $|E_i^j| \leq \epsilon \frac{|E_i|}{2}$, it cannot be bad, even if $E_i^j$ is monochromatic.
- Find connected components of **bad** edges.
- Recolor the vertices of these components by exhaustive search, s.t. no bad remains. Such a coloring exists by the LLL.
- This 2-coloring satisfies:

$$\forall E_i : R(E_i) = R(E_i^1) + R(E_i^2) + R(E_i^3) \approx (1 \pm 3\epsilon)\frac{|E_i|}{2}$$
\textbullet The connected components of bad edges are called 1-components.
• The connected components of bad edges are called 1-components.
- The connected components of bad edges are called 1-components.
- Each edge that is not bad but is intersecting too many 1-components is *dangerous*. 
• The connected components of bad edges are called 1-components.

• Each edge that is not bad but is intersecting too many 1-components is **dangerous**.
• The connected components of bad edges are called 1-components.

• Each edge that is not bad but is intersecting too many 1-components is dangerous.

• Re-coloring 1-components that are intersecting a common dangerous edge may create a new bad edge.
The connected components of bad edges are called 1-components.

Each edge that is not bad but is intersecting too many 1-components is dangerous.

Re-coloring 1-components that are intersecting a common dangerous edge may create a new bad edge.

Therefore, we find maximal connected components of 1-components and dangerous edges; These are 2-components.
• The connected components of bad edges are called 1-components.

• Each edge that is not bad but is intersecting too many 1-components is *dangerous*.

• Re-coloring 1-components that are intersecting a common dangerous edge may create a new bad edge.

• Therefore, we find maximal connected components of 1-components and dangerous edges; These are 2-components.

• We can consider each 2-component independently.
• The connected components of bad edges are called 1-components.

• Each edge that is not bad but is intersecting too many 1-components is \textit{dangerous}.

• Re-coloring 1-components that are intersecting a common dangerous edge may create a new bad edge.

• Therefore, we find maximal connected components of 1-components and dangerous edges; These are 2-components.

• We can consider each 2-component independently.

• Using the LLL there exists a partitioning of the edges of 2-components.
• The connected components of bad edges are called 1-components.
• Each edge that is not bad but is intersecting too many 1-components is *dangerous*.
• Re-coloring 1-components that are intersecting a common dangerous edge may create a new bad edge.
• Therefore, we find maximal connected components of 1-components and dangerous edges; These are 2-components.
• We can consider each 2-component independently.
• Using the LLL there exists a partitioning of the edges of 2-components.
• With prob at least $1 - \frac{1}{m^\epsilon}$, no 1-component has size larger than $O(\log m)$.
The connected components of bad edges are called 1-components.

Each edge that is not bad but is intersecting too many 1-components is "dangerous."

Re-coloring 1-components that are intersecting a common dangerous edge may create a new bad edge.

Therefore, we find maximal connected components of 1-components and dangerous edges; These are 2-components.

We can consider each 2-component independently.

Using the LLL there exists a partitioning of the edges of 2-components.

With prob at least $1 - \frac{1}{m^\epsilon}$, no 1-component has size larger than $O(\log m)$.

We repeat the same procedure on the new 2-components; with high probability all 2-components will have size $O(\log \log m)$. 

\[ A (1 + \epsilon) - \text{Approximation Algorithm for Partitioning Hypergraphs} \]
Open problems

- All the known algorithms have a loss in exponent of dependencies.
Open problems

- All the known algorithms have a loss in exponent of dependencies.

Example:

- For a $k$-uniform hypergraph: a 2-coloring exists with $2^{k-3}$ dependencies.
- We can find a 2-coloring with only $2^{\frac{k}{16}}$ dependencies.
Open problems

- All the known algorithms have a loss in exponent of dependencies.

Example:

- For a $k$-uniform hypergraph: a 2-coloring exists with $2^{k-3}$ dependencies.
- We can find a 2-coloring with only $2^{k/16}$ dependencies.

Find an algorithm that finds a 2-coloring when the number of dependencies is $2^{k-O(1)}$. 
Open problems

- All the known algorithms have a loss in exponent of dependencies.

Example:

- For a $k$-uniform hypergraph: a 2-coloring exists with $2^{k-3}$ dependencies.
- We can find a 2-coloring with only $2^{k/16}$ dependencies.

Find an algorithm that finds a 2-coloring when the number of dependencies is $2^{k-O(1)}$.

- These algorithms work when the number of colors is $O(Polylog(m + n))$. What if not?
Open problems

- All the known algorithms have a loss in exponent of dependencies.

Example:

★ For a $k$-uniform hypergraph: a 2-coloring exists with $2^k - 3$ dependencies.
★ We can find a 2-coloring with only $2^{k/16}$ dependencies.

Find an algorithm that finds a 2-coloring when the number of dependencies is $2^k - O(1)$.

- These algorithms work when the number of colors is $O(Polylog(m + n))$. What if not?

- How about other problems that none of these algorithms apply directly?
Open problems

- All the known algorithms have a loss in exponent of dependencies.

Example:

- For a $k$-uniform hypergraph: a 2-coloring exists with $2^{k-3}$ dependencies.
- We can find a 2-coloring with only $2^{\frac{k}{16}}$ dependencies.

Find an algorithm that finds a 2-coloring when the number of dependencies is $2^{k-O(1)}$.

- These algorithms work when the number of colors is $O(\text{Polylog}(m + n))$. What if not?

- How about other problems that none of these algorithms apply directly?

- How about a completely different approach?