# Towards Perfect and Compact Symmetry Breaking: Computing Unavoidable Subgraphs Using SAT Solvers

Cuong Chau & Marijn Heule

{*ckcuong,marijn*}*@cs.utexas.edu*

Department of Computer Science

The University of Texas at Austin

November 9, 2015

# Outline

# Outline

# Introduction

Symmetry-breaking methods have been highly crucial for solving **Boolean satisfiability problems (SAT)** with large numbers of symmetries.

However, existing state-of-the-art symmetry-breaking methods, such as **shatter**, are not powerful enough to help SAT solvers successfully solving hard graph-related problems.

| $n$ | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|
| #graphs | $2^1$ | $2^3$ | $2^6$ | $2^{10}$ | $2^{15}$ | $2^{21}$ | $2^{28}$ |
| #isomorphism_classes | 2 | 4 | 11 | 34 | 156 | 1,044 | 12,346 |
| #graphs using **shatter** | 2 | 4 | 11 | 46 | 325 | 4,045 | 53,806 |

# Introduction

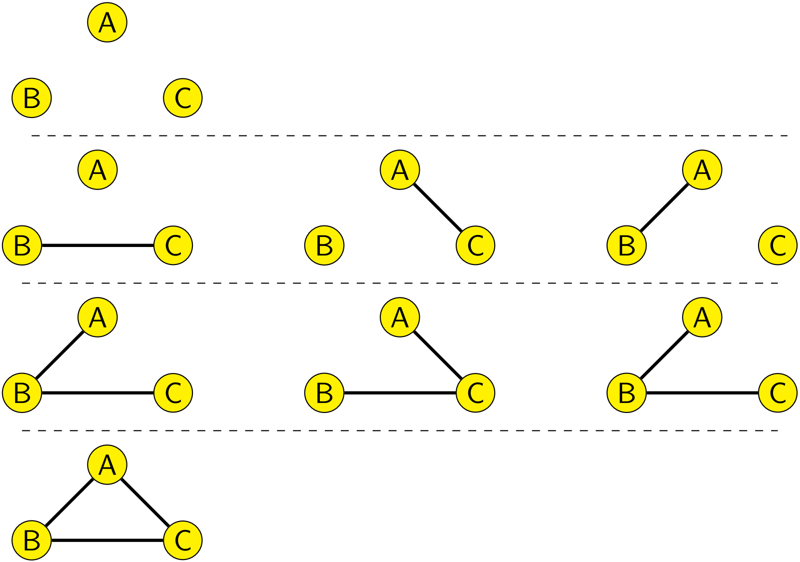
Symmetry-breaking methods have been highly crucial for solving **Boolean satisfiability problems (SAT)** with large numbers of symmetries.

However, existing state-of-the-art symmetry-breaking methods, such as **shatter**, are not powerful enough to help SAT solvers successfully solving hard graph-related problems.

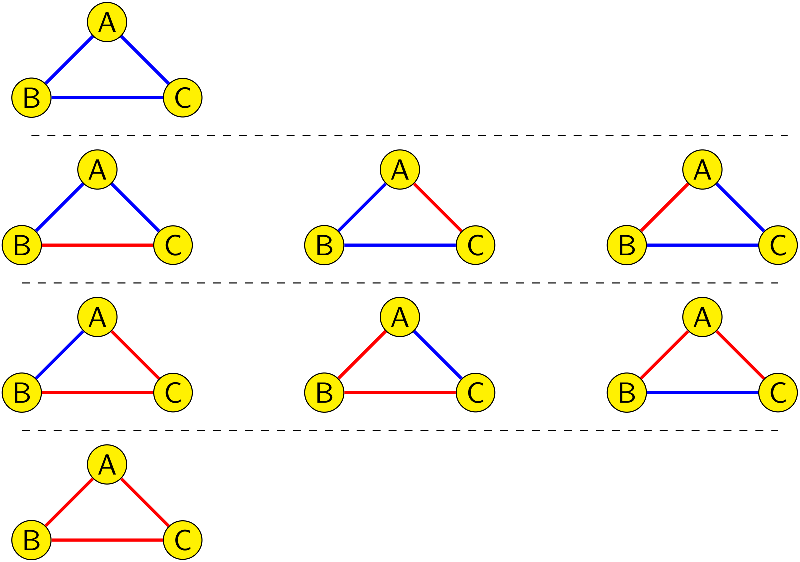
| $n$ | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|
| #graphs | $2^1$ | $2^3$ | $2^6$ | $2^{10}$ | $2^{15}$ | $2^{21}$ | $2^{28}$ |
| #isomorphism_classes | 2 | 4 | 11 | 34 | 156 | 1,044 | 12,346 |
| #graphs using **shatter** | 2 | 4 | 11 | 46 | 325 | 4,045 | 53,806 |

Yet, lots of symmetries are not broken by the existing symmetry-breaking methods.

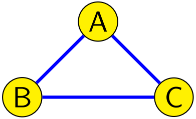
# Introduction



# Introduction

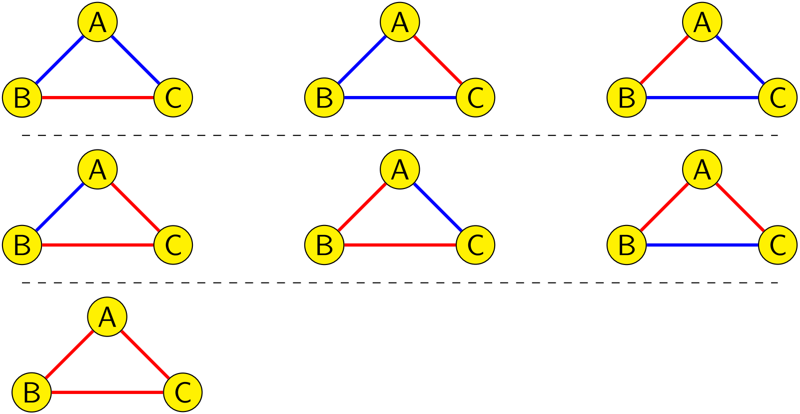


# Introduction

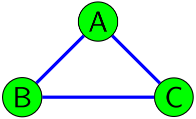



A path of two edges has the same color.

(Unavoidable subgraph)



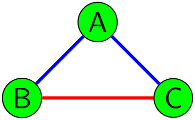
# Introduction

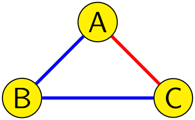


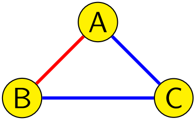

Convert an unavoidable subgraph to

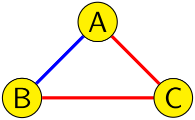
a constraint (an isolator):

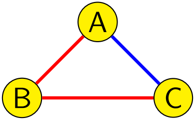
Path B-A-C has the same color.

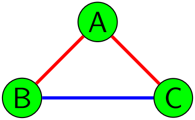


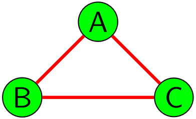



# Introduction

Let $E_G$ denote the edge set of a graph $G$,
$K_n$ denote the complete graph of order $n$.

$$|E_{K_n}| = \binom{n}{2}$$

# Introduction

Let $E_G$ denote the edge set of a graph $G$,
$K_n$ denote the complete graph of order $n$.

$$|E_{K_n}| = \binom{n}{2}$$

The number of all possible two-color edge-labelings of $K_n$ is $2^{|E_{K_n}|}$.

# Introduction

Let $E_G$ denote the edge set of a graph $G$,
$K_n$ denote the complete graph of order $n$.

$$|E_{K_n}| = \binom{n}{2}$$

The number of all possible two-color edge-labelings of $K_n$ is $2^{|E_{K_n}|}$.

Using a constraint derived from an unavoidable subgraph $G$ with $m$ connected components, the search space is reduced to $2^{|E_{K_n}|-(|E_G|-m)}$.

# Introduction

In this talk, I will focus on answering the following three questions:

1. What is an unavoidable subgraph?
2. How to efficiently compute unavoidable subgraphs of a given complete graph?
3. How to derive a constraint, say an isolator, from unavoidable subgraphs so that it can help to avoid examining isomorphic graphs?

# Outline

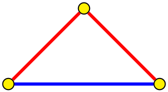
# Unavoidable Subgraph

**Definition:** An unavoidable subgraph $G$ of a fully-connected graph $K$ is a graph such that any red/blue edge-labeling of $K$ contains each connected component in $G$ either in red or blue.

# Unavoidable Subgraph

**Definition:** An unavoidable subgraph $G$ of a fully-connected graph $K$ is a graph such that any red/blue edge-labeling of $K$ contains each connected component in $G$ either in red or blue.

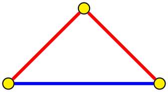
**Examples:** A path of two edges is an unavoidable subgraph of a complete graph of order 3.



# Unavoidable Subgraph

**Definition:** An unavoidable subgraph $G$ of a fully-connected graph $K$ is a graph such that any red/blue edge-labeling of $K$ contains each connected component in $G$ either in red or blue.

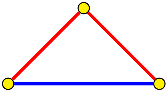
**Examples:** A path of two edges is an unavoidable subgraph of a complete graph of order 3. It is also the largest unavoidable subgraph (measured in the number of edges) of a complete graph of order 3.



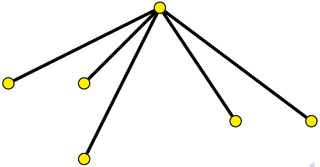
# Unavoidable Subgraph

**Definition:** An unavoidable subgraph $G$ of a fully-connected graph $K$ is a graph such that any red/blue edge-labeling of $K$ contains each connected component in $G$ either in red or blue.

**Examples:** A path of two edges is an unavoidable subgraph of a complete graph of order 3. It is also the largest unavoidable subgraph (measured in the number of edges) of a complete graph of order 3.



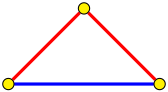
A triangle is an unavoidable subgraph of a complete graph of order 6.



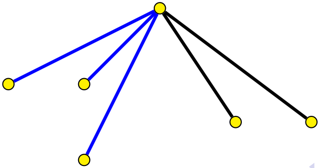
# Unavoidable Subgraph

**Definition:** An unavoidable subgraph $G$ of a fully-connected graph $K$ is a graph such that any red/blue edge-labeling of $K$ contains each connected component in $G$ either in red or blue.

**Examples:** A path of two edges is an unavoidable subgraph of a complete graph of order 3. It is also the largest unavoidable subgraph (measured in the number of edges) of a complete graph of order 3.

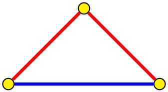


A triangle is an unavoidable subgraph of a complete graph of order 6.

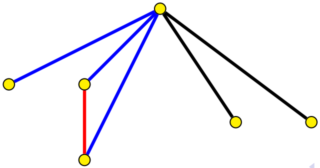


**Definition:** An unavoidable subgraph $G$ of a fully-connected graph $K$ is a graph such that any red/blue edge-labeling of $K$ contains each connected component in $G$ either in red or blue.

**Examples:** A path of two edges is an unavoidable subgraph of a complete graph of order 3. It is also the largest unavoidable subgraph (measured in the number of edges) of a complete graph of order 3.



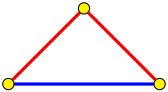
A triangle is an unavoidable subgraph of a complete graph of order 6.



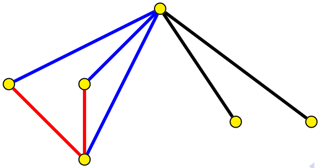
# Unavoidable Subgraph

**Definition:** An unavoidable subgraph $G$ of a fully-connected graph $K$ is a graph such that any red/blue edge-labeling of $K$ contains each connected component in $G$ either in red or blue.

**Examples:** A path of two edges is an unavoidable subgraph of a complete graph of order 3. It is also the largest unavoidable subgraph (measured in the number of edges) of a complete graph of order 3.



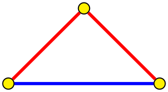
A triangle is an unavoidable subgraph of a complete graph of order 6.



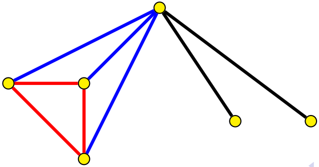
# Unavoidable Subgraph

**Definition:** An unavoidable subgraph $G$ of a fully-connected graph $K$ is a graph such that any red/blue edge-labeling of $K$ contains each connected component in $G$ either in red or blue.

**Examples:** A path of two edges is an unavoidable subgraph of a complete graph of order 3. It is also the largest unavoidable subgraph (measured in the number of edges) of a complete graph of order 3.

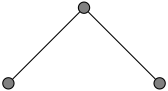


A triangle is an unavoidable subgraph of a complete graph of order 6.

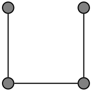


# Largest Connected Unavoidable Subgraph Examples

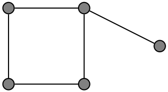
$K_3$ & $K_4$:



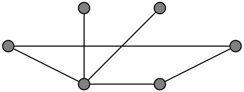
$K_5$:



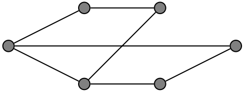
$K_6$:



$K_7$:



$K_8$:



# Unavoidable Subgraph

Many "nicely" structured unavoidable subgraphs have been studied. E.g., cliques (Ramsey numbers), cycles, paths, stars, trees, wheels, etc.

Our approach tries to compute a largest unavoidable subgraph for a given complete graph. The subgraph's structure doesn't have to be "nice".

# Unavoidable Subgraph

Many "nicely" structured unavoidable subgraphs have been studied. E.g., cliques (Ramsey numbers), cycles, paths, stars, trees, wheels, etc.

Our approach tries to compute a largest unavoidable subgraph for a given complete graph. The subgraph's structure doesn't have to be "nice".

Determining unavoidable subgraph problems can be encoded as SAT problems straightforwardly. Hence, they can be solved using SAT solvers.

# Unavoidable Subgraph

Many "nicely" structured unavoidable subgraphs have been studied. E.g., cliques (Ramsey numbers), cycles, paths, stars, trees, wheels, etc.

Our approach tries to compute a largest unavoidable subgraph for a given complete graph. The subgraph's structure doesn't have to be "nice".

Determining unavoidable subgraph problems can be encoded as SAT problems straightforwardly. Hence, they can be solved using SAT solvers. However, symmetries are the main obstacle for solving these problems using SAT solvers.

# Unavoidable Subgraph

Many “nicely” structured unavoidable subgraphs have been studied. E.g., cliques (Ramsey numbers), cycles, paths, stars, trees, wheels, etc.

Our approach tries to compute a largest unavoidable subgraph for a given complete graph. The subgraph’s structure doesn’t have to be “nice”.

Determining unavoidable subgraph problems can be encoded as SAT problems straightforwardly. Hence, they can be solved using SAT solvers. However, symmetries are the main obstacle for solving these problems using SAT solvers.

**Approach:** Start with computing unavoidable subgraphs for small graphs, which can be computed easily by SAT solvers. Then, these unavoidable subgraphs will be converted into an isolator for computing unavoidable subgraphs of bigger graphs. And so on.

# Outline

# SAT Encoding of Unavoidable Subgraphs

We employ a SAT solver to check whether a given graph $G$ of order $k$ is an unavoidable subgraph of a complete graph $K_n$ of order $n$ ($k \leq n$).

# SAT Encoding of Unavoidable Subgraphs

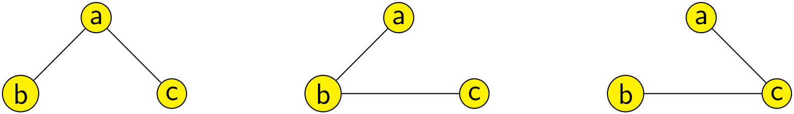
We employ a SAT solver to check whether a given graph $G$ of order $k$ is an unavoidable subgraph of a complete graph $K_n$ of order $n$ ($k \leq n$).

**Encoding:** Let's see how we encode the problem into SAT through the following example: *Check if a path of two edges is an unavoidable subgraph of* $K_3$*.*

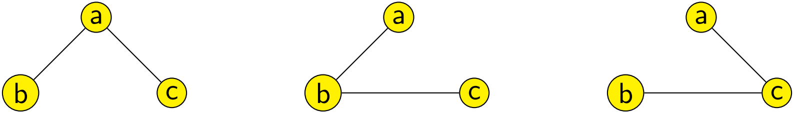
# SAT Encoding of Unavoidable Subgraphs

We employ a SAT solver to check whether a given graph $G$ of order $k$ is an unavoidable subgraph of a complete graph $K_n$ of order $n$ ($k \leq n$).

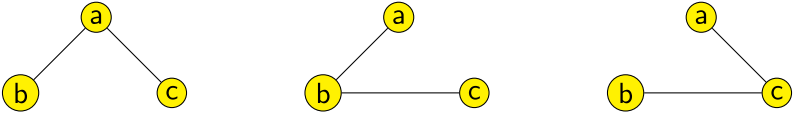
**Encoding:** Let's see how we encode the problem into SAT through the following example: *Check if a path of two edges is an unavoidable subgraph of* $K_3$.



# SAT Encoding of Unavoidable Subgraphs

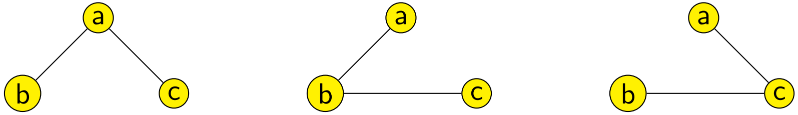


# SAT Encoding of Unavoidable Subgraphs




Let *ab*, *ac*, and *bc* be the Boolean variables representing the color of the edge connecting vertices *a* and *b*, *a* and *c*, and *b* and *c*, respectively. If a Boolean variable has value *T*, the corresponding edge has color red. Otherwise it has color blue.

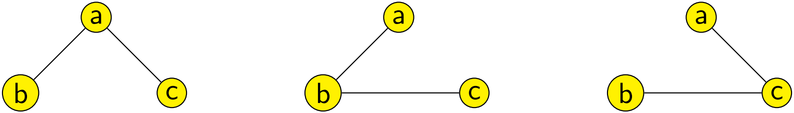
# SAT Encoding of Unavoidable Subgraphs




Let *ab*, *ac*, and *bc* be the Boolean variables representing the color of the edge connecting vertices *a* and *b*, *a* and *c*, and *b* and *c*, respectively. If a Boolean variable has value $T$, the corresponding edge has color red. Otherwise it has color blue.

$$(ab \wedge ac) \vee (\overline{ab} \wedge \overline{ac})$$

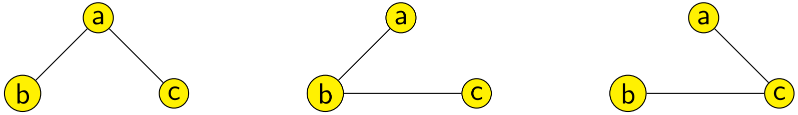
# SAT Encoding of Unavoidable Subgraphs




Let $ab$, $ac$, and $bc$ be the Boolean variables representing the color of the edge connecting vertices $a$ and $b$, $a$ and $c$, and $b$ and $c$, respectively. If a Boolean variable has value $T$, the corresponding edge has color red. Otherwise it has color blue.

$$(ab \wedge ac) \vee (\overline{ab} \wedge \overline{ac}) \quad (ab \wedge bc) \vee (\overline{ab} \wedge \overline{bc})$$

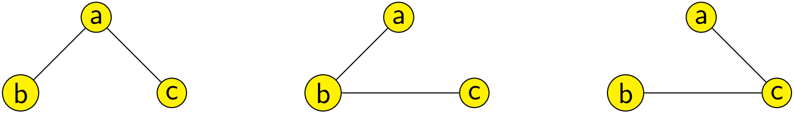
# SAT Encoding of Unavoidable Subgraphs




Let $ab$, $ac$, and $bc$ be the Boolean variables representing the color of the edge connecting vertices $a$ and $b$, $a$ and $c$, and $b$ and $c$, respectively. If a Boolean variable has value $T$, the corresponding edge has color red. Otherwise it has color blue.

$$(ab \land ac) \lor (\overline{ab} \land \overline{ac}) \quad (ab \land bc) \lor (\overline{ab} \land \overline{bc}) \quad (ac \land bc) \lor (\overline{ac} \land \overline{bc})$$

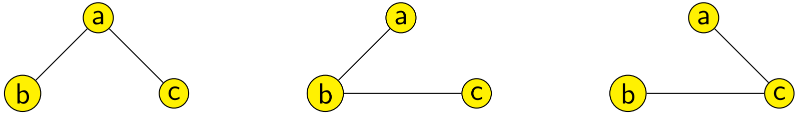
# SAT Encoding of Unavoidable Subgraphs




Let $ab$, $ac$, and $bc$ be the Boolean variables representing the color of the edge connecting vertices $a$ and $b$, $a$ and $c$, and $b$ and $c$, respectively. If a Boolean variable has value $T$, the corresponding edge has color red. Otherwise it has color blue.

$$(ab \wedge ac) \vee (\overline{ab} \wedge \overline{ac}) \vee (ab \wedge bc) \vee (\overline{ab} \wedge \overline{bc}) \vee (ac \wedge bc) \vee (\overline{ac} \wedge \overline{bc})$$

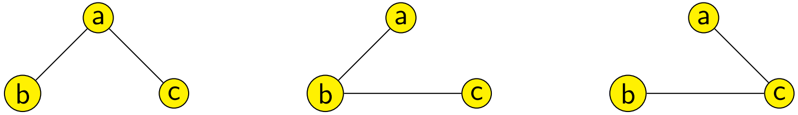
# SAT Encoding of Unavoidable Subgraphs




Let *ab*, *ac*, and *bc* be the Boolean variables representing the color of the edge connecting vertices *a* and *b*, *a* and *c*, and *b* and *c*, respectively. If a Boolean variable has value *T*, the corresponding edge has color red. Otherwise it has color blue.

$$\mathcal{F}_G = (ab \wedge ac) \vee (\overline{ab} \wedge \overline{ac}) \vee (ab \wedge bc) \vee (\overline{ab} \wedge \overline{bc}) \vee (ac \wedge bc) \vee (\overline{ac} \wedge \overline{bc})$$

# SAT Encoding of Unavoidable Subgraphs

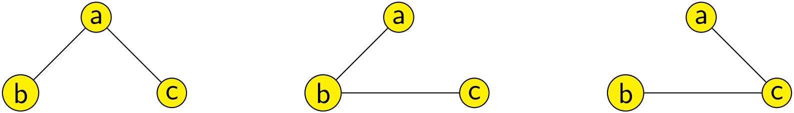



Let *ab*, *ac*, and *bc* be the Boolean variables representing the color of the edge connecting vertices *a* and *b*, *a* and *c*, and *b* and *c*, respectively. If a Boolean variable has value $T$, the corresponding edge has color red. Otherwise it has color blue.

$$\mathcal{F}_G = (ab \wedge ac) \vee (\overline{ab} \wedge \overline{ac}) \vee (ab \wedge bc) \vee (\overline{ab} \wedge \overline{bc}) \vee (ac \wedge bc) \vee (\overline{ac} \wedge \overline{bc})$$

$$\overline{\mathcal{F}_G} = (\overline{ab} \vee \overline{ac}) \wedge (ab \vee ac) \wedge (\overline{ab} \vee \overline{bc}) \wedge (ab \vee bc) \wedge (\overline{ac} \vee \overline{bc}) \wedge (ac \vee bc)$$

# SAT Encoding of Unavoidable Subgraphs



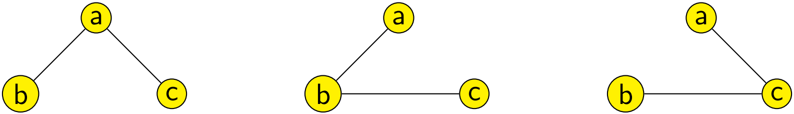

Let *ab*, *ac*, and *bc* be the Boolean variables representing the color of the edge connecting vertices *a* and *b*, *a* and *c*, and *b* and *c*, respectively. If a Boolean variable has value $T$, the corresponding edge has color red. Otherwise it has color blue.

$$\mathcal{F}_G = (ab \wedge ac) \vee (\overline{ab} \wedge \overline{ac}) \vee (ab \wedge bc) \vee (\overline{ab} \wedge \overline{bc}) \vee (ac \wedge bc) \vee (\overline{ac} \wedge \overline{bc})$$

$$\overline{\mathcal{F}_G} = (\overline{ab} \vee \overline{ac}) \wedge (ab \vee ac) \wedge (\overline{ab} \vee \overline{bc}) \wedge (ab \vee bc) \wedge (\overline{ac} \vee \overline{bc}) \wedge (ac \vee bc)$$

$\overline{\mathcal{F}_G}$ is UNSATISFIABLE $\Leftrightarrow$ $\mathcal{F}_G$ is VALID $\Leftrightarrow$ A path of two edges is an unavoidable subgraph of $K_3$.

# SAT Encoding of Unavoidable Subgraphs




Let $ab$, $ac$, and $bc$ be the Boolean variables representing the color of the edge connecting vertices $a$ and $b$, $a$ and $c$, and $b$ and $c$, respectively. If a Boolean variable has value $T$, the corresponding edge has color red. Otherwise it has color blue.

$$\mathcal{F}_G = (ab \wedge ac) \vee (\overline{ab} \wedge \overline{ac}) \vee (ab \wedge bc) \vee (\overline{ab} \wedge \overline{bc}) \vee (ac \wedge bc) \vee (\overline{ac} \wedge \overline{bc})$$

$$\overline{\mathcal{F}_G} = (\overline{ab} \vee \overline{ac}) \wedge (ab \vee ac) \wedge (\overline{ab} \vee \overline{bc}) \wedge (ab \vee bc) \wedge (\overline{ac} \vee \overline{bc}) \wedge (ac \vee bc)$$

$\overline{\mathcal{F}_G}$ is UNSATISFIABLE $\Leftrightarrow$ $\mathcal{F}_G$ is VALID $\Leftrightarrow$ A path of two edges is an unavoidable subgraph of $K_3$.

Since $\overline{\mathcal{F}_G}$ is in CNF format, SAT solvers can solve it directly.

# SAT Encoding of Unavoidable Subgraphs

**Encoding algorithm:** Construct $\overline{\mathcal{F}_G}$.

1. Compute the set $\mathcal{G}$ of all subgraphs of $K_n$ that are isomorphic to $G$.

**Encoding algorithm:** Construct $\overline{\mathcal{F}_G}$.

1. Compute the set $\mathcal{G}$ of all subgraphs of $K_n$ that are isomorphic to $G$.
2. Initiate $\overline{\mathcal{F}_G} = \varnothing$. For each $H \in \mathcal{G}$, compute:
3. $\mathcal{F}_r$ = disjunction of all positive literals representing the red color of the edges in $H$.
4. $\mathcal{F}_b$ = disjunction of all negative literals representing the blue color of the edges in $H$.
5. $\overline{\mathcal{F}_G} = \overline{\mathcal{F}_G} \wedge \mathcal{F}_r \wedge \mathcal{F}_b$.

# SAT Encoding of Unavoidable Subgraphs

**Encoding algorithm:** Construct $\overline{\mathcal{F}_G}$.

1. Compute the set $\mathcal{G}$ of all subgraphs of $K_n$ that are isomorphic to $G$.
2. Initiate $\overline{\mathcal{F}_G} = \emptyset$. For each $H \in \mathcal{G}$, compute:
3. $\mathcal{F}_r$ = disjunction of all positive literals representing the red color of the edges in $H$.
4. $\mathcal{F}_b$ = disjunction of all negative literals representing the blue color of the edges in $H$.
5. $\overline{\mathcal{F}_G} = \overline{\mathcal{F}_G} \wedge \mathcal{F}_r \wedge \mathcal{F}_b$.
6. Return $\overline{\mathcal{F}_G}$.

# Computing Isomorphic Graphs

**Inputs:**

1. An adjacency matrix $A_G$ representing a graph $G$ of order $k$.
2. $n$ (number of nodes of $K_n$).

**Output:**

A set $\mathcal{G}$ of all subgraphs of $K_n$ that are isomorphic to $G$.

# Computing Isomorphic Graphs

**Inputs:**

1. An adjacency matrix $A_G$ representing a graph $G$ of order $k$.
2. $n$ (number of nodes of $K_n$).

**Output:**

A set $\mathcal{G}$ of all subgraphs of $K_n$ that are isomorphic to $G$.

**Algorithm:**

1. $\mathcal{G} = \varnothing$. For each combination of $k$ nodes taken from $n$ nodes:
2. $\quad ACC = \varnothing$.
3. $\quad$ Compute all permutations $P$ of $k$ nodes. For each $P_i \in P$:
4. $\quad\quad$ Construct a graph $H$ by applying $A_G$ to $P_i$. $H$ is isomorphic to $G$.
5. $\quad\quad$ If $H \notin ACC$, then add $H$ to $ACC$.
6. $\quad \mathcal{G} = \mathcal{G} \cup ACC$.
7. Return $\mathcal{G}$.

# Computing Unavoidable Subgraphs of $k$ Nodes for $K_n$

The tool **nauty** can generate all non-isomorphic graphs of $k$ ($\leq 10$) nodes in adjacency matrix format very quickly.

When $n$ is small ($\leq 20$), we are able to compute all unavoidable subgraphs of order $k$ ($\leq 6$) automatically with the help of symmetry-breaking predicates generated from **shatter** (We haven't tried with subgraphs from 7 to 10 nodes). These experiments were performed on TACC (Texas Advanced Computing Center).

# Computing Unavoidable Subgraphs of $k$ Nodes for $K_n$

**Inputs:**

1. $k$ (number of nodes of $G$).
2. $n$ (number of nodes of $K_n$).

**Output:**

A set $\mathcal{H}$ of all non-isomorphic graphs of order $k$ that are unavoidable in $K_n$.

# Computing Unavoidable Subgraphs of $k$ Nodes for $K_n$

**Inputs:**

1. $k$ (number of nodes of $G$).
2. $n$ (number of nodes of $K_n$).

**Output:**

A set $\mathcal{H}$ of all non-isomorphic graphs of order $k$ that are unavoidable in $K_n$.

**Algorithm:**

1. Generate symmetry-breaking predicates $SB$ for $K_n$ using **shatter**.

2. Generate all non-isomorphic graphs of order $k$ in adjacency matrix format using the **nauty** package.

3. $\mathcal{H} = \varnothing$. For each graph $G$ generated in Step 2:

4. Construct $\overline{\mathcal{F}_G}$ (as described).

5. Check the satisfiability of $(\overline{\mathcal{F}_G} \wedge SB)$ using a SAT solver.

6. If $(\overline{\mathcal{F}_G} \wedge SB)$ is UNSATISFIABLE, then add $G$ to $\mathcal{H}$.

7. Return $\mathcal{H}$.

# Outline

# Deriving An Isolator from Unavoidable Subgraphs

Converting an unavoidable subgraph $G$ into an isolator by forcing all edges in each connected component of $G$ to have the same color.

$$e_1 \leftrightarrow e_2 \leftrightarrow e_3 \leftrightarrow \cdots \leftrightarrow e_m \equiv e_1 \rightarrow e_2 \rightarrow e_3 \rightarrow \cdots \rightarrow e_m \;(e_m \rightarrow e_1)$$

$$\equiv (\overline{e_1} \vee e_2) \wedge (\overline{e_2} \vee e_3) \wedge \ldots \wedge (\overline{e_m} \vee e_1)$$

# Deriving An Isolator from Unavoidable Subgraphs

Converting an unavoidable subgraph $G$ into an isolator by forcing all edges in each connected component of $G$ to have the same color.

$$e_1 \leftrightarrow e_2 \leftrightarrow e_3 \leftrightarrow \cdots \leftrightarrow e_m \quad \equiv \quad e_1 \rightarrow e_2 \rightarrow e_3 \rightarrow \cdots \rightarrow e_m \ (\rightarrow e_1)$$

$$\equiv \quad (\overline{e_1} \vee e_2) \wedge (\overline{e_2} \vee e_3) \wedge \ldots \wedge (\overline{e_m} \vee e_1)$$

Each monochromatic connected component of $m$ edges can be encoded as a CNF formula consisting of $m$ binary clauses.

# Deriving An Isolator from Unavoidable Subgraphs

**Example:** Convert a path of two edges into an isolator for $K_3$.

$\mathcal{I}_3 = ab \leftrightarrow ac = (\overline{ab} \vee ac) \wedge (\overline{ac} \vee ab)$

# Deriving An Isolator from Unavoidable Subgraphs

**Example:** Convert a path of two edges into an isolator for $K_3$.

$\mathcal{I}_3 = ab \leftrightarrow ac = (\overline{ab} \vee ac) \wedge (\overline{ac} \vee ab)$

$\mathcal{I}_3$ has 4 satisfying assignments:

1. $ab := F, ac := F, bc := F$ (a graph with zero edges).
2. $ab := F, ac := F, bc := T$ (a graph with one edge).
3. $ab := T, ac := T, bc := F$ (a graph with two edges).
4. $ab := T, ac := T, bc := T$ (a graph with three edges).

# Deriving An Isolator from Unavoidable Subgraphs

**Example:** Convert a path of two edges into an isolator for $K_3$.

$\mathcal{I}_3 = ab \leftrightarrow ac = (\overline{ab} \vee ac) \wedge (\overline{ac} \vee ab)$

$\mathcal{I}_3$ has 4 satisfying assignments:

1. $ab := F, ac := F, bc := F$ (a graph with zero edges).
2. $ab := F, ac := F, bc := T$ (a graph with one edge).
3. $ab := T, ac := T, bc := F$ (a graph with two edges).
4. $ab := T, ac := T, bc := T$ (a graph with three edges).

$\Rightarrow \mathcal{I}_3$ is also a perfect isolator for $K_3$, i.e., only one graph from each isomorphism class of all graphs of order 3 satisfies $\mathcal{I}_3$.

# Computing Unavoidable Subgraphs for Big Graphs

**Fact:** For a given $K_n$, the bigger the subgraph we try to determine its unavoidability, the more expensive the computation is involved.

# Computing Unavoidable Subgraphs for Big Graphs

**Fact:** For a given $K_n$, the bigger the subgraph we try to determine its unavoidability, the more expensive the computation is involved.

However, the computation costs for big subgraphs can be reduced by incorporating isolators encoded from smaller unavoidable subgraphs, which can be computed much easier.

# Computing Unavoidable Subgraphs for Big Graphs

**Fact:** For a given $K_n$, the bigger the subgraph we try to determine its unavoidability, the more expensive the computation is involved.

However, the computation costs for big subgraphs can be reduced by incorporating isolators encoded from smaller unavoidable subgraphs, which can be computed much easier.

**Claim:** If $G$ is an unavoidable subgraph of $K_n$, then $G$ is also an unavoidable subgraph of $K_m$ where $n \leq m$.

# Computing Unavoidable Subgraphs for Big Graphs

**Fact:** For a given $K_n$, the bigger the subgraph we try to determine its unavoidability, the more expensive the computation is involved.

However, the computation costs for big subgraphs can be reduced by incorporating isolators encoded from smaller unavoidable subgraphs, which can be computed much easier.

**Claim:** If $G$ is an unavoidable subgraph of $K_n$, then $G$ is also an unavoidable subgraph of $K_m$ where $n \leq m$.

$\Rightarrow$ The costs of computing unavoidable subgraphs for $K_m$ can be reduced by incorporating isolators of $K_n$.

# Outline

# Summary

We are in the first step towards constructing perfect isolators (i.e., isolators that break all symmetries) with reasonable sizes (polynomial in the size of the corresponding graph-related problem): computing initial isolators from unavoidable subgraphs.

We have presented a method for automatically computing all unavoidable subgraphs of order $k$ for a given complete graph of order $n$ using SAT solvers.

Deriving an isolator from unavoidable subgraphs is straightforward as presented.

Isolators derived from unavoidable subgraphs of small graphs can be used to compute unavoidable subgraphs of bigger graphs.

# Summary

We are in the first step towards constructing perfect isolators (i.e., isolators that break all symmetries) with reasonable sizes (polynomial in the size of the corresponding graph-related problem): computing initial isolators from unavoidable subgraphs.

We have presented a method for automatically computing all unavoidable subgraphs of order $k$ for a given complete graph of order $n$ using SAT solvers.

Deriving an isolator from unavoidable subgraphs is straightforward as presented.

Isolators derived from unavoidable subgraphs of small graphs can be used to compute unavoidable subgraphs of bigger graphs.

Have some "crazy" moments with C programming.

# Future Work

We plan to compute perfect isolators for complete graphs up to 8 vertices. This can be divided into two sub-tasks:

# Future Work

We plan to compute perfect isolators for complete graphs up to 8 vertices. This can be divided into two sub-tasks:

1. Compute asymmetric unavoidable subgraphs.

An asymmetric unavoidable subgraph of a fully-connected graph $K$ is a pair of graphs $G$ and $H$ such that any red/blue edge-labeling of $K$ contains either a red $G$ or blue $H$.

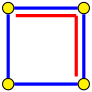
# Future Work

We plan to compute perfect isolators for complete graphs up to 8 vertices. This can be divided into two sub-tasks:

1. Compute asymmetric unavoidable subgraphs.

An asymmetric unavoidable subgraph of a fully-connected graph $K$ is a pair of graphs $G$ and $H$ such that any red/blue edge-labeling of $K$ contains either a red $G$ or blue $H$.

**Example:** The graph below is an asymmetric unavoidable subgraph of $K_4$:



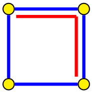
# Future Work

We plan to compute perfect isolators for complete graphs up to 8 vertices. This can be divided into two sub-tasks:

1. Compute asymmetric unavoidable subgraphs.

An asymmetric unavoidable subgraph of a fully-connected graph $K$ is a pair of graphs $G$ and $H$ such that any red/blue edge-labeling of $K$ contains either a red $G$ or blue $H$.

**Example:** The graph below is an asymmetric unavoidable subgraph of $K_4$:



2. Study a technique for constructing a perfect isolator from an initial isolator, which is derived from (asymmetric) unavoidable subgraphs.

# Future Work

Combine (1) symmetry-breaking predicates generated from **shatter** with (2) isolators derived from known unavoidable subgraphs in computing unknown unavoidable subgraphs.

# Future Work

Combine (1) symmetry-breaking predicates generated from **shatter** with (2) isolators derived from known unavoidable subgraphs in computing unknown unavoidable subgraphs.

**Requirement:** Constraints in (2) have to be consistent with constraints in (1). Otherwise, they can remove completely certain isomorphism classes of the original problem.

Thank You!