Computing Maximum Unavoidable Subgraphs Using SAT Solvers

Cuong Chau and Marijn Heule

Department of Computer Science
The University of Texas at Austin
Austin, TX, USA

{ckcuong,marijn}@cs.utexas.edu

Abstract. Unavoidable subgraphs have been widely studied in the context of Ramsey Theory. The research in this area focuses on highly structured graphs such as cliques, cycles, paths, stars, trees, and wheels. We propose to study *maximum unavoidable subgraphs* measuring the size in the number of edges. We computed maximum unavoidable subgraphs for graphs up to order nine via SAT solving and observed that these subgraphs are less structured, although all are bipartite. Additionally, we found large unavoidable bipartite subgraphs up to order twelve. We also present the concept of multi-component unavoidable subgraphs and show that large multi-component subgraphs are unavoidable in small graphs. We envision that maximum unavoidable subgraphs can be exploited using an alternative approach to breaking graph symmetries.

Keywords: satisfiability solving, unavoidable subgraph, combinatorics, graph theory, symmetry breaking

1 Introduction

Satisfiability (SAT) solvers have become very powerful tools to solve hard-combinatorial problems that have only *few* symmetries. Recent successes in this direction are solving Erdős discrepancy problem [10] and the Boolean Pythagorean Triples problem [9]. However, for hard-combinatorial problems with *many* symmetries, such as Ramsey numbers [7], SAT solvers may not be the strongest tools around. For example, the most impressive result regarding Ramsey numbers, solving $R(4,5)$ [12], is two decades old and cannot be reproduced with SAT solving yet. In this paper, we propose to study a special kind of hard-combinatorial problems that could be helpful to bridge this gap.

Consider the fully connected undirected graph of order n, in short K_n. A graph G is called an *unavoidable subgraph* of K_n if G occurs as a fully red or fully blue subgraph in any red/blue edge-coloring of K_n. Unavoidable subgraphs have been widely studied, as can be observed in a survey paper [13] that cites over 600 papers on the subject. This research area focuses on highly structured graphs such as cliques, cycles [3], paths [6], stars [8], trees [4], and wheels [14].

We propose to investigate less structured graphs: the *maximum* unavoidable subgraphs with the size measured in the number of edges.

We compute the maximum unavoidable subgraphs via SAT solving. We observe that the maximum unavoidable subgraphs for small graphs are all bipartite and conjecture that this is also the case for large graphs. Another interesting observation made within the experimented range is that K_{n+1} always has a strictly larger unavoidable subgraph than K_n for $n > 3$. The difference in size between the maximum unavoidable subgraphs of K_n and K_{n+1} is typically one edge and sometimes two edges. Consequently, the size of the maximum unavoidable subgraphs (measured in the number of edges) grows faster than the size of K_n (measured in the number of vertices).

The conventional notion of unavoidable subgraphs considers only connected (single-component) subgraphs. We introduce the concept of *multi-component* unavoidable subgraphs: each component occurs in either red or blue in all red/blue edge-colorings of K_n — although some components may occur in blue, while others occur in red. Starting with K_6, some interesting patterns can be observed in the largest found multi-component unavoidable subgraphs.

The state-of-the-art symmetry-breaking methods for SAT [1] or specifically for graphs [5] are not powerful enough to make some reasonably simple unavoidable subgraph problems solvable via SAT techniques. For example, consider the problem whether a star of eight edges, in short S_8, is an unavoidable subgraph of K_{15}. Using a short combinatorial argument one can solve this problem: K_{15} has an odd number of edges (105), so not all vertices can have exactly seven red and seven blue edges. Hence, one vertex must have at least eight red or at least eight blue edges, or equivalently S_8 as a monochromatic subgraph.

We envision that knowledge about the maximum unavoidable subgraphs, both the single and the multi-component variants, could be a basis for novel symmetry-breaking techniques for SAT solvers. All edges in a component can be replaced by a single edge in the component, allowing significant simplification of graph problems.

The remainder of this paper is structured as follows. First we present some background information on unavoidable subgraphs in Section 2. In Section 3, we describe how to encode unavoidable subgraph problems into SAT. Computing single and multi-component unavoidable subgraphs is discussed in Section 4 and 5, respectively. Section 6 presents a method to exploit unavoidable subgraphs and we draw some conclusions in Section 7.

2 Unavoidable Subgraphs and Motivation

All graphs mentioned in the paper are undirected. Before presenting the definition of an unavoidable subgraph, we first introduce the concept of *graph isomorphism*: Two graphs G and H are isomorphic if there exists an edge-preserving bijection from the vertices of G to the vertices of H. We say that two isomorphic graphs occur in the same *isomorphism class*.

Definition 1 (Unavoidable subgraph). *A graph G is called an unavoidable subgraph of the fully-connected graph K_n if for all red/blue edge-colorings EC of K_n, there exists a subgraph H of K_n such that*

1. *H is isomorphic to G.*
2. *H is monochromatic, either in red or blue, under the coloring EC of K_n.*

We want to emphasize that Definition 1 does not require that H is the same graph for different red/blue edge-colorings of K_n. A small example of unavoidable subgraphs is shown in Figure 1. This figure lists all red/blue edge-colorings of K_3. Notice that a monochromatic path of two edges occurs in all graphs. Hence a path of two edges is an unavoidable subgraph of K_3.

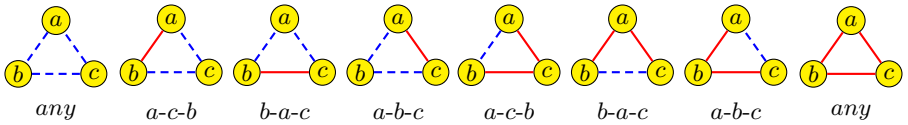


Fig. 1. All red/blue edge-colorings of K_3. For readability, we draw red and blue edges using solid and dashed lines, respectively. Observe that all colored graphs contain a monochromatic path of two edges (as shown below the graphs).

The two propositions below follow from the definition of unavoidable subgraphs. We will refer to them in Section 4:

Proposition 1. *If G is an unavoidable subgraph of K_n, then G is also an unavoidable subgraph of K_m for all $m \geq n$.*

Proposition 2. *If G is an unavoidable subgraph of K_n, then all subgraphs of G are also unavoidable subgraphs of K_n.*

Many "nicely" structured unavoidable subgraphs have been heavily studied [13] such as cliques, cycles [3], paths [6], stars [8], trees [4], and wheels [14]. We propose to study unavoidable subgraphs somewhat differently compared to existing work. Instead of searching for graphs with a well-defined structure, we want to compute *maximum unavoidable subgraphs* (measured in the number of edges). Such maximum unavoidable subgraphs may not have a clear structure.

We argue that maximum unavoidable subgraphs are interesting, because they allow for an alternative symmetry-breaking approach for graph problems: given an avoidable subgraph, we can simplify graph problems by enforcing that all edges in the unavoidable subgraph are either all present or all absent. The larger the unavoidable subgraph, the stronger the symmetry-breaking predicate that can be derived from it. In Section 6 we will explain this in more detail.

3 SAT Encoding of Unavoidable Subgraph Problems

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

is illustrated in the following example: check whether a path of two edges is an unavoidable subgraph of K_3. There are three paths of two edges in K_3 as shown in Figure 2. A path of two edges is an unavoidable subgraph of K_3 if and only if for any red/blue edge-coloring of K_3, at least one of the three paths in Figure 2 is monochromatic. Let ab, ac, and bc denote the Boolean variables representing the color of the edges connecting vertices a and b, a and c, and b and c, respectively. If a Boolean variable has value *true*, the corresponding edge has color red, otherwise it has color blue. Then, the following Boolean formula represents the fact that at least one of the three paths in Figure 2 is monochromatic:

$$\mathcal{F}_{G,K_3} = (ab \wedge bc) \vee (\overline{ab} \wedge \overline{bc}) \vee (ab \wedge ac) \vee (\overline{ab} \wedge \overline{ac}) \vee (ac \wedge bc) \vee (\overline{ac} \wedge \overline{bc})$$

Determining whether a path of two edges is an unavoidable subgraph of K_3 is equivalent to checking the validity of $\mathcal{F}_{G,K_3}$, i.e., checking if $\mathcal{F}_{G,K_3}$ holds for all truth assignments to the variables. This is then equivalent to checking if the negation of $\mathcal{F}_{G,K_3}$ is *unsatisfiable*.

$$\overline{\mathcal{F}_{G,K_3}} = (\overline{ab} \vee \overline{bc}) \wedge (ab \vee bc) \wedge (\overline{ab} \vee \overline{ac}) \wedge (ab \vee ac) \wedge (\overline{ac} \vee \overline{bc}) \wedge (ac \vee bc)$$

Since $\overline{\mathcal{F}_{G,K_3}}$ is in conjunctive normal form (CNF), SAT solvers can solve it directly. Thus, determining an unavoidable subgraph problem can be converted into a SAT problem as illustrated. The construction of $\overline{\mathcal{F}_{G,K_n}}$ is described in Algorithm 1. In particular, given the complete graph K_n and its subgraph G that we want to check if G is unavoidable in K_n, for each subgraph H of K_n that is isomorphic to G, we construct two clauses: (1) disjunction of *positive* literals representing *red* color of edges in H, and (2) disjunction of *negative* literals representing *blue* color of edges in H. The formula $\overline{\mathcal{F}_{G,K_n}}$ is then the conjunction of all of these clauses.

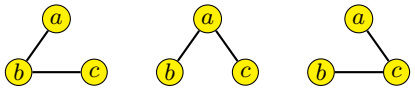


Fig. 2. All paths of two edges in K_3.

The most expensive computation in Algorithm 1 is the function call at line 2, which generates all subgraphs of K_n that are isomorphic to G. A naive approach for generating all isomorphic graphs is to apply all permutations on every set of k nodes taken from n nodes, where k is the number of nodes in G. Each permutation application will of course produce an isomorphic graph to G. However, duplicate graphs can be produced since different permutations can return the same graph. The number of duplicate graphs can be huge in comparison with the number of non-duplicate isomorphic graphs, depending on the graph's structure and size. More importantly, the number of permutations grows rapidly as the number of nodes in G increases. And the number of non-duplicate isomorphic

Algorithm 1 SAT Encoding of An Unavoidable Subgraph Problem

1: **function** UNAVOID-SUBGRAPH-SAT-ENCODING(G, K_n)
2: $\mathcal{G} \leftarrow$ ISOMORPHIC-SUBGRAPHS-GEN(G, K_n)
3: $\overline{\mathcal{F}_{G,K_n}} \leftarrow \varnothing$
4: **for** each $H \in \mathcal{G}$ **do**
5: $C_r \leftarrow$ disjunction of *positive* literals representing *red* color of edges in H
6: $C_b \leftarrow$ disjunction of *negative* literals representing *blue* color of edges in H
7: $\overline{\mathcal{F}_{G,K_n}} \leftarrow \overline{\mathcal{F}_{G,K_n}} \wedge C_r \wedge C_b$
8: **end for**
9: **return** $\overline{\mathcal{F}_{G,K_n}}$
10: **end function**

graphs is usually very small in comparison with the number of permutations. Consequently, we will come up with spending most of the time computing duplicate graphs, which are unnecessary. We want to avoid these computations. Instead we just apply all permutations on the first set of k nodes taken from n nodes. After this step, we will recognize which permutations generate duplicate graphs. We can then avoid applying these permutations on all remaining sets of k nodes. The details of this approach are described in Algorithm 2. The output of this algorithm is a collection of all non-duplicate subgraphs of K_n that are isomorphic to the input graph G.

Algorithm 2 only applies the set P_k of all permutations on the first set of k nodes taken from n nodes in the first loop (lines 7-14). This loop also constructs a *minimal* set P'_k of permutations that generates all non-duplicate isomorphic graphs to G for any set of k nodes (recall that k is the number of nodes in G). After this loop, the algorithm will apply P'_k instead of P_k for all remaining sets of k nodes (lines 16-22).

4 Computing Unavoidable Subgraphs Using SAT Solvers

We are interested in computing unavoidable subgraphs mechanically. By exploiting the `gtools` programs in the `nauty` package [11], we are able to generate all non-isomorphic graphs of order k (≤ 10) very quickly. For each generated graph G, we can mechanically check whether it is unavoidable for a given complete graph K_n by converting into a SAT formula $\overline{\mathcal{F}_{G,K_n}}$ as described in the previous section, and then employing a SAT solver to check the formula's satisfiability. We use the `glucose 3.0` SAT solver [2] to do the satisfiability check.

4.1 Breaking Symmetries

Symmetries are the main obstacle for checking the satisfiability of $\overline{\mathcal{F}_{G,K_n}}$ efficiently. To counter this obstacle, we add symmetry-breaking predicates (SBP) to $\overline{\mathcal{F}_{G,K_n}}$ before calling a SAT solver. Unavoidable subgraph problems have two symmetries: any permutation of the vertices and swapping the edge colors. The

Algorithm 2 Computing Isomorphic Subgraphs

function ISOMORPHIC-SUBGRAPHS-GEN(G, K_n)
  $P_k \leftarrow$ all permutations of $\{0, 1, 2, ..., k-1\}$, where k is the order of G
  $Q_k^n \leftarrow$ all k-combinations of k nodes taken from the vertex set of K_n
  $\mathcal{G} \leftarrow \varnothing$ ▷ Output: all subgraphs of K_n that are isomorphic to G
  $P_k' \leftarrow \varnothing$ ▷ A subset of P_k s.t. its application generates non-duplicate graphs
  Pick the first k-combination $Q \in Q_k^n$
  for each $\pi \in P_k$ **do**
    $Q' \leftarrow \pi(Q)$
    $G' \leftarrow Q'(G)$ ▷ Construct an isomorphic graph G' of G
    if $G' \notin \mathcal{G}$ **then** ▷ Check if G' is not duplicate in $\mathcal{G}$
      $\mathcal{G} \leftarrow \mathcal{G} \cup \{G'\}$
      $P_k' \leftarrow P_k' \cup \{\pi\}$
    end if
  end for
  $Q_k^n \leftarrow Q_k^n \backslash \{Q\}$ ▷ We are done with the first combination of Q_k^n
  for each $Q \in Q_k^n$ **do**
    for each $\pi \in P_k'$ **do**
      $Q' \leftarrow \pi(Q)$
      $G' \leftarrow Q'(G)$ ▷ Construct an isomorphic graph G' of G
      $\mathcal{G} \leftarrow \mathcal{G} \cup \{G'\}$ ▷ G' is guaranteed to be non-duplicate in $\mathcal{G}$
    end for
  end for
  return $\mathcal{G}$
end function

edge color symmetry can easily be broken by selecting an edge and forcing it to a color by adding a unit clause.

Several methods have been developed to break such graph symmetries [1, 5]. Existing techniques can be viewed as enforcing a lexicographic order on the rows of the adjacency matrix. Let A_G be the adjacency matrix representing a graph G of order n. The predicate $A_G[i] \preceq_{\{i,j\}} A_G[j]$ states that the binary representation of the i-th row of A_G is less than or equal to the binary representation of the j-th row of A_G when excluding the i-th and j-th columns of A_G.

Definition 2 (Linear symmetry break). *We define the linear symmetry-breaking predicates (L-SBP) for graph G as follows:*

$$\textit{L-SBP}(G) = \bigwedge_{i=1}^{n-1} A_G[i] \preceq_{\{i,i+1\}} A_G[i+1]$$

The symmetry-breaking tool `shatter` [1] adds L-SBP to graph problems such as unavoidable subgraph formulas.

Table 1. The number of graphs that satisfy the symmetry-breaking predicates L-SBP and Q-SBP as compared to the number of isomorphism classes of graphs of order 5 to 12. The number of satisfying assignments were computed using `sharpSAT` [15].

n	L-SBP	Q-SBP	# isomorphism classes
5	46	43	34
6	325	276	156
7	4,045	3,158	1,044
8	89,812	66,595	12,346
9	3,583,903	2,587,488	274,668
10	258,518,959	184,193,025	12,005,168
11	33,859,710,152	23,962,961,317	1,018,997,864
12	8,086,937,704,176	5,700,915,311,729	165,091,172,592

Codish et al. [5] demonstrated that the comparator $\preceq_{\{i,j\}}$ is not transitive: $A_G[h] \preceq_{\{h,i\}} A_G[i] \wedge A_G[i] \preceq_{\{i,j\}} A_G[j]$ does not imply $A_G[h] \preceq_{\{h,j\}} A_G[j]$. Enforcing $A_G[i] \preceq_{\{i,j\}} A_G[j]$ for all $1 \leq i < j \leq n$ is a valid symmetry-breaking predicate for graph problems [5]. We will refer to this method as *quadratic symmetry-breaking method* since it adds a quadratic number of constraints to graph problems.

Definition 3 (Quadratic symmetry break). *We define the quadratic symmetry-breaking predicates (Q-SBP) for graph G as follows:*

$$\textit{Q-SBP}(G) = \bigwedge_{1 \leq i < j \leq n} A_G[i] \preceq_{\{i,j\}} A_G[j]$$

Table 1 shows the number of graphs / assignments that satisfy the symmetry-breaking predicates L-SBP and Q-SBP as well as the number of isomorphism classes. Notice that Q-SBP is slightly better than L-SBP. Both numbers are not close to the number of isomorphism classes meaning that many symmetries are not broken by both methods.

The impact of symmetry-breaking predicates on the time required to solve some large unavoidable subgraph problems is shown in Table 2. Notice that the runtime is reduced by orders of magnitude for the larger unavoidable subgraphs. However, there is no clear difference between the L-SBP and Q-SBP methods.

Table 2. Runtime in seconds to compute the maximum or largest found unavoidable subgraphs of K_6 to K_{12} shown in Figure 3 using `glucose` 3.0. The experiments were run on 3.5GHz Intel Xeon E31280 processors with 8MB L3 cache size. A '-' means a timeout after 24 hours.

n	6	7	8	9	10	11	12
No SBP	0	0.025	0.38	4.85	11,690.70	-	-
L-SBP	0	**0**	**0.01**	**0.11**	**4.77**	**18.73**	312.60
Q-SBP	0	**0**	**0.01**	**0.11**	7.98	19.40	**303.40**

Algorithm 3 Computing Unavoidable Subgraphs of k Nodes for K_n

function UNAVOID-SUBGRAPHS-ORDER-K-GEN(k, K_n)
  $\mathcal{G} \leftarrow$ all non-isomorphic graphs of order k generated using `nauty`
  $\mathcal{H} \leftarrow \varnothing$ ▷ Output: all unavoidable subgraphs of order k in K_n
  for each $G \in \mathcal{G}$ **do**
    $\overline{\mathcal{F}_{G,K_n}} \leftarrow$ UNAVOID-SUBGRAPH-SAT-ENCODING(G, K_n)
    if UNSAT($\overline{\mathcal{F}_{G,K_n}} \wedge SBP(\overline{\mathcal{F}_{G,K_n}})$) **then**
      $\mathcal{H} \leftarrow \mathcal{H} \cup \{G\}$
    end if
  end for
  return $\mathcal{H}$
end function

4.2 Enumerating Unavoidable Subgraphs

Algorithm 3 shows the pseudo-code of enumerating unavoidable subgraphs. In principle, this algorithm can be applied to compute all unavoidable subgraphs of a given complete graph K_n by setting k equal to n. Nonetheless, the number of non-isomorphic subgraphs of K_n grows rapidly as n increases, so it is impractical to exhaustedly check the unavoidability of all non-isomorphic subgraphs of large complete graphs.

We reduced the number of evaluated subgraphs by ignoring several kinds of graphs that cannot be unavoidable in a given complete graph. For example, if the maximum degree of a graph exceeds some threshold, it cannot be unavoidable in a given complete graph. One may think this threshold degree is $\lfloor n/2 \rfloor$ for all K_n by reasoning that there always exists a red/blue edge-coloring on any complete graph K_n such that: among edges connected to each vertex v in K_n, the number of edges of the same color is at most $\lfloor n/2 \rfloor$. As a result, subgraphs with the maximum degree higher than $\lfloor n/2 \rfloor$ cannot be unavoidable in K_n. However, this is not true when $n \equiv 3 \pmod 4$ as stated in Theorem 1.

Theorem 1. *For every K_n with $n \equiv 3$ (mod 4), there exists a graph G such that G is unavoidable in K_n and the maximum degree of G is at least $\lfloor n/2 \rfloor + 1$.*

Proof. We prove by contradiction. Suppose there exists a red/blue edge-coloring of K_n such that among edges connected to each vertex v in K_n, the number of edges of the same color is at most $\lfloor n/2 \rfloor$. (1)

Since $n \equiv 3 \pmod 4$, n is an odd number. For each vertex v in K_n, there are $(n-1)$ edges connected to v, which is therefore an even number. (2)

From (1) and (2), we can claim that the number of red and blue edges connected to each v in K_n are both equal to $\lfloor n/2 \rfloor$. As a result, the number of red and blue edges in K_n must be equal. (3)

Since $n \equiv 3 \pmod 4$, K_n has an odd number of edges, or exactly $8l^2 + 10l + 3$ with $l = \lfloor n/4 \rfloor$. Hence, the number of red and blue edges in K_n must be different, which contradicts (3). Thus there must exist a subgraph G in K_n with the maximum degree at least $(\lfloor n/2 \rfloor + 1)$ and G is unavoidable in K_n.

By using our SAT solving approach, we proved that for all $n \leq 18$, the star $S_{\lfloor n/2 \rfloor+2}$ is not an unavoidable subgraph of K_n when $n \equiv 3 \pmod 4$, and $S_{\lfloor n/2 \rfloor+1}$ is not an unavoidable subgraph of K_n for other values of n[1]. Since the star S_k (i.e., the complete bipartite graph $K_{1,k}$) is the smallest graph with the maximum degree k, if S_k is not unavoidable in K_n, then no other graphs with the maximum degree at least k is unavoidable in K_n by the contrapositive of Proposition 2. For that reason, given a complete graph K_n such that $n \leq 18$, we only need to check the unavoidability of subgraphs with the maximum degree at most $(\lfloor n/2 \rfloor + 1)$ if $n \equiv 3 \pmod 4$, and subgraphs with the maximum degree at most $\lfloor n/2 \rfloor$ for the other case of n.

Since we are interested in discovering maximum unavoidable subgraphs — instead of all unavoidable subgraphs of a given complete graph K_n — we skipped evaluating subgraphs that are smaller than the largest known unavoidable subgraph of K_n. For example, when we know that certain graph G is a maximum unavoidable subgraph of K_n, then we only need to evaluate subgraphs with at least $(|E(G)| + 1)$ edges for K_{n+1}, because G is also unavoidable in K_{n+1} by Proposition 1. In particular, we first check subgraphs satisfying the maximum degree requirement as stated above and their number of edges equal to $(|E(G)|+1)$. If at least one of them is unavoidable in K_n, we then check subgraphs with the number of edges equal to $(|E(G)|+2)$, and so on. The process will stop if neither one of subgraphs with the number of edges $(|E(G)| + i)$ is unavoidable in K_n. From the contrapositive of Proposition 2, we can claim that there does not exist a subgraph H in K_n s.t. $|E(H)| \geq (|E(G)| + i)$ and H is unavoidable in K_n. In other words, a maximum unavoidable subgraph of K_n must have $(|E(G)|+i-1)$ edges. Using this approach, we are able to find maximum unavoidable subgraphs for K_3 to K_9 effectively by reducing the number of subgraphs to be checked a substantial amount (see Table 3).

Table 3. The number of graphs to be evaluated while searching for maximum unavoidable subgraphs of K_n as compared to the number of isomorphism classes of graphs of order n with $3 \leq n \leq 9$. Our approach imposes an upper bound on the maximum degree of graphs and a lower bound on the number of edges. We start with the fact that K_2 is the maximum unavoidable subgraph for itself.

n	3	4	5	6	7	8	9
# isomorphism classes	4	11	34	156	1,044	12,346	274,668
# checked graphs	2	2	6	35	97	291	904

[1] Currently, our system is not able to prove this property for $n > 18$ due to out of computational resources.

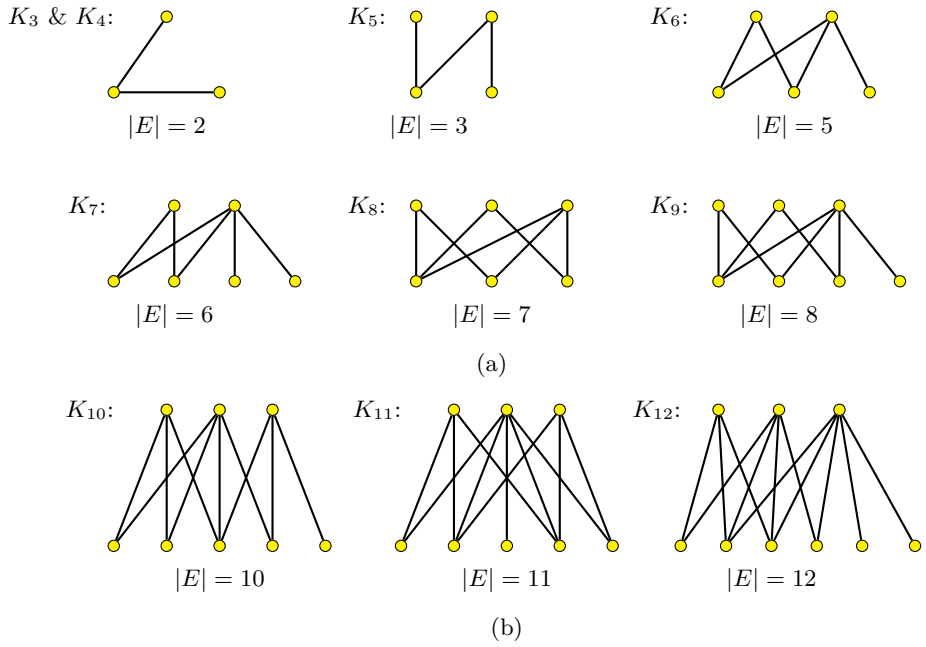


Fig. 3. Some large single-component unavoidable subgraphs of K_3 to K_{12}: (a) maximum unavoidable subgraphs of K_3 to K_9, all of them are bipartite; (b) maximum bipartite unavoidable subgraphs discovered of K_{10} to K_{12}.

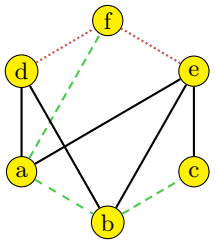
4.3 Results on Single-Component Unavoidable Subgraphs

Figure 3 (a) shows the maximum unavoidable subgraphs for each K_n with $3 \leq n \leq 9$ [2]. Observe that all maximum unavoidable subgraphs are *bipartite*. We used this observation to find large unavoidable subgraphs for K_{10} to K_{12}. The bipartite restriction was required to make the number of graphs to be evaluated manageable — apart from the other restrictions of the maximum degree and the minimum number of edges. Figure 3 (b) shows the largest found unavoidable subgraphs for K_{10} to K_{12}. We are unable to determine whether these unavoidable bipartite subgraphs are maximal: the evaluation of many graphs with one more edge required more than 24 hours SAT solving time (the limit on our cluster).

5 Computing Multi-Component Unavoidable Subgraphs

The concept unavoidable subgraphs as stated in Definition 1 can be generalized to multiple components, such that each component must occur monochromatic in all red/blue edge-colorings of K_n. We will represent a multiple-component

[2] We found multiple maximum unavoidable subgraphs for K_9. Figure 3 (a) shows one of them.



	a	b	c	d	e	f
a	0	2	0	1	1	2
b	2	0	2	1	1	0
c	0	2	0	0	1	0
d	1	1	0	0	0	3
e	1	1	1	0	0	3
f	2	0	0	3	3	0

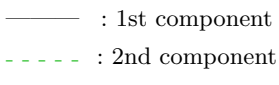


Fig. 4. An example illustrating the adjacency matrix of a 3-component graph.

graph G by a graph with edge labels. Edges are in the same component of G if and only if they have the same label. In this section, we show how to compute multi-component unavoidable subgraphs using SAT solvers. We will discuss in Section 6, how multi-component unavoidable subgraphs can be helpful in constructing symmetry-breaking predicates for graph problems. Given a multi-component graph G, we write G_i to denote the subgraph of G that has only the edges with label i. The adjacency matrix of a multi-component graphs is constructed as follows: $A_G(i,j) = A_G(j,i) = k$ if the edge connecting nodes i and j has label k and $A_G(i,j) = A_G(j,i) = 0$ if no edge connects nodes i and j. Figure 4 shows a 3-component graph and its adjacency matrix. In the visualizations, edges with label 1 are shown as solid lines, edges with label 2 as dashed lines, and edges with label 3 as dotted lines.

Algorithm 4 describes the SAT encoding algorithm for the case $m = 2$. Given a graph G consisting of two components, first all occurrences of G in K_n are computed using Isomorphic-Subgraphs-Gen(G, K_n). Notice that we had to generalize this procedure to deal with graphs that have labelled edges. For each possible presence of G in K_n, we have 4 clauses stating that at least one component must be non-monochromatic. The algorithm can be easily generalized to m-component unavoidable subgraphs by adding 2^m clauses for each possible presence of the graph in K_n.

Our mechanism for computing m-component unavoidable subgraphs is incremental from some $(m-1)$-component unavoidable subgraph ($m \geq 2$). Algorithm 5 describes our mechanism for computing 2-component unavoidable subgraphs from a given single-component unavoidable subgraph. In this algorithm, we use the notation $V(G)$ to denote the vertex set of G, and $E_i(G)$ to denote the edges with label i in G. Given an unavoidable subgraph G of K_n, we generate all possible supergraphs that have one additional edge in the second component. For each of those generated graphs, we check whether it is also unavoidable in K_n. If it is unavoidable, we recursively call the same procedure again (line 7).

The idea of this algorithm can be generalized to m-component unavoidable subgraph computation for all $m \geq 2$. Let $S(n)$ denote the size of maximum single-component unavoidable subgraphs of K_n. As heuristic to reduce the vast number of possible multi-component graphs, we start with a single component with at least $S(n) - 1$ edges. Second, we compute which one of them can be extended

Algorithm 4 SAT Encoding of A 2-Component Unavoidable Subgraph Problem

function TWO-COMP-UNAVOID-SUBGRAPH-SAT-ENCODING(G, K_n)
  $\mathcal{G} \leftarrow$ ISOMORPHIC-SUBGRAPHS-GEN(G, K_n)
  $\overline{\mathcal{F}_{G,K_n}} \leftarrow \varnothing$
  for each $H \in \mathcal{G}$ **do** ▷ Suppose H consists of 2 components H_1 and H_2
    $C_{r_1} \leftarrow$ disjunction of *positive* literals representing *red* color of edges in H_1
    $C_{b_1} \leftarrow$ disjunction of *negative* literals representing *blue* color of edges in H_1
    $C_{r_2} \leftarrow$ disjunction of *positive* literals representing *red* color of edges in H_2
    $C_{b_2} \leftarrow$ disjunction of *negative* literals representing *blue* color of edges in H_2
    $\overline{\mathcal{F}_{G,K_n}} \leftarrow \overline{\mathcal{F}_{G,K_n}} \wedge (C_{r_1} \vee C_{r_2}) \wedge (C_{r_1} \vee C_{b_2}) \wedge (C_{b_1} \vee C_{r_2}) \wedge (C_{b_1} \vee C_{b_2})$
  end for
  return $\overline{\mathcal{F}_{G,K_n}}$
end function

Algorithm 5 Computing 2-Component Unavoidable Subgraphs of K_n from a Given Subgraph

function TWO-COMP-UNAVOID-SUBGRAPHS-GEN(G, K_n)
  $\mathcal{G} \leftarrow$ all supergraphs H of G s.t. $|V(H)| = |V(G)|$, $|E_1(H)| = |E_1(G)|$, and $|E_2(H)| = |E_2(G)| + 1$
  $\mathcal{H} \leftarrow \varnothing$ ▷ Output: all 2-component graphs (extended from G) ▷ that are unavoidable in K_n.
  for each $H \in \mathcal{G}$ **do**
    $\overline{\mathcal{F}_{H,K_n}} \leftarrow$ TWO-COMP-UNAVOID-SUBGRAPH-SAT-ENCODING(H, K_n)
    if UNSAT($\overline{\mathcal{F}_{H,K_n}} \wedge SBP(\overline{\mathcal{F}_{H,K_n}})$) **then**
      $\mathcal{H} \leftarrow \mathcal{H} \cup \{H\} \cup$ TWO-COMP-UNAVOID-SUBGRAPHS-GEN(H, K_n)
    end if
  end for
  return $\mathcal{H}$
end function

using a second component consisting of two edges (not necessarily connected). Notice that components consisting of a single edge can be ignored as each single edge is always monochromatic. This is repeated by trying to extend the graphs with two components to three components by adding another component with two edges. Once the starting points of unavoidable subgraphs have been determined, we try to extend them by adding edges to each component. We repeat this until no multi-component graphs can be further extended. Figure 5 displays our discovery of largest found multi-component unavoidable subgraphs for K_6, K_7, and K_8. One interesting observation is that the maximum single-component unavoidable subgraph of K_6 is an induced subgraph of the maximum single-component unavoidable subgraphs for K_7 and K_8. The same observation can also be made for the largest found multi-component unavoidable subgraphs.

The above procedure does not necessarily produce maximum multi-component unavoidable subgraphs. The used heuristics, i.e., starting with a single component of at least $S(n) - 1$ edges, may prevent us finding a maximum multi-

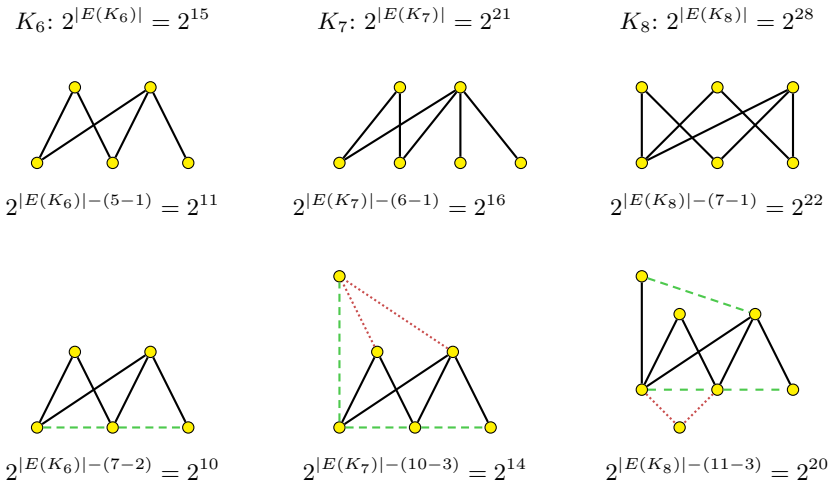


Fig. 5. Comparison between maximum single-component and largest found multi-component unavoidable subgraphs of K_6, K_7, and K_8. The top row displays single-component graphs. The bottom row displays multi-component graphs: 2-component for K_6, 3-component for K_7 and K_8. For all these three cases, symmetry-breaking predicates derived from largest found multi-component unavoidable subgraphs result in smaller search spaces (measured in the number of graphs) as compared to the ones derived from maximum single-component unavoidable subgraphs.

component unavoidable subgraph. Also, we restricted the search to graphs that have at most three components. In future work, we want to compute maximum multi-component unavoidable subgraphs by applying the methods without heuristics and restrictions.

6 Deriving Symmetry-Breaking Predicates from Unavoidable Subgraphs

In order to compute unavoidable subgraphs efficiently, symmetry-breaking techniques were used as discussed in Section 4.1. Although adding symmetry-breaking predicates reduced the solving costs substantially, we also observed that these predicates are not strong enough to solve some relatively easy unavoidable subgraph problems using SAT. For example, the problem whether the star S_8 is unavoidable in K_{15} cannot be solved even after symmetry breaking. This section describes how knowledge about some unavoidable subgraphs could be turned into an alternative approach to symmetry breaking.

The method works as follows. Given a known unavoidable subgraph G of K_n consisting of m components, construct a predicate that forces for each component that all of its edges are equal. For example, recall that a path of two edges is an unavoidable subgraph of K_3. First, we pick a concrete path, say b-a-c, and force both edges (a-b and a-c) to be equal, i.e., either both present or both

absent in graphs of order 3. There are only four graphs of order 3 that satisfy this constraint and these are shown in Figure 6. Moreover, these four graphs represent exactly the four isomorphism classes of graphs of order 3. This implies that the symmetry-breaking predicate derived from a path of two edges is *perfect* for graphs of order 3, i.e., exactly one graph from each isomorphism class satisfies the symmetry-breaking predicate.

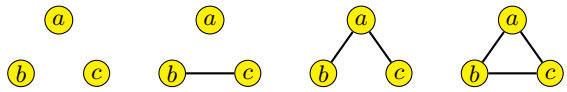


Fig. 6. All graphs of order 3 satisfying the symmetry-breaking predicate that forces the path b-a-c to be either present or absent.

More concretely, given an unavoidable subgraph G of K_n. Let $|E_1(G)| = l$ and let e_1, e_2,..., e_l denote the Boolean variables representing the edges with label 1. The symmetry-breaking predicate forcing all these variables (edges) to be equivalent is

$$e_1 \leftrightarrow e_2 \leftrightarrow e_3 \leftrightarrow \cdots \leftrightarrow e_l$$

The $(l-1)$ equivalence relations above can be expressed using a cycle of l implications, which can be represented using l binary clauses as follows:

$$e_1 \leftrightarrow e_2 \leftrightarrow e_3 \leftrightarrow \cdots \leftrightarrow e_l \quad \equiv \quad e_1 \rightarrow e_2 \rightarrow e_3 \rightarrow \cdots \rightarrow e_l$$

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

Thus, a monochromatic component consisting of l edges can be encoded with l binary clauses. Applying the method for all components of G results in a symmetry-breaking predicate of $|E(G)|$ binary clauses. Since any unavoidable subgraph G of K_n is also an unavoidable subgraph of K_m where $m \geq n$, we can apply the symmetry-breaking predicate derived from G in checking the unavoidability of other subgraphs of K_m. How useful are these symmetry-breaking predicates? Using the largest found unavoidable subgraph of K_{11} as symmetry-breaking predicate to compute the largest found unavoidable subgraph of K_{12} reduces the runtime to 532.48 (s). Not as strong as the L-SBP and Q-SBP predicates (see Table 2), but still reasonably well.

In general, given an unavoidable subgraph G of K_n with m components, the number of graphs of order n that satisfy the symmetry-breaking predicate is $2^{|E(K_n)|+m-|E(G)|}$, because $|E(G)|-m$ edges will be depending on m edges (the representatives of the components). Figure 5 shows how much the search space is reduced when converting the maximum or largest known unavoidable subgraphs for K_6, K_7, and K_8 into symmetry-breaking predicates. In all these three cases, the largest known multi-component unavoidable subgraphs are more effective

Table 4. Number of graphs of order n that satisfy symmetry-breaking predicates. L-SBP-bin: the binary clauses of L-SBP; Q-SBP-bin: the binary clauses of Q-SBP; and USG-SBP: the symmetry-breaking predicates based on unavoidable subgraphs.

n	L-SBP-bin	Q-SBP-bin	USG-SBP
6	5,210	4,672	1,024
7	196,608	181,248	16,384
8	14,680,064	13,664,256	1,048,576

than the maximum single-component unavoidable subgraphs. When comparing these numbers with Table 1, the results do not look impressive. However, the existing symmetry-breaking methods use many long clauses. If we only consider the binary clauses in symmetry-breaking predicates of existing techniques, then symmetry-breaking predicates derived from unavoidable subgraphs look much more interesting, see Table 4. The number of graphs that satisfy the predicates is much smaller.

The main question that arises is: how can we improve the unavoidable subgraph based predicates? A possible answer is finding non-binary clauses that can be added similar to the L-SBP and Q-SBP methods. Alternatively, one can search for *asymmetric unavoidable subgraphs*, such as: is a cycle of four edges occurring in red or a path of two edges occurring in blue unavoidable in K_4? We observed that this asymmetric subgraph is indeed unavoidable. Converting such asymmetric unavoidable subgraphs into a symmetry-breaking predicate makes it stronger. These topics will be the focus of future work.

7 Conclusions

We studied and computed maximum unavoidable subgraphs using SAT solvers. During our experiments we observed that all maximum unavoidable subgraphs are bipartite and conjecture that this holds in general. Also, it appears that the maximum unavoidable subgraphs of K_{n+1} are strictly larger than the maximum unavoidable subgraphs of K_n for $n > 3$. Symmetry breaking was crucial to obtain our results. However, we also observed that current symmetry-breaking techniques are not strong enough to compute some relatively simple unavoidable subgraph problems using SAT. We demonstrated how unavoidable subgraphs can be converted into symmetry-breaking predicates. We are hopeful that this approach helps to improve symmetry-breaking techniques for SAT.

Acknowledgements

The authors are supported by the National Science Foundation under grant number CCF-1526760 and acknowledge the Texas Advanced Computing Center (TACC) at The University of Texas at Austin for providing grid resources that have contributed to the research results reported within this paper.

References

1. Fadi A. Aloul, Karem A. Sakallah, and Igor L. Markov. Efficient symmetry breaking for boolean satisfiability. In Georg Gottlob and Toby Walsh, editors, *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003*, pages 271–276. Morgan Kaufmann, 2003.
2. Gilles Audemard and Laurent Simon. Predicting learnt clauses quality in modern SAT solvers. In Craig Boutilier, editor, *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*, pages 399–404, 2009.
3. John A. Bondy and Paul Erdős. Ramsey numbers for cycles in graphs. *Journal of Combinatorial Theory, Series B*, 14(1):46 – 54, 1973.
4. Stefan A Burr and Paul Erdős. Extremal Ramsey theory for graphs. *Utilitas Mathematica 9*, pages 247–258, 1976.
5. Michael Codish, Alice Miller, Patrick Prosser, and Peter J. Stuckey. Breaking symmetries in graph representation. In *Proceedings of IJCAI 2013*, pages 510–516. IJCAI/AAAI, 2013.
6. László Gerencsér and András Gyárfás. On Ramsey-type problems. *Annales Universitatis Scientiarum Budapestinensis*, 10:167–170, 1967.
7. Ronald L. Graham, Bruce L. Rothschild, and Joel H. Spencer. *Ramsey Theory*. A Wiley-Interscience publication. Wiley, 1990.
8. Frank Harary. Recent results on generalized ramsey theory for graphs. In Y. Alavi, D.R. Lick, and A.T. White, editors, *Graph Theory and Applications*, volume 303 of *Lecture Notes in Mathematics*, pages 125–138. Springer Berlin Heidelberg, 1972.
9. Marijn J. H. Heule, Oliver Kullmann, and Victor W. Marek. Solving and verifying the boolean pythagorean triples problem via cube-and-conquer, 2016. Accepted for SAT 2016, appearing in the same volume.
10. Boris Konev and Alexei Lisitsa. A sat attack on the erdos discrepancy conjecture. In Carsten Sinz and Uwe Egly, editors, *Theory and Applications of Satisfiability Testing SAT 2014*, volume 8561 of *Lecture Notes in Computer Science*, pages 219–226. Springer International Publishing, 2014.
11. Brendan D. McKay and Adolfo Piperno. Practical graph isomorphism, {II}. *Journal of Symbolic Computation*, 60(0):94 – 112, 2014.
12. Brendan D. McKay and Stanislaw P. Radziszowski. R(4, 5) = 25. *Journal of Graph Theory*, 19(3):309–322, 1995.
13. Stanisław P. Radziszowski. Small Ramsey numbers. *The Electronic Journal of Combinatorics*, #DS1, 2014.
14. Stanisław P. Radziszowski and Xia Jin. Paths, cycles and wheels in graphs without antitriangle. *Australasian Journal of Combinatorics*, 9:221–232, 1994.
15. Marc Thurley. Theory and applications of satisfiability testing - sat 2006: 9th international conference, seattle, wa, usa, august 12-15, 2006. proceedings. chapter sharpSAT – Counting Models with Advanced Component Caching and Implicit BCP, pages 424–429. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.