

ON HADWIGER'S CONJECTURE WITH INDEPENDENCE NUMBER 2

DANIEL CARTER

A SENIOR THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
BACHELOR OF ARTS IN MATHEMATICS AT
PRINCETON UNIVERSITY

ADVISER: MARIA CHUDNOVSKY

MAY 1, 2023

Abstract

We prove a variety of conditions that counterexamples to Hadwiger’s Conjecture with independence number 2 must satisfy. This thesis is split into two parts. In the first part, we prove that $\{\overline{K_3}, H\}$ -free graphs are not counterexamples to Hadwiger’s Conjecture, where H is any one of 33 graphs on seven, eight, or nine vertices, or $H = K_8$. This improves on past results of Plummer-Stiebitz-Toft, Kriesell, and Bosse.

In the second part, we introduce the notion of a graph G satisfying a “local condition;” roughly speaking, this means that every induced subgraph of G isomorphic to a given graph H is in fact found in a larger induced subgraph of G isomorphic to a given H' . By modifying the techniques in the first part, we prove that counterexamples to Hadwiger’s Conjecture with independence number 2 satisfy many specific local conditions.

The proofs of the main theorems in each part are mostly computer-assisted. We prove that the algorithm in the second part is “complete” in the sense that any statement of the form “all graphs (including infinite graphs) that satisfy local condition Q also satisfy local condition R ” can be proved by this algorithm. Therefore we expect these techniques to be widely applicable to problems in structural graph theory.

Acknowledgements

I am deeply grateful to Maria Chudnovsky for introducing me to this problem and for very many helpful discussions, as well as for teaching me most of what I know about graph theory. Thank you as well to Paul Seymour, Zach Hunter, Jan Goedgebeur, and Aleksa Milojevic for helpful comments and discussions at various stages of this work. This work was partially supported by funding from Princeton University for undergraduate summer research.

Declaration

I declare that I have not violated the Honor Code during the composition of this work. This paper represents my own work in accordance with University regulations.

I authorize Princeton University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purposes of scholarly research.

Contents

Abstract	ii
Acknowledgements	iii
I Forbidden Induced Subgraphs	1
1 Introduction	2
1.1 Relationship to Past Results	3
2 Algorithms	5
2.1 Basic Algorithm: No Dominating Edges	5
2.2 Getting Unstuck: Four-Clique Covers	7
2.2.1 Assuming No Vertex Dominates H	7
2.2.2 Weaker or Stronger Assumptions	10
2.3 Combined Algorithm	12
2.4 Behavior of the Algorithm	13
2.4.1 Ordering	13
2.4.2 Limitations	19
3 Results	20
3.1 Some Simple Cases	21
3.2 Data for Computer Verification	24
4 Final Remarks	25
II Local Conditions	29
5 Introduction	30
6 Local Conditions	32
6.1 Definitions	32
6.2 Remarks	35

7	Proving One Local Condition Given Another	37
7.1	The Algorithm	37
7.2	Some Properties of the Algorithm	41
7.3	Extensions to the Algorithm	44
8	Strategy	44
8.1	Step 1: Starting Point	47
8.2	Step 2(a): Pruning	48
8.3	Step 2(b): Further Alterations	49
8.4	Step 3: Final Proof	50
9	Data For Computer Verification	54
10	Discussion of the Main Theorem	58
11	Other Problems	58
11.1	Seymour’s Strengthening	58
11.2	Parameterized Hadwiger’s Conjecture	60
12	Miscellaneous Results	63
12.1	Computational Complexity	63
12.2	Extremal Properties	64
13	Final Remarks	67
	Bibliography	68
	Appendix	70
A.1	Further Implementation Details and Code	70
A.1.1	In Part I	70
A.1.2	In Part II	71
A.2	More Data	72
A.3	Proving HC- $\{\overline{K_3}, K_8\}$	74

Part I

Forbidden Induced Subgraphs

1 Introduction

All graphs in this part are finite and simple. Let $h(G)$ be the maximum t such that K_t is a minor of G , $\chi(G)$ the chromatic number of G , and $\omega(G)$ and $\alpha(G)$ respectively the clique number and independence number of G . *Hadwiger's Conjecture* is:

Conjecture 1. *For all graphs G , $h(G) \geq \chi(G)$.*

For a graph H , we say G is H -free if no induced subgraph of G is isomorphic to H . Likewise for a set of graphs S , we say G is S -free if G is H -free for all H in S . We denote by $HC-S$ the statement that Hadwiger's Conjecture holds for all S -free graphs.

There is much interest in the specific case of Hadwiger's Conjecture for graphs with no stable set of size 3, i.e. $\overline{K_3}$ -free graphs. Paul Seymour writes about this case:

This seems to me to be an excellent place to look for a counterexample. My own belief is, if it is true for graphs with stability number two then it is probably true in general. [Sey16]

A well-known result is:

Theorem 1.1. [PST03] *$HC-\{\overline{K_3}\}$ is equivalent to the statement that all $\overline{K_3}$ -free graphs G have a $K_{\lceil |V(G)|/2 \rceil}$ minor.*

Previously, it was proved:

Theorem 1.2. *$HC-\{\overline{K_3}, H\}$ holds where H is any graph on five vertices or the graphs H_6 or H_7 in Figure 1 [PST03, Kri10], or the wheel graph on six vertices W_5 , $\overline{K_{1,5}}$, or K_7 [Bos19].*

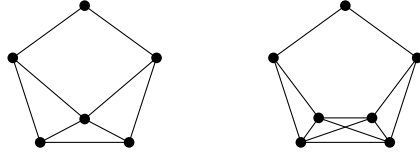


Figure 1: The graphs H_6 (left) and H_7 (right).

We improve this results using three main facts:

Lemma 1.3. [PST03] *Any minimal counterexample to $HC-\{\overline{K_3}\}$ has no dominating edges.*

Here, a *dominating edge* is an edge uv such that all vertices are adjacent to u or v (or both), and by *minimal counterexample* we mean that no proper induced subgraph is also a counterexample.

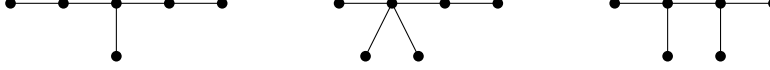


Figure 2: The trees T_1 , T_2 , and T_3 (left-to-right) associated with Lemma 1.5. Sometimes T_1 is called “E,” T_2 is called “cross,” and T_3 is called “H.”

Lemma 1.4. [CS12] Any $\overline{K_3}$ -free graph with

$$\omega(G) \geq \begin{cases} \lceil \frac{|G|}{4} \rceil & \text{if } |G| \text{ is even,} \\ \lceil \frac{|G|+3}{4} \rceil & \text{if } |G| \text{ is odd} \end{cases}$$

has $h(G) \geq \chi(G)$.

Lemma 1.5. $\{K_3, H\}$ -free graphs have chromatic number at most k , where (H, k) is any of the pairs $(K_1, 0)$, $(K_2, 1)$, $(P_4, 2)$, or $(T_1, 3)$, $(T_2, 3)$, or $(T_3, 3)$ [Ran04],¹ where T_i are shown in Figure 2.

By starting from a graph that is known to be an induced subgraph of any counterexample to $\text{HC-}\{\overline{K_3}\}$ and iteratively applying these lemmas, it is sometimes possible to find a new graph that must be an induced subgraph of any counterexample to $\text{HC-}\{\overline{K_3}\}$. Specifically, Lemma 1.3 implies, given a dominating edge uv , the existence of a vertex adjacent to neither u or v , which can attach onto the graph in some number of ways. If there are no dominating edges, we can use the other two lemmas to get the existence of one or more vertices that attach onto the graph in such a way as to introduce new dominating edges.

In particular, we prove:

Theorem 1.6. $\text{HC-}\{\overline{K_3}, H\}$ holds where $H \in \{H'_1, H'_2, \dots, H'_{33}\}$ is any of the graphs in Figure 3.

A small number of these cases are easily proved by hand, but most require hundreds of steps and are proved with computer assistance. We describe the relevant algorithms, prove their correctness, and prove some facts about their behavior and limitations in Section 2. We then provide the data necessary to verify Theorem 1.6 in Section 3, and we prove a few of the simple cases by hand for demonstration. We make some final remarks in Section 4.

1.1 Relationship to Past Results

Theorem 1.6 improves dramatically on Theorem 1.2. In fact, the only graphs in the statement of Theorem 1.2 that are not an induced subgraph of any graph in the statement of Theorem 1.6 are H_7 and K_7 .

¹In fact, $(T'_1, 3)$ is also such a pair where T'_1 is T_1 with a second leaf added to the third vertex in the 5-path [SWY21]. The author was not aware of this stronger result until after finding the main theorem of this part. Replacing T_1 by T'_1 in the relevant algorithm could potentially lead to slightly better results. In Part II, we do use the stronger version, restated as Lemma 5.1.

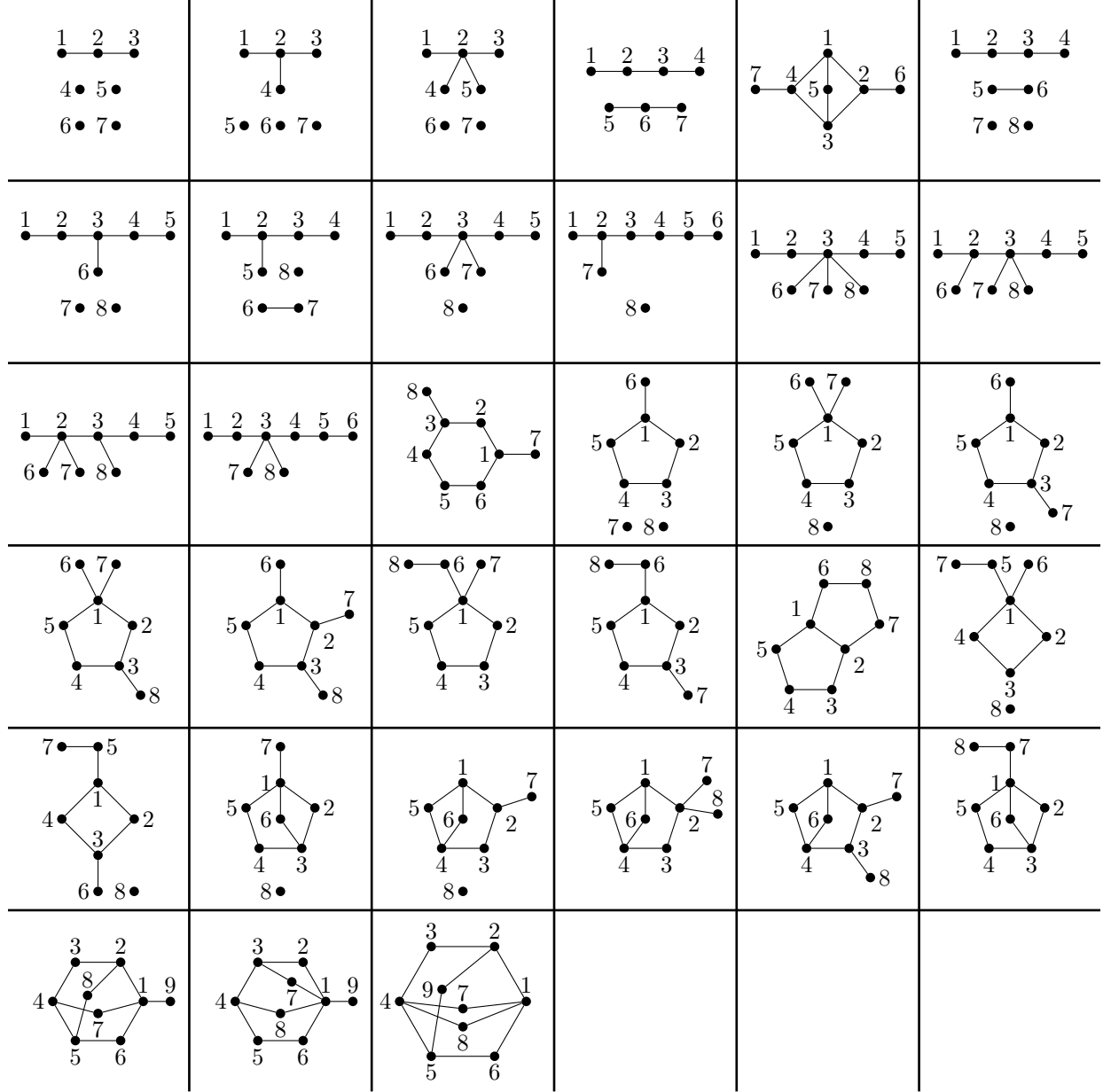


Figure 3: The complements of the graphs $H'_1, \dots, H'_{33}$, when read from left-to-right, top-to-bottom. We display the complements because these graphs have many edges, so it is almost always easier to recognize them from their complements. The graphs are first ordered by number of vertices, then by common structural elements. The labelings are used to choose edges in a consistent manner for algorithms.

By Proposition 2.7, if $\text{HC}\{\overline{K_3}, K_8\}$ holds then Algorithm 4 can prove this fact, though it may take an extremely long time. In fact, since the Ramsey number $R(3, 8) = 28$, the algorithm will most likely have to consider 27-vertex graphs at some point. Each such graph accrues a “weight” of $2^{27} = 134217728$ (see Section 4), which is already nearly 6 times the highest total weight of any of the proof steps in Figure 5 used to prove Theorem 1.6, and there are 477142 $\{\overline{K_3}, K_8\}$ -free graphs with 27 vertices [BGSP12], so one should not expect to improve K_7 to K_8 by running Algorithm 4 for any reasonable amount of time. Instead, one should just check these 477142 graphs directly for counterexamples to $\text{HC}\{\overline{K_3}\}$, since all graphs with 26 or fewer vertices have large enough clique number to satisfy the hypothesis of Lemma 1.4 due to the exact known values of the Ramsey numbers $R(3, k)$ with $k \leq 7$. Our initial hope was that Theorem 1.6 would make this check very easy, but this could not be further from the truth. In fact, *all* $\{\overline{K_3}, K_8\}$ -free graphs on 27 vertices contain *all* of the H'_i (and H_7), so Theorem 1.6 is no help at all. Instead, we perform a check using other means in Appendix A.3, and get:

Theorem 1.7. *$\text{HC}\{\overline{K_3}, K_8\}$ holds.*

and also

Corollary 1.8. *A minimal counterexample to $\text{HC}\{\overline{K_3}\}$ has at least 31 vertices.*

Proof. By the theorem above, a minimal counterexample must have a K_8 . If such a graph has fewer than 31 vertices, Lemma 1.4 implies it is not a counterexample to $\text{HC}\{\overline{K_3}\}$. $\square$

Since $R(3, 9) = 36$, it seems computationally infeasible to improve K_8 to K_9 by brute-force.

The fact that H_7 was not improved to a larger graph is due primarily to the combination of the facts that the “ $\overline{C_4}$ -core” of H_7 is all of H_7 (see Section 2.4.2), and H_7 has no dominating edges. The only way H_7 could be improved by the algorithms in this part is directly starting from H_7 itself, but this is a poor starting place given the lack of dominating edges.

2 Algorithms

2.1 Basic Algorithm: No Dominating Edges

Suppose all counterexamples to $\text{HC}\{\overline{K_3}\}$ are known to contain induced subgraph H , i.e. $\text{HC}\{\overline{K_3}, H\}$ holds. If H has a dominating edge uv , then by Lemma 1.3, all minimal counterexamples to $\text{HC}\{\overline{K_3}\}$ have an induced H plus a vertex w adjacent to neither u nor v . Since all minimal counterexamples have this property, any graph containing any minimal counterexample as an induced subgraph will also have this property, i.e. this holds for all counterexamples, not just minimal ones. Denote by H_{uv} the set of graphs

formed by attaching a new vertex w to H in all possible ways while making sure w is not adjacent to u or v (there are only finitely many such attachments). We see that $\text{HC}-(\{\overline{K_3}\} \cup H_{uv})$ holds. In fact, we can remove from H_{uv} all graphs containing $\overline{K_3}$ and the result will still hold, since of course no $\overline{K_3}$ -free graph could contain a graph that itself contains $\overline{K_3}$. Let H_{uv}^- denote this “reduced” set of graphs containing no $\overline{K_3}$.

Suppose that some $H' \in H_{uv}^-$ has a dominating edge $u'v'$. Applying the same logic as before, we see that $\text{HC}-(\{\overline{K_3}\} \cup (H_{uv}^- \setminus H') \cup (H')_{u'v'}^-)$ holds. We can repeat this process for as long as we like, at least until we run into graphs with no more dominating edges.

If we impose some additional restriction on counterexamples, say that they satisfy some property P , and modify this process to also remove from H_{uv} all graphs satisfying P , in some cases we can completely run out of graphs, concluding that all counterexamples $\text{HC}-(\{\overline{K_3}\})$ satisfy P . This is possible when P is *monotone*, that is, if H satisfies P and G has induced subgraph H then G also satisfies P . Consider then Algorithm 1.

Algorithm 1: Prove that all counterexamples to $\text{HC}-(\{\overline{K_3}\})$ satisfy monotone property P .

Input: Monotone property P , graph H that is known to be an induced subgraph of any counterexample to $\text{HC}-(\{\overline{K_3}\})$.

Let $A \leftarrow \{H\}$;

while A is nonempty **do**

 Let $H' \in A$;

 Let $A \leftarrow A \setminus \{H'\}$;

if H' has no dominating edges **then**

return FAILURE;

else

 Let uv be a dominating edge of H' ;

for $N \in 2^{V(H') \setminus \{u,v\}}$ **do**

 Let G be H' plus a new vertex w such that the neighbors of w are N ;

if G has no stable set of size 3, G does not satisfy P , and G is A -free **then**

 Let $A \leftarrow A \cup \{G\}$;

return SUCCESS;

We now prove the correctness of the algorithm.

Lemma 2.1. *If Algorithm 1 returns SUCCESS, then all counterexamples to $\text{HC}-(\{\overline{K_3}\})$ satisfy property P .*

Proof. We claim that, inductively, at the start of each iteration of the main **while** loop, A is always set of graphs such that all counterexamples to $\text{HC}-(\{\overline{K_3}\})$ either satisfy P or contains one of the graphs in A as an induced subgraph. The base case, when $A = \{H\}$, is clear.

Suppose $H' \in A$ has a dominating edge uv . Then if a counterexample to $\text{HC}-(\{\overline{K_3}\})$ contained H' as induced subgraph, it must also have a vertex w that is nonadjacent to both u and v . Thus if a counterexample contained H' , it must actually contain one of the graphs G obtained by attaching to H in all possible ways

while being nonadjacent to u and v . If G has a stable set of size 3 then it is obviously not a counterexample to $\text{HC-}\{\overline{K_3}\}$ (and any graph containing G is also not a counterexample to $\text{HC-}\{\overline{K_3}\}$). Additionally, if G satisfies P , then since P is monotone, any counterexample containing G will also satisfy P . Finally, note that the set of A -free graphs is the same as the set of $(A \cup \{G\})$ -free graphs if G contains one of the graphs in A as an induced subgraph. Thus if a counterexample to $\text{HC-}\{\overline{K_3}\}$ contained H' as induced subgraph, either it satisfies P or it contains as an induced subgraph one of the graphs contained in A at the end of the inner **for** loop (which is the same as A at the beginning of the next iteration of the **while** loop). This completes the inductive step.

The condition for the **while** loop to terminate and the algorithm to return **SUCCESS** is that A is empty at the start of the **while** loop. If this is the case, that means that all counterexamples to $\text{HC-}\{\overline{K_3}\}$ satisfy P by the claim just proven. $\square$

Of interest to us, the property of containing a particular induced subgraph H' is monotone, so in some cases this can be used to prove that all counterexamples to $\text{HC-}\{\overline{K_3}\}$ have an induced H' , i.e. $\text{HC-}\{\overline{K_3}, H'\}$ holds, so long as the graphs H and H' are chosen carefully and we get lucky. For an example, see the proof of case I_{21} in Section 3 (and see Figure 4 for a picture of the complement of this graph). However, it is quite likely that one will run into a graph with no dominating edges, no stable set of size three, and with no induced H' , in which case the algorithm will return **FAILURE**. We can often deal with these cases using Lemmas 1.4 and 1.5, by a process described in the next subsection.

2.2 Getting Unstuck: Four-Clique Covers

2.2.1 Assuming No Vertex Dominates H

The restriction that a graph has no stable set of size three means that if two vertices have a common nonneighbor, they must be adjacent. Thus if H is an induced subgraph of G and no vertex in $G \setminus H$ is adjacent to all vertices in H , then the vertices of $G \setminus H$ partition into cliques based on the intersection of their neighborhood with H . Additionally, to avoid stable sets of size 3 and monotone property P , many pairs of these cliques must be *complete* to each other, i.e. there must be no nonedges between vertices in those two cliques. In many cases, one can find four cliques Q_1, Q_2, Q_3, Q_4 that cover G such that

$$\sum_{i=1}^4 |Q_i \cap V(H)| \geq |H| + 2,$$

so by Pigeonhole Principle, $\omega(G) \geq \lceil \frac{|G|+2}{4} \rceil$. Note that

$$\left\lceil \frac{|G|+2}{4} \right\rceil \geq \begin{cases} \lceil \frac{|G|}{4} \rceil & \text{if } |G| \text{ is even,} \\ \lceil \frac{|G|+3}{4} \rceil & \text{if } |G| \text{ is odd,} \end{cases}$$

so by Lemma 1.4, G is not a counterexample to $\text{HC-}\{\overline{K_3}\}$. If this is the case, we conclude that all counterexamples with an induced H either satisfy property P or have a vertex adjacent to every vertex in H . The graph obtained by adding to H a vertex adjacent to all vertices in H is known as the *coning* of H , and we denote it by $H \vee K_1$ (for we will generalize this construction in a later subsection). In other words, all counterexamples with an induced H either satisfy property P or have an induced $H \vee K_1$.¹

The usefulness of this is that $H \vee K_1$ has a dominating vertex, and therefore dominating edges, for all H . Hence it can be used to “unstick” Algorithm 1 when one encounters a graph with no dominating edges, i.e. when the algorithm would otherwise return FAILURE. More importantly, this check can be done completely automatically by Algorithm 2.

Algorithm 2: Prove that all counterexamples to $\text{HC-}\{\overline{K_3}\}$ that have an induced H either satisfy monotone property P or have an induced $H \vee K_1$.

Input: Monotone property P , graph H .

Let $A \leftarrow \emptyset$;

for $N \in 2^{V(H)} \setminus \{V(H)\}$ **do**

 Let G be H plus a new vertex w such that the neighbors of w are N ;

if G has no stable set of size 3 and G does not satisfy P **then**

 Let $A \leftarrow A \cup \{N\}$;

Let $B \leftarrow \emptyset$;

for $(N_1, N_2) \in \binom{A}{2}$ **do**

 Let G be H plus new vertices u and v such that the neighbors of u are N_1 and neighbors of v are N_2 ;

if G has a stable set of size 3 or G satisfies P **then**

 Let $B \leftarrow B \cup \{(N_1, N_2)\}$;

Let G be H plus vertices $\{u_i\}_{i=1}^{|A|}$ such that the neighbors of each u_i are N_i (where $A = \{N_1, \dots, N_{|A|}\}$), and u_i and u_j are adjacent iff $\{N_i, N_j\} \in B$;

if there are four cliques Q_1, Q_2, Q_3, Q_4 covering G such that $\sum_{i=1}^4 |Q_i \cap V(H)| \geq |H| + 2$ **then**

return SUCCESS;

else

return FAILURE;

We now prove the correctness of this algorithm.

Lemma 2.2. *If Algorithm 2 returns SUCCESS, then all counterexamples to $\text{HC-}\{\overline{K_3}\}$ with an induced H*

¹This is the main idea in Bosse’s proof of $\text{HC-}\{\overline{K_3}, W_5\}$, with $H = C_5$ and P omitted [Bos19], although that proof used some additional cleverness beyond the algorithm we describe.

either satisfy property P or have an induced $H \vee K_1$.

Proof. Suppose for contradiction Algorithm 2 returned SUCCESS but there is a counterexample G' to HC- $\{\overline{K_3}\}$ with an induced H but no induced $H \vee K_1$ that does not satisfy property P .

Since there is no vertex in G' adjacent to all vertices in the induced H , the vertices of $G' \setminus H$ partition into at most $2^{|H|} - 1$ cliques based on their neighbors in H . Specifically, if two vertices u, v have identical neighbors in H , then by assumption that G' has no induced $H \vee K_1$, u and v have a common nonneighbor $w \in H$ and must be therefore be adjacent to avoid $\{u, v, w\}$ being a stable set of size 3. Some of these potential cliques may be forced to be empty; for instance if u, v are not adjacent in H , all vertices in $G' \setminus H$ must be adjacent to at least one of u and v . It is clear that, after the first **for** loop, A contains the set of possible neighborhoods in H of vertices in $G' \setminus H$, since the algorithm filters out all neighborhoods such that if there was a vertex with that neighborhood, G' would either have a stable set of size 3 or satisfy property P (here using the fact that P is assumed to be monotone). Note that not all possible neighborhoods are achieved in G' , but all vertices in $G' \setminus H$ have one of the neighborhoods in A . We identify the cliques in the clique partition of $G' \setminus H$ with the elements of A .

Next, suppose $u \in V(G)$ is in clique N_1 and $v \in V(G)$ is in clique N_2 , for some $N_1 \neq N_2$. If the graph formed by adding to H the vertices u and v with neighborhoods N_1 and N_2 , respectively, has either a stable set of size 3 or satisfies P , then u and v must be adjacent in G' . Since this holds for all vertices in N_1 and N_2 , in fact N_1 and N_2 are complete to each other in G' . Then it is clear that, after the second **for** loop, if $\{N_1, N_2\} \in B$, then N_1 and N_2 are complete in G' (though it is possible other pairs of cliques in $G' \setminus H$ could be complete to each other but are not captured by this **for** loop).

Since the algorithm returned SUCCESS, there is a covering of the graph G formed after the second **for** loop by four cliques Q_1, Q_2, Q_3, Q_4 such that $\sum_{i=1}^4 |Q_i \cap V(H)| \geq |H| + 2$. Then there is an analogous clique cover of G' by four cliques Q'_1, Q'_2, Q'_3, Q'_4 such that $\sum_{i=1}^4 |Q'_i \cap V(H)| \geq |H| + 2$. In particular,

$$Q'_i = (Q_i \cap V(H)) \cup \bigcup_{u_j \in Q_i} (\text{clique in } G' \text{ corresponding to } N_j)$$

for $i = 1, 2, 3, 4$ is such a clique partition. The inequality $\sum_{i=1}^4 |Q'_i \cap V(H)| \geq |H| + 2$ is clear, so we just need to check that the Q'_i are indeed cliques. For two $u, v \in Q'_i$ with $u \neq v$, there are four cases to check:

- If $u, v \in V(H)$, then since Q_i is a clique in G , uv is an edge of H , so an edge in G' .
- If $u \in V(H)$ and $v \in (\text{clique corresponding to } N_j)$ for some j (or symmetrically $v \in V(H)$ and $u \in \text{clique corresponding to } N_j$), then since Q_i is a clique in G , uu_j is an edge of G , which means $u \in N_j$ and uv is an edge in G' by definition of N_j .

- If $u, v \in$ (clique corresponding to N_j) for some j , then obviously uv is an edge in G' .
- If $u \in$ (clique corresponding to N_j) and $v \in$ (clique corresponding to N_k) for some $j \neq k$, then since Q_i is a clique in G , $u_j u_k$ is an edge of G , so $\{N_j, N_k\} \in B$ and the cliques corresponding to N_j and N_k are complete to each other in G' , i.e. uv is an edge in G' .

This implies by Pigeonhole Principle that $\omega(G') \geq \lceil \frac{|G'|+2}{4} \rceil$, but by Lemma 1.4 this contradicts that G' is a counterexample to $\text{HC-}\{\overline{K_3}\}$. $\square$

2.2.2 Weaker or Stronger Assumptions

Sometimes we can get different results than Algorithm 2 with different assumptions than there being no vertex adjacent to all vertices in an induced H . For instance, if we allow G to have vertices complete to the induced H , but assume that they form a clique, then all vertices in G still partition into cliques based on the intersection of their neighborhood with H , only now this neighborhood is allowed to be all of H . This assumption implies that $c(G, H)$ is $\overline{K_2}$ -free, where $c(G, H)$ is the subgraph of $G \setminus H$ consisting of vertices complete to H .

We can make an even weaker assumption on G if we enforce only that $c(G, H)$ has clique cover number equal to at most some constant k . This is equivalent to the assumption that $\overline{c(G, H)}$ is k -colorable, since a cover of $c(G, H)$ by k cliques is a cover of $\overline{c(G, H)}$ by k independent sets, i.e. color classes. By Lemma 1.5, $c(G, H)$ having clique cover number at most k is implied by $c(G, H)$ being F_k -free, where $F_0 = \overline{K_1} \cong K_1$, $F_1 = \overline{K_2}$, $F_2 = \overline{P_4} \cong P_4$, and $F_3 = \overline{T_1}$.² Denote by $G \vee H$ (read “ G join H ”) the graph formed by adding to $G \cup H$ all edges between G and H . In other words, $G \vee H = \overline{\overline{G} \sqcup \overline{H}}$, with $\sqcup$ the disjoint union of graphs. Then consider Algorithm 3.

The correctness of this algorithm follows from essentially the same logic as Algorithm 2, so we skip some details of the proof.

Lemma 2.3. *If Algorithm 3 returns SUCCESS, then all counterexamples to $\text{HC-}\{\overline{K_3}\}$ with an induced H either satisfy property P or have an induced $H \vee F_k$.*

Proof. The crux of the argument, as in the previous proof, is that if the G constructed by Algorithm 3 has a cover by cliques $\{Q_1, Q_2, Q_3, Q_4\}$ with $\sum_{i=1}^4 |Q_i \cap V(H)| \geq |H| + 2$, then any potential counterexample G' also has a cover by cliques $\{Q'_1, Q'_2, Q'_3, Q'_4\}$ with $\sum_{i=1}^4 |Q'_i \cap V(H)| \geq |H| + 2$, and therefore has clique number at least $\frac{|G'|+2}{4}$. Now $c(G', H)$ is assumed to have clique cover number at most k , so it is covered by

²One could also take $F_3 = \overline{T_2}$ or $\overline{T_3}$. This leads to slight variations in the algorithm and could potentially lead to different results. We have not investigated these alternatives.

Algorithm 3: Prove that all counterexamples to $\text{HC-}\{\overline{K_3}\}$ that have an induced H either satisfy monotone property P or have an induced $H \vee F_k$, for chosen k .

Input: Monotone property P , graph H , integer $k \in \{0, 1, 2, 3\}$.
 Let $A \leftarrow \emptyset$;
for $N \in 2^{V(H)} \setminus \{V(H)\}$ **do**
 Let G be H plus a new vertex w such that the neighbors of w are N ;
 if G has no stable set of size 3 and G does not satisfy P **then**
 Let $A \leftarrow A \cup \{N\}$;
 Let $B \leftarrow \emptyset$;
for $(N_1, N_2) \in \binom{A}{2}$ **do**
 Let G be H plus new vertices u and v such that the neighbors of u are N_1 and neighbors of v are N_2 ;
 if G has a stable set of size 3 or G satisfies P **then**
 Let $B \leftarrow B \cup \{\{N_1, N_2\}\}$;
 Let $C \leftarrow \emptyset$;
for $N \in A$ **do**
 Let G be H plus new vertices u and w such that the neighbors of u are N and neighbors of w are $V(H)$;
 if G has a stable set of size 3 or G satisfies P **then**
 Let $C \leftarrow C \cup \{\{N\}\}$;
 Let G be H plus vertices $\{u_i\}_{i=1}^{|A|} \cup \{w_i\}_{i=1}^k$ such that the neighbors of each u_i are N_i (where $A = \{N_1, \dots, N_{|A|}\}$), u_i and u_j are adjacent iff $\{N_i, N_j\} \in B$, the neighbors of each w_i are $V(H)$, none of the w_i and w_j are adjacent, and w_i and w_j are adjacent iff $N_j \in C$;
if there are four cliques Q_1, Q_2, Q_3, Q_4 covering G such that $\sum_{i=1}^4 |Q_i \cap V(H)| \geq |H| + 2$ **then**
 return SUCCESS;
else
 return FAILURE;

k cliques $R_1, \dots, R_k$. The cliques of G' are given by

$$Q'_i = (Q_i \cap V(H)) \cup \bigcup_{u_j \in Q_i} (\text{clique in } G' \text{ corresponding to } N_j) \cup \bigcup_{w_j \in Q_i} R_j.$$

We must check that these are cliques. For $u, v \in Q'_i$ with $u \neq v$, the new cases not in the previous proof are:

- If $u \in V(H)$ and $v \in R_j$ for some j , then obviously uv is an edge of G .
- If $u \in$ clique corresponding to N_j and $v \in R_\ell$ for some j, ℓ , then $N_j \in C$, so if uv was a nonedge of G' , G' would either have a stable set of size 3 or would satisfy P .
- If $u, v \in R_j$ for some j , then uv is an edge of G by assumption that R_j was a clique.

It is not possible for $u \in R_j$ and $v \in R_\ell$ for some $j \neq \ell$ to both be in the same Q'_i since the $w_i \in V(G)$ form a stable set.

Therefore if $c(G', H)$ has clique cover number at most k , G' is not a counterexample to $\text{HC-}\{\overline{K_3}\}$. So by Lemma 1.5, $c(G', H)$ has an induced F_k , so G' has an induced $H \vee F_k$. $\square$

Recall that the purpose of Algorithms 2 and 3 is to “unstick” Algorithm 1 by introducing new dominating edges. However, the results of these algorithms in general are much stronger than what is necessary to unstick Algorithm 1. Define $\text{dom}(H)$ to be the set of graphs formed from H by adding one vertex, such that the resulting graph has at least one dominating edge. Then weakest possible variation of this idea would be Algorithm 2 but replacing P with the logical-or of P and the negation of being $\text{dom}(H)$ -free. Since $H \vee K_1 \in \text{dom}(H)$ for any H , if this modified algorithm returns SUCCESS then all counterexamples to $\text{HC-}\{\overline{K_3}\}$ that have an induced H either satisfy P or have one of the members of $\text{dom}(H)$ as an induced subgraph.

2.3 Combined Algorithm

The algorithms in Section 2.2 can be easily incorporated into Algorithm 1 to form Algorithm 4.

The correctness of this algorithm follows almost immediately from Lemmas 2.1, 2.2, and 2.3. There are just two parts of the algorithm that have not been explained. First, we add a check at the beginning of the algorithm, and if H already satisfies P then we immediately return success. Second, the final check performed is that if H' is not a counterexample to $\text{HC-}\{\overline{K_3}\}$, then any counterexample to $\text{HC-}\{\overline{K_3}\}$ that has an induced H' must actually have an H' plus one vertex, though the edges incident to this vertex have no restriction other than that the resulting graph is $\overline{K_3}$ -free and does not satisfy P .

In actuality, Algorithm 4 never had to perform the final check on any input we have given it in the process of discovering Theorem 1.6. Thus we never actually implemented the final check. However, it is not difficult to construct examples of H' such that the second-to-last check (Algorithm 2 run with modified P) returns FAILURE; for instance this will happen if it is given the complement of any triangle-free 5-chromatic graph. The smallest such graphs have 22 vertices [JR95]. Thus if one was to extend our computations substantially farther, one might want to add this additional check.

This algorithm’s behavior is dependent on the method of choosing H' and a dominating edge uv in cases that A has more than one element or H' has more than one dominating edge. There are theoretical reasons to prefer some methods over others, which we describe in the next subsection. We describe some miscellaneous implementation details in Appendix A.1.1.

Algorithm 4: Prove that all counterexamples to $\text{HC-}\{\overline{K_3}\}$ satisfy monotone property P .

Input: Monotone property P , graph H that is known to be an induced subgraph of any counterexample to $\text{HC-}\{\overline{K_3}\}$, boolean E .

if H satisfies P **then**
 return SUCCESS;

Let $A \leftarrow \{H\}$;

while A is nonempty **do**
 Let $H' \in A$;
 Let $A \leftarrow A \setminus \{H'\}$;
 if H' has a dominating edge **then**
 Let uv be a dominating edge of H' ;
 for $N \in 2^{V(H') \setminus \{u,v\}}$ **do**
 Let G be H' plus a new vertex w such that the neighbors of w are N ;
 if G has no stable set of size 3, G does not satisfy P , and G is A -free **then**
 Let $A \leftarrow A \cup \{G\}$;

else
 for $k \in \{3, 2, 1, 0\}$ (in the order written) **do**
 if Algorithm 3 returns SUCCESS when given input (P, H', k) **then**
 if $H' \vee F_k$ has no stable set of size 3, $H' \vee F_k$ does not satisfy P , and $H' \vee F_k$ is A -free **then**
 Let $A \leftarrow A \cup \{H' \vee F_k\}$;
 Break out of the inner **for** loop;

if the previous **for** loop was not broken out of **then**
 if Algorithm 2 returns SUCCESS when given input $(P \vee \neg \text{"dom}(H')\text{-free"}, H')$ **then**
 for $H'' \in \text{dom}(H')$ **do**
 if H'' has no stable set of size 3, H'' does not satisfy P , and H'' is A -free **then**
 Let $A \leftarrow A \cup \{H''\}$;

else
 if H' is not a counterexample to $\text{HC-}\{\overline{K_3}\}$ **then**
 for $N \in 2^{V(H')}$ **do**
 Let G be H' plus a new vertex w such that the neighbors of w are N ;
 if G has no stable set of size 3, G does not satisfy P , and G is A -free **then**
 Let $A \leftarrow A \cup \{G\}$;

else
 return FAILURE;

return SUCCESS;

2.4 Behavior of the Algorithm

2.4.1 Ordering

As we mentioned, the intermediate steps of the algorithm depend on the method employed to choose H' from A and one dominating edge from H' in the event there are multiple possibilities. In this paper we are only interested in deterministic ways of doing this. Let $\mathcal{A}$ be one particular implementation of Algorithm 4, i.e.

one possible method of choosing graphs and edges. More precisely, $\mathcal{A}$ consists of a collection of (computable, deterministic) functions: one that chooses an element $H' \in A$, one that chooses a dominating edge $uv \in H'$ if H' has at least one such edge, and some that determine the order to consider graphs in the innermost **for** loops. These functions are allowed to take as input auxiliary memory that may be modified at any point in the algorithm. For instance, $\mathcal{A}$ may remember some vertex labeling in graphs, remember the order that graphs were added or removed from A , remember the number of iterations of various loops, etc. and use this information to determine which graph or dominating edge to consider next.

Let us write $H \vdash_{\mathcal{A}} P$ (read “ H proves P (under implementation $\mathcal{A}$)”) if the $\mathcal{A}$ implementation of Algorithm 4 returns SUCCESS given input (H, P) , and write $H \vdash_{\mathcal{A}} G$ if $H \vdash_{\mathcal{A}} P$ when P is the property of having an induced G . There are several highly desirable traits that we would like to be true:

1. For any H , $H \vdash_{\mathcal{A}} H$, i.e. $\vdash_{\mathcal{A}}$ is reflexive.
2. If $H \vdash_{\mathcal{A}} G$ and $G \vdash_{\mathcal{A}} P$ then $H \vdash_{\mathcal{A}} P$, i.e. $\vdash_{\mathcal{A}}$ is transitive.
3. If H is an induced subgraph of H' and $H \vdash_{\mathcal{A}} P$ then $H' \vdash_{\mathcal{A}} P$, i.e. $\vdash_{\mathcal{A}}$ is monotone decreasing on the left with respect to taking induced subgraphs.
4. If G is an induced subgraph of G' and $H \vdash_{\mathcal{A}} G'$ then $H \vdash_{\mathcal{A}} G$, i.e. $\vdash_{\mathcal{A}}$ is monotone increasing on the right with respect to taking induced subgraphs. More generally, if P is a stricter property than P' in the sense that any graph satisfying P' also satisfies P , and $H \vdash_{\mathcal{A}} P'$, then $H \vdash_{\mathcal{A}} P$.

This would make $\vdash_{\mathcal{A}}$, when restricted to the case that P is the property of having an induced G , a quasi-order on graphs that respects the induced subgraph partial order. The first property is obviously true for any $\mathcal{A}$ due to the check at the very beginning of Algorithm 4. Additionally, there is a simple condition on $\mathcal{A}$ needed to make property (4) hold:

Theorem 2.4. *If $\mathcal{A}$ has the following properties, then if P is a stricter property than P' and $H \vdash_{\mathcal{A}} P'$, then $H \vdash_{\mathcal{A}} P$:*

- $\mathcal{A}$ always chooses the most recently added H' from A ,
- the order that graphs are added to A depends only on the H' chosen in that iteration of the **while** loop (not on A or P or anything else), and³
- the dominating edge chosen when there are multiple possibilities depends on H' only as well.

³Note that this is only a condition on the order that the graphs are considered. Obviously whether or not they are added to A depends on P and A , but they must be considered in the same order no matter what.

Proof. Suppose $H \vdash_{\mathcal{A}} P'$, and consider the set of graphs S' that are ever a member of A when the algorithm is run with input (H, P') . Since $H \vdash_{\mathcal{A}} P'$, this set is finite. Construct a directed acyclic graph D' with vertex set S' and an edge from v and u if v is an induced subgraph of u . Then the order that graphs are removed from A corresponds to a depth-first search on D' , by virtue of always choosing the latest-added graph to A in each iteration.

If P is a stricter property than P' , then the corresponding set S when the algorithm is run with input (H, P) is a subset of S' , and the corresponding directed acyclic graph D is a subgraph of D' . To see this, consider the sequences $\mathcal{S}_1 = \{A_1, \dots, A_k\}$ and $\mathcal{S}_2 = \{A'_1, \dots, A'_\ell\}$ consisting of the set A snapshotted at the beginning of each iteration of the main **while** loop when the algorithm is run with input (H, P) and (H, P') , respectively.

We will show there is a injection $f : \mathcal{S}_1 \rightarrow \mathcal{S}_2$ such that $A \subset f(A)$ for all $A \in \mathcal{S}_1$. More strongly, the relative order that graphs were added to A is the same as the relative order they were added in $f(A)$. This injection is constructed as follows: $f(A_i)$ is the A'_j such that if $H_i \in A_i$ is the graph chosen by $\mathcal{A}$ at the beginning of the **while** loop at that iteration (i.e. the most recently added graph to A at that point in time), then H_i is also the graph chosen in A'_j . To see that this is well-defined and has the desired properties, we use induction. This is obviously true for $A_1 = \{H\} = A'_1 = f(A_1)$.

To see that $f(A_{i+1})$ exists and has the desired properties given $f(A_i)$ does, either H_{i+1} is in A_i or not. If it is, then H_{i+1} is also in $A'_j = f(A_i)$. On some iteration $j_2 > j$ of the algorithm run on (H, P') , H_{i+1} must be the most recently added graph in A_{j_2} . This is the chosen value of $f(A_{i+1})$. The fact that the rest of A_{i+1} is in $f(A_{i+1})$ with the same relative order follows from inductive hypothesis because the rest of A_{i+1} is older than H'_{i+1} and also appears in A_i in the same relative order. Thus the rest of A_{i+1} appears in $f(A_i)$ and, since the rest is older than H'_{i+1} , is untouched until after iteration j_2 .

If H_{i+1} is not in A_i , then it was added to A in iteration i under input (H, P) , possibly alongside some other graphs. Therefore if $A'_j = f(A_i)$, H_{i+1} was added to A in iteration j under input (H, P') , along with the same other graphs as in input (H, P) in the same order, plus possibly some additional graphs interspersed with these (since P is stricter than P'). Thus at some iteration $j_2 > j$, H_{i+1} will be the most recently added graph under input (H, P') , and the rest of the logic follows similarly to the case in the previous paragraph. This completes the inductive step.

Thus we can conclude $S \subseteq S'$ and, more weakly, $H \vdash_{\mathcal{A}} P$. □

Unfortunately, we are unable to find a version of Algorithm 4 that makes all of the remaining properties true. However, restricted to only the constituent parts of Algorithm 4 (namely Algorithms 1 and 3), we can come up with results close to the above properties. We write $H \vdash_e P$ if either H satisfies P or Algorithm 1

returns SUCCESS given input (H, P) and the implementation described in the following two paragraphs. Similarly we write $H \vdash_e P$ if either H satisfies P or Algorithm 2 returns SUCCESS given input (H, P) . Note that the output of Algorithm 2 does not depend on implementation details.

There is a method of choosing dominating edges and elements of A that makes $\vdash_e$ have these properties. Namely, we first label the vertices of H with integers 1 through $|V(H)|$. Whenever a vertex is added to a graph, it gets the smallest integer label not yet used in that graph. For $H' \vee F_k$, the labels of the vertices in the F_k part are the next $|V(F_k)|$ integers, and relative ordering of vertices in the F_k part is always the same (though this is not strictly necessary). Edges uv are identified with the tuple $(\max(u, v), \min(u, v))$, where vertices are identified by their label. The chosen dominating edge is the minimum edge under the lexicographical ordering with this identification. In essence, the dominating edge chosen is the one whose younger endpoint is as old as possible, breaking ties by looking at the other endpoint.

Next, whenever multiple graphs are to be added to A , notice that they are all just H' plus one vertex, possibly with some restrictions on the neighbors of the new vertex. We add these to A in the order such that if N_1 and N_2 are two different sets of the labels of the neighbors of the newly-added vertex, the graph corresponding to N_1 is added before N_2 iff, after sorting N_1 and N_2 and writing them as tuples, N_1 comes lexicographically after N_2 . If N_1 is a prefix of N_2 , it comes after N_2 ; otherwise the one that comes second is the one with the smallest element in the symmetric difference of N_1 and N_2 .

Theorem 2.5. *Under the method of choosing dominating edges and graphs described in the previous paragraphs:*

1. *If $H \vdash_e G$ and $G \vdash_e P$ then $H \vdash_e P$.*
2. *If H is an induced subgraph of H' and $H \vdash_e P$ then $H' \vdash_e P$.*
3. *If $H \vdash_e P$ then $H \vdash_{\mathcal{A}} P$, where the method for choosing graphs and dominating edges in $\mathcal{A}$ is also as in the previous paragraphs.*

Proof. First, (3) is obvious since Algorithm 4 performs exactly the same computations as Algorithm 1 given that implementation $\mathcal{A}$ and the fact that $H \vdash_e P$.

For a particular graph H , consider the set $S(H)$ of minimal graphs H' containing H so that no dominating edge of H is a dominating edge of H' . If $H \vdash_e P$, then for each $H' \in S(H)$, either H' satisfies P or H' has a dominating edge, because if H' did not have a dominating edge and did not satisfy P , then Algorithm 1 would construct it from H and then return FAILURE. In particular, because the order for choosing dominating edges considers all dominating edges of H before any edges introduced by adding vertices to H , all members

H' of $S(H)$ will be constructed by the algorithm unless a proper induced subgraph of H' satisfies P . Let $S_P(H)$ be the subset of $S(H)$ that does not satisfy property P .

If $H \vdash_e P$, then for each $H' \in S_P(H)$, for each $H'' \in S(H')$, either H'' satisfies P or H'' has a dominating edge. We continue in this manner, but because $H \vdash_e P$, the algorithm terminates at some point, and there must be a k such that $S_P^k(H)$ is empty, where $S_P^0(H) = \{H\}$ (unless H already satisfies P , in which case $S_P^0(H) = \emptyset$) and

$$S_P^i(H) = \bigcup_{H' \in S_P^{i-1}(H)} S_P(H')$$

for $i \geq 1$.

Now suppose H is an induced subgraph of I and $H \vdash_e P$. Then for any graph $I' \in S(I)$, there is a graph $H' \in S(H)$ so that H' is an induced subgraph of I' . This is because in I' none of the dominating edges of I are dominating, which means none of the dominating edges of the H contained in the I are dominating in I' , so I' must contain one of the graphs in $S(H)$. Similarly, for any $I' \in S_P(I)$, there is an $H' \in S_P(H)$ such that H' is an induced subgraph of I' . Inductively one can then see that for any i and $I' \in S_P^i(I)$, there is an $H' \in S_P^i(H)$ such that H' is an induced subgraph of I' . Setting $i = k$, since $S_P^k(H)$ is empty, $S_P^k(I)$ must also be empty, which means $I \vdash_e P$. This is (2).

For (1), let k_G be the minimum k so that $S_P^k(G) = \emptyset$ and k_H the minimum k so that $S_G^k(H) = \emptyset$, where the subscript G means to consider the property of having an induced G . Any graph $I \in S_P^{k_H}(H)$ has an induced G and therefore (by the reasoning in the previous paragraph), $S_P^{k_G}(I) = \emptyset$. But it is to prove by induction on b that

$$S_P^{a+b}(H) = \bigcup_{I \in S_P^a(H)} S_P^b(I),$$

which proves (1) by setting $a = k_H$ and $b = k_G$.

□

Similarly, for Algorithm 2:

Theorem 2.6. *If H is an induced subgraph of H' , $H \vee K_1$ satisfies P , and $H \vdash_c P$ then $H' \vdash_c P$.*

Proof. Note if H' has $H \vee K_1$ as an induced subgraph, $H' \vdash_c P$ is immediate. If not, then consider first the case that H' is H plus one vertex v , and consider the sets A and B constructed for the algorithm given input (H, P) and corresponding sets A' and B' constructed given input (H', P) . Then since $H \vee K_1$ satisfies P , we find every member of A' is either N_i or $N'_i = N_i \cup \{v\}$ for some $N_i \in A$. Also, if the neighbors of v are N , then $N \neq V(H)$, and N is not in A' (only possibly $N \cup \{v\}$) due to the restriction that there are no stable sets of size 3.

Now the following are members of B' , assuming the relevant sets are members of A' :

- $\{N_i, N'_i\}$, since there is some $v \in H$ with $v \notin N_i$ and $v \notin N'_i$.
- $\{N_i, N_j\}$, $\{N_i, N'_j\}$, and $\{N'_i, N'_j\}$ if $\{N_i, N_j\} \in B$, since if H plus two vertices with neighbor sets N_i and N_j has a stable set of size 3 or satisfies P , H' plus two vertices with neighbor sets $(N_i \text{ or } N'_i)$ and $(N_j \text{ or } N'_j)$ also has a stable set of size 3 or satisfies P , which can be seen just by deleting v from the graph after adding the two new vertices.

Thus if Q_1, Q_2, Q_3, Q_4 are four cliques covering $G = H \cup \{u_i\}$ with $\sum_{i=1}^4 |Q_i \cap V(H)| \geq |H| + 2$, then there are Q'_1, Q'_2, Q'_3, Q'_4 are four cliques covering $G' = H' \cup \{u_i\} \cup \{u'_i\}$ with $\sum_{i=1}^4 |Q'_i \cap V(H')| \geq |H'| + 2$. Let the neighbors of v in H be N , which we can take without loss of generality to be in Q_1 . Then set

$$Q'_i = \{u_i \mid u_i \in Q_i\} \cup \{u'_i \mid u_i \in Q_i\} \cup \begin{cases} \{v, u'\} & \text{if } i = 1 \\ \emptyset & \text{otherwise} \end{cases}$$

and u'_i is the vertex in G' corresponding to set $N'_i \in A'$ and u' the vertex corresponding to $N \cup \{v\}$ where N is the neighbor set of v (unless $N \cup \{v\} \notin A'$, in which case just drop u' from Q'_1). The fact these are cliques follows from Q_i being cliques and each of the pairs of sets noted above being in B' , and

$$\begin{aligned} \sum_{i=1}^4 |Q'_i \cap V(H')| &\geq 1 + \sum_{i=1}^4 |Q'_i \cap V(H)| && (Q_1 \text{ contains } v) \\ &\geq 1 + \sum_{i=1}^4 |Q_i \cap V(H)| \\ &\geq |V(H)| + 3 \\ &= |V(H')| + 2. \end{aligned}$$

The full theorem statement then follows easily by induction on $|V(H')|$. □

Unfortunately, it does not seem like one can generalize the above theorem to be valid for Algorithm 3 run with the third input $k > 0$. For instance, one can verify that the algorithm returns SUCCESS on the input $(F_1, F_1 \vee F_2, 2)$, but FAILURE on $(F_2, F_2 \vee F_1, 1)$. Therefore if one wanted to prove $\text{HC-}\{\overline{K_3}, F_1 \vee F_2\}$ using Algorithm 3, you would succeed if you started from F_1 but not F_2 , despite F_2 containing F_1 . There are additional complications if one wanted to combine Theorems 2.5 and 2.6 to say something about the behavior of Algorithm 4, but this specific failure means that there is probably no implementation or simple modification to Algorithm 4 that makes all the desired properties hold unless one severely weakened the algorithm by removing the calls to Algorithm 3 with $k > 0$ and the additional checks if Algorithm 3 fails.

2.4.2 Limitations

We begin this section with a simple observation:

Proposition 2.7. *If $HC\text{-}\{\overline{K_3}, K_n\}$ holds for some n , the $H \vdash_{\mathcal{A}} K_n$ for all H and $\mathcal{A}$.*

Proof. By Ramsey's Theorem there are only finitely many $\{\overline{K_3}, K_n\}$ -free graphs, so Algorithm 4 must terminate. If $HC\text{-}\{\overline{K_3}, K_n\}$ holds then Algorithm 4 will not return FAILURE, so it must return SUCCESS. $\square$

For some H and G , we can actually prove the reverse: there is no $\mathcal{A}$ for which $H \vdash_{\mathcal{A}} G$. Given a graph I , define the I -core of a graph H to be the subgraph of H induced by all vertices that are in an induced G in H . Then:

Theorem 2.8. *If $H \vdash_{\mathcal{A}} G$ for some $\mathcal{A}$, then the I -core of G is an induced subgraph of the I -core of H , where I is either:*

1. $\overline{C_4}$,
2. any graph with minimum codegree at least 3, i.e. the complement of a graph with minimum degree at least 3.

Proof. We prove the contrapositive: if the I -core of G is not an induced subgraph of the I -core of H , then not $H \vdash_{\mathcal{A}} G$ for any $\mathcal{A}$. In particular, we will prove inductively that at each iteration of the **while** loop, A contains at least one graph whose I -core does not contain the I -core of G . The base case (in either possibility for I) is just the hypothesis of the contrapositive. Also, in either possibility for I , if the I -core of the chosen $H' \in A$ contains the I -core of G , then by inductive hypothesis there is another $H'' \in A$ whose I -core does not contain the I -core of G , and this H'' will still be present in A in the next iteration (unless the algorithm returned FAILURE). So we may assume the chosen $H' \in A$ has an I -core that does not contain the I -core of G .

1. If H' has a dominating edge, say $\mathcal{A}$ chooses edge uv , then one possible neighbor set of the newly added vertex w is $V(H') \setminus \{u, v\}$. Let H'' be the graph formed by adding w to H' . In this case, w cannot possibly be in an induced $\overline{C_4}$ in H'' . If it was, say $\{a, b, c, w\}$ induces $\overline{C_4}$ with a adjacent to b and c adjacent to w . Since w is not adjacent to a or b , we must have $\{a, b\} = \{u, v\}$, but then uv was not dominating since c is also nonadjacent to both a and b . Thus the $\overline{C_4}$ -core of H'' is the same as H' . Therefore G is not an induced subgraph of H' . Also H'' is $\overline{K_3}$ -free if H' is, since the only nonneighbors of w are u and v , which are adjacent. Therefore either H'' is added to A or else there is another $H''' \in A$ that is an induced subgraph of H'' . In either case, there is a graph whose $\overline{C_4}$ -core does not contain the $\overline{C_4}$ -core of G .

If H' has no dominating edge, then if any of the calls to Algorithm 3 succeed, one can verify the new graph to be added to A has the same $\overline{C_4}$ -core as H' , so the inductive step follows in this case as well by similar logic to above.

If none of the calls to Algorithm 3 succeed, but the call to Algorithm 2 succeeds, then one of the graphs to be added to A is $H \vee K_1$. By similar logic to above, the inductive step holds here.

Finally, if even the call to Algorithm 2 fails, but the final check that H' is not a counterexample to $\text{HC-}\{\overline{K_3}\}$ succeeds, then one of the graphs to be added to A is $H \vee K_1$. By the same logic, the inductive step holds.

If all checks fail then the algorithm returns failure and it is not the case that $H \vdash_{\mathcal{A}} G$.

2. If H' has a dominating edge, say $\mathcal{A}$ chooses edge uv , then one possible neighbor set of the newly added vertex w is $V(H') \setminus \{u, v\}$. Noting that w has codegree 2 in H'' , we see that the I -core of H'' is the same as the I -core of H' . Then the inductive step follows from the same logic as in case (1).

If H' has no dominating edge, then if any of the calls to Algorithm 3 succeed, one can verify the new graph to be added to A has the same $\overline{C_4}$ -core as H' , so the inductive step follows in this case as well by similar logic to above. Some care must be taken in the $k = 3$ case, since F_3 has a vertex w of codegree

3. However, none of the nonneighbors of this vertex have codegree 3 or greater, so it is not part of any induced subgraph of $H' \vee F_3$ that has minimum codegree 3 (either an induced subgraph of $H' \vee F_3$ contains a nonneighbor of w with codegree less than 3 or $H' \vee F_3$ does not contain all nonneighbors of w and w has codegree less than 3 in the induced subgraph).

The last two cases, if the call to Algorithm 2 succeeds or if it fails, are identical to case (1).

□

This means that it is impossible for the techniques in this part to improve Theorem 1.2 to all graphs on six vertices, since the graph $\overline{K_{3,3}}$ (among others) is equal to its $\overline{C_4}$ -core. It also helps explain the general patterns in the graphs H'_i , such as the relative lack of induced $\overline{C_4}$'s (compared to induced $\overline{C_5}$'s, say) and abundance of low-codegree vertices (which appear as leaves and isolated vertices in the drawings of the complements in Figure 3).

3 Results

The graphs used in intermediate steps in proving Theorem 1.6 and in the example proofs are shown in Figure 4.

3.1 Some Simple Cases

First we present a proof of $\text{HC-}\{\overline{K_3}, I_{21}\}$ that uses only Algorithm 1, i.e. only using the fact that minimal counterexamples to $\text{HC-}\{\overline{K_3}\}$ have no dominating edges. This will involve several steps starting from the graph W_5 .

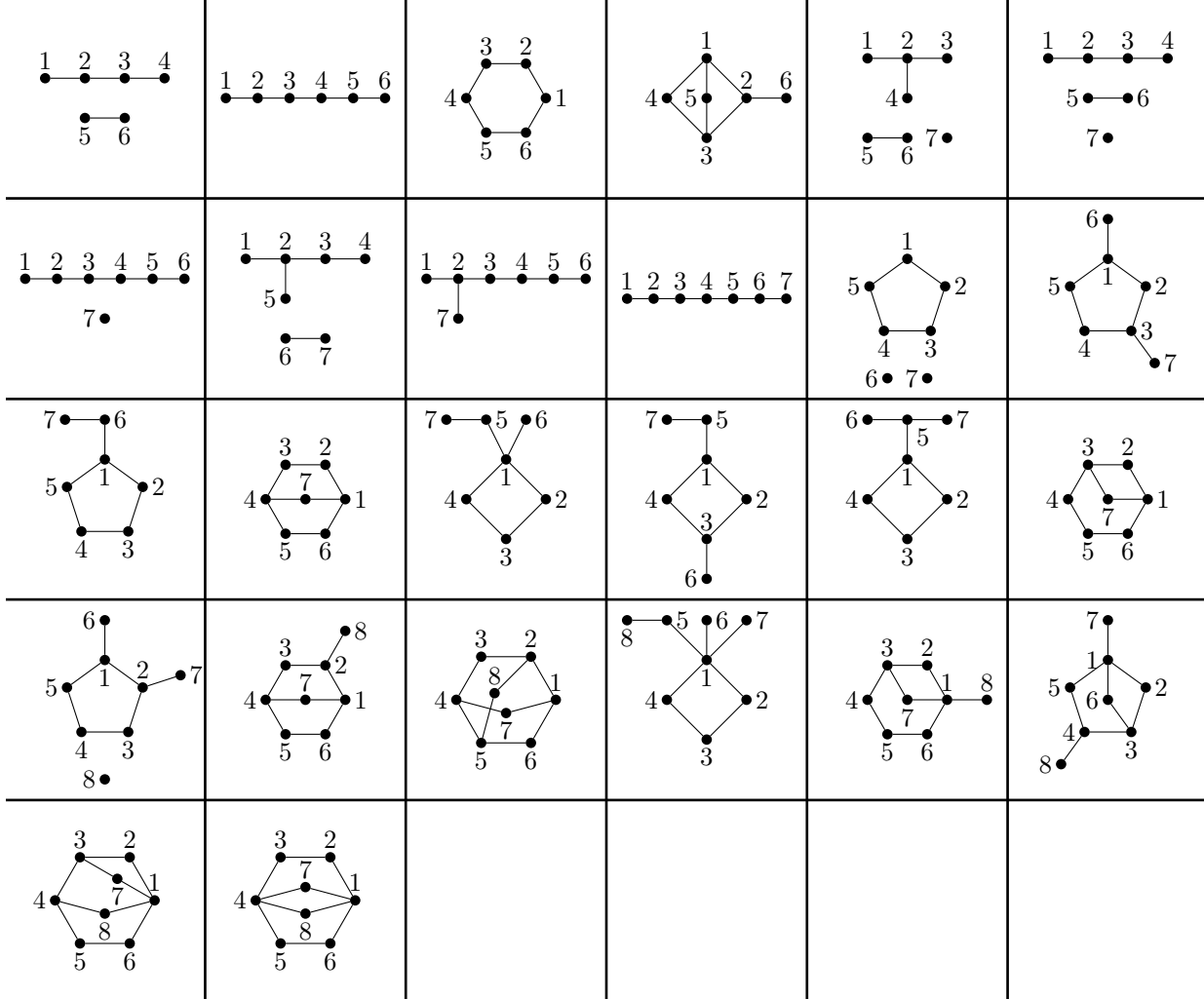


Figure 4: The complements of the graphs $I_1, \dots, I_{26}$ when read left-to-right, top-to-bottom. These graphs are used in the proof of Theorem 1.6. As in Figure 3, the labelings are used to choose edges consistently in algorithms.

Lemma 3.1. $\text{HC-}\{\overline{K_3}, I_3\}$ holds.

Proof. By Theorem 1.2, any minimal counterexample G to $\text{HC-}\{\overline{K_3}\}$ has an induced W_5 , say induced by vertices 1 through 6 with 1-2-3-4-5-1 a 5-cycle and 6 adjacent to the other five vertices. For edge $(1, 6)$ to not be dominating, there must be a vertex 7 in G adjacent to neither 1 nor 6. Then 7 is adjacent to 3 and 4 (nonneighbors of 1), and at least one of 2 and 5 since G has no stable set of size 3. WLOG, 7 is adjacent

to 2 and either (case 1) adjacent or (case 2) nonadjacent to 5.

(Case 1) For edge $(2, 6)$ to not be dominating, G must have a vertex 8 adjacent to neither 2 nor 6. Vertex 8 must be adjacent to 4, 5, and 7. If it is adjacent to 1, then $\{1, 4, 2, 8, 6, 7\}$ induces I_3 . If 8 is not adjacent to 1, then delete vertex 7 and witness that the resulting graph is isomorphic to the graph to be considered in case 2, so we reduce to that case.

(Case 2) For edge $(2, 6)$ to not be dominating, G must have a vertex 8 adjacent to neither 2 nor 6. Vertex 8 must be adjacent to 4, 5, and 7. If it is adjacent to 1, then $\{1, 4, 2, 8, 6, 7\}$ induces I_3 . If not, then it is adjacent to 3. For edge $(3, 6)$ to not be dominating, G must have a vertex 9 adjacent to neither 3 nor 6. Vertex 9 must be adjacent to 1, 5, 7, and 8. If 9 is adjacent to 2 then $\{2, 5, 3, 9, 6, 8\}$ induces I_3 . If it is not, then it is adjacent to 4 and $\{2, 4, 1, 7, 6, 9\}$ induces I_3 .

Therefore any minimal counterexample (and therefore every counterexample) to $HC-\{\overline{K_3}\}$ has an induced I_3 . $\square$

Lemma 3.2. $HC-\{\overline{K_3}, I_{14}\}$ holds.

Proof. By the previous lemma, any minimal counterexample $G \not\models HC-\{\overline{K_3}\}$ has an induced I_3 , say induced by vertices 1 through 6 with 1-2-3-4-5-6-1 a 6-cycle in the complement graph. For edge $(1, 4)$ to not be dominating, G must have a vertex 7 adjacent to neither 1 nor 4. But then since G has no stable set of size 3, 7 must be adjacent to all four other vertices, and $\{1, 2, 3, 4, 5, 6, 7\}$ induces I_{14} . $\square$

Lemma 3.3. $HC-\{\overline{K_3}, I_{21}\}$ holds.

Proof. Starting from the endpoint of the proof of the previous lemma, in order for $(2, 5)$ to not be dominating, there must be a vertex 8 adjacent to neither 2 nor 5. Then 8 must be adjacent to all other vertices except possibly 7. If it is adjacent to 7 then $\{1, 2, 3, 4, 5, 6, 7, 8\}$ induces I_{21} . If not, then for $(3, 6)$ to not be dominating, there must be a vertex 9 adjacent to neither 3 nor 6; this vertex is adjacent to 1, 2, 3, 4, 5, 6, and at least one of 7 and 8. If 7 then $\{1, 2, 3, 4, 5, 6, 7, 9\}$ induces I_{21} ; if 8 then $\{1, 2, 3, 4, 5, 6, 8, 9\}$ induces I_{21} instead. $\square$

This is probably the best result that can be obtained starting from Theorem 1.2 using Algorithm 1 alone without involving a tremendous amount of casework.

The graph H_7 in Figure 1 has no dominating edges, but we can use Algorithm 3 to introduce new vertices and dominating edges into this graph, and then use Algorithm 1 afterwards. Here is an example, proving eventually $HC-\{\overline{K_3}, I_{12}\}$.

Lemma 3.4. $HC-\{\overline{K_3}, I_2\}$ holds.

Proof. By Theorem 1.2, any minimal counterexample G to $\text{HC-}\{\overline{K_3}\}$ has an induced H_7 , say induced by vertices 1 through 7 with 1-2-3-4-5-1 a 5-cycle, 6 adjacent to $\{1, 2, 3, 7\}$, and 7 adjacent to $\{2, 3, 4, 6\}$. Suppose for contradiction G had no I_2 .

Then the vertices v of $G \setminus H_7$ have $N(v) \cap \{1, \dots, 7\}$ equal to one of the following sets:

$$\begin{array}{lll} N_0 = \{3, 4, 5, 7\}, & N_1 = \{3, 4, 5, 6, 7\}, & N_2 = \{2, 3, 4, 6, 7\}, \\ N_3 = \{2, 3, 4, 5, 7\}, & N_4 = \{2, 3, 4, 5, 6, 7\}, & N_5 = \{1, 4, 5, 7\}, \\ N_6 = \{1, 4, 5, 6, 7\}, & N_7 = \{1, 4, 5, 6\}, & N_8 = \{1, 4, 5\}, \\ N_9 = \{1, 3, 4, 5, 7\}, & N_{10} = \{1, 3, 4, 5, 6, 7\}, & N_{11} = \{1, 3, 4, 5, 6\}, \\ N_{12} = \{1, 3, 4, 5\}, & N_{13} = \{1, 2, 5, 6, 7\}, & N_{14} = \{1, 2, 5, 6\}, \\ N_{15} = \{1, 2, 4, 5, 7\}, & N_{16} = \{1, 2, 4, 5, 6, 7\}, & N_{17} = \{1, 2, 4, 5, 6\}, \\ N_{18} = \{1, 2, 4, 5\}, & N_{19} = \{1, 2, 3, 6, 7\}, & N_{20} = \{1, 2, 3, 5, 6, 7\}, \\ N_{21} = \{1, 2, 3, 5, 6\}, & N_{22} = \{1, 2, 3, 4, 6, 7\}, & N_{23} = \{1, 2, 3, 4, 5, 7\}, \\ N_{24} = \{1, 2, 3, 4, 5, 6, 7\}, & N_{25} = \{1, 2, 3, 4, 5, 6\}, & N_{26} = \{1, 2, 3, 4, 5\}. \end{array}$$

(This is easily verified by brute-force; any other neighbor set leads either to a stable set of size 3 or an I_2 .)

The set of vertices S_i for which $N(v) \cap \{1, \dots, 7\} = N_i$ forms a clique in G for all i except possibly $i = 24$ since G has no stable set of size 3 and such vertices share a nonneighbor except in the case $i = 24$. Suppose, for the moment, that G had no induced $H_7 \vee \overline{K_2}$. Then S_{24} is also a clique in G .

For each pair (i, j) with marked with an X in Table 3.1, S_i and S_j must be complete to each other in G , since if they weren't, G would either have a stable set of size 3 or an I_2 (again, this can be verified by brute force by adding two vertices to G with the appropriate neighbors and making those vertices nonadjacent, then witnessing that the resulting graph has a $\overline{K_3}$ or I_2).

Then the sets given by

- $Q_1 = \{4, 5\} \cup S_0 \cup S_1 \cup S_5 \cup S_6 \cup S_7 \cup S_8 \cup S_9 \cup S_{10} \cup S_{11} \cup S_{12}$,
- $Q_2 = \{2, 3\} \cup S_3 \cup S_4 \cup S_{20} \cup S_{21} \cup S_{23} \cup S_{24} \cup S_{25} \cup S_{26}$,
- $Q_3 = \{2, 3, 6, 7\} \cup S_2 \cup S_{19} \cup S_{22}$, and
- $Q_4 = \{1, 2\} \cup S_{13} \cup S_{14} \cup S_{15} \cup S_{16} \cup S_{17} \cup S_{18}$

are a clique cover of G , and $|Q_1| + |Q_2| + |Q_3| + |Q_4| \geq |V(G)| + 2$, so $\omega(G) \geq \frac{|V(G)|+2}{4}$, so by Lemma 1.4, G is not a counterexample to $\text{HC-}\{\overline{K_3}\}$. This means our supposition that G was $(H_7 \vee \overline{K_2})$ -free was incorrect, so G has a vertex 8 and 9 both adjacent to $\{1, \dots, 7\}$ but not adjacent to each other.

Then for $(1, 8)$ to not be a dominating edge, G must have a vertex 10 adjacent to neither 1 nor 8. Vertex 10 must be adjacent to 3 and 4. If 10 is adjacent to 2 or 5 then $\{9, 8, 10, 1, 4, 2\}$ or $\{9, 8, 10, 1, 3, 5\}$ induces I_2 , respectively. But 10 cannot be nonadjacent to both 2 and 5, so we have a contradiction and G does have an I_2 . $\square$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
1	X																									
2	X	X																								
3	X	X	X																							
4	X	X	X	X																						
5	X	X		X																						
6	X	X			X	X																				
7	X	X			X	X	X																			
8	X	X		X		X	X	X																		
9	X	X		X		X	X	X	X																	
10	X	X			X	X	X	X	X	X																
11	X	X			X	X	X	X	X	X	X															
12	X	X		X		X	X	X	X	X	X	X														
13		X		X	X	X	X	X	X	X	X	X														
14						X	X	X	X			X	X	X												
15	X			X	X	X	X	X	X	X	X		X	X	X											
16		X		X	X	X	X	X	X	X	X	X		X	X	X										
17		X		X	X	X	X	X	X		X	X	X	X	X	X	X									
18	X			X	X	X	X	X	X	X		X	X	X	X	X	X	X								
19			X											X	X											
20		X		X	X	X	X			X	X	X	X	X	X	X	X				X					
21		X		X	X			X	X	X	X	X	X	X	X			X	X	X	X					
22			X			X	X	X		X	X	X	X			X	X	X	X	X						
23	X			X	X	X	X		X	X	X	X	X	X		X	X	X	X		X	X	X			
24		X		X	X	X	X	X		X	X	X	X	X		X	X	X	X		X	X	X	X		
25		X		X	X		X	X	X	X	X	X	X		X	X	X	X	X		X	X	X	X	X	
26	X			X	X	X		X	X	X	X	X	X		X	X	X	X	X		X	X	X	X	X	X

Table 3.1: Pairs of sets S_i and S_j that must be complete to each other. Each such pair with $i > j$ is marked with an X in row i , column j .

The proof above essentially used Algorithm 3 with $k = 1$, finishing by eliminating a dominating edge. As can be seen, this clique cover technique is computationally intensive and rather impractical to verify by hand, even for this relatively small example.

Lemma 3.5. $HC-\{\overline{K_3}, I_{12}\}$ holds.

Proof. From the previous lemma, any minimal counterexample G of $HC-\{\overline{K_3}\}$ has an induced I_2 , say by vertices 1 through 6 with 1-2-3-4-5-6 a 6-path in the complement graph. For $(2, 5)$ to not be dominating, G must have a vertex 7 adjacent to neither 2 nor 5, but this vertex must be adjacent to 1, 3, 4, and 6, so $\{2, 7, 5, 4, 3, 1, 6\}$ induces I_{12} . $\square$

3.2 Data for Computer Verification

As we discussed in Section 2.4, in Algorithm 1 and Algorithm 4, one needs to choose a dominating edge whenever a graph has multiple. We describe our method for this in more detail in Section 2.4 and Appendix A.1.1. The vertex labeling of the initial graph H is given by the labelings in Figures 3 and 4.

Whenever a vertex is added to a graph, it gets the smallest positive integer label not yet used in that graph. The chosen dominating edge is the one that minimizes the larger label of its endpoints.

We then present the necessary steps to verify Theorem 1.6 as a directed forest. Each edge $G \rightarrow H$ means to run Algorithm 4 starting from graph G to prove $\text{HC}\{-\overline{K_3}, H\}$, i.e. it is the assertion $G \vdash_{\mathcal{A}} H$ for our implementation $\mathcal{A}$ of Algorithm 4.

There are four starting points used: W_5 , $\overline{K_{2,3}}$, K_7 , and H_7 . All four are known to be induced subgraphs of any counterexample to $\text{HC}\{-\overline{K_3}\}$ due to Theorem 1.2. The labeling to use for W_5 is to label the vertices in the 5-cycle as 1-2-3-4-5 and the hub as 6. The labeling to use for $\overline{K_{2,3}}$ is that the part with three vertices gets labels 1, 3, and 5 and the part with two vertices gets labels 2 and 4. All labelings of K_7 are equivalent due to symmetry. The labeling to use for H_7 is to label a 5-cycle 1-2-3-4-5 and vertex 6 is adjacent to 1, 2, 3, 7 and vertex 7 is adjacent to 2, 3, 4, 7.

The directed forest is shown in Figure 5. There are many other possible connections between the graphs in the diagram; the arrows we display seek to minimize the time it takes for a computer to verify each step. We describe this in more detail in Section 4, as well as explain briefly how this directed forest was found starting from Theorem 1.2.

4 Final Remarks

Call the *weight* (relative to an implementation $\mathcal{A}$ of Algorithm 4) of input (H, G) the sum over the graphs H' added to A at any point in the algorithm (including H) of $2^{|V(H')|}$, assuming $H \vdash_{\mathcal{A}} G$ so that this weight is finite. This measure is designed to give a very rough estimate of the amount of time it takes for Algorithm 4 to prove $\text{HC}\{-\overline{K_3}, G\}$ starting from H . This is because there are several parts of the algorithm that scale exponentially (or at least super-polynomially) in the size of the graph H' considered at some point:

- When one constructs graphs by adding a vertex to H' while either avoiding or introducing a dominating edge, there may be exponentially many such graphs.
- In several places we must solve the subgraph isomorphism problem, which is NP-complete in general. Even though this is only polynomial-time for fixed G , recall that in some places we check if graphs are A -free with A containing possibly arbitrarily large graphs, so we are still potentially subject to the full NP-completeness of this problem.
- Running Algorithm 3 requires possibly constructing an auxiliary graph with exponentially many vertices. Worse, we then try to find a clique cover of this graph with four cliques, which is equivalent



to 4-coloring the complement graph and is thus NP-complete in general. The total running time of Algorithm 3 is therefore actually potentially doubly-exponential in the number of vertices of the input.

Weight the directed edges of Figure 5 according to this metric. The proof data expressed in Figure 5 is optimized to minimize the largest weight in any individual edge from a graph H'_i to one of the roots using our specific implementation of Algorithm 4. In fact, the way these proofs were found was by running in parallel all inputs (H, G) with H a maximal graph for which $\text{HC}\{-\overline{K_3}, H\}$ is known and G a minimal graph that is not an induced subgraph of any graph H' for which $\text{HC}\{-\overline{K_3}, H'\}$ is known, where by “parallel” we mean at each point in time to step forward whichever instance currently has the lowest running total weight. In this way, when an instance returns SUCCESS, we can be sure it is the most efficient possible improvement.

This was performed starting from the graphs in Theorem 1.2 until no new improvements were found with a weight less than $2.3 \cdot 10^7$. There are 69 minimal graphs that are not a subgraph of any H'_i or K_7 or H_7 , though several of these (such as $\overline{K_{3,3}}$) could not possibly be proved by the methods in this part due to Theorem 2.8. We expect if one continues to run this process, one will continue to find new graphs for which $\text{HC}\{-\overline{K_3}, H\}$ holds for a long time, though the improvements become less frequent over time.

The weights for each edge in Figure 5 are given in Appendix A.2.

There are several routes that could directly improve the results obtained in this part. First, as mentioned in a footnote in the introduction, there is a recent improvement to Lemma 1.5 [SWY21], so one can replace the tree T_1 with the larger T'_1 obtained by adding a second leaf attached to the third vertex of the 5-path of T_1 , and replace F_3 accordingly. We expect perhaps a few more results can be obtained with this substitution.

Second, Lemma 1.3 has a stronger version:

Lemma 4.1. *[PST03] Any minimal counterexample to $\text{HC}\{-\overline{K_3}\}$ has no connected dominating matching.*

Here, a *connected matching* is a set of vertex-disjoint edges that are pairwise adjacent. It is *dominating* if additionally every vertex outside the matching is adjacent to every edge in the matching. A dominating edge is just a connected dominating matching with one edge. This substitution theoretically allows you to sidestep Theorem 2.8, especially the restriction on $\overline{C_4}$ -cores; for instance it was the main ingredient used to prove that $\text{HC}\{-\overline{K_3}, H_7\}$ holds in [PST03]. It is clear how to extend Algorithm 1 to incorporate this stronger result. However, it is not immediately clear how to reconcile it with Algorithm 2/3, since it is conjectured:

Conjecture 2. *All $\overline{K_3}$ -free graphs on n vertices have a set of $\lceil n/2 \rceil$ pairwise adjacent vertices or edges.*

This conjecture is called *Seymour’s Strengthening* of $\text{HC}\{-\overline{K_3}\}$ in [Bla07] or the *matching version* of $\text{HC}\{-\overline{K_3}\}$ in [CS12], and it is often abbreviated SSH. It is very close to conjecturing that all $\overline{K_3}$ -free graphs have a connected dominating matching (which is a slightly stronger statement), so at the very least one

should expect “most” $\overline{K_3}$ -free graphs to have a connected dominating matching. Thus, naïvely replacing the check for dominating edges with a check for connected dominating matchings likely completely removes the impact of Algorithms 2 and 3.

Third, if one marks vertices in the graph and only checks for subgraphs containing the set of marked vertices, rather than subgraphs in the whole graph, one can prove theorems of the form “In any minimal counterexample to $\text{HC-}\{\overline{K_3}\}$, every induced H must be part of an induced H' ” where G' contains G as an induced subgraph. For example:

Theorem 4.2. *In any minimal counterexample to $\text{HC-}\{\overline{K_3}\}$, every vertex must be found in some induced C_5 .*

Proof. Suppose G is a minimal counterexample to $\text{HC-}\{\overline{K_3}\}$. Let a be an arbitrary vertex to G . Obviously a must have a nonneighbor c , and G must be connected so we may take c to be adjacent to a neighbor of a , say b . The edge ab is not a dominating edge, so there is a vertex d adjacent to neither a nor b ; this vertex must be adjacent to c to avoid $\overline{K_3}$. Then for bc to not be a dominating edge, there is a vertex e adjacent to neither b nor c ; this vertex must be adjacent to a and d . Then $\{a, b, c, d, e\}$ is an induced C_5 containing a . □

In the Part II, we perform all of these modifications and improvements, as well as several others.

Part II

Local Conditions

5 Introduction

All graphs in this part are still simple, but unlike the previous part, some graphs in this part are infinite.

In this part we modify the techniques used in the proofs in Part I in two major ways. First, some modifications allow one to prove results about labeled graphs instead of unlabeled graphs. This allows one to make assertions such as the one in Theorem 4.2. This leads to the general notion of a *local condition* introduced in Section 6. Second, we generalize the proof steps involving dominating edges in Part I to instead use any local condition. This modification theoretically makes the main algorithm in this part stronger than the algorithms in Part I, though there are some practical constraints that make this difficult to realize in practice, discussed in Section 10.

As in Part I, we make use of the Lemma 1.4, as well as the stronger version of Lemma 1.5:

Lemma 5.1. *$\{K_3, H\}$ -free graphs have chromatic number at most k , where (H, k) is any of the pairs $(K_1, 0)$, $(K_2, 1)$, $(P_4, 2)$, or $(T_1, 3)$ [SWY21] or $(T_2, 3)$ [Ran04], where T_i are shown in Figure 6.*

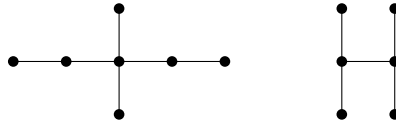


Figure 6: The trees T'_1 (left) and T'_2 (right) associated with Lemma 5.1. The graph T'_1 is sometimes called “fork” or “trident.” Note $T'_2 = T_3$ defined in Figure 2.

The main theorem of this part is:

Theorem 5.2. *Let G be a minimal counterexample to $HC\text{-}\{\overline{K_3}\}$. Then G satisfies the meet of the basic local conditions in Figure 7. Additionally, G satisfies the local condition Q drawn in Figures 8 and 9.*

Combined with a simple result in Section 12, this implies, for example:

Corollary 5.3. *Let G be a minimal counterexample to $HC\text{-}\{\overline{K_3}\}$ with n vertices. Then G contains at least $n/5$ induced copies of W_5 , the wheel graph on 6 vertices.*

This represents a modest improvement over [Bos19], where it was proven that counterexamples to $HC\text{-}\{\overline{K_3}\}$ have at least one copy of W_5 . (It also arguably improves on Part I, where one can glean that counterexamples must have at least *two* copies of W_5 , since H'_{16} has two copies of W_5 .) Many similar results to the one above can be obtained for other graphs using Theorem 5.2.

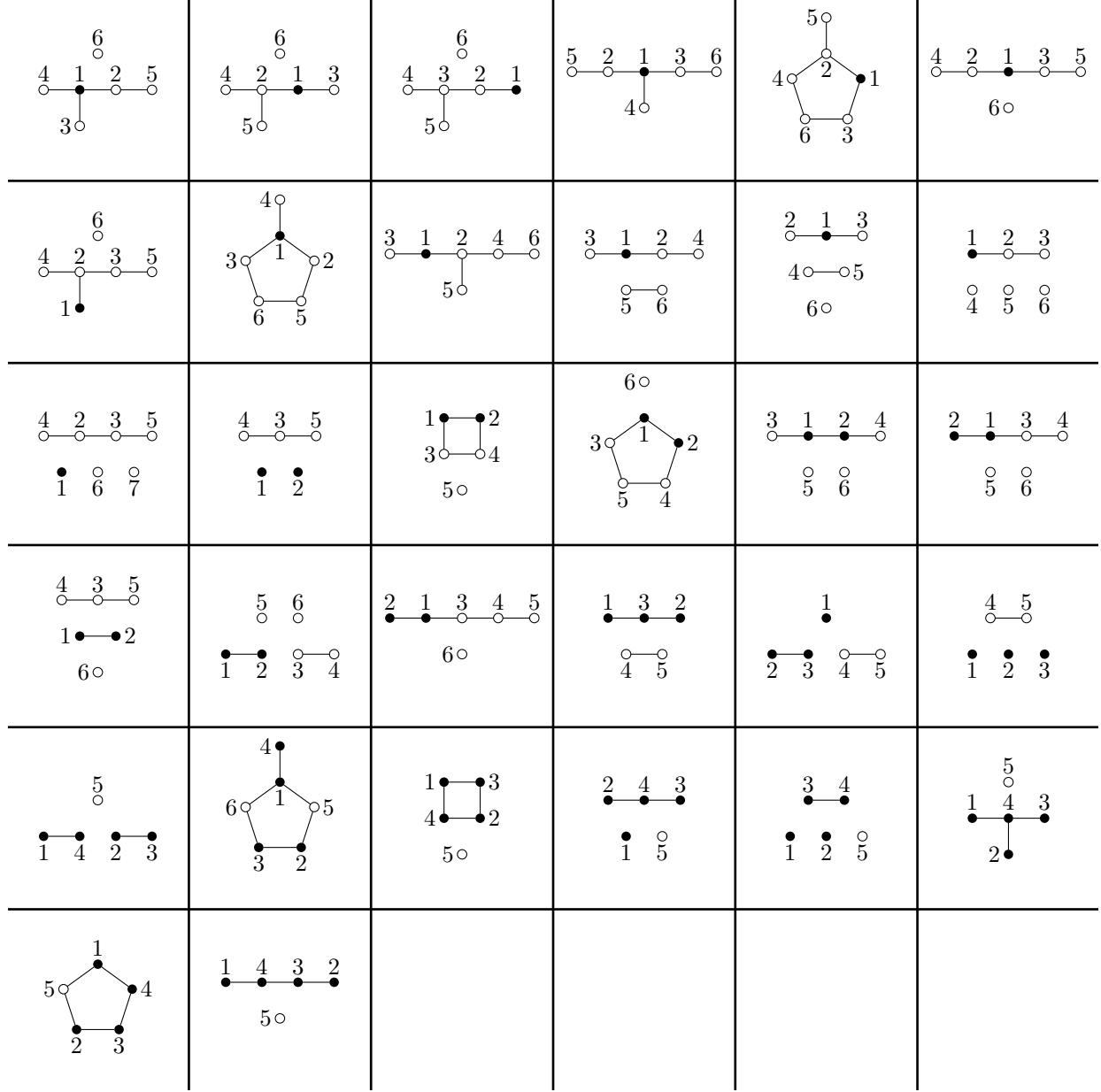


Figure 7: Basic local conditions (H, k) associated with Theorem 5.2. The complements of the graphs are drawn. Any minimal counterexample to $\text{HC-}\{K_3\}$ must satisfy the meet of these conditions. The black vertices are vertices in the pattern $H[k]$ of the local condition, and the white vertices are the other vertices of H .

We also prove that the main algorithm in this part is “complete” in the sense that any logical implication between local conditions can be proven using our algorithm. See Theorem 7.6 for the precise statement of this assertion. We suspect that this algorithm, perhaps with small modifications, will be useful in the study of many other problems in structural graph theory, not just Hadwiger’s Conjecture.

6 Local Conditions

6.1 Definitions

For a given (labeled) graph H and vertex subset $U \subseteq V(H)$, we say that G satisfies *basic local condition* (H, U) if for every $A \subseteq V(G)$ and isomorphism $\phi : G[A] \rightarrow H[U]$, there is a superset $B \supseteq A$ and isomorphism $\psi : G[B] \rightarrow H$ with $\psi|_A = \phi$. By possibly modifying the underlying labeling of the vertices of H , it suffices to consider the case when $U = \{1, \dots, k\}$ for some k ; we denote the corresponding basic local condition (H, k) . If k is a positive integer, $H[k]$ means $H[\{1, \dots, k\}]$. Note that G vacuously satisfies (H, k) if it has no subgraph isomorphic to $H[k]$. We sometimes call $H[k]$ the *pattern* and H the *target*, and we say that a particular set A with $G[A]$ isomorphic to $H[k]$ *extends to H* if there is a B and ψ satisfying the property in the first sentence of this paragraph (or we may say “ $G[A]$ extends to H ,” rather than “ A extends to H ,” they mean the same thing).

We define some conventions for working with labeled graphs. When we write the path graph P_n , the labeling is assumed to be such that the path is 1-2-...- n , and when we write the cycle graph C_n , the labeling is assumed to be such that the cycle is 1-2-...- n -1. For a labeled graph G and vertex subset A , the induced subgraph $G[A]$ simply inherits the labeling from G ; for example $C_5[\{2, 3, 5\}]$ has vertices labeled 2, 3, and 5 with an edge between 2 and 3.

We say that two labeled graphs G and H are *labeled-isomorphic* if they are not just isomorphic, but their labelings are the same, i.e. $V(G) = V(H)$ and the identity map $[V(G)] \rightarrow [V(H)]$ induces an isomorphism $G \rightarrow H$. We write $G \approx H$ if G and H are isomorphic and $G = H$ if they are labeled-isomorphic. Furthermore, we say that a graph G is a *labeled-induced subgraph* of H if there is a subset $U \subseteq V(H)$ with $G = H[U]$. We write $G \lesssim H$ if G is an induced subgraph of H and $G \leq H$ if G is a labeled-induced subgraph of H . There is one more relation between labeled graphs that we consider, called $\preceq$, but we delay the definition of that until Section 7.

We define two operations for combining local conditions, and the set of all *local conditions* is the closure of basic local conditions under these operations. These operations are:

- For a (possibly infinite) set of basic local conditions $(H_1, k), \dots$ with $H_i[k] = H_1[k]$ for all i , we say

that G satisfies the *join* $\bigvee_i(H_i, k)$ if for every $A \subseteq V(G)$ and isomorphism $\phi : G[A] \rightarrow H_1[k]$, there is a superset $B \supseteq A$, positive integer i , and isomorphism $\psi : G[B] \rightarrow H_i$ with $\psi|_A = \phi$. If $Q = \bigvee_i(H_i, k)$ and $R = \bigvee_i(H'_i, k)$ with $H_i[k] = H'_i[k]$, we define $Q \vee R$ in the obvious way.

- For a (possibly infinite) set of local conditions $Q_1, \dots$, we say that G satisfies the *meet* $\bigwedge_i Q_i$ if G satisfies all Q_i .

We define $\mathcal{G}$ to be the set of all finite graphs, and for local condition Q , we define $\mathcal{G}_Q$ to be the set of finite graphs satisfying Q , or the *class* of Q . We say that two local conditions Q and R are *equivalent* and write $Q \approx R$ if $\mathcal{G}_Q = \mathcal{G}_R$, and *equal* and write $Q = R$ if one may write $Q = \bigwedge_i \bigvee_j(H_{i,j}, k_i)$ and $R = \bigwedge_i \bigvee_j(H'_{i,j}, k_i)$ with $H_{i,j} = H'_{i,j}$ for all i, j . Note that equality is a stronger condition than equivalence. If we only require that $H_{i,j} \approx H'_{i,j}$ and $H_{i,j}[k] = H'_{i,j}[k]$, then it is easy to see that Q and R are equivalent. It is also easy to see:

Proposition 6.1. *The following facts hold:*

1. Any local condition may be written in the form $\bigwedge_i \bigvee_j(H_{i,j}, k_i)$ where $H_{i,j}[k_i] = H_{i,j'}[k_i]$ for any i, j, j' .
2. $\mathcal{G}_{Q \wedge R} = \mathcal{G}_Q \cap \mathcal{G}_R$ for any local conditions Q, R . This also implies that if $Q \approx R$, then for any local condition S , $Q \wedge S \approx R \wedge S$.
3. $\mathcal{G}_{Q \vee R} \supseteq \mathcal{G}_Q \cup \mathcal{G}_R$ for any local conditions Q, R when $Q \vee R$ is defined.¹
4. $\mathcal{G}_{(H, V(H))} = \mathcal{G}$ for any graph H .
5. $\vee$ and $\wedge$ are commutative and associative (when defined), and $Q \vee Q = Q \wedge Q = Q$.
6. If $H_1[k_1] = H_2$ and $H_1[k_2] = H_2[k_2]$ then $\mathcal{G}_{(H_1, k_2)} \subseteq \mathcal{G}_{(H_1, k_1) \wedge (H_2, k_2)}$. Similar statements can be made for more than two conditions.

□

Note that so far we have only defined the join on basic local conditions with the same pattern. It is not clear if it is possible to extend the definition to joins of basic local conditions with different patterns, and in any case we will not need such a definition for the purposes of this paper.

Consider a local condition Q . If Q may be written in the form described in Proposition 6.1(1) where the meet is finite (but the joins are not necessarily finite), then we call Q a *finite* local condition; otherwise, we call it an *infinite* local condition. If the meet and all the joins are finite, then we call Q a *superfinite* local

¹Note that it is *not* the case in general that if $Q \approx R$, then $Q \vee S \approx R \vee S$; a good example is Proposition 6.2(5). However, if $Q = R$ then $Q \vee S = R \vee S$, and the same for meets.

condition; these are primarily of interest for computational reasons, since there is an obvious algorithm to check if a graph satisfies such a local condition for superfinite local conditions.

We end this subsection by listing several common graph properties that are local conditions.

Proposition 6.2. *The following properties are finite local conditions:*

1. *The property of containing a given graph H as an induced subgraph, or as a subgraph, or as a minor. More generally, any “monotone increasing” property, that is, any graph property P such that if G satisfies P and H contains G as an induced subgraph, then H satisfies P .*
2. *The property of having mindegree at least k , for any k .*
3. *The property of having diameter at most k , for any $k \geq 2$.*
4. *Connectedness.*
5. *k -connectedness, for any k .*
6. *Twinfreeness: no two vertices have the same neighborhood. Also, the related condition that there are no two vertices u, v with $N(u) \subseteq N(v)$.*

Also, the property of being S -free for any set S is an infinite local condition. The same goes for any “monotone decreasing” property, that is, any graph property P such that if G satisfies P and H is an induced subgraph of G , then H satisfies P .

Proof. Finite local conditions:

1. Any monotone increasing property is equivalent to containing at least one graph in a (possibly infinite) set S as an induced subgraph. Then $Q = \bigvee_{H \in S} (H, 0)$.
2. $Q = \bigvee_{H \in S} (H, 1)$ where S is the set of (labeled) graphs on $k + 1$ vertices where vertex 1 is dominating.
3. $Q = \bigvee_{i=3}^{k+1} (P_i, \{1, i\})$.
4. $Q = \bigvee_{i=3}^{\infty} (P_i, \{1, i\})$. Note that connectedness is the same as having finite diameter, and writing these properties as local conditions makes this immediately clear.
5. Let S be the set of all graphs formed by making vertices 1 and 2 nonadjacent and adding k vertex-disjoint paths of any length with endpoints 1 and 2 (note there are allowed to be edges between the paths). Then $Q = \bigvee_{G \in S} (G, 2)$.

6. Let H_1 be the graph on three vertices with an edge between vertex 1 and 3 and H_2 the graph on three vertices with an edge between vertex 2 and 3. Then $Q = (H_1, 2) \vee (H_2, 2)$ for the twinfreeness property and $Q = (H_1, 2)$ for the other property. Note that $(H_1, 2) \approx (H_2, 2)$, so this shows in general that $Q \approx R$ does not mean $Q \vee S \approx R \vee S$ (since $(H_1, 2) \approx (H_1, 2) \vee (H_1, 2)$).

Finally, any monotone decreasing property is equivalent to being S -free for a (possibly infinite) set S . For a graph $H \in S$ let L_H be the set of all graphs H' with $H'[V(H)] = H$. Then $Q = \bigwedge_{H \in S} \bigwedge_{H' \in L_H} (H', V(H))$ (note that this is a meet of meets, not a meet of joins). $\square$

Also, at the beginning of Section 8, we show there is an infinite local condition that basically captures the property of being a minimal counterexample to $\text{HC-}\{\overline{K_3}\}$, and a similar local condition for SSH in Section 11.1.

6.2 Remarks

We have the following simple result:

Theorem 6.3. *Let Q be a finite local condition. Then $\mathcal{G}_Q$ is nonempty; in fact large random graphs satisfy Q with high probability.*

Proof. Write $Q = \bigwedge_i \bigvee_j (H_{i,j}, k_i)$ and $Q' = \bigwedge_i (H_{i,1}, k_i)$. Let $K_i = |V(H_{i,1})|$ and write

$$Q'' = \bigwedge_i \bigwedge_{\ell=1}^{K_i - k_i} (H_{i,1}[k_i + \ell], k_i + \ell - 1).$$

Note that $\mathcal{G}_{Q''} \subseteq \mathcal{G}_{Q'} \subseteq \mathcal{G}_Q$, the first inclusion by Proposition 6.1(6) and the second by Proposition 6.1(2). Therefore it suffices to prove that a finite meet of basic local conditions (H_i, k_i) where $|V(H_i)| = k_i + 1$ has nonempty class.

Let $G = G(n, 1/2)$ be a random graph. Let E_i be the expected number of subsets $A \subseteq V(G)$ and isomorphisms $\phi : G[A] \rightarrow H_i[k_i]$. This is equal to the number of automorphisms of $H_i[k_i]$ times the expected number of induced subgraphs of G isomorphic to $H_i[k_i]$; explicitly, we have

$$E_i = \binom{n}{k_i} k_i! \left(\frac{1}{2}\right)^{\binom{k_i}{2}}.$$

Suppose $\phi : G[A] \rightarrow H_i[k_i]$ is an isomorphism. Let $F_{i,A}$ be the probability that there is no superset $B \supseteq A$ and isomorphism $\psi : G[B] \rightarrow H_i$ where $\psi|_A = \phi$. We compute (because $|V(H_i)| = k_i + 1$)

$$F_{i,A} = \left(\frac{1}{2}\right)^{k_i(n-k_i)}.$$

Therefore the expected number of places G fails to satisfy the local condition (H_i, k_i) is

$$\binom{n}{k_i} k_i! \left(\frac{1}{2}\right)^{\binom{k_i}{2}} \cdot \left(\frac{1}{2}\right)^{k_i(n-k_i)}$$

which goes to zero as $n \rightarrow \infty$. Therefore the probability that G fails to satisfy the finite meet also goes to zero (it is not more than the sum of the probabilities G fails to satisfy each constituent basic local condition). Thus, large random graphs satisfy a given local condition with high probability. $\square$

Let $\text{Forb}(S)$ be the set of S -free graphs. One might try to generalize the previous theorem as follows:

(False) Conjecture 3. *Let $Q = \bigwedge_i \bigvee_j (H_{i,j}, k_i)$ be a finite local condition and S a set of graphs such that every $H_{i,j} \in \text{Forb}(S)$. Then $\mathcal{G}_Q \cap \text{Forb}(S)$ is nonempty.*

In other words, if the local condition does not directly force the graph to contain a graph in S , then one might expect there is S -free graph satisfying it. The previous theorem is the case $S = \emptyset$, since all graphs are vacuously $\emptyset$ -free. However, this conjecture is easily seen to be false, even for finite S :

Theorem 6.4. *Any graph satisfying local condition $(K_1, 0) \wedge (P_3, \{2\})$ has an induced cycle. Any graph satisfying $(K_1, 0) \wedge (P_3, \{2\}) \wedge (P_3, \{1, 3\})$ has an induced C_3, C_4 , or C_5 .*

Proof. The first basic local condition considered, $(K_1, 0)$, simply excludes the graph with no vertices. The second basic local condition forces the graph to have minimum degree 2, so it cannot be a forest. This proves the first claim. The third basic local condition in the second claim forces the graph to have diameter at most 2 and therefore girth at most 5, so the shortest (induced) cycle has at most 5 vertices, proving the second claim. $\square$

With slightly more work one can improve the above and show that any graph satisfying $(K_1, 0) \wedge (P_3, \{2\}) \wedge (P_3, \{1, 3\})$ has an induced C_4 or C_5 ; we omit the details.

The natural question is: for which S does Conjecture 3 hold? In our case, we are interested specifically in $\overline{K_3}$ -free graphs.

Conjecture 4. *Let $Q = \bigwedge_i \bigvee_j (H_{i,j}, k_i)$ be a finite local condition such that every $H_{i,j} \in \text{Forb}(\overline{K_3})$. Then $\mathcal{G}_Q \cap \text{Forb}(\overline{K_3})$ is nonempty.*

This conjecture seems quite hard. Obviously it is equivalent to the same statement replacing $\overline{K_3}$ by K_3 ; call this modification Conjecture 4'. It is not necessarily the case that large random triangle-free graphs satisfy such Q with high probability. For example, a classical result of Erdős, Kleitman, and Rothschild is

that large random triangle-free graphs are bipartite with high probability, so they do not satisfy the local condition $(C_5, 0)$ [EKR76].

A graph G is called *k-existentially complete triangle-free (k-ECTF)* if for every pair of subsets $A, B \subseteq V(G)$ with $|A| + |B| = k$ and A is a stable set, there is a vertex of G adjacent to every vertex in A and no vertex in B . We have the following result:

Theorem 6.5. *The following are equivalent:*

1. *Conjecture 4'.*
2. *There are finite k-ECTF graphs for all k.*

Proof. The implication $(1) \Rightarrow (2)$ follows from the observation that being k -ECTF is a finite local condition satisfying the hypothesis of (1). For $(2) \Rightarrow (1)$, write $Q = \bigwedge_i \bigvee_j (H_{i,j}, k_i)$ and note that k -ECTF graphs satisfy Q for all $k \geq \max_i \min_j |V(H_{i,j})|$. $\square$

The question of whether there are k -ECTF graphs for all k was originally asked by Cherlin in 1993 [Che92]. It remains open for all $k \geq 4$, and it is suspected that there are no finite k -ECTF graphs for sufficiently large k [LS20]. In Part I, we were only proving that counterexamples contain certain induced subgraph, but it is the case that for any finite set of $\overline{K_3}$ -free graphs S there are $\overline{K_3}$ -free graphs containing all of the graphs in S , so no matter how much you prove using the techniques in that part, you will never prove $\text{HC-}\{\overline{K_3}\}$. However, if Conjecture 4' fails, then the techniques in this part may at some point in the future resolve Hadwiger's Conjecture for graphs with independence number 2, since there may be a finite local condition that no finite $\overline{K_3}$ -free graph satisfies.

We provide some miscellaneous results about local conditions in Section 12.

7 Proving One Local Condition Given Another

7.1 The Algorithm

In this section, we describe an algorithm that, given local conditions Q and R , sometimes proves that all graphs satisfying Q also satisfy R . It suffices to consider the case that R is a join of basic local conditions, since if R is a meet, one can simply prove that all graphs satisfying Q also satisfy each constituent join of basic local conditions making up R .

The idea behind the algorithm is simple, and it is essentially a direct generalization of Algorithm 1 from Part I. Suppose, for simplicity, we are trying to prove that graphs satisfying some local condition (H, k) also

satisfy (I, ℓ) . A counterexample to this statement would be a graph containing an induced $I[\ell]$ that does not extend to an induced I . Suppose we had such a graph G , and label the vertices in the induced $I[\ell]$ with labels 1 through ℓ . We will reveal other vertices in G using (H, k) . First, find an induced $H[k]$ in the vertices of G revealed so far (i.e. the vertices 1 through ℓ). Second, check that this $H[k]$ does not extend to an induced H in the vertices revealed so far. Since G is assumed to satisfy (H, k) , this means there are some other vertices in G that attach to the identified induced $H[k]$ in a particular way. There will, in general, be many ways that these vertices attach to the other vertices of G revealed so far. We consider all possible minimal ways to attach vertices to those revealed so far to make it so that (H, k) is satisfied at the identified $H[k]$. Here, “minimal” means that attaching any proper subset of the vertices does not make the induced $H[k]$ extend to H . For each of these possibilities, we repeat the same process: identify an induced $H[k]$, check that it does not yet extend to an induced H , then consider all ways of attaching vertices to remedy this, and forking on these possibilities.

The key is that we assumed $G[\ell]$ does not extend to I . If ever a possibility attached vertices in a way to make $G[k]$ extend to I , we can prune that possibility from further consideration. If we exhaust all possibilities, then we have proven that $G[\ell]$ actually does extend to I . Since the choice of induced $I[\ell]$ in G at the beginning was arbitrary, the same statement holds for all induced $I[\ell]$ in G , so in fact G satisfies (I, ℓ) .

There are some optimizations that can be made. As described right now, the algorithm essentially traverses a search tree in a poset of labeled graphs according to the local condition (H, k) , where $G' \preceq_\ell G''$ in the poset if there is a graph isomorphism $\phi : V(G') \rightarrow A \subseteq V(G'')$ such that ϕ is the identity when restricted to $\{1, \dots, \ell\}$ (and two isomorphic graphs with labels differing only outside $\{1, \dots, \ell\}$ are equivalent, since in this case $G' \preceq_\ell G''$ and $G'' \preceq_\ell G'$).¹ It is worth noting here that this ordering is very useful and will appear many times in proofs. The optimization to the algorithm comes from the fact that some graphs can be reached in multiple ways traveling up this poset, but we only need to consider one path, so we can prune certain graphs early if we detect this.

There are at least two ways to do this. One is to keep track of graphs whose descendants were all eventually pruned and then immediately prune any future graphs that “contain” any of these graphs according to the partial order described above; we know the algorithm will determine that these graphs will eventually have all descendants pruned anyway. The second way to enforce this is the reverse: immediately prune any graph G that contains any of the other graphs H in the list of graphs to consider in the future. Here, if H has all descendants pruned eventually, so will G , so we might as well skip it. The latter method turned out

¹This is also the partial order on σ -flags in flag algebras. See Section 12 for further discussion of how this work related to flag algebras.

to be slightly easier to code (especially given its similarity to an optimization used in Part I), so we just consider that one. Intuitively, these two methods should have basically identical performance since the set of all graphs considered at any point during the algorithm should be the same, but we have not checked this carefully.

The generalization to the case that the graph is known to satisfy a general local condition, say $Q = \bigwedge_i \bigvee_j (H_{i,j}, k_i)$, and we are trying to prove it satisfies a join, say $R = \bigvee_j (I_j, \ell)$, is clear. At each step, we now just find an induced $H_{i,1}[k_i]$ in the revealed vertices for some i , check that it does not extend to $H_{i,j}$ for any j , and consider all possible (minimal) ways of attaching vertices so that it does extend to $H_{i,j}$ for some j . We prune any possibility that makes $G[\ell]$ extend to I_j for any j .

With this in mind, we consider Algorithm 5.

Algorithm 5: Prove that all graphs that satisfy $Q = \bigwedge_i \bigvee_j (H_{i,j}, k_i)$ also satisfy $R = \bigvee_j (I_j, \ell)$.

Input: Superfinite local conditions $Q = \bigwedge_i \bigvee_j (H_{i,j}, k_i)$ and $R = \bigvee_j (I_j, \ell)$.

Let $A \leftarrow \{I_1[\ell]\}$;

while A is nonempty **do**

 Let $H' \in A$;

 Let $A \leftarrow A \setminus \{H'\}$;

if H' satisfies Q **then**

return FAILURE;

else

 Let S be a subset of $V(H')$, i be an integer, and $\phi : S \rightarrow H_{i,1}[k_i]$ be a graph isomorphism such that S does not extend to $H_{i,j}$ for any j ;

for X in $\{1, \dots, \max_j (|V(H_{i,j})|) - k_i\}$ **do**

 Let $v_1, \dots, v_X$ be vertices (to be attached to H');

for $N \in \prod_{x=1}^X 2^{V(H') \cup \{v_1, \dots, v_{x-1}\}}$ **do**

 Let G be H' plus new vertices $v_1, \dots, v_x$ such that the neighbors of each v_j are N_j , the j th entry of N ;

if $G[S]$ extends to $H_{i,j}$ for some j , $G[\ell]$ does not extend to I_j for any j , and there is no $H'' \in A$ with $H'' \preceq_\ell G$ **then**

 Let $A \leftarrow A \cup \{G\}$;

return SUCCESS;

Lemma 7.1. *If Algorithm 5 returns SUCCESS, then all graphs satisfying Q also satisfy R .*

Proof. We prove by induction that at the beginning of each iteration of the **while** loop, we have proved that graphs satisfying Q also satisfy $R \vee \bigvee_{I \in A} (I, \ell)$. Let $A_0, A_1, \dots, A_j, \dots$ be the set A snapshotted at the beginning of the j th iteration of the **while** loop. The base case is obvious since all graphs satisfy $\bigvee_{I \in A_0} (I, \ell) = (I[\ell], \ell)$.

Between each iteration of the **while** loop, one graph H' is removed from the set A , and some graphs $G_1, \dots, G_m$ are possibly added to A . If the inductive step fails at iteration j , that is, there is a graph F

satisfying Q and $R \vee \bigvee_{I \in A_j} (I, \ell)$ but not $R \vee \bigvee_{I \in A_{j+1}} (I, \ell)$, then F must satisfy (H', ℓ) but not $\bigvee_{j=1}^m (G_j, \ell)$. Since F does not satisfy $\bigvee_{j=1}^m (G_j, \ell)$, it must contain $G_1[\ell] = I_1[\ell]$ as an induced subgraph, which we may assign labels 1 through ℓ . Since F does satisfy (H', ℓ) and $H'[\ell] = I_1[\ell]$ also, this induced $I_1[\ell]$ must extend to an induced H' in F , which we may assign the same labels as in H' ; in other words we may assume $F[V(H')] = H'$ as labeled graphs.

Since the algorithm was assumed to return SUCCESS, H' does not satisfy Q , and we can consider the S, i, ϕ chosen in the first line of the **else** statement. Since F was assumed to satisfy Q , $F[S]$ must extend to $H_{i,j}$ for some j . Thus there is some set T of up to $\max_j (|V(H_{i,j})|) - k_i$ vertices such that those vertices plus the vertices in S induce $H_{i,j}$ for some j (and the relevant graph isomorphism ψ has $\psi|_S = \phi$). It is possible some of these vertices to be among the $|V(H')|$ revealed vertices of F , but at least one vertex must not be revealed. The outer **for** loop goes over the possibilities for the number of vertices X of T that are not yet revealed. The inner **for** loop then goes over every possible way these vertices could be adjacent to the revealed vertices ($V(H)$) and each other ($\{v_1, \dots, v_{x-1}\}$). The graph G is constructed using this information.

We then check if $G[S] (= F[S])$ extends to $H_{i,j}$ for some j ; if this is not the case then the vertices of G did not actually contain the set T . It is clear then that we have shown that any graph satisfying $R \vee \bigvee_{I \in A_j} (I, \ell)$ also satisfies

$$R \vee \bigvee_{I \in A_j \setminus \{H'\} \cup \bigcup (G'_j, \ell)} (I, \ell),$$

where G'_j here goes over all the G constructed during the **for** loops that satisfy just the first condition of the **if** statement, that is, that $G'_j[S]$ extends to $H_{i,j}$ for some j .

But notice that

$$R \vee \bigvee_{I \in A_j \setminus \{H'\} \cup \bigcup (G'_j, \ell)} (I, \ell) = R \vee \bigvee_{I \in A_j \setminus \{H'\} \cup \bigcup (G_j, \ell)} (I, \ell)$$

where $\{G_j\}$ is the set of graphs that satisfy all three conditions of the **if** statement. This is because if $G'_j[\ell]$ extends to I_j for some j , then (G'_j, ℓ) is at least as strong of a condition as (I_j, ℓ) , and so $(I_j, \ell) \vee (G'_j, \ell) = (I_j, \ell)$. The same holds if there is some $H'' \in A$ with $H'' \preceq_\ell G'_j$ (which is equivalent to the statement that $G'_j[\ell]$ extends to H''). Thus there is no point in adding such G'_j to A . This proves the inductive step.

The condition to exit the **while** loop is that A is empty. If this occurs, then we have proven that graphs that satisfy Q also satisfy R . $\square$

Note that nowhere in the proof did we assume that F was finite. Thus the proof is valid for infinite graphs just as well as finite graphs. Therefore:

Corollary 7.2. *If there is any (possibly infinite) graph that satisfies Q but not R , then Algorithm 5 either*

returns FAILURE or does not halt. □

This is unfortunate for us. For example, it implies the following:

Corollary 7.3. *Algorithm 5 cannot be used to prove that finite k -ECTF graphs do not exist (for any k).*

Proof. The Henson graph $\mathcal{H}_3$ is a triangle-free (infinite) graph that is k -ECTF for all k . □

Note also that the “algorithm” as we have written it is not actually an algorithm that could be implemented yet. This is because we have not specified how to choose H' and S, i, ϕ when there are multiple options. In the following subsection we prove that there is a way to do this that is “at least as good” as any other method, in the sense that if *any* method proves that graphs satisfying Q also satisfy R , then the method described also does (though possibly less efficiently).

Finally, note that if we are trying to prove a meet of conditions using Q , all we have to do is prove each constituent join using Algorithm 5. From now on, if the input R is a meet, we understand that it means to run the algorithm on each constituent join of R separately; if all runs return SUCCESS, then any graph satisfying Q also satisfies R .

7.2 Some Properties of the Algorithm

Let $\mathcal{A}$ be a method to choose H' and S, i, ϕ given A and any other information. Specifically, $\mathcal{A}$ is any pair $(\mathcal{A}_1, \mathcal{A}_2)$ of (not necessarily computable) functions from “histories” of the algorithm to H' in $\mathcal{A}_1$ ’s case and to (S, i, ϕ) in $\mathcal{A}_2$ ’s case, such that $H' \in A$ and ϕ is a graph isomorphism from $H'[S]$ to $H_{i,1}[k_i]$. This represents any way of choosing H' and S, i, ϕ using any information that is available to the algorithm or was available to it at any point in time. We say $Q \vdash_{\mathcal{A}} R$ if the version of the algorithm that chooses H' and S, i, ϕ according to $\mathcal{A}$ returns SUCCESS on input (Q, R) . We call such $\mathcal{A}$ *choice functions*.

We show there is a particular (computable) choice function $\mathcal{A}^*$ such that if there is any $\mathcal{A}$ such that $Q \vdash_{\mathcal{A}} R$, then $Q \vdash_{\mathcal{A}^*} R$. This is defined as follows. The function $\mathcal{A}_1^*$ simply chooses the H' that was most recently added to A . The function $\mathcal{A}_2^*$ chooses the smallest valid (S, i, ϕ) under the following order: (S_1, i_1, ϕ_1) is preferred to (S_2, i_2, ϕ_2) if $\max(S_1) < \max(S_2)$, or $\max(S_1) = \max(S_2)$ and $i_1 < i_2$, or $\max(S_1) = \max(S_2)$ and $i_1 = i_2$ and $\phi_1 < \phi_2$ under “inverse lexicographical ordering,” meaning the smallest v such that $\phi_1^{-1}(v) \neq \phi_2^{-1}(v)$ has $\phi_1^{-1}(v) < \phi_2^{-1}(v)$. Actually, the order on ϕ_1 and ϕ_2 can be anything; only the order on S_1, S_2, i_1 , and i_2 matters, but this ordering on ϕ ’s is easy to code, so we chose this definition. Additionally, the behavior of $\mathcal{A}_1^*$ is not really important.

Theorem 7.4. *If there is any $\mathcal{A}$ such that $Q \vdash_{\mathcal{A}} R$, then $Q \vdash_{\mathcal{A}^*} R$.*

Proof. Recall the proof of Lemma 7.1, where we argued (essentially) that one can remove the check that there is no $H'' \in A$ with $H'' \preceq_\ell G$ from the innermost **if** statement, and whether the algorithm returns SUCCESS or not does not change by this modification. We will consider the version without this check, because it makes the analysis a bit simpler. In particular, it removes any dependence on $\mathcal{A}_1$ and $\mathcal{A}_1^*$.

During each iteration of the **while** loop, one graph H' is removed from A , and possibly many graphs are added to A . Call these graphs the $\mathcal{A}$ -children of H' . We show that every $\mathcal{A}$ -child D of any graph H has an *iterated* $\mathcal{A}^*$ -child D^* of H such that $D \preceq_\ell D^*$. An iterated child is just a child of a child of a ... of H . This result implies the theorem, because since $Q \vdash_{\mathcal{A}} R$, there is a bounded number of times you can take $\mathcal{A}$ -children of $I_1[\ell]$, so there is also a bounded number of times you can take $\mathcal{A}^*$ -children of $I_1[\ell]$. This implies the set A eventually becomes empty when the algorithm is run with choice function $\mathcal{A}^*$, so $\mathcal{A}^*$ also returns SUCCESS on the same input.

To do this, suppose that $\mathcal{A}$ chooses (S', i', ϕ') for graph H , while $\mathcal{A}^*$ chooses (S^*, i^*, ϕ^*) . Then we have $(S^*, i^*, \phi^*) \leq (S', i', \phi')$ (where $<$ is the order given in the definition of $\mathcal{A}^*$), and, importantly, there are finitely many possible (S, i, ϕ) between (S^*, i^*, ϕ^*) and (S', i', ϕ') , say k of them. Then every $\mathcal{A}$ -child D of H has a k' -iterated $\mathcal{A}^*$ -child D^* of H with $D \preceq_\ell D^*$, for some $k' \leq k$. To see this, note the number of (S, i, ϕ) between (S^*, i^*, ϕ^*) and (S', i', ϕ') does not increase when vertices are added to H to form a child. Each iteration of taking an $\mathcal{A}$ -child, the smallest (S, i, ϕ) between (S^*, i^*, ϕ^*) and (S', i', ϕ') that has not yet been “satisfied” is then satisfied in that iteration. Therefore, the choice (S', i', ϕ') must have been satisfied in at most k iterations, that is, in any k -iterated $\mathcal{A}^*$ -child D^* , S' (with the given isomorphism ϕ') extends to $H_{i',j}$ for some j , which implies that D^* contains (in the sense of $\preceq_\ell$) some $\mathcal{A}$ -child of H . $\square$

Due to the theorem above, we can drop the $\mathcal{A}$ quantifier from $\vdash$, and just write $Q \vdash R$ if Algorithm 5 returns SUCCESS when run with input (Q, R) and choice function $\mathcal{A}^*$.

Lemma 7.1 is then restated to be:

Theorem 7.5. *If $Q \vdash R$ then all graphs that satisfy Q satisfy R .* $\square$

The converse of this statement is also true:

Theorem 7.6. *If all graphs that satisfy Q satisfy R , then $Q \vdash R$.*

Note that in these two theorems, by “all graphs” we include infinite graphs.

Proof (Theorem 7.6). The contrapositive is that if it is not the case that $Q \vdash R$, then there is a (possibly infinite) graph G that satisfies Q but not R . If it is not the case that $Q \vdash R$, then either the algorithm returns FAILURE, in which case the algorithm gives an explicit finite graph satisfying Q but not R , or the

algorithm does not halt. In the latter case, it is easy to see there is an infinite sequence of graphs

$$J_1 < J_2 < J_3 < \dots$$

added to A while the algorithm runs. The meaning of “ $<$ ” here is that each graph contains the previous graph as a proper induced subgraph with the same labels. In particular, we can take J_{i+1} to be some $\mathcal{A}^*$ -child of J_i for all i , using the notion of children from the previous proof. Then one can define $J_\infty = \lim_{i \rightarrow \infty} J_i$. In this infinite graph, to find out whether vertex a is adjacent to vertex b , you just need to look at an J_i with i large enough so that J_i contains both vertex a and b .

We claim that J_∞ satisfies Q but not R . It does not satisfy R because vertices $\{1, \dots, \ell\}$ induces $I_1[\ell]$ but does not extend to any I_j ; if it did extend, this would also be the case for some J_i with finite index, but we assumed that J_i was added to A , so $J_i[\ell]$ does not extend to any I_j . If J_∞ did not satisfy Q , then there is some (finite) set $S \subseteq V(J_\infty)$ that shows this; this S' has $J_\infty[S]$ isomorphic to some $H_{i',1}[k_{i'}]$ with isomorphism ϕ' , but it does not extend to $H_{i',j}$ for any j . But this also holds for all the J_i , a contradiction by similar logic to the previous proof: there is some k such that all k -iterated children of J_1 satisfy (S', i', ϕ') , so J_{k+1} must satisfy (S', i', ϕ') , for instance. This proves the contrapositive statement. $\square$

We find this result philosophically interesting. It means essentially that Algorithm 5 is “complete”: any true statement about implications between local conditions can be proven by this algorithm. Unfortunately, it has one major downside, which is that there may be no finite graph satisfying Q but not R , but there is an infinite graph. This is suspected to be the case for $Q = “k\text{-ECTF}”$ and $R = (K_3, 0)$, for example, for sufficiently large k . Also, this algorithm only works for superfinite local conditions, which is a rather severe limitation, since many interesting graph properties are infinite local conditions (or finite but not superfinite local conditions).

This result also immediately implies another nice property of this algorithm, namely:

Corollary 7.7. *If $Q \vdash R$ then $R \vdash S$ then $Q \vdash S$.*

Proof. Use Theorem 7.5 to turn the statements “ $Q \vdash R$ ” and “ $R \vdash S$ ” into “all graphs that satisfy Q satisfy R ” and “all graphs that satisfy R satisfy S ,” which shows that all graphs that satisfy Q satisfy S . Then apply Theorem 7.6. $\square$

We may write $Q \vdash R \vdash S$ to indicate the situation in the corollary above. We have additionally:

Corollary 7.8. *If $Q \approx R$ and $S \approx T$ and $Q \vdash S$ then $R \vdash T$.*

Proof. By Theorem 7.6, $R \vdash Q$ and $S \vdash T$, so then by the previous corollary $R \vdash Q \vdash S \vdash T$. $\square$

Therefore, roughly speaking, the specific labeling of the vertices in the input local conditions to Algorithm 5 does not affect the end result of the algorithm. It may affect the runtime, so we still include such labeling information in the various figures in this part. Similar results were given in Part I about the behavior of the algorithms considered in that part, but the proofs were much more complicated because almost every proof had to “look into” the details of the algorithm like we do in the proofs of Lemma 7.1 and Theorem 7.6. The insight in this part is that Theorems 7.5 and 7.6 are really the key reasons that algorithm has the properties in the corollaries above.

7.3 Extensions to the Algorithm

In the context of $\text{HC-}\{\overline{K_3}\}$, we are primarily interested in proving things about $\overline{K_3}$ -free graphs. We can modify Algorithm 5 to capture this setting. In particular, all we need to do is **and** the condition “ G has no stable set of size 3” to the innermost **if** statement. We call the resulting algorithm Algorithm 5’.

Theorem 7.9. *If Algorithm 5’ returns SUCCESS on input (Q, R) , then all graphs satisfying Q either also satisfy R or have a stable set of size 3.*

Proof. The proof is the basically same as Lemma 7.1, so we omit the details. $\square$

Clearly the same thing can be done for any set S of forbidden subgraphs,² or indeed for any monotone increasing property so long as the property has an algorithm for testing it.

All of the theorems from the previous subsection also hold for Algorithm 5’ (with the appropriate modifications to restrict to graphs with no stable set of size 3); we omit listing their statements and proofs. We define $Q \vdash' R$ to mean that Algorithm 5’ returns SUCCESS when run with input (Q, R) and choice function $\mathcal{A}^*$ (the same special choice function defined previously).

8 Strategy

Consider the following local condition. Let $\mathcal{H}$ be the set of graphs satisfying $\text{HC-}\{\overline{K_3}\}$. For $H \in \mathcal{H}$ let X_H be the set of all models of $K_{\lceil |V(H)|/2 \rceil}$ in H , i.e. X_H is all sets of $\lceil |V(H)|/2 \rceil$ pairwise disjoint, pairwise adjacent, connected subsets of $V(H)$. For $S \in M \in X_H$ let $H_{M,S}$ be the set of $\overline{K_3}$ -free graphs consisting of H plus a vertex that is not adjacent to S . Then the local condition we consider is

$$Q^{\text{HC}} = \bigwedge_{H \in \mathcal{H}} \bigwedge_{M \in X_H} \bigvee_{S \in M} \bigvee_{H' \in H_{M,S}} (H', V(H)).$$

²This is also a familiar setting in flag algebras, which are often defined with in terms of a given set of forbidden subgraphs.

We show that this local condition basically captures the property of being a counterexample to $\text{HC-}\{\overline{K_3}\}$. Specifically:

Theorem 8.1. *If G is a graph that satisfies Q^{HC} with no stable set of size 3 that is minimal among such graphs, then G is a minimal counterexample to $\text{HC-}\{\overline{K_3}\}$. The converse also holds.*

Proof. ($\Rightarrow$) Suppose G satisfies Q^{HC} , has no stable set of size 3, and is minimal among such graphs. Then G cannot have a model of $K_{\lceil |V(G)|/2 \rceil}$, since if it did, it would appear in the set $\mathcal{H}$, and the local condition forces G to have at least one more vertex than it does. So G is a counterexample to $\text{HC-}\{\overline{K_3}\}$.

We prove inductively that no proper subgraph is a counterexample to $\text{HC-}\{\overline{K_3}\}$. The base case (the empty graph) is trivial. Suppose G' is a proper subgraph of G , and inductively all proper subgraphs of G' are not counterexamples to $\text{HC-}\{\overline{K_3}\}$. Since G' does not satisfy Q^{HC} , there is some subgraph G'' of G' and model M of $K_{\lceil |V(G'')|/2 \rceil}$ such that every vertex of G' not in G'' is adjacent to every set in M . The set of vertices of $G' \setminus G''$ has a $K_{\lceil |V(G' \setminus G'')|/2 \rceil}$ minor M' by inductive hypothesis, so G' has a $K_{\lceil |V(G')|/2 \rceil}$ minor, namely $M \cup M'$. This proves the inductive step.

($\Leftarrow$) Suppose G is a minimal counterexample to $\text{HC-}\{\overline{K_3}\}$. Suppose G also did not satisfy Q^{HC} . Then there is some subgraph G' that has a $K_{\lceil |V(G')|/2 \rceil}$ model M such that every vertex of $G \setminus G'$ is adjacent to every class in the model. Since G was minimal, $G \setminus G'$ has a $K_{\lceil |V(G \setminus G')|/2 \rceil}$ model M' ; then $M \cup M'$ is a $K_{\lceil |V(G)|/2 \rceil}$ model in G , contradicting that G was a counterexample to $\text{HC-}\{\overline{K_3}\}$. So G must satisfy Q^{HC} . Also G obviously has no stable set of size 3. Finally, it is easy to see that no proper subgraph satisfies Q^{HC} , since any proper subgraph of G is in $\mathcal{H}$. $\square$

One can check:

Proposition 8.2. *The graph consisting of two nonadjacent cliques with infinitely many vertices each satisfies Q^{HC} and has no stable set of size 3.* $\square$

This does not refute Hadwiger's Conjecture, of course, but it does provide a barrier to proving local conditions using Q^{HC} (or the local condition Q we have derived so far). For example, Q^{HC} by itself cannot prove that counterexamples to $\text{HC-}\{\overline{K_3}\}$ contain any induced subgraphs except disjoint unions of two cliques. In Part I, Lemma 1.4 was the ingredient that let us get around this barrier (though the use of Lemma 1.4 was not framed this way in that part). The same more or less holds in this part, with some differences.

In particular, we derive a second local condition, not provable from Q^{HC} , that finite counterexamples to $\text{HC-}\{\overline{K_3}\}$ must satisfy. We call this condition $Q^{4\text{CC}}$, defined as follows. Let G be a graph with no stable set of size 3. Let S_G be the set of $N \subsetneq V(G)$ such that the graph G' formed by adding one vertex v to G with $N_{G'}(v) = N$ has no stable set of size 3. For a subset $S' \subseteq S_G$, define $G_{S'}$ to be the following graph. The

vertices of $G_{S'}$ are $V(G) \cup \{v_N \mid N \in S'\}$. The edges are $E(G) \cup \{(u, v_N) \mid u \in N\} \cup \{(v_N, v_{N'}) \mid N \cup N' \neq V(G)\}$. If $G_{S'}$ admits four cliques $C_1, \dots, C_4$ such that $C_1 \cup \dots \cup C_4$ cover $V(G_{S'})$ and

$$\sum_{i=1}^4 |C_i \cap V(G)| \geq |V(G)| + 2,$$

then S' is called *valid* (relative to the graph G). For brevity we call such C_i a *four-clique cover* of G' , like we did in Part I. Then we define

$$Q^{4CC} = \bigwedge_{\overline{K_3}\text{-free}} \bigwedge_{G \text{ valid}} \bigwedge_{S'} \left((G \vee K_1, |V(G)|) \vee \bigvee_{\substack{G'=G+v \\ N_{G'}(v) \in S_G \setminus S'}} (G', |V(G)|) \right).$$

Recall from Part I that $G \vee K_1 = \overline{G \sqcup \overline{K_1}}$ is the graph formed by adding one vertex to G adjacent to all other vertices, called the join of G and K_1 . (We apologize for the unfortunate clash of notation with joins of local conditions.) We have:

Theorem 8.3. *If H is a counterexample to $HC\text{-}\{\overline{K_3}\}$, then H satisfies Q^{4CC} .*

Proof. Consider any subgraph G of H and valid subset S' . If H did not satisfy the join

$$(G \vee K_1, |V(G)|) \vee \bigvee_{\substack{G'=G+v \\ N_{G'}(v) \in S_G \setminus S'}} (G', |V(G)|)$$

then the vertices of H are the vertices of G , plus vertices whose subset of neighbors in $V(G)$ is equal to N for each $N \in S'$. The set of such vertices for each N form a clique in H since they have a common nonneighbor (namely any vertex of $V(G) \setminus N$, which is nonempty since the definition of $S \supseteq S'$ excluded $V(G)$). Additionally, many pairs of these cliques must be complete to each other, or else H would have a stable set of size 3; these pairs correspond to the third set in the union forming the edges of G' ($\{(v_N, v_{N'}) \mid N \cup N' \neq V(G)\}$).

Thus a four-clique cover $C_1, \dots, C_4$ of the relevant G' gives a four-clique cover of H , namely this is formed by replacing $v_N \in (V(G') \setminus V(G)) \cap C_i$ by the clique formed by vertices with neighborhood in $V(G)$ equal to N . But by Pigeonhole Principle this means that there is a clique of H containing at least $\lceil \frac{|V(H)|+2}{4} \rceil$ vertices; by Lemma 1.4, this contradicts that H was a counterexample to $HC\text{-}\{\overline{K_3}\}$. The choice of G and S' was arbitrary, so this means H must satisfy Q^{4CC} . $\square$

Obviously Q^{4CC} is equivalent to the same local condition but restricting the inner meet to only maximal valid S' (with respect to inclusion).

Let Q_k^{HC} be the local condition Q^{HC} but where the first (outer) meet is restricted only to graphs in $\mathcal{H}$ with at most k vertices. Similarly, define $Q_k^{4\text{CC}}$ to be the fragment of $Q^{4\text{CC}}$ where the outermost meet is restricted to graphs with k or fewer vertices. Then the general strategy for proving Theorem 5.2 is as follows:

1. Choose a particular k_1, k_2 and let $Q = Q_{k_1}^{\text{HC}} \wedge Q_{k_2}^{4\text{CC}}$. We chose $k_1 = k_2 = 4$, mostly due to computational constraints.
2. Perform some simplification steps on Q to replace Q by an equivalent local condition with fewer constituent basic local conditions. This step (probably) speeds up the algorithms. This is done in two substeps: (a) “pruning” superfluous conditions from Q , then (b) using Algorithm 5’ to prove seemingly stronger joins than Q contained to begin with (but that are actually equivalent in strength).
3. Use the final Q , along with some additional results, to prove Theorem 5.2.

These steps are elaborated in the following subsections.

8.1 Step 1: Starting Point

Note:

Lemma 8.4. $Q_{2k}^{\text{HC}} \vdash' Q_{2k+1}^{\text{HC}}$ for all k .

Proof. Let $H \in \mathcal{H}$ have $2k+1$ vertices. Note in a model M of $K_{\lceil |V(H)|/2 \rceil}$ in H , since H has an odd number of vertices, at least one of the models must have just one vertex, say $v \in V(H)$. Then $H \setminus v$ has the model $M \setminus \{v\}$ of $K_{\lceil |V(H)-1|/2 \rceil}$. If a graph G satisfies Q_{2k}^{HC} and contains H , say with the same labels as H , there must be a vertex in G that is not adjacent to one of the classes in $M \setminus \{v\}$. This vertex cannot be v since v is adjacent to all of these classes. Therefore G satisfies

$$\bigvee_{S \in M} \bigvee_{H' \in H_{M,S}} (H', V(H))$$

for this particular H, M . But this holds for all relevant H, M , so if G satisfies Q_{2k}^{HC} , it also satisfies Q_{2k+1}^{HC} . This theorem then follows from Theorem 7.6. $\square$

This implies, due to Corollary 7.7, that there is no benefit in choosing an odd k as our starting point, since it is equally strong as the even k just below it. At the point of $k = 8$, the initial local condition $Q = Q_8^{\text{HC}}$ becomes infeasible to compute, at least for our (admittedly unoptimized) code. So the choice is really between $k = 4$ and $k = 6$. If one does the pruning step described in the next subsection (without including the fragment of $Q^{4\text{CC}}$), $k = 6$ gives a meet of 346 joins, containing over 13000 basic local conditions

in total, while $k = 4$ has just 17 joins and 120 basic local conditions. To reduce the complexity of Theorem 5.2 to something that can be drawn in a reasonable amount of space, we ultimately decided on $k = 4$. Note that $k = 4$ is already more than we used in Part I; in that part, we used just that counterexamples have no dominating edges, which turns out to be (basically) equivalent to Q_2^{HC} .

There is a second potential reason to choose $k = 4$ due to a result regarding SSH. However, this is not that relevant given our use of Lemma 1.4, which is not known to be valid for SSH, so we relegate the discussion to Section 11.1.

The largest k for which we could compute in a reasonable amount of time $Q_k^{4\text{CC}}$ is $k = 4$. The main computational hurdle is that finding four-clique covers is difficult, and some 5-vertex graphs G have over 100 maximal valid S' , such as the graph formed by removing two incident edges from K_5 . More optimized code could probably compute $Q_5^{4\text{CC}}$, but we have the same problem as Q_6^{HC} that the resulting local condition is too large to reasonably draw in a paper.

8.2 Step 2(a): Pruning

The basis of the “pruning” simplification step is the following observation:

Proposition 8.5. *If $S = \{(S_1, k), (S_2, k), \dots\}$ and $T = \{(T_1, k), (T_2, k), \dots\}$ with $S_1[k] = S_i[k] = T_j[k]$ for all i, j , and $T \subseteq S$, then $S \wedge T \approx T$.* $\square$

Then we consider the following algorithm:

Algorithm 6: Given a local condition $Q = \bigwedge_{i \in I} \bigvee_j (H_{i,j}, k_i)$, produce a local condition $Q' = \bigwedge_{i \in I'} \bigvee_j (H_{i,j}, k_i)$ such that $Q \approx Q'$ and $I' \subseteq I$.

Input: Superfinite local condition $Q = \bigwedge_i \bigvee_j (H_{i,j}, k_i)$.

Let $I' \leftarrow I$;

for $i_1 \in I$ **do**

for $i_2 \in I' \setminus \{i_1\}$ **do**

if $H_{i_1,1}[k_{i_1}] \approx H_{i_2,1}[k_{i_2}]$ **then**

for $\pi \in S_{k_1}$ **do**

 For each j let $H'_{i_1,j}$ be the graph formed by permuting the vertices of $H_{i_1,j}$ according to π ;

if $(|\{H_{i_1,j}\}_j| \neq |\{H_{i_2,j}\}_j| \text{ or } i_1 > i_2) \text{ and } H'_{i_1,1}[k_{i_1}] = H_{i_1,1}[k_{i_2}] \text{ and for all } j \text{ there is a } j' \text{ such that } H'_{i_1,j} \preceq_{k_i} H_{i_2,j'} \text{ then}$

 Let $I' \leftarrow I' \setminus \{i_1\}$;

 Break out of the inner two **for** loops;

return $\bigwedge_{i \in I'} \bigvee_j (H_{i,j}, k_i)$;

We have:

Theorem 8.6. *Let Q' be the output of Algorithm 6 run with input Q . Then $Q \approx Q'$.*

Proof. We omit a detailed proof and just sketch it. For each pair of joins J_1, J_2 of Q , corresponding to I and I' in the algorithm, the algorithm computes whether J_2 is at least as strong as J_1 . To do this, it tries every possibility for permuting the vertices of the pattern of one of the joins so that it equals the pattern for the other join (as labeled graphs). It then computes whether every target in J_1 is contained in the sense of $\preceq$ in a target of J_2 . If so, then J_2 is at least as strong as J_1 . If J_1 and J_2 have the same number of constituent basic local conditions then they are actually equivalent, and we make sure to keep the one with smaller index in Q . Some additional modifications speed up the algorithm slightly, like only checking J_1 against the J_2 that have not yet been pruned, and including a check at the start to make sure their patterns are isomorphic. $\square$

After pruning $Q_4^{\text{HC}} \wedge Q_4^{\text{4CC}}$, Q is left with 32 joins and 138 constituent basic local conditions.

8.3 Step 2(b): Further Alterations

Let J be a constituent join of Q and S a proper subset of the constituent basic local conditions of J . If $Q \vdash \bigvee_{B \in S} B$, then obviously $Q \approx (\bigwedge_{J' \neq J} J') \wedge \bigvee_{B \in S} B$; i.e. we may replace J' by a simpler (and seemingly stronger) local condition. This gives an obvious algorithm for further simplifying Q , namely to take a subset of some join J of Q and try to prove the relevant stronger join from Q . If successful, we replace Q by the simplified local condition and repeat. This is close to but not quite what we do in this step.

Instead, for each graph G on 4 or fewer vertices, we consider all joins J of basic local conditions (H, k) with pattern G and $k = |V(H)| - 1$. All of the $\bigvee_{B \in S} B$ in the previous paragraph are of this form for our particular Q , but this set contains some additional joins. If we are successful for a particular J , we replace Q by $Q \wedge J$ then prune as in the previous substep. The resulting local condition after doing this is still equivalent to Q , of course, due to Corollary 7.7 and Theorem 8.6.

If a particular J fails, on the other hand, then either Algorithm 5' returns FAILURE or it loops. In the latter case, we need to stop at some point and move on to a different J . To do this, we keep track of two values while running the algorithm. The first is the total count of the number of graphs added to the set A at any point. The second is weight, which has the same definition as in Part I, namely it is the sum of $2^{|V(G)|}$ where G ranges over all of the graphs ever added to A . We chose somewhat arbitrarily to stop the algorithm if ever at least 100 graphs were added to A or the weight up to that point was 10^5 or higher.

We refer the reader to Section 9 for all of the alterations performed in this substep. After this step, Q consists of 49 joins and 115 basic local conditions, shown in Figures 8 and 9. Note that there are more joins

than the end of step 2(a), but fewer constituent local conditions; both of these point to the local condition at the end of step 2(b) appearing stronger than the one at the end of step 2(a).

8.4 Step 3: Final Proof

At this point, the Q obtained is not quite what is used to prove Theorem 5.2. Rather, this Q serves as the input for a modified version of Algorithm 5'. Before describing that algorithm, we describe a slightly more general use of four-clique covers relying on Lemma 5.1. This is similar to Part I. First we restate the four-clique cover argument. Consider a graph G with no stable set of size 3, and an induced subgraph H , say induced by vertices $U \subseteq V(G)$. We saw that the set of vertices that are not complete to H can be partitioned into cliques based on their neighborhood in H , and some pairs of these cliques must be complete to each other. If we assume G has no vertices complete to H , and possibly also forbid some other neighborhood sets, and if the resulting graph had a cover using four cliques with at least two vertices of overlap, then G is not a counterexample to $\text{HC-}\{\overline{K_3}\}$. Thus any induced H in a minimal counterexample extends to $H \vee K_1$ (the graph formed by adding one vertex to H complete to H), or H plus one vertex with one of the forbidden neighborhoods.

The more general version instead assumes that the vertices complete to H form some k cliques, where $k = 0$ is the original version. The conclusion is that a minimal counterexample extends to $H \vee F_k$ for some particular graph F_k (or H extends to H plus one vertex with one of the forbidden neighborhoods as before) since $\{\overline{K_3}, F_k\}$ -free graphs have clique cover number at most k . Lemma 5.1 tells us what the graphs F_k should be:

$$F_0 = K_1, \quad F_1 = \overline{K_2}, \quad F_2 = P_4, \quad F_3 = \overline{T'_1}$$

where T'_1 is given in Figure 6. (Note that Lemma 5.1 is framed in terms of the chromatic number of graphs, but this is the same as the clique cover number of the complement.) One could take $F_3 = \overline{T'_2}$ instead and likely get slightly different results.

So we consider a modified version of Algorithm 5, Algorithm 7.

We have:

Theorem 8.7. *If Algorithm 7 returns SUCCESS, then all graphs satisfying $Q \wedge (I^*, \ell)$ also satisfy R or are not counterexamples to $\text{HC-}\{\overline{K_3}\}$.*

Proof. We just sketch the argument. The more general four-clique cover argument is easily seen to give a local condition that all counterexamples to $\text{HC-}\{\overline{K_3}\}$ must satisfy (similar to $Q^{4\text{CC}}$). The check for such four-clique covers with $k > 0$ is equivalent to computing if the graph satisfies some join contained in this

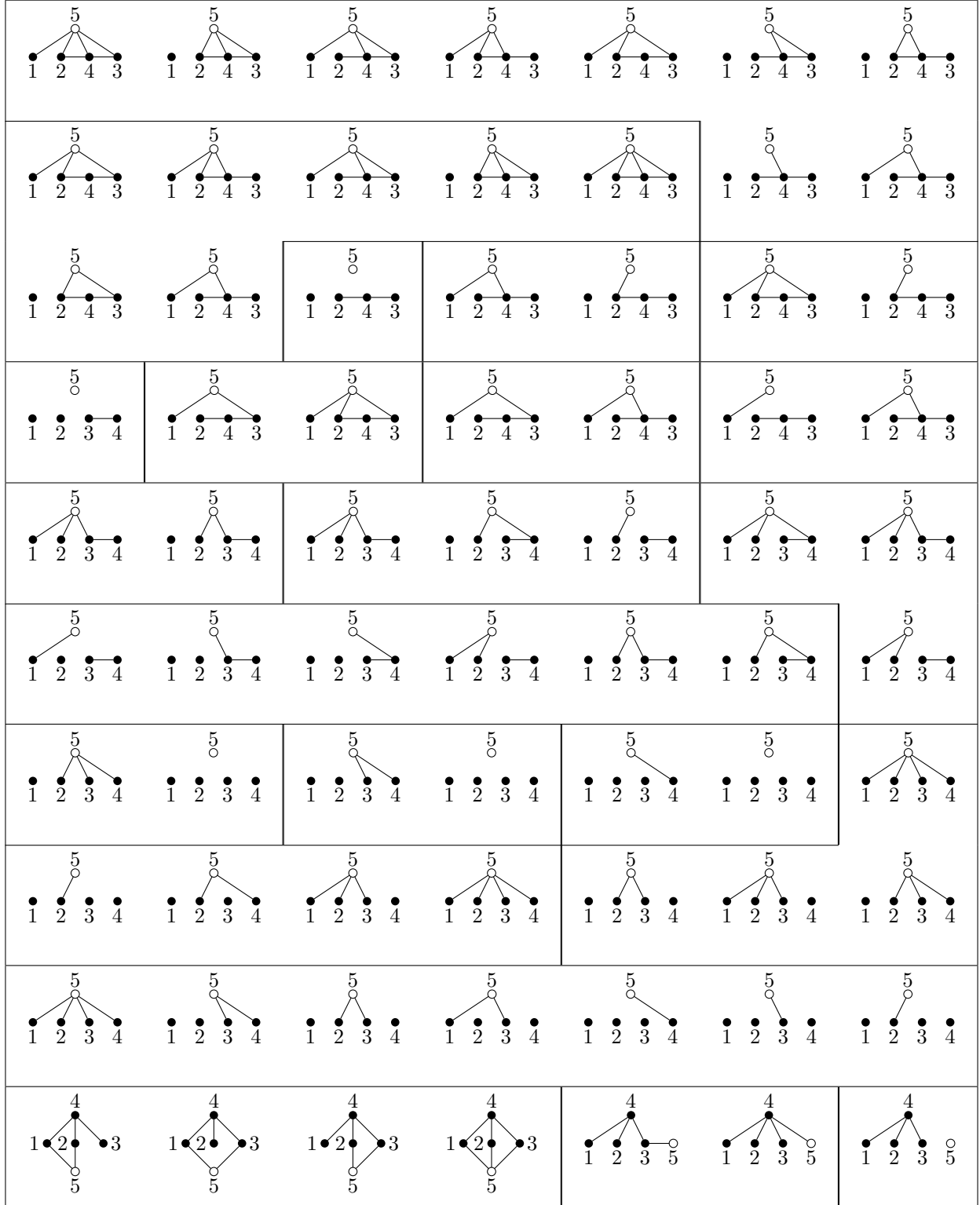


Figure 8: The local condition Q at the end of step 2. Each box is a join, and the local condition Q is the meet of all these joins. We display the complements of every graph. The order of the joins in the meet and graphs within each join are written in boustrophedon in this figure: the rows alternate between left-to-right and right-to-left. This is part 1; the figure continues on the next page.

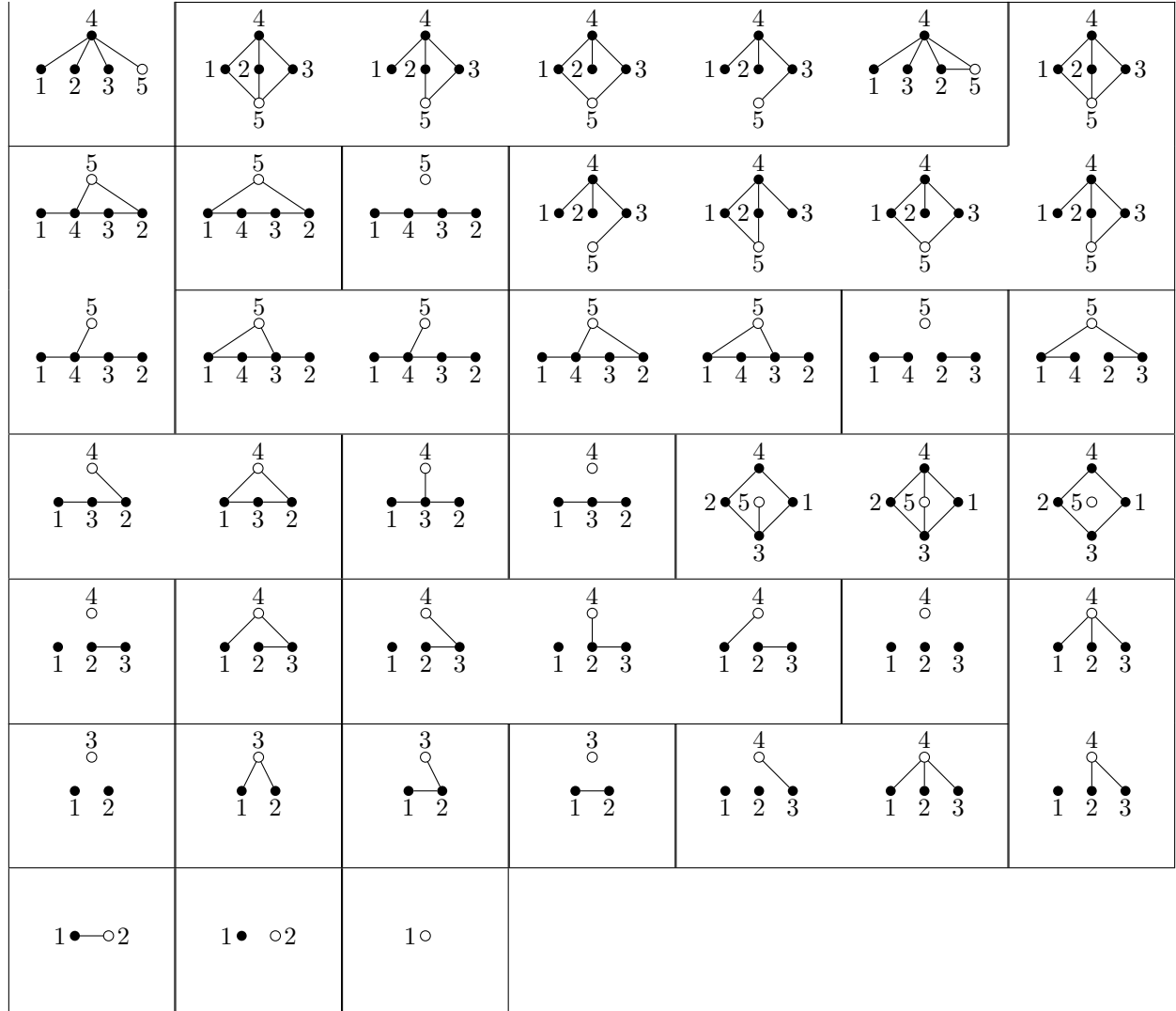


Figure 9: The local condition Q at the end of step 2. This is part 2; continuing from the previous page. Note the top-left graph is part of a join starting from the previous page.

Algorithm 7: Prove that all graphs that satisfy $Q = \bigwedge_i \bigvee_j (H_{i,j}, k_i)$ and (I^*, ℓ) also satisfy $R = \bigvee_j (I_j, \ell)$, where $I^*[\ell] = I_j[\ell]$, or are not counterexamples to $\text{HC-}\{\bar{K}_3\}$.

Input: Superfinite local condition $Q = (\bigwedge_i \bigvee_j (H_{i,j}, k_i))$, basic local condition (I^*, ℓ) , and superfinite local condition $R = \bigvee_j (I_j, \ell)$.

Let $A \leftarrow \{I^*[\ell]\}$;

while A is nonempty **do**

 Let $H' \in A$;

 Let $A \leftarrow A \setminus \{H'\}$;

if H' satisfies Q **then**

return FAILURE;

else

 Let S be a subset of $V(H')$, i be an integer, and $\phi : S \rightarrow H_{i,1}[k_i]$ be a graph isomorphism such that S does not extend to $H_{i,j}$ for any j , such that $(\max(S), i, \phi^{-1})$ is minimized among all such possibilities;

if $\max(S)$ is greater than the $\max(S')$ chosen for any parent of H' during the process of running the algorithm **then**

for $k \in \{3, 2, 1, 0\}$ (in this order) **do**

 Construct the graph G' consisting of H' plus one vertex v_N for each neighborhood $N \subsetneq V(H')$ such that $(H + v_N)[\ell]$ does not extend to I_j for any j and there is no $H'' \in A$ with $H'' \preceq_\ell H + v_N$ and $H + v_N$ has no stable set of size 3. The neighborhood of each v_N in H' is N . For each N, N' , if $G' = H' + v_N + v_{N'}$ has $G'[\ell]$ extend to I_j for some j or there is an $H'' \in A$ with $H'' \preceq_\ell G'$ or G' has a stable set of size three, add the edge $(v_N, v_{N'})$ to G ;

 Add k vertices $v_1, \dots, v_k$ to G that are complete to the H' in H' . Additionally for each N if $G' = H' + v_1 + v_N$ has $G'[\ell]$ extend to I_j for some j or there is an $H'' \in A$ with $H'' \preceq_\ell G'$, add the edge (v_i, v_N) to G for each $i = 1, \dots, k$;

if there are four cliques $C_1, \dots, C_4$ covering G with $\sum_i |C_i \cap V(H')| \geq |V(H')| + 2$ **then**

if $(H' \vee F_k)[\ell]$ does not extend to I_j for any j , and there is no $H'' \in A$ with $H'' \preceq_\ell H' \vee F_k$ **then**

 Let $A \leftarrow A \cup \{H' \vee F_k\}$;

 Break out of the **for** loop;

else

for X in $\{1, \dots, \max_j (|V(H_{i,j})|) - k_i\}$ **do**

 Let $v_1, \dots, v_X$ be vertices (to be attached to H');

for $N \in \prod_{x=1}^X 2^{V(H') \cup \{v_1, \dots, v_{x-1}\}}$ **do**

 Let G be H' plus new vertices $v_1, \dots, v_x$ such that the neighbors of each v_j are N_j , the j th entry of N ;

if $G[S]$ extends to $H_{i,j}$ for some j , $G[\ell]$ does not extend to I_j for any j , and there is no $H'' \in A$ with $H'' \preceq_\ell G$, and G has no stable set of size 3 **then**

 Let $A \leftarrow A \cup \{G\}$;

return SUCCESS;

local condition, and the resulting graphs (actually, just one graph) added to A are exactly the graphs that Algorithm 5' would add to A given this local condition. Thus, the piece of the algorithm inside the outermost **else** is equivalent to Algorithm 5', just with an extra local condition baked in. The loop of the algorithm

(and corresponding inductive step in the proof) are thus correct.

The basic local condition (I^*, ℓ) serves as an alternative initialization of A in the algorithm, which is the base case in the proof. It is easy to see that setting $A \leftarrow \{I^*[\ell]\}$ is equivalent to combining Q and (I^*, ℓ) in the input to $Q \wedge (I^*, \ell)$, setting $A \leftarrow \{I_1[\ell]\}$ to start, and in the very first iteration of the loop choosing $S = \{1, \dots, \ell\}$, i the appropriate index to get (I^*, ℓ) , and $\phi = \text{id}_{\{1, \dots, \ell\}}$, then in future iterations pretending Q does not actually contain (I^*, ℓ) . Therefore the modified algorithm is equivalent to Algorithm 5', just run with slightly different input; in particular, it is correct. $\square$

Note that finding four-clique covers is far more computationally intensive than the other branch of Algorithm 7, so prove Theorem 5.2, we use Algorithm 5' rather than Algorithm 7 in the early stages, only switching to the more advanced algorithm when it starts becoming necessary.

The strategy for proving Theorem 5.2 is to first prove some weaker results involving subgraphs of the basic local conditions of Figure 7. These graphs then become the starting point (I^*, ℓ) to feed into the algorithms.

9 Data For Computer Verification

As we explained at the end of the previous section, the proof of Theorem 5.2 proceeds in a few steps. The code to compute $Q_4^{\text{HC}} \wedge Q_4^{\text{4CC}}$ (step 1) is relatively straightforward, as is step 2(a). In step 2(b), we modify the local condition in many ways. In total, 30 alterations are made. Each individual alteration is given in Figure 10 in the order we perform them. To prove each alteration, one should run Algorithm 5' with the input local condition Q being the current Q computed before the alteration in question and the input R being the join shown in that row of Figure 10. Then one should replace the working Q with the output of Algorithm 6 run on $Q \wedge R$ before moving on to the next row.

Finally, in step 3, we incrementally prove larger basic local conditions using Algorithm 5' and eventually Algorithm 7. The logic for this is shown in Figure 11: each arrow $R \rightarrow S$ means to run Algorithm 5' with input $(Q \wedge S, T)$, where S is the starting point (I^*, ℓ) (replacing (I_1, ℓ) at the beginning of Algorithm 5'). Similarly, each arrow $R \Rightarrow S$ means to run Algorithm 7 with input (Q, S, T) .

In Appendix A.1.2 we discuss some implementation details. We have not yet compiled a complete verifier program for Theorem 5.2 since we wish to run the code for longer and improve Theorem 5.2 further.

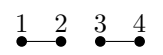
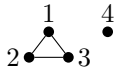
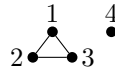
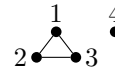
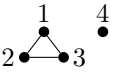
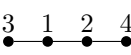
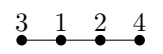
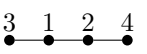
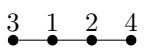
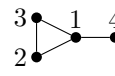
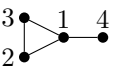
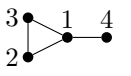
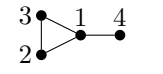
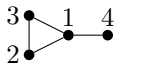
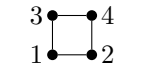
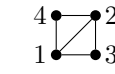
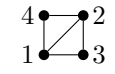
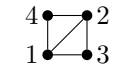
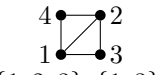
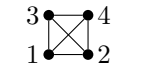
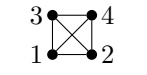
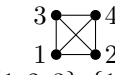
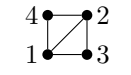
 $\{1\}$	 $\{1, 2\}$	 $\{1, 3\}, \{3\}$	 $\{2\}$	 $\emptyset, \{1\}$	 $\emptyset, \{1, 2\}$
 $\{1, 2\}, \{1, 2, 4\}$	 $\{1, 2, 3\}, \{1, 2, 4\}$	 $\{1, 2, 3\}, \{1, 4\}, \{2, 4\}, \{3, 4\}, \{4\}$	 $\{1, 2, 4\}, \{1, 4\}, \{1, 3, 4\}, \{2, 4\}, \{4\}$	 $\{1, 2, 4\}, \{1, 4\}, \{2, 4\}, \{3, 4\}, \{4\}$	 $\{3, 4\}$
 $\{2, 3\}, \{3\}$	 $\{1, 2, 3\}, \{2, 4\}$	 $\{1, 3\}, \{2, 4\}$	 $\{1, 3, 4\}, \{2, 3\}$	 $\{1, 3, 4\}, \{2, 4\}$	 $\{2, 3, 4\}, \{2, 3\}$
 $\{2, 3\}, \{2, 4\}$	 $\{2, 4\}, \{4\}$	 $\{2, 3\}, \{2, 4\}$	 $\{1, 4\}, \{4\}$	 $\{1, 3, 4\}, \{1, 3\}, \{4\}$	 $\{3, 4\}, \{3\}, \{4\}$
 $\{1, 2, 3\}, \{1, 3\}, \{1, 2, 4\}, \{1, 4\}, \{1, 3, 4\}, \{3, 4\}, \{2, 3, 4\}$	 $\{1, 4\}, \{1\}, \{4\}, \emptyset$	 $\{1, 3, 4\}, \{1, 3\}, \{4\}, \emptyset$	 $\{1, 2, 3\}, \{1, 2\}, \{1, 2, 4\}, \{1, 4\}, \{1, 3, 4\}, \{2, 4\}, \emptyset$	 $\{1, 2\}, \{1, 3\}, \{2, 3\}$	 $\{1, 2, 3\}, \{1, 3\}, \{1, 2, 4\}, \{1, 4\}, \{2, 3, 4\}, \{3, 4\}$

Figure 10: The sequence of joins proven in step 2(b). Each entry gives the pattern $H[k]$ and then, below that, the neighborhood of vertex $k + 1$ in each basic local condition of the join. For example, the first entry corresponds to the local condition $(\overline{P_3}, 2)$, which is also the third join in the second-to-last line in Figure 9. The joins should be read left-to-right, top-to-bottom. Unlike other figures, the actual graphs are drawn here, not their complements; this is because the graphs are so small there is not much benefit to drawing their complements.

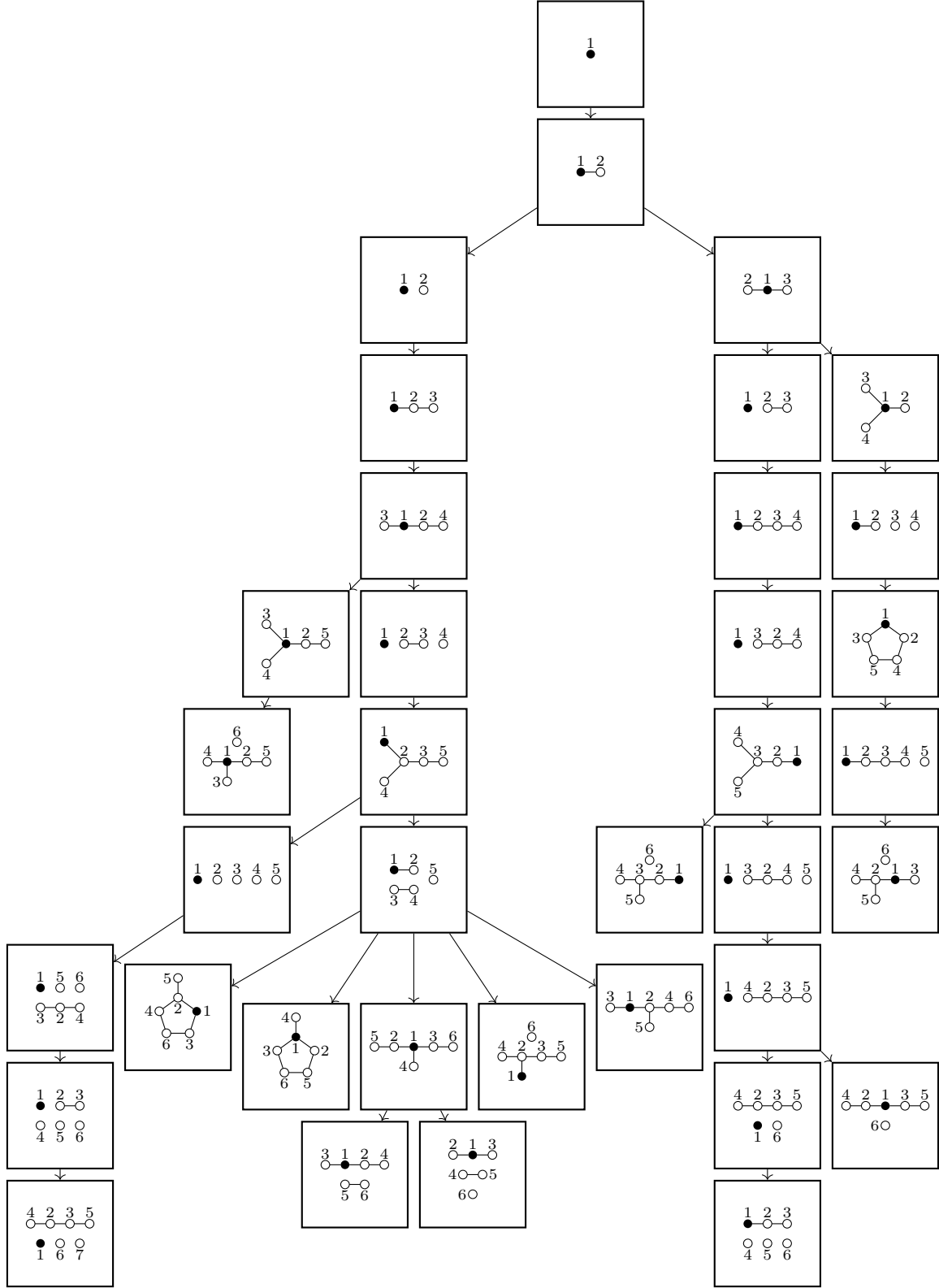


Figure 11: The logic for proving Theorem 5.2 using Algorithm 7 and the local condition Q obtained in steps 1 and 2. An arrow $R \rightarrow S$ means to run Algorithm 7 with input (Q, R, S) . The complements of the graphs are drawn. This is part 1. Part 2 is Figure 12.

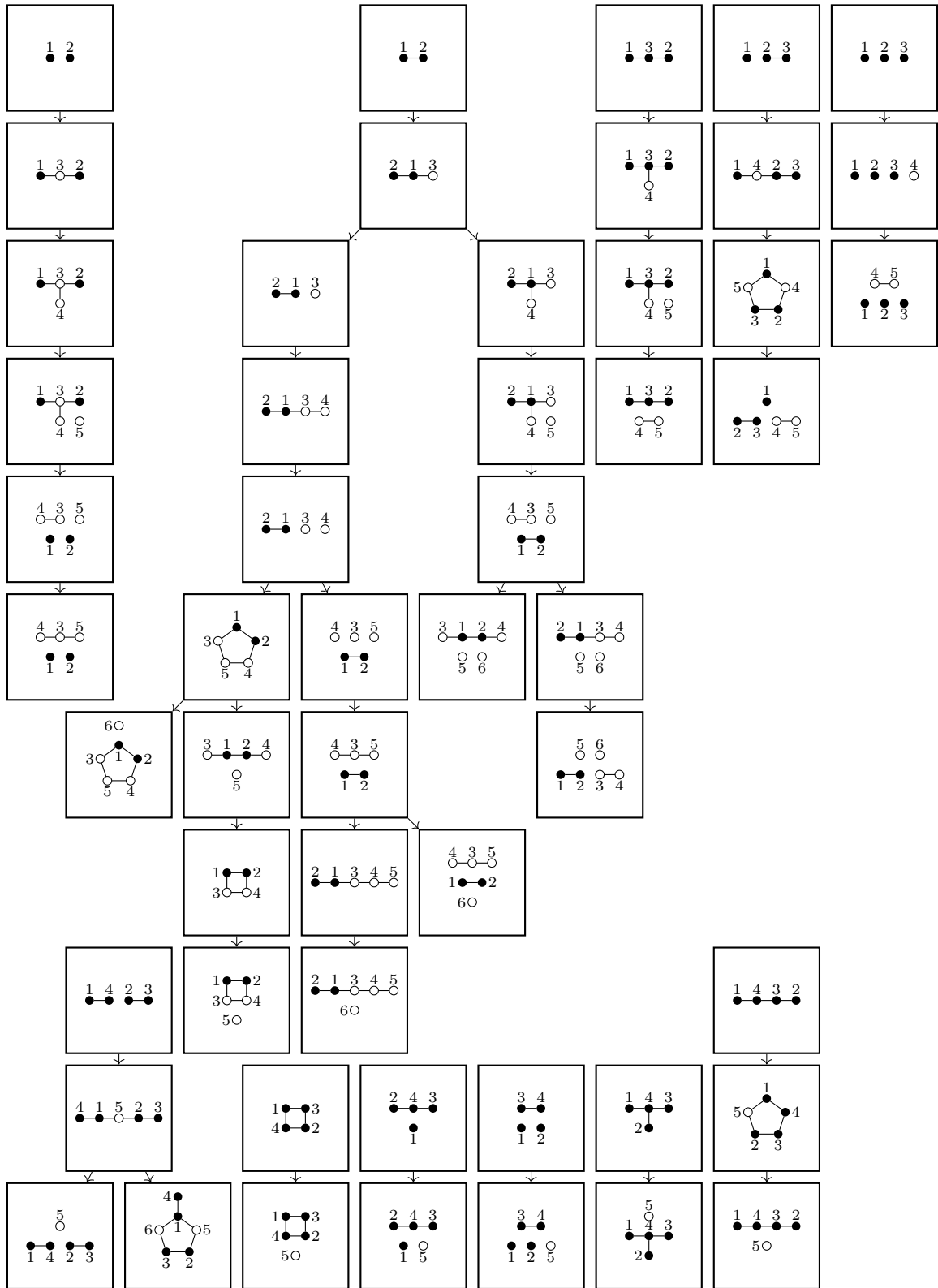


Figure 12: Part 2 of the logic for proving Theorem 5.2. Part 1 is Figure 11.

10 Discussion of the Main Theorem

The algorithms in this part are generalized versions of the algorithms in Part I. Despite this, we are unable to improve the main theorem of Part I; that is, we are unable to prove that counterexamples to $\text{HC-}\{\overline{K_3}\}$ have any specific induced subgraphs beyond those in Part I. The main issue is that although the algorithms in this part are theoretically stronger and have the potential to prove more results, in practice they are so much slower that they cannot improve Part I. One needs to look no further than the limiting weight for the results in this part compared to Part I: In this part, the limit was set to 10^5 , but in Part I, the limit was 2.6×10^7 , 260 times higher. The code for this part was run for a similar amount of time to Part I.

Looking at the specific local condition Q that formed the basis for the results in this part, the vast majority of joins and constituent basic local conditions have a pattern with four vertices. In a graph with n vertices, one should expect $\Theta(n^4)$ ordered subsets of vertices to match one of these patterns, compared to $\Theta(n^2)$ if you use only dominating edges as in Part I. Hence, the main algorithm in this part is simply fundamentally significantly slower than the main algorithm in Part I. It may be of interest in the future to take some fragment of Q that excludes most of the joins whose patterns have four vertices and see what local conditions it proves, and in particular to see what induced subgraphs it shows counterexamples to $\text{HC-}\{\overline{K_3}\}$ must contain.

We observe that as the size of the pattern increases, Q proves fewer and fewer local conditions with that pattern. This is no surprise, since if Q proves (H, k) it should prove $(H, k - 1)$ as well (assuming Q also proves $(H[k], 0)$, which is the case for every pattern we considered), but the converse doesn't hold.

11 Other Problems

11.1 Seymour's Strengthening

Similarly to $\text{HC-}\{\overline{K_3}\}$, there is a local condition that basically captures the property of being a counterexample to SSH. Let $\mathcal{S}$ be the set of graphs satisfying SSH. For $H \in \mathcal{S}$ let Y_H be the set of models of $K_{\lceil |V(H)|/2 \rceil}$ in H such that all classes have 1 or 2 vertices. Then the condition is

$$Q^{\text{SSH}} = \bigwedge_{H \in \mathcal{S}} \bigwedge_{M \in Y_H} \bigvee_{S \in M} \bigvee_{H' \in H_{M,S}} (H', V(H)).$$

As before:

Theorem 11.1. *If G is a graph that satisfies Q^{SSH} with no stable set of size 3 that is minimal among such graphs, then G is a minimal counterexample to SSH. The converse also holds.*

Proof. We omit the proof; it is essentially identical to Theorem 8.1. □

Additionally, Lemma 8.4 holds replacing everywhere “HC” with “SSH,” with identical proof.

It would be quite remarkable if all graphs that satisfied Q^{SSH} also satisfied Q^{HC} or had a stable set of size 3, since that would imply that SSH and HC- $\{\overline{K_3}\}$ are actually equivalent. Since Q^{HC} and Q^{SSH} are infinite local conditions, we cannot expect to be able to prove this algorithmically. However, we might be able to prove that some finite fragment of Q^{SSH} implies a finite fragment of Q^{HC} . Define Q_k^{SSH} similarly to Q_k^{HC} . Then we have the following extremely surprising result:

Proposition 11.2. $Q_k^{SSH} \vdash' Q_k^{HC}$ for all $k \leq 5$.

Proof. Run Algorithm 5' with the relevant input. □

It takes a bit of time to explain why this result is so surprising. In Q_4^{HC} , there is the join $(P_4 + v, 4) \vee (P_4 + v', 4)$ where $P_4 + v$ is the graph formed by adding one vertex to P_4 adjacent to vertices 3 and 4, and $P_4 + v'$ is the graph formed by adding one vertex adjacent to vertices 2, 3, and 4. This arose from the model $\{\{1\}, \{2, 3, 4\}\}$ of K_2 in P_4 . The interpretation is that if a graph G has an induced P_4 , say by vertices 1 through 4, such that every other vertex of G is adjacent vertex 1, then G is not a minimal counterexample to HC- $\{\overline{K_3}\}$. This is because a model M of $K_{\lceil |V(G)-4|/2 \rceil}$ in $G \setminus P_4$ immediately gives a model of $K_{\lceil |V(G)|/2 \rceil}$ in G , namely $M \cup \{\{1\}, \{2, 3, 4\}\}$.

The fact that $Q_4^{SSH} \vdash' Q_4^{HC}$ means that this G is also not a minimal counterexample to SSH. *But why is this?* It is not clear at all how one would extend a large enough clique minor in $G \setminus P_4$ to one in G if you are only allowed to use classes of 1 or 2 vertices, but Algorithm 5' shows that this is actually possible to do. In fact, the algorithm as we have implemented it considers over 800 intermediate graphs during its operation, so extending a clique minor of $G \setminus P_4$ to one of G is (probably) an extremely complicated procedure, with a huge amount of casework.

Based on this result, one might conjecture:

Conjecture 5. $Q_k^{SSH} \vdash' Q_k^{HC}$ for all k .

This would imply SSH and HC- $\{\overline{K_3}\}$ are equivalent, and not only that, but finding a $K_{n/2}$ minor with only classes of size 1 or 2 is in some sense not much more difficult than finding a $K_{n/2}$ minor using classes of any size. Unfortunately, some experiments using Algorithm 5' suggest this conjecture might be false for

$k = 6$. One of the hard cases is the join $(2K_3 + e + v_1, 6) \vee (2K_3 + e + v_2, 6) \vee \dots$, where $2K_3 + e$ is the graph formed by two triangles joined with one edge, and the $v_1, \dots$ are all the ways of attaching a vertex to $2K_3 + e$ such that the vertex is nonadjacent to vertex 1 or nonadjacent to vertex 2. This local condition comes from the model $\{\{1\}, \{2\}, \{3, 4, 5, 6\}\}$ of $2K_3 + e$. On the face of it, this shows some similarities to the hard case with $k = 4$ (that join can be written $(2K_2 + e + v_1, 4) \vee \dots$), but despite running the algorithm for much longer than the $k = 4$ case, we are still unable to prove this join from Q_6^{SSH} .

11.2 Parameterized Hadwiger's Conjecture

We can generalize further. For constant $c \leq 1/2$ and $m \in \mathbb{Z}_+ \cup \{\infty\}$, denote by $\text{PHC}(c, m)$ (standing for Parameterized Hadwiger's Conjecture with parameters c and m) the statement “All $\overline{K_3}$ -free graphs G have a set of $\lceil c|V(G)| \rceil$ pairwise adjacent vertex-disjoint connected subgraphs each of size at most m .” Note $\text{HC-}\{\overline{K_3}\}$ is equivalent to $\text{PHC}(1/2, \infty)$ and SSH is equivalent to $\text{PHC}(1/2, 2)$. Also, the statement gets stronger with increasing c and weaker with increasing m . Like SSH and $\text{HC-}\{\overline{K_3}\}$, there is also a local condition that captures the property of being a minimal counterexample to $\text{PHC}(c, m)$, namely

$$Q^{\text{PHC}(c, m)} = \bigwedge_{H \in \mathcal{P}_{c, m}} \bigwedge_{M \in X_H^{c, m}} \bigvee_{S \in M} \bigvee_{H' \in H_{M, S}} (H', V(H))$$

where $\mathcal{P}_{c, m}$ is the set of graphs satisfying $\text{PHC}(c, m)$ and $X_H^{c, m}$ is the set of models M of $K_{\lceil c|V(H)| \rceil}$ in H with every $S \in M$ having $|S| \leq m$.

Theorem 11.3. *If G is a graph that satisfies $Q^{\text{PHC}(c, m)}$ with no stable set of size 3 that is minimal among such graphs, then G is a minimal counterexample to $\text{PHC}(c, m)$. The converse also holds.*

Proof. As with Theorem 11.1, the proof is essentially identical to Theorem 8.1, so we omit the details. $\square$

Thus for any c, m , we can investigate $\text{PHC}(c, m)$ using local conditions almost just as well as $\text{HC-}\{\overline{K_3}\}$. We say “almost” since the result about $Q^{4\text{CC}}$ was only valid for $\text{HC-}\{\overline{K_3}\}$. There is actually a generalization of Lemma 1.4 that lets us get a result similar to the one about $Q^{4\text{CC}}$ for different c , namely:

Theorem 11.4. *Suppose G is a graph with no stable set of size three and $c \leq 1/2$. Let $t = \lceil c|V(G)| \rceil$ and further assume G is t -connected. If $\omega(G) \geq \frac{3t - |V(G)|}{2}$, then $h(G) \geq t$; in fact, G has a model of K_t using only sets of size at most 3.*

To prove this we rely on the following result from [CS12]:

Lemma 11.5. *Let G be a graph with no stable set of size 3 and k an integer such that it is not the case that $G \approx W_5$ and $k = 2$. Then G has k vertex-disjoint induced P_3 's if and only if:*

1. G has at least $3k$ vertices,
2. G is k -connected,
3. every clique of G has capacity at least k , and
4. $\overline{G}$ has a matching of size k .

Here, the *capacity* of a clique C in G is equal to $|D| + |A \cup B|/2$ where $A, B, D \subseteq V(G) \setminus C$ are the subsets of vertices that are complete to C , anticomplete to C , and neither complete nor anticomplete to C , respectively.

Proof (Theorem 11.4). The proof is essentially an straightforward generalization of the proof of Lemma 1.4 in [CS12]. Let $n = |V(G)|$ for brevity. If $\omega(G) \geq t$ then the result is trivial. Let Z be a clique of maximum size in G ; we may assume $|Z| < t$. Set $H = G \setminus Z$ and $k = t - |Z|$. We show that there are k vertex-disjoint induced P_3 's in H using Lemma 11.5.

Note $|V(H)| = n - |Z| \geq 3k$ by assumption, verifying (1) from the lemma. Since G is t -connected, H is k -connected, verifying (2).

Suppose for contradiction C was a clique of H that had capacity less than k , and let A, B, D be the vertices of $H \setminus C$ that are complete, anticomplete, and neither complete nor anticomplete to C , respectively. Then $|A| + |B| + 2|D| \leq 2k - 1$, so

$$\begin{aligned}
t/c - 1 + |D| &\leq |V(G)| + |D| \\
&= |A| + |B| + |C| + 2|D| + |Z| \\
&\leq 2k - 1 + 2|Z| \\
&= 2t - 1
\end{aligned}$$

since $c \leq 1/2$. This means all of the inequalities in the lines above are actually equalities, so $|D| = 0$ and $|C| = |Z|$. Thus C is a maximum clique, so $|A| = 0$ as well, so $V(H) = C \cup B$. But since H is connected, we must have $|B| = 0$ as well, and $V(H) = C$. But this implies $n = |C| + |Z| = 2|Z|$, contradicting that $|Z| < t$. This verifies (3).

Suppose for contradiction that $\overline{H}$ had no matching of size k . Let M be a maximal matching of $\overline{H}$; there are at least $|V(H)| - 2(k - 1)$ vertices in $\overline{H}$ not incident to any edge of M . Since M was maximal, these vertices are a stable set of $\overline{H}$, so a clique in H . Thus $|V(H)| - 2(k - 1) \leq |Z|$, implying $n \leq 2(t - 1) \leq 2cn$, a contradiction since $c \leq 1/2$. This verifies (4).

This proves the theorem assuming H is not isomorphic to W_5 (or $k \neq 2$), since a model of K_t in G is given by the k induced P_3 's plus the individual vertices of Z . If $H \approx W_5$, then G has at most 9 vertices since otherwise the individual vertices of Z plus one induced P_3 in H gives a K_t minor. One can then check by brute force that every graph with no stable set of size 3 and at most 9 vertices satisfies $\text{PHC}(1/2, 3)$; there are just under 2500 such graphs. Alternatively, one can repeat the argument at the end of the proof of Lemma 1.4 in [CS12] verbatim to dispose of this case. $\square$

Corollary 11.6. *If G is a graph with no stable set of size 3, $c \leq 1/2$, and $\omega(G) \geq c'(|V(G)| + c'')$, where $c' = \frac{3c-1}{2}$ and $c'' = 2$ if $c = 1/2$, $c'' = 0$ if $c \leq 1/3$, and $c'' = \lceil 1/c' \rceil$ otherwise, then G satisfies $\text{PHC}(c, 3)$.*

Proof. The case $c = 1/2$ follows from Lemma 1.4. It is easily seen that every graph satisfies $\text{PHC}(1/3, 3)$ by taking a maximal set of vertex-disjoint induced P_3 's and the largest clique in the graph remaining from the removal of these P_3 's (the remaining graph is the union of two cliques); this argument is found in [PST03]. This gives the $c \leq 1/3$ case. The remaining case follows from Theorem 11.4 with some careful analysis of the rounding on t caused by the ceiling function. $\square$

This allows one to generalize $Q^{4\text{CC}}$. In particular, say $S' \subseteq S$ is (p/q) -valid if $G_{S'}$ admits p cliques $C_1, \dots, C_p$ such that every vertex of $V(G_{S'})$ is in at least q of the C_i and

$$\sum_{i=1}^4 |C_i \cap V(G)| \geq |V(G)| + c''$$

where c'' is given in the statement of Corollary 11.6. Define

$$Q^{\text{CC}(p/q)} = \bigwedge_{\substack{\overline{K_3}\text{-free} \\ G}} \bigwedge_{(p/q)\text{-valid } S'} \left((G \vee K_1, |V(G)|) \vee \bigvee_{\substack{G' = G + v \\ N_{G'}(v) \in S_G \setminus S'}} (G', |V(G)|) \right).$$

We have:

Theorem 11.7. *If H is a counterexample to $\text{PHC}(\frac{2p/q+1}{3}, 3)$, then H satisfies $Q^{\text{CC}(p/q)}$.*

Proof. The proof is the same as Theorem 8.3, just replacing Lemma 1.4 with Corollary 11.6; we omit the details. $\square$

The local condition $Q^{\text{CC}(p/q)}$ is nearly a “fractional” version of $Q^{4\text{CC}}$. If you could take $c'' = 0$ and Corollary 11.6 still held, then finding a set of p cliques that cover $V(G_{S'})$ q times is just the same as proving that the fractional clique cover number $\chi^*(\overline{G_{S'}})$ is at most p/q .

We remark that Theorem 5.2 is valid not just for $\text{HC}-\{\overline{K_3}\}$, but also for the slightly stronger statement $\text{PHC}(1/2, 3)$, since the starting point $Q_4^{\text{HC}} \wedge Q_4^{4\text{CC}}$ is actually the same as $Q_4^{\text{PHC}(1/2, 3)} \wedge Q_4^{\text{CC}(4/1)}$ with the obvious definition of these local conditions. Also, we remark that $Q_3^{\text{PHC}(1/3, 3)} \vdash \overline{K_3}$ due to containing the basic local condition $(P_3 \sqcup K_1, 3)$ (alone in its join); this gives a proof that $\text{PHC}(1/3, 3)$ holds, although this was already essentially known in [PST03]. In fact, in [PST03] it was proved that every graph G with no stable set of size 3 has a $K_{(|V(G)|+\omega(G))/3}$ minor; Corollary 11.6 improves this to almost $K_{(|V(G)|+2\omega(G))/3}$ (just a clique minor slightly smaller than this due to the constant c'').

At present, we leave the investigation of $\text{PHC}(c, m)$ for other parameters than $1/2, 3$ relatively unexplored. One can repeat exactly the same strategy to prove that minimal counterexamples to $\text{PHC}(c, 3)$ satisfy some certain local condition, like Theorem 1.6, for any fixed $c \in (1/3, 1/2]$.

12 Miscellaneous Results

12.1 Computational Complexity

Given a graph G and local condition $Q = \bigwedge_i \bigvee_j (H_{i,j}, k_i)$, the LOCAL CONDITION decision problem is to determine if G satisfies Q . Here, for simplicity, Q is assumed to be a superfinite local condition so there is an obvious (brute force) algorithm for checking if G satisfies Q . In general one might consider non-superfinite local conditions or even infinite local conditions as long as there is a convenient algorithm or oracle for the necessary induced subgraph checks in the case that there are infinitely many possible induced subgraphs to check.

Let INDUCED SUBGRAPH be the decision problem of determining if a given graph G contains another given graph H as an induced subgraph. This is well-known to be NP-complete, and LOCAL CONDITION contains this problem as a special case, so LOCAL CONDITION is certainly NP-hard. We can actually place LOCAL CONDITION in a precise location in the polynomial hierarchy:

Theorem 12.1. *LOCAL CONDITION is Π_2^P -complete.*

Proof. LOCAL CONDITION is Π_2^P : Suppose we have access to an NP oracle. Then negative instances of LOCAL CONDITION can be verified in polynomial time given a witness, that witness being an index i and ordered subset $A \subseteq V(G)$ such that $G[A] = H_{i,1}[k_i]$ but there is no (ordered) $B \subseteq V(G)$ containing A with $G[B] = H_{i,j}$ for any j .

The subproblem of checking, for a given G, Q, A , if there is such a B is clearly NP-complete: if there is a B , the witness is B itself and it is easily verified in polynomial time, so it is NP, and if contains INDUCED SUBGRAPH as the case $A = \emptyset$, so it is NP-hard. Therefore if we are given an NP oracle, we can verify

negative instances of LOCAL CONDITION in polynomial time, by asking the oracle if there is such a B for the given witness A . Thus LOCAL CONDITION is in $\text{co-NP}^{\text{NP}} = \Pi_2^{\text{P}}$.

LOCAL CONDITION is Π_2^{P} -hard: The MINMAX CLIQUE problem, proven to be Π_2^{P} -complete in [KL95], is as follows. Given an integer k , graph G , and partition of $V(G)$ into sets $V_{i,j}$, $1 \leq i \leq I$ and $1 \leq j \leq J$, is there a clique of size k in $G[\bigcup_i V_{i,f(i)}]$ for all $f : [I] \rightarrow [J]$? We reduce this problem to LOCAL CONDITION to prove that LOCAL CONDITION is Π_2^{P} -hard.

Construct a graph G' by, for each i , adding to G some J disjoint copies of $C_{|V(G)|+i+1}$, with edges between one chosen vertex of the j th copy of $C_{|V(G)|+i+1}$ and all the vertices in $V_{i,j}$, and no other edges added. Then $G, \{V_{i,j}\}, k$ is accepted by MINMAX CLIQUE if and only if G' satisfies the basic local condition

$$\left(\left(\bigcup_{i=1}^I C_{|V(G)|+i+1} \right) \cup K_k, I(|V(G)|+1) + \frac{I(I+1)}{2} \right)$$

where the vertices are labeled in the order they appear in the graph as written above (so $C_{|V(G)|+2}$ gets labels 1 through $|V(G)|+1$, and so on, and the K_k gets labels $I(|V(G)|+1) + \frac{I(I+1)}{2} + 1$ through $I(|V(G)|+1) + \frac{I(I+1)}{2} + k$), and all possible edges from the first vertex in each $C_{|V(G)|+i+1}$ to the K_k are present. It is simple to see the equivalence, since the only induced $C_{|V(G)|+i+1}$ in G' are the ones that were added when forming G' from G , so if G' does not satisfy this local condition, one can find, for each i , which of the J copies of $C_{|V(G)|+i+1}$ demonstrates that G' does not satisfy the local condition, and the corresponding function $f : [I] \rightarrow [J]$ shows that $G[\bigcup_i V_{i,f(i)}]$ does not have a clique of size k . The same holds in reverse: if $G[\bigcup_i V_{i,f(i)}]$ does not have a clique of size k , then this f tells you which copy of each $C_{|V(G)|+i+1}$ can be used to show that G' does not satisfy the local condition. Thus LOCAL CONDITION is Π_2^{P} -hard. $\square$

It is clear from the proof above that one can replace “local condition” with “basic local condition” and everything still holds. Thus the problem BASIC LOCAL CONDITION (with the obvious meaning) is also Π_2^{P} -complete.

12.2 Extremal Properties

A natural question to ask is if G satisfies a given local condition, what does this imply about the extremal properties of G ; for example, how few vertices can G have, how many edges can G have, or more generally how many copies of H appear as induced subgraphs of G ? We have the following obvious result. Denote by $f(G, H)$ the number of times H appears as an ordered induced subgraph in G , i.e. the number of subsets $A \subseteq V(G)$ and $\phi : A \rightarrow V(H)$ that are isomorphisms between $G[A]$ and H . Note that this is exactly equal to $|\text{Aut}(H)|$ times the number of induced subgraphs of G isomorphic to H . Then:

Lemma 12.2. *If G satisfies basic local condition (H, k) , $f(G, H) \geq cf(G, H[k])$ for some c depending on (H, k) .*

Proof. Each induced copy of H in G can only make a constant C number of *ordered* copies of $H[k]$ in G “satisfied” in the obvious sense of the word. The constant C here is equal to the number of equivalence classes of automorphisms of H where two automorphisms π, τ are equivalent if $\pi(i) = \tau(i)$ for all $i \in [k]$. Hence there must be at least $1/C$ times as many induced H ’s as *ordered* induced $H[k]$ ’s in G , so

$$f(G, H) \geq f(G, H[k]) \cdot |\text{Aut}(H)|/C$$

as desired. □

Here is a restatement of the previous proof using a form of discharging:

Proof. For each $A \subseteq V(G)$ with $G[A]$ isomorphic to H , give a charge of C to that A , where C is defined the same as the previous proof. For each equivalence class $[\pi] \in \text{Aut}(A)/\sim$ as defined in the previous proof, A then gives charge 1 to the subset $\pi([k]) \subseteq V(G)$. Since G satisfies the local condition (H, k) , every B with $B \subseteq V(G)$ ends with charge at least $|\text{Aut}(H[k])|$, so

$$\text{total charge} = C \frac{f(G, H)}{|\text{Aut}(H)|} \geq f(G, H[k])$$

as before. □

In some cases, one can use a form of discharging closer to “typical” discharging proofs, where the vertices have charge rather than subsets of vertices. For instance, for the local condition $(W_5, 1)$ where vertex 1 is not the hub of the wheel, put a charge of x on each vertex v where x is 5 times the number of induced subgraphs of G isomorphic to W_5 for which v is the hub of the wheel. Then for each induced W_5 , each hub gives charge 1 to each vertex of the cycle; all vertices end with charge at least 1, so the number of induced W_5 ’s is at least $|V(G)|/5$. This proves Corollary 5.3 from the introduction; note every vertex has a nonneighbor and every nonedge is a nonedge in a W_5 by Theorem 5.2, so minimal counterexamples satisfy $(W_5, 1)$. Note that this matches the bound in both proofs since $C = 5$ here. We suspect that more sophisticated discharging arguments might be able to prove more results about graphs satisfying local conditions, but we leave this idea relatively unexplored.

Local conditions as we have defined them are very closely related to *flag algebras* introduced by Razborov [Raz07]. In particular, a graph satisfying a given basic local condition can be stated as the condition that

a certain flag or combination of flags has positive density everywhere in G , roughly speaking. The join operation $\vee$ corresponds roughly to addition of flags and the meet operation $\wedge$ corresponds roughly to multiplication of flags. One would hope that proving that a graph satisfies a certain local condition can be used to bound the number of a particular substructure appearing in G by converting to flag algebras, but this seems difficult to realize. A major barrier to doing this is that having positive density of a flag does not (necessarily) imply a positive value in a limit functional, so the local condition tells you essentially nothing in the limit in general. This seems to lock off a large portion of the theory of flag algebras from being useful in our context.

Still, we can give a third proof of the previous proposition using the language of flag algebras, giving the same constant again:

Proof. Recall the identity (from [dCSdOFS16])

$$\mathbb{E}[p(F; (G, \theta))] = \frac{q_\sigma(F)p(\downarrow F; G)}{q_\sigma(\sigma)p(\downarrow \sigma; G)}$$

where F is a σ -flag with $|\sigma| = k$, p is the density of flags, $q_\sigma(F)$ is the probability that a random injective map $\theta : [k] \rightarrow V(F)$ has image in F isomorphic to σ (in the sense of flags, so the image must both be isomorphic to σ in a graph-theoretic sense and “appear the same” as σ relative to the rest of F), $\downarrow$ is the downward operator sending σ -flags to $\emptyset$ -flags in the natural way, and the expectation is over a uniform embedding $\theta : [k] \rightarrow V(G)$ of σ into G . Notice that p is just a probabilistic version of f , so we can convert between them easily. Define $\sigma = H[k]$ and σ -flag $F = (H, \text{id}_{[k]})$ in the obvious way. We have

$$\begin{aligned} f(G, H) &= \left(\frac{|V(G)|}{|V(H)|} \right) |\text{Aut}(H)| p(\downarrow F; G) \\ &= \left(\frac{|V(G)|}{|V(H)|} \right) |\text{Aut}(H)| \frac{q_\sigma(\sigma)p(\downarrow \sigma; G) \mathbb{E}[p(F; (G, \theta))]}{q_\sigma(F)} \\ &= \frac{\left(\frac{|V(G)|}{|V(H)|} \right) |\text{Aut}(H)|}{\left(\frac{|V(G)|}{|V(H)|} \right) |\text{Aut}(H[k])|} \frac{q_\sigma(\sigma)}{q_\sigma(F)} f(G, H[k]) \cdot \mathbb{E}[p(F; (G, \theta))] \\ &\geq \frac{\left(\frac{|V(G)|}{|V(H)|} \right) |\text{Aut}(H)|}{\left(\frac{|V(G)|}{|V(H)|} \right) |\text{Aut}(H[k])|} \frac{q_\sigma(\sigma)}{q_\sigma(F)} f(G, H[k]) \cdot \frac{1}{\binom{|V(G)|-k}{|V(H)|-k}} \\ &= \frac{|\text{Aut}(H)|}{|\text{Aut}(H[k])| \binom{|H|}{k}} \frac{q_\sigma(\sigma)}{q_\sigma(F)} f(G, H[k]) \\ &= \frac{|\text{Aut}(H)|}{C} f(G, H[k]) \end{aligned}$$

as desired, where C has the same definition as in the previous two proofs; one can check $q_\sigma(F) = \frac{(|V(H)|-k)!}{|V(H)|!} C$ and $q_\sigma(\sigma) = \frac{|\text{Aut}(H[k])|}{k!}$. $\square$

13 Final Remarks

Suppose there existed an algorithm that is “complete for finite graphs,” i.e. if it is the case that all finite graphs that satisfy Q satisfy R , then the algorithm will prove that. Such an algorithm would be extremely powerful for resolving graph theory problems. For example, such an algorithm would immediately resolve Cherlin’s question, assuming the answer is that there is no finite k -ECTF graph for sufficiently large k . It would also likely resolve $\text{HC-}\{\overline{K_3}\}$ or maybe even SSH, for example by using some fragment of Q^{HC} or Q^{SSH} .

In fact, it would also probably give a new proof of the four-color theorem (4CT). This is because:

Theorem 13.1. *[AH77, RSST97] Any finite planar triangulation contains one of finitely many reducible configurations.*

The dual version is:

Theorem 13.2. *Any finite graph with minimum degree 3 either contains a K_5 minor, a $K_{3,3}$ minor, a vertex with degree 4, or one of finitely many dual reducible configurations.*

Here, the “dual reducible configurations” may be defined in terms of reducible configurations in an obvious way; we omit the details. Note that having minimum degree 3 is a superfinite local condition, and the other conditions in the theorem constitute a finite local condition; it is not superfinite, but importantly it has an algorithm to test for it. Thus a simple modification of an algorithm that is complete for finite graphs would prove the theorem above and therefore prove 4CT aside from showing the configurations are actually reducible.

It seems plausible there would be some modification to the above that would allow an algorithm that is complete for finite graphs to prove Hadwiger’s Conjecture with $t = 4$ directly (note that this is equivalent to 4CT by Wagner’s Theorem), and indeed it seems plausible that there would be some modification that proves Hadwiger’s Conjecture for any finite t if you only run the algorithm for long enough.

Bibliography

- [AH77] K. Appel and W. Haken. Every planar map is four colorable. *Illinois Journal of Mathematics*, 21(3):429–567, 1977.
- [BGSP12] Gunnar Brinkmann, Jan Goedgebeur, and Jan-Christoph Schlage-Puchta. Ramsey numbers $R(K_3, G)$ for graphs of order 10. *The Electronic Journal of Combinatorics*, 19(4), 2012.
- [Bla07] Jonah Blasiak. A special case of Hadwiger’s conjecture. *Journal of Combinatorial Theory, Series B*, 97(6):1056–1073, 2007.
- [Bos19] Christian Bosse. A note on Hadwiger’s conjecture for W_5 -free graphs with independence number two. *Discrete Mathematics*, 342, 2019.
- [Che92] Gregory L. Cherlin. Combinatorial problems connected with finite homogeneity. In *Contemporary Mathematics*, volume 131, pages 3–30, 1992.
- [CN06] Gabor Csardi and Tamas Nepusz. The igraph software package for complex network research. *InterJournal, Complex Systems*:1695, 2006.
- [CS12] Maria Chudnovsky and Paul Seymour. Packing seagulls. *Combinatorica*, 32:251–282, 2012.
- [dCSdOFS16] Marcel de Carli Silva, Fernando de Oliveira Filho, and Cristiane Sato. Flag algebras: a first glance. *Nieuw Archief voor Wiskunde*, 5/17(3):193–199, 2016.
- [EKR76] P. Erdős, D. J. Kleitman, and B. L. Rothschild. Asymptotic enumeration of K_n -free graphs. In *Colloquio Internazionale sulle Teorie Combinatorie*, volume 2, pages 19–27, 1976.
- [HSS08] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring Network Structure, Dynamics, and Function using NetworkX. In Gaël Varoquaux, Travis Vaught, and Jarrod Millman, editors, *Proceedings of the 7th Python in Science Conference*, pages 11 – 15, Pasadena, CA USA, 2008.
- [IMM18] Alexey Ignatiev, Antonio Morgado, and Joao Marques-Silva. PySAT: A Python toolkit for prototyping with SAT oracles. In *SAT*, pages 428–437, 2018.
- [JR95] Tommy Jensen and Gordon Royle. Small graphs with chromatic number 5: A computer search. *Journal of Graph Theory*, 19(1):107–116, 1995.

- [KL95] Ker-I Ko and Chih-Long Lin. On the complexity of min-max optimization problems and their approximation. In *Minimax and Applications*, volume 4, pages 219–239, 1995.
- [Kri10] Matthias Kriesell. On Seymour’s strengthening of Hadwiger’s conjecture for graphs with certain forbidden subgraphs. *Discrete Mathematics*, 310:2714–2724, 2010.
- [LS20] Shoham Letzter and Julian Sahasrabudhe. On existentially complete triangle-free graphs. *Israel Journal of Mathematics*, 236:591–601, 2020.
- [McK] Brendan McKay. Combinatorial data. <http://users.cecs.anu.edu.au/~bdm/data/ramsey.html>.
- [PST03] Michael D. Plummer, Michael Stiebitz, and Bjarne Toft. On a special case of Hadwiger’s conjecture. *Discussiones Mathematicae Graph Theory*, 23:333–363, 2003.
- [Ran04] Bert Randerath. 3-Colorability and forbidden subgraphs. I: Characterizing pairs. *Discrete Mathematics*, 276(1):313–325, 2004. 6th International Conference on Graph Theory.
- [Raz07] Alexander A. Razborov. Flag algebras. *The Journal of Symbolic Logic*, 72(4):1239–1282, 2007.
- [RSST97] Neil Robertson, Daniel Sanders, Paul Seymour, and Robin Thomas. The four-colour theorem. *Journal of Combinatorial Theory, Series B*, 70(1):2–44, 1997.
- [Sey16] Paul Seymour. Hadwiger’s conjecture. In *Open Problems in Mathematics*, pages 417–437, 2016.
- [SWY21] Joshua Schroeder, Zhiyu Wang, and Xingxing Yu. On the 3-colorability of triangle-free and fork-free graphs, 2021.

Appendix

A.1 Further Implementation Details and Code

A.1.1 In Part I

Graph computations were done using the igraph Python package ([CN06]). In many places one must check if a graph has a particular induced subgraph (either $\overline{K_3}$ for the independence number 3 checks, or some given graph H for the the property P checks whenever P is the property of containing an induced H). This problem is NP-complete, but it can be solved quickly enough for small graphs such as ours. In our case, we used igraph’s implementation of the LAD algorithm.

To perform the check for a four-clique cover of G at the end of Algorithm 2 (and in Algorithm 7), we model it as a SAT instance and use a SAT solver. Specifically, there are four variables v_1, v_2, v_3, v_4 for each vertex v in G corresponding to the four cliques that v can be included in. The boolean formula is the conjunction of the following clauses:

- $(v_1 \vee v_2 \vee v_3 \vee v_4)$ for each $v \in V(G)$, ensuring that Q_1, Q_2, Q_3, Q_4 cover G ;
- $(\neg v_i \vee \neg u_i)$ for each $uv \in E(G)$ and $i = 1, 2, 3, 4$, ensuring that Q_1, Q_2, Q_3, Q_4 are cliques;
- $(\sum_{i=1}^4 \sum_{v \in V(H)} v_i \geq |H| + 2)$, ensuring the additional condition $\sum_{i=1}^4 |Q_i \cap V(H)| \geq |H| + 2$.

We used PySAT ([IMM18]) to solve these SAT instances.

Also, often one needs to generate the set of graphs formed by adding one vertex v to a graph H with some restrictions on the result. Instead of trying to add v to H , we instead start with v and then add the vertices of H one at a time, at each step checking the imposed restrictions. This allows one to “bail out early,” so to speak, if a potential attachment does not satisfy the restrictions. Some care must be taken afterwards to fix the vertex labelings so that H gets labels 1 through $|V(H)|$ and v gets label $|V(H)| + 1$. This modification is only done in Part I, but something similar is done for Part II, explained later.

Finally, much computation can be reused when running Algorithm 3 with the same input aside from k . In fact, the sets A , B , and C are the same regardless of k , so we temporarily cache them whenever Algorithm 4 reaches the part where Algorithm 3 should be run.

We provide code at <https://github.com/dcartermath/hc-forbidden-subgraphs> which can be used to verify Theorem 1.6. We used Python version 3.9.6, igraph version 0.9.11, and PySAT version 0.1.7.dev19, though the code most likely works fine with newer versions. The program reads the TeX source of Figures 3, 4,

and 5, respectively copied into files `hp_figure`, `i_figure`, and `edges_figure` put in the same directory as this program, to determine the graphs, labelings, and steps to verify Theorem 1.6, thus making absolutely sure that the data provided in the main body of Part I is correct. Each line of output is one arrow of Figure 5 consisting of the name of the input graph, the name of the output graph, the total number of graphs ever added to the set A , and the weight.

In the process of discovering Theorem 1.6, we did not immediately use the optimizations discussed above. Initially, we simply brute-force checked for four-clique covers with the desired properties, and computed the set of graphs formed by adding a single vertex to a given graph H by simply trying all $2^{|V(H)|}$ possibilities for the neighbor set of the new vertex. Whenever we made an optimization, we verified the newer version gave an identical state to the older version run for one hour. Thus, we are confident in the correctness of our implementation.

A.1.2 In Part II

Most of the comments about the implementation of the algorithms in Part I also holds for Part II. All discussion about four-clique covers is the same. In Part II, the basic notion of graph containment is not induced subgraphs, but the partial order $\preceq$ defined in Section 7; we need an algorithm to compute this order. Luckily, `igraph`'s LAD algorithm implementation has the optional `domains` parameter, which does exactly what we need. In particular, to check if $H \preceq_k G$, we just need to call `subisomorphic_lad` with

$$\text{domains} = [[0], [1], \dots, [k-1], \underbrace{[k, \dots, |V(G)|], \dots, [k, \dots, |V(G)|]}_{|V(H)|-k \text{ copies}]$$

noting that the vertices are 0-indexed.

In this part, we use a slightly different optimization method for attaching vertices than in Part I, which turned out to be necessary due to the fact that the $\preceq$ check is sensitive to vertex labels, as opposed to the subgraph isomorphism of Part I. The optimization for Part II is to keep track directly which “partial” neighborhoods are legal neighborhoods for attaching the new vertex. Here a partial neighborhood is an assignment of adjacency or nonadjacency of v to a subset of the rest of the graph H . We iteratively take all legal partial neighborhoods computed so far and, for each, seeing if it would be legal to make v adjacent, or nonadjacent, to the next vertex not included in the partial neighborhood. Say that the new larger partial neighborhood to test includes vertices $S \subseteq V(H)$. We check all of the relevant subgraph isomorphisms and $\preceq$ checks on $(H+v)[S]$; if any fail, then we discard that partial neighborhood from further consideration.

We have not yet finalized the verifier for Theorem 5.2. The author wishes to search for further optimizations and run the code for longer in the future, the version of Part II that will eventually be submitted for

publication will have a different main theorem and possibly a slightly different strategy.

A.2 More Data

Running the code in Appendix A.1.1, after some minutes or hours of computation, produces the following output:

```
K23c I4 2 544
K7 H2p 121 317568
W5 I3 5 832
H7 I15 2 640
K7 H3p 600 6222976
H7 I2 2 640
I2 I12 4 576
I15 I18 27 194688
I3 I14 1 64
H2p H1p 532 2454144
I4 H5p 26 423488
I18 I25 1 128
H5p I26 65 324736
I14 I21 2 384
I12 I1 7 5248
I1 I13 2 192
I25 I5 336 3440896
I26 H33p 348 20723968
I25 H24p 134 715008
I21 I6 20 42752
I25 I8 76 527616
I21 I11 33 102144
I21 H16p 139 768768
I21 I19 35 84736
I19 H21p 121 336128
I5 H13p 524 7547520
I13 I7 17 19584
```

I11 H20p 73 407168
H24p I22 522 3486976
H24p I16 95 604928
H24p I23 989 22616320
H24p H26p 136 611072
I6 I20 32 28800
I8 H12p 131 411776
I7 H18p 112 345216
I23 H32p 1 256
I20 H31p 45 1067776
I22 H29p 333 6226688
I16 I24 1 128
H31p H4p 338 5561856
I24 H25p 1348 15871232
H18p H14p 566 14731520
I24 H27p 94 621824
H18p I10 265 1639680
I10 H7p 256 1474432
I10 H17p 218 1423232
H27p I17 354 8121088
I10 H15p 265 11495296
I10 H8p 756 10430336
H27p H30p 327 9972480
H4p H10p 899 10312832
I10 H9p 305 1976192
H15p H23p 479 15760640
I17 H28p 1 128
H17p H11p 341 1031424
H7p I9 369 3561216
H23p H6p 723 13390592
I9 H19p 19 8320
H23p H22p 74 17303296

The four columns here separated by spaces are the in-vertex G and out-vertex H of an arrow of Figure 5, then the total number of graphs added to A when Algorithm 4 is run with input (G, H) , and finally the weight of the arrow, which is the sum over H' ever added to A of $2^{|V(H')|}$.

A.3 Proving HC- $\{\overline{K_3}, K_8\}$

As we mentioned in the introduction, Theorem 1.6 is no help for proving Theorem 1.7 because all 477142 $\{\overline{K_3}, K_8\}$ -free graph on 27 vertices contains all H'_i , and K_7 and H_7 . However, the vast majority of these graphs have a dominating edge, and it turns out that the ones that don't have a connected dominating matching with two edges. Specifically:

Proposition A.3. *Of the 477142 $\{\overline{K_3}, K_8\}$ -free graph on 27 vertices, 455344 have a dominating edge and the remaining 21798 have a connected dominating matching of two edges.* $\square$

Code to verify this is available at <https://github.com/dcartermath/hc-forbidden-subgraphs>. This code, `k8.py`, simply checks each graph in turn, using the file containing such graphs available at [McK] (though note that the relevant file contains all $\{K_3, \overline{K_8}\}$ -free graph on 27 vertices, so we must take the complement of them). The graph data is parsed using NetworkX version 2.8.3 ([HSS08]), but the graphs are immediately converted to `igraph`. The program outputs several lines of intermediate computation before eventually the line `477142 / 477142 [455344, 21798]`, indicating that all 477142 graphs were read, 455344 have a dominating edge, and the remaining 21798 have a connected dominating matching of two edges.

The check for connected dominating matchings of two edges is simply hard-coded in; a pair of edges uv, xy is a connected dominating matching if all of the following conditions are met:

- $\{u, v\}$ and $\{x, y\}$ are disjoint.
- $\{u, v\} \cap N(x) \neq \emptyset$ or $\{u, v\} \cap N(y) \neq \emptyset$.
- For each $z \in V(G) \setminus \{u, v, x, y\}$, $z \in N(u)$ or $z \in N(v)$, and $z \in N(x)$ or $z \in N(y)$.

Theorem 1.7 follows from Proposition A.3 due to Lemma 4.1 and the fact that HC- $\{\overline{K_3}, K_7\}$ holds. The latter fact implies (similarly to the reasoning in Corollary 1.8) that any counterexample to HC- $\{\overline{K_3}\}$ has at least 27 vertices. Thus the graphs considered in Proposition A.3 either satisfy Hadwiger's Conjecture or are minimal counterexamples.