Wesleyan University The Honors College

Forbidden Pairs Make Problems Hard

by

Eli Fox-Epstein
Class of 2011

A thesis submitted to the
faculty of Wesleyan University
in partial fulfillment of the requirements for the
Degree of Bachelor of Arts
with Departmental Honors in Computer Science

Middletown, Connecticut April, 2011

Acknowledgements

First and foremost, I thank my thesis advisor, Professor Danny Krizanc. As my thesis topic mutated from what it was to what it is, you were always supporting and encouraging my curiosity. Thanks for tirelessly proofreading and discussing my ideas. I also thank Professor Eric Aaron and Professor Jim Lipton for their advice over the past four years and their feedback on my thesis.

A big thanks to my roommates, friends, and neighbors for putting up with me. Thanks for providing distraction when needed, keeping it quiet, listening to me ramble, and being ridiculously supportive. Special thanks to Qianqian for great feedback, constant encouragement, and keeping me fed.

I thank my parents for unending support, encouragement, and inspiration for the past 21 years. Their wisdom, sensible advice, and high expectations made this thesis possible.

Thank you, dear reader, for taking interest.

Abstract

This thesis introduces a decision problem in graph theory that we call the forbidden pairs problem by generalizing from similar, more specific prior work. We focus on the following decision problems: given a graph property, a graph G, and a set F of pairs of vertices (or edges) from G, can we find a subgraph of G that has the property subject to the condition that no two of its vertices (or edges) form a pair in F?

Our main result proves that for many graph properties, such as cyclic and nonplanar, that are described by their members necessarily having specific minors, the corresponding forbidden pairs decision problems are **NP**-complete.

Additional results determine the computational complexity of more forbidden pairs problems. Future work could more completely classify further sets of graph properties or attempt to approximate solutions to the hard problems introduced here.

Contents

CHAPTER 1

Introduction

An important focus of theoretical computer science is computational efficiency. Problems are often categorized by the efficiency of the worst-case running times of algorithms and separated into two broad groups: those computationally feasible and those that are believed to be infeasible. This is formalized through the notion of **NP**-completeness, which categorizes a large class of problems that are believed, but not known, to take time that grows exponentially with the size of the input.

Graph theory is an area of mathematics and computer science that provides pure, abstract representations of countless real-world problems, such as network simulations and road maps. These lead to a plethora of computational tasks, many of which have been shown to be **NP**-complete.

Many proofs of the **NP**-completeness of problems use methods applicable only to the specific problem at hand and do not generalize well. One exception to this in graph theory is the work of Yannakakis and Lewis ([**21**], [**16**]) in which they prove that a large swath of graph theory problems are **NP**-complete. In a similar fashion, we investigate the complexity of many related problems at once.

This thesis is centered around graph problems that restrict particular choices of edges or vertices available to demonstrate that a subgraph of an input graph has a particular property. We study the computational complexity of these problems en masse.

1. Complexity Preliminaries

The area of computational complexity in computer science has largely focused on **NP**-completeness, a concept that was introduced in [**6**] and [**15**] (which is translated into English in [**20**]), and developed in [**11**]. The following is a summary of the concepts from computational complexity that we use. Further information can be found in complexity theory textbooks, such as [**17**] or [**10**].

The Turing Machine is the basis of measuring computational complexity. This abstract machine performs computations one step at a time on an infinite tape of memory. Turing machines capture the standard notion of computation, and are equally as powerful as many other bases for computation.

A *decision problem* is a 'yes-or-no' question, or more mathematically, a function from arbitrary inputs to $\top$ (true) or $\perp$ (false). One can classify decision problems into various complexity classes, which are groups of problems each decidable within the same time complexity.

The class **P**, or polynomial time, introduced in [**5**] and [**7**], is the set of decision problems that can be solved on a deterministic Turing Machine in a number of computational steps that grows polynomially with the size of the input.

We often reason about the complexity of problems through *reduction*. We show that through a polynomial-time transformation, we can convert between problems. For example, if we can convert in polynomial time a problem of known complexity into another of unknown complexity, solve the result, and then convert back to the original problem in polynomial time, then we can infer a lower bound on the complexity of the unknown problem.

The complexity class **NP** is the set of all decision problems that have proofs verifiable in polynomial time. This means that given evidence of an affirmative decision, known as a certificate, one can verify in time polynomial in the size of the input that the problem should answer ‘yes’.

For example, consider the decision problem for whether a natural number is composite. If one is given an integer strictly between one and the input as a certificate, one could ensure that the input is evenly divided by it in polynomial time.

This can be shown to be equivalent to those problems solvable in polynomial time by a nondeterministic Turing Machine. If a problem can be solved by a deterministic Turing machine, then a nondeterministic Turing machine can solve it, so $\mathbf{P} \subseteq \mathbf{NP}$.

Intuitively, a problem is **NP**-hard if it is at least as computationally difficult as any problem in **NP**. In other words, any problem in **NP**-hard can be reduced in polynomial time to any other problem that is **NP**-hard. Thus, the problems in **NP** are a subset of those that are **NP**-hard. A problem is **NP**-complete if it is in **NP** and **NP**-hard.

Reduction is used to prove that a problem X is **NP**-hard or **NP**-complete. We show that one can solve another problem known to be **NP**-hard with an algorithm that solves X, allowing a polynomial amount of work to be performed to convert between the problems. By transitivity, by showing that it is at least as hard as one problem in **NP**, this establishes that all problems in **NP** can be reduced to X. **NP**-completeness is proven by then demonstrating that a proof of X can be verified in polynomial time.

A fundamental, open question in computer science is whether or not $\mathbf{P} = \mathbf{NP}$. For conciseness, we say a problem is *hard* if it is **NP**-complete, and *easy* if it is in **P**, despite the possibility that $\mathbf{P} = \mathbf{NP}$.

One of the earliest and most important **NP**-complete problems is that of boolean satisfiability, SAT. k-SAT, proved to be **NP**-complete for all $k \geq 3$ in [**6**], is the decision property of the satisfiability of a boolean formula consisting of the conjunction of disjunctive clauses each with k literals. For example, instances of 3-SAT with n clauses have the form

$$(l_{1,1} \vee l_{1,2} \vee l_{1,3}) \wedge \ldots \wedge (l_{n,1} \vee l_{n,2} \vee l_{n,3})$$

where each literal $l_{i,j}$ is an element of some set of variables and their negations. An example instance of 3-SAT is $(a \vee b \vee \bar{c}) \wedge (\bar{a} \vee \bar{b} \vee c)$.

MONOTONE 1-IN-3 SAT (M1-3SAT) is a variation on 3-SAT that is particularly useful for reductions between graph decision problems. 1-IN-3 SAT differs from 3-SAT in that all solutions must have exactly one true value per disjunction. MONOTONE 1-IN-3 SAT is true exactly when an an instance of 1-IN-3 SAT is true and contains no negated variables as literals. The following folklore theorem will be useful in the following chapters:

THEOREM 1. MONOTONE 1-IN-3 SAT *is **NP**-complete.*

One proof of this theorem is provided in **Appendix A**.

2. Graph Preliminaries

For the most part, these preliminaries agree with those found in any standard text on graph theory, such as [**3**]. An undirected *graph* G is a tuple (V, E) of a set of vertices and a set of edges. We write V_G and E_G to be the vertices and edges of graph G, respectively. Edges are unordered pairs and a subset of $\{\{u, v\} \mid u, v \in V_G; u \neq v\}$. An edge $\{u, v\}$ is written more concisely as uv. The size of a graph, denoted $|G|$, is equal to $|V_G|$.

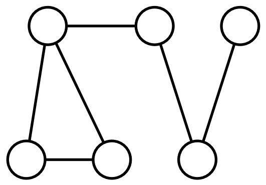

FIGURE 1. An example representation of a graph, circles denote vertices and lines show edges.

The *complement* of G, denoted $\overline{G}$, is the graph with vertices V_G and edges $\{uv \mid uv \notin E_G\}$.

Graph H is isomorphic to G if one can define a bijection f between the vertices of the two graphs such that $uv \in E_G$ if and only if $f(u)f(v) \in E_H$. Whenever graph equality is referred to, we are actually asking if the graphs are isomorphic.

Two vertices u and v in a graph are *adjacent* if the graph contains the edge uv. Two edges in a graph are *adjacent* if they share a vertex. A vertex is *incident* to each edge containing it. An *endpoint* of an edge is one of the vertices in the pair. The *degree* of a vertex is the number of edges incident to it. The maximum degree of a graph is the the maximum of the degrees of all vertices in the graph.

A *path* in a graph is a sequence of vertices such that each is adjacent to its successor, if present. Two vertices are *connected* if there is a path between them. A graph is connected if each pair of vertices is connected. A *cycle* is a path whose first vertex is its last. A *simple path* is a path with no duplicate vertices and a *simple cycle* is a cycle with no duplicate vertices.

H is a *subgraph* of G (denoted $H \subseteq G$) if $V_H \subseteq V_G$ and $E_H \subseteq E_G$. In this instance, G is a *supergraph* of H. H is an *induced subgraph* of G if $E_H = \{uv|u, v \in V_H, uv \in E_G\}$. Subgraphs and induced subgraphs are not considered to be strict; graphs are subgraphs of themselves.

A graph containing an edge between each pair of distinct vertices is *complete*. A complete graph with m vertices is denoted K_m. The complement of a complete graph is an *empty* or *null* graph, denoted $\overline{K_m}$ if it has m vertices. A complete subgraph is a *clique* and a null induced subgraph is an *independent set*.

The *components* of a graph is the set of largest possible subgraphs such the vertices in each are pairwise connected. A component is k*-connected* if one can delete any $k-1$ vertices without splitting the component into multiple components. A component is k*-edge-connected* if one can delete any $k-1$ edges without splitting the component into multiple components.

A *subdivision* is an operation on an edge uv that creates a new vertex w, deletes uv, and adds edges uw and wv. A graph H is a subdivision of G if one can subdivide an edge in G to produce H or another graph of which H is a subdivision. In other words, H is a subdivision of G if it is in the closure of the edge subdivision applied to G. The action of undoing a subdivision is *smoothing*.

An *edge contraction* is an operation on a graph that collapses two adjacent vertices. If $u \neq v$ are adjacent vertices, we can contract on uv by taking the subgraph induced by $V_G - \{u, v\}$, adding a new vertex w, and adding the edges $\{wx \mid vx \in E_G \vee ux \in E_G\}$.

G is a *minor* of graph H if one can perform some number of edge contractions on a subgraph of H to obtain G.

The *disjoint union* of a set S of graphs is a new graph defined by

$$\left(\bigcup_{s \in S} V_s, \bigcup_{s \in S} E_s \right).$$

3. Graph Properties

A *graph property* Π is a function from graphs to $\{\top, \bot\}$. When $\Pi(G) = \top$, we may write 'G satisfies Π', '$G \in \Pi$', '$\Pi(G)$ holds' or just '$\Pi(G)$'. We also refer to a property as the set of graphs $\{G \mid \Pi(G)\}$ when convenient. The *graph property decision problem* takes as input a graph and a property and asks if that graph satisfies the property. We refer to this problem when saying that a property can be *recognized* in a graph. The *subgraph property decision problem* asks if the input graph contains a subgraph with the property.

A property Π is *trivial* if finitely many graphs do or do not satisfy Π. The *complement* of a property Π, denoted $\overline{\Pi}$, is the set of all graphs not satisfying Π.

Let S be a possibly infinite set of graphs. From this, we can define 6 properties, each *characterized* by S. We say S is the set of *characteristic graphs* for each property.

A property is *edge-hereditary*, *vertex-hereditary*, or *minor-hereditary* if each graph in the property has no characteristic graph as a subgraph, induced subgraph, or minor, respectively.

A property is *edge-ancestral*, *vertex-ancestral*, or *minor-ancestral* if each graph in the property does have a characteristic graph as a subgraph, induced subgraph, or minor, respectively.

Note that the complement of a hereditary property is ancestral. To specify which properties we intend, we write *characteristic subgraphs*, *characteristic induced subgraphs*, or *characteristic minors.* Edge-hereditary properties are closed under subgraphs, vertex-hereditary properties are closed under induced subgraphs, and minor-hereditary properties are closed under minors.

If there are finitely many characteristic graphs for a property, it is *finitely characterized.* Otherwise, it is *infinitely characterized.*

4. Some Important Graph Properties

A graph is *bipartite* if its vertices can be partitioned into two independent sets such that each vertex is in one or the other. A bipartite graph is complete if it contains all edges possible without invalidating the bipartite property. We denote complete bipartite graphs as $K_{m,n}$ if one independent set has m vertices and the other has n. Bipartiteness is vertex-hereditary. A *star*, denoted S_k, is the complete bipartite graph $K_{1,k}$. S_3 is known as a *claw*.

A graph is *planar* if it can be embedded on the plane. It is well known that planar graphs are those containing no subdivisions of K_5 or $K_{3,3}$. Planarity is edge-hereditary.

A graph is *acyclic* if it does not have a cycle as a subgraph. This property is edge-hereditary.

A graph is *biconnected* if one can delete one vertex and its adjacent edges and the graph remains connected. Biconnected graphs are cyclic.

The subgraph recognition problem for each of the above properties can be decided in time polynomial in the size of the input graph. There are many properties for which deciding the subgraph recognition problem is known to be **NP**-complete, such as *Hamiltonian*, which is the set of graphs that contain a cycle including every vertex exactly once.

5. Forbidden Pairs Problems

Informally, we wish to see if a given graph has a subgraph that has a property and does not violate a number of constraints on its edges or vertices. For example, one might wish to see if there is a cyclic subgraph subject to the constraints on which vertices can be simultaneously in a cycle.

Let G be a graph, F any subset of $V_G \times V_G$, F' any subset of $E_G \times E_G$, and Π a graph property.

The *forbidden vertex pairs problem*, $\mathrm{FVP}_\Pi(G, F)$, is true when:

(1) There exists an $H \subseteq G$ with $\Pi(H)$.
(2) $V_H \times V_H \cap F = \emptyset$.

The *forbidden edge pairs problem*, $\mathrm{FEP}_\Pi(G, F')$, is true when:

(1) There exists an $H \subseteq G$ with $\Pi(H)$.
(2) $E_H \times E_H \cap F' = \emptyset$.

We call the H found in these problems a *witness*. F and F' denote sets of *forbidden pairs*. If the witness does not conflict with the forbidden pairs (i.e., it satisfies the second condition), we say it 'respects' the forbidden pairs.

There are numerous variants on the forbidden pairs problems. For example, one might be interested in an optimization version. Let $\mathrm{FVP}_\Pi(G, F, k)$ be true if and only if there is a subgraph of G with at least k vertices that respects the forbidden pairs and satisfies Π. Alternatively, one could restrict witnesses to only induced subgraphs that satisfy the property and respect the pairs.

6. Prior Work

Let Π be a graph property. The node-deletion problem for Π is the optimization problem of finding the minimum number of nodes one must delete from an input graph to arrive at a resulting induced subgraph satisfying Π. There also is a corresponding edge-deletion problem and edge-contraction problem.

The complexity of the node-deletion problem has been worked on for quite some time, such as in [**14**]. Yannakakis and Lewis, in [**21**] and [**16**], proved a number of important theorems about node-deletion and edge-deletion problems.

In particular, they prove that the node-deletion problem on any nontrivial vertex-hereditary property is **NP**-complete. Lewis and Yannakakis also prove that several common graph properties, such as planarity, form hard edge-deletion problems, but no general theorem is proved.

Asano and Hirata in [**2**] work on the edge-deletion and edge-contraction problem. In minor-hereditary properties with characteristic minors that are all made of 3-connected components, the edge-deletion and edge-contraction problems are **NP**-complete. Planarity is an example of a property that fits the necessary conditions.

Much more recently, in [**1**], Alon, Shapira, and Sudakov show that for any vertex-hereditary property, it can be approximated well if and only if all bipartite graphs satisfy the property. Alon et al. also prove that for hereditary properties that include all bipartite graphs, the problem is **NP**-hard. This partially answered a question Yannakakis posed almost 30 years before.

In a series of 23 major papers starting with [**19**] on graph minors, Robertson and Seymour produced scores of important results in the field. This ultimately culminated with the Robertson-Seymour theorem, which states that every family of graphs that is minor-ancestral can be described by a *finite* set of characteristic minors. In [**8**], the power of the Robertson-Seymour theorem is demonstrated by nonconstructively proving polynomial-time decision algorithms exist for many problems that can be generalized into graph minor problems. [**18**] proves a very important theorem that, given a finite, fixed set of graphs, one can tell in cubic time if a graph contains any member of the set of graphs as a minor. Unfortunately, the proof is nonconstructive.

The concept of forbidden pairs of vertices comes from the field of software testing and automated analysis, and was introduced in [**13**]. Additional motivation is derived from computational biology, such as [**4**]. There exist pairs of mutations, called *synthetic lethals*, either of which allows the organism to survive but both together are fatal. This could be modeled as a forbidden pairs problem.

Gabow, Maheshwari, and Osterweil in [**9**] prove that deciding if a path that respects forbidden pairs exists between two vertices is **NP**-complete. Much later, [**22**] shows that slight restrictions on the forbidden pairs once again make the path-finding problem easy. In this case, a condition was imposed that for any vertices a, b, c, d, when $(a, b), (a, c), (c, d)$ are forbidden, then (b, d) also must be forbidden. [**12**] shows that other restrictions on the forbidden pairs can either keep the forbidden pairs problem hard, or make them polynomial. Despite all of these specific papers, the subject of forbidden pairs is lacking any general results. Specific problems have been classified but no prior work categorizes the complexity of whole swaths of forbidden pairs problems.

7. Outline of Thesis

We attempt to extend and generalize the earlier work on forbidden pairs from [**13**], [**9**], [**22**], and [**12**], much in the way Yannakakis and Lewis did for the node-deletion problems.

In **Chapter 2**, we present a widget that is used in several of our proofs. **Chapter 3** presents two examples illustrating how forbidden pairs problems can be proven to be **NP**-complete for individual properties. **Chapter 4** presents the main results: that for each member of a large class of graph properties, the corresponding forbidden pairs problem is **NP**-complete. We classify a few additional classes of graph properties in **Chapter 5**. The final chapter shows that much work remains to be done on forbidden pairs problems.

We concentrate primarily on forbidden *vertex* pairs problem below. In general, a straightforward argument shows that equivalent theorems follow for the forbidden edge pairs problem.

CHAPTER 2

Preliminary Results

When exploring the proofs that FVP_Π is **NP**-complete for various individual properties, similar constructions seemed to work for each. Here, we construct the **SAT Widget**, which is immensely useful for reducing Monotone 1-in-3 Sat (see **Appendix A** for definition and proof of **NP**-completeness) to forbidden pairs problems.

Let $C = C_1 \wedge ... \wedge C_n$ be an instance of M1-3Sat. We construct a graph $\mathcal{G}$ and forbidden pairs $\mathcal{F}$ as follows:

(1) Let $\mathcal{G}$ be an empty graph of $3n$ vertices.

(2) Label the vertices $v_{i,j}$ with $i \in [1, n]$ and $j \in \{1, 2, 3\}$.

(3) Add an edge to $\mathcal{G}$ between each $v_{i,j}$ and each $v_{i+1,k}$.

(4) Let $V(v_{i,j})$ be the jth variable in C_i.

(5) Let i be the *rank* of $v_{i,j}$.

(6) Add the pairs $(v_{i,x}, v_{i,y})$ where $1 \leq x \neq y \leq 3$ to $\mathcal{F}$.

(7) If $V(v_{i,j}) = V(v_{l,m})$, add all $(v_{i,j}, v_{l,x})$ with $x \neq m$ to $\mathcal{F}$.

This concludes the construction of the **SAT Widget**. Following are several lemmas that prove the widget's worth.

Lemma 1. *A subgraph of $\mathcal{G}$ can have no more than n vertices and respect $\mathcal{F}$.*

Proof. Suppose, for the sake of contradiction, that a subgraph $\mathcal{G}' \subseteq \mathcal{G}$ had more than n vertices. By the pigeonhole principle, there must be an i such that $\mathcal{G}'$ contains two distinct vertices $v_{i,j}$ and $v_{i,k}$ (with $j \neq k$). This is a contradiction;

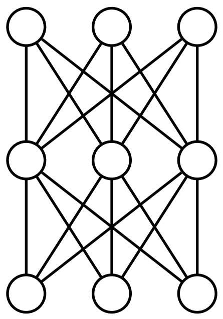

FIGURE 1. An example **SAT Widget**.

step 6 above explicitly forbids two vertices with the same rank from appearing together in a subgraph.

Therefore, all subgraphs of $\mathcal{G}$ that respect $\mathcal{F}$ must have at most n vertices. □

LEMMA 2. *If C is satisfiable, there exists a subgraph of $\mathcal{G}$ with exactly n vertices that respects $\mathcal{F}$. Furthermore, this subgraph is a path.*

PROOF. Suppose that C is satisfiable. Let ϕ be a satisfying assignment for C. We will now construct $\mathcal{G}' \subseteq \mathcal{G}$ that satisfies $\mathcal{F}$ such that $|V_{\mathcal{G}'}| = n$.

Create $\mathcal{G}'$ by discarding every $v_{i,j}$ where $V(v_{i,j}) = \bot$. It should be apparent that exactly n vertices remain. Suppose, for the sake of contradiction, that fewer than n vertices remained. Then there would exist an i such that no $V(v_{i,j})$ equaled $\top$. This means that clause i did not have a true literal in it, so ϕ was not a satisfying assignment for C. Thus, at least n vertices must remain. By **Lemma 1**, $\mathcal{G}'$ cannot have more than n vertices and satisfy $\mathcal{F}$.

There are two ways that $\mathcal{G}'$ could potentially violate a forbidden pair, and both are impossible.

(1) Violating item 6 above is impossible, since by construction, for each i there is exactly one $v_{i,j}$ remaining.

(2) Disrespecting item 7 above is impossible, for if $V(v_{i,j}) = V(v_{l,m})$, then by construction both are in $\mathcal{G}'$ and item 4 ensures that they are the only vertices of ranks i and l respectively.

We have shown that with a valid satisfying assignment for C, we can produce a subgraph with n vertices that respects $\mathcal{F}$. The subgraph will have exactly one vertex of each rank, and each is adjacent to one vertex of greater and one vertex of lesser rank. This forms a path. □

LEMMA 3. *If there exists a subgraph of $\mathcal{G}$ that satisfies $\mathcal{F}$ with exactly n vertices, then C is satisfiable.*

PROOF. Suppose that $\mathcal{G}' \subseteq \mathcal{G}$ satisfies $\mathcal{F}$ and has n vertices. For each vertex $v_{i,j}$ in $V_{\mathcal{G}'}$, assign $V(v_{i,j})$ to $\top$. Assign all other literals to $\bot$.

Due to the forbidden pairs, no rank may have more than one true literal, and since there are n vertices, each clause will have exactly one true literal. Now we need to ensure that no literal has been assigned both $\top$ and $\bot$.

If some literal had been assigned to both $\top$ and $\bot$ then there would be a $v_{i,j}$ and a $v_{l,m}$ with $V(v_{i,j}) = V(v_{l,m})$. The final rule of the construction of $\mathcal{F}$ ensures that both or neither of these two vertices would be in $\mathcal{G}$. Therefore, this situation cannot occur. Thus, the truth assignment extracted from $\mathcal{G}'$ must satisfy C. □

LEMMA 4. *Any subgraph of $\mathcal{G}$ that respects $\mathcal{F}$ will be a single path or disconnected paths.*

PROOF. For each rank, the forbidden pairs ensure that there can be at most one vertex. Suppose, for the sake of contradiction, that some vertex $v_{i,j}$ in a subgraph that respects $\mathcal{F}$ had degree greater than 2.

Consider the adjacent vertices: $v_{i-1,1}, v_{i-1,2}, v_{i-1,3}, v_{i+1,1}, v_{i+1,2},$ and $v_{i+1,3}$. By the pigeonhole principle (or manual verification), any pair of those violates the forbidden pairs that prevents two vertices with the same rank. A contradiction has been reached, so no vertex has degree greater than two. Thus, the subgraph must be components of paths or cycles.

Next, suppose for the sake of contradiction that a cycle existed. Consider a vertex of least rank in the cycle. Since it is part of a cycle, it must have two adjacent vertices of equal or greater rank. We know that no vertex has any edges to other vertices of the same rank. Additionally, each vertex has at most 3 edges to vertices of greater rank, but they are pairwise forbidden. Thus, no vertex in a subgraph that respects $\mathcal{F}$ can have two adjacent vertices of equal or greater rank, so a contradiction has been reached. Thus, no cycles exist in subgraphs that respect $\mathcal{F}$.

Therefore, since all components of a valid subgraph must have degree 2 or less and are not cycles, each component is a path. □

LEMMA 5. *There is a path from $v_{1,j}$ to $v_{n,k}$ for some j, k that respects $\mathcal{F}$ if and only if C is satisfiable.*

PROOF. By **Lemma 3**, if C is not satisfiable, then no subgraph of $\mathcal{G}$ has n vertices while respecting $\mathcal{F}$. For the sake of contradiction, suppose that a path had less than n vertices and was a path from $v_{1,j}$ to $v_{n,k}$. Since there are n ranks between the two but fewer than $n-1$ edges, by the pigeonhole principle, some edge must traverse more than one rank. By construction, this is trivially impossible and we have reached a contradiction. Thus no path with fewer than n vertices will connect the two endpoints.

So, by **Lemma 3**, C must be satisfiable if the path exists. **Lemma 2** proves the other direction. □

CHAPTER 3

Two Illustrative Examples

The following examples are proofs that forbidden pairs problems for two specific graph properties are **NP**-complete. They illustrate the techniques that we generalize in the following chapter.

1. Cyclic

Let Π be the ancestral property *cyclic*, which is true of all graphs containing a cycle. It is important to note that Π can be recognized in polynomial time.

THEOREM 2. FVP_Π *is **NP**-complete.*

PROOF. First, we need to verify that FVP_Π is in **NP**. Let $H \subseteq G$ be a certificate to $\text{FVP}_\Pi(G, F)$. We can check that H is valid by ensuring that each member of $V_H \times V_H$ is not in F. This takes at most quadratic (polynomial) time.

We now reduce MONOTONE 1-IN-3 SAT to FVP_Π. Let $C = C_1 \wedge ... \wedge C_n$ be conjunctive clauses each with 3 literals, an instance of M1-3SAT. Construct a graph Γ and forbidden pairs F as follows:

(1) Let $(\mathcal{G}, \mathcal{F})$ be an instance of the **SAT Widget** for C.
(2) Add $\mathcal{G}$ to Γ and let $F = \mathcal{F}$.
(3) Add a disconnected vertex s.
(4) For each $v_{1,j}$, add an edge $sv_{1,j}$.
(5) For each $v_{n,j}$, add an edge $sv_{n,j}$.

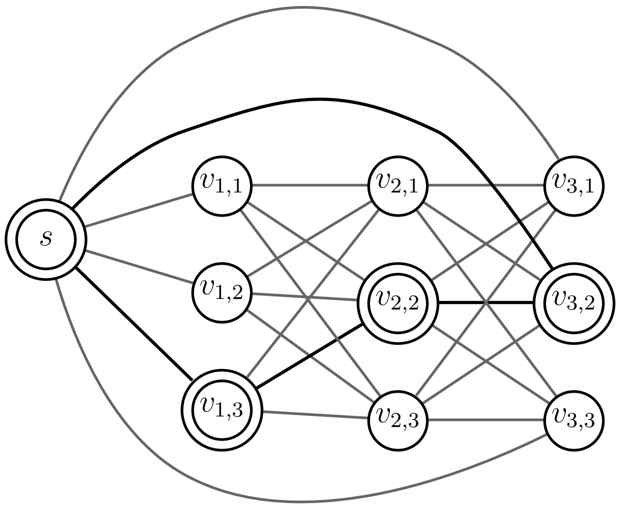

FIGURE 1. Cyclic subgraph highlighted in the **SAT Widget** for example clause $(w \vee x \vee y) \wedge (x \vee y \vee z) \wedge (w \vee y \vee z)$.

Consider a subgraph of Γ that satisfies F. By **Lemma 4**, the widget has no cycles in it. By **Lemma 5**, there is a path of length n through the entirety of the widget that respects $\mathcal{F}$ if and only if C is satisfiable.

Pick the subgraph that consists of the path through the widget, s, and edges $sv_{1,j}$ and $sv_{n,k}$. This is a cycle. Therefore, the answer to $\text{FVP}_{\Pi}(\Gamma, F)$ decides the satisfiability of C.

As an example, **Figure 1** shows the construction for the reduction from an instance of M1-3SAT, $(w \vee x \vee y) \wedge (x \vee y \vee z) \wedge (w \vee y \vee z)$. We are able to extract a cyclic subgraph that respects the forbidden pairs, as denoted by the bolder edges and double-circled vertices. The corresponding truth assignment is $w = \bot, x = \bot, y = \top, z = \bot$. $\square$

2. Matching

A *matching* is a set of edges that are pairwise non-adjacent. Finding a matching in a graph is a common problem with several known polynomial time algorithms. The *matching number* of a graph is the size of the matching with the most edges. Finding matchings and the matching number of a graph is in **P**.

We can adapt the forbidden pairs problem to show that finding matchings in graphs with the presence of forbidden pairs is hard.

A matching is perfect if every vertex in the graph is incident to exactly one edge in the matching. In this case, the matching number is equal to half the number of vertices in the graph. Let Π be the property 'has a perfect matching'. Trivially, there is a polynomial time algorithm for FVP_{Π}: examine every pair of vertices and select the first pair that is adjacent and not forbidden. However:

THEOREM 3. *Given a graph G, forbidden pairs F, and a natural number k, the optimization problem $\mathrm{FVP}_{\Pi}(G, F, k)$ is **NP**-complete.*

PROOF. Whether or not a graph has a perfect matching can be determined in polynomial time, so clearly the problem is in **NP**. Now, we must show that it is also **NP**-hard.

We reduce from M1-3SAT, with $C = C_1 \wedge ... \wedge C_n$ as an instance. Let $(\mathcal{G}, \mathcal{F})$ be a copy of the **SAT Widget** for C. For each i, attach all $v_{i,j}$ to a new vertex w_i to create a new graph $\mathcal{G}'$.

By **Lemmas 2 and 3**, we can find a subgraph of n from the widget vertices if and only if C is satisfiable.

We will now show that $\mathrm{FVP}_{\Pi}(\mathcal{G}', \mathcal{F}, 2n)$ is true if and only if C is satisfiable. Pick a subgraph containing $2n$ vertices. Since there are n clauses and n additional vertices, by **Lemmas 2** and **3**, this is possible if and only if C is satisfiable. Pick a subgraph containing the edges incident to each w_i, but none from the original widget. These edges form n connected components of two vertices each. This is a perfect matching as it contains n edges and $2n$ vertices. Thus, we can use the optimization version of the forbidden vertex pairs problem to decide MONOTONE 1-IN-3 SAT, so the problem is **NP**-complete. $\square$

CHAPTER 4

Main Result

Our main result proves that an infinite number of related graph properties, each decidable in polynomial time, become **NP**-complete under the forbidden vertex pairs decision problem.

Let Π be a minor-ancestral graph property defined by a known set of characteristic graphs. In this chapter, we seek to use the nature of the characteristic graphs of Π to determine the complexity of FVP_Π. We call a graph that is a disjoint union of paths and claws *clawful*.

THEOREM 4. FVP_Π *is* ***NP****-complete if* Π *has no clawful characteristic graph.*

PROOF. Robertson and Seymour showed that any minor-ancestral property can be recognized in polynomial time, so the problem is in **NP**.

To show that it is **NP**-complete, we reduce from Monotone 1-in-3 SAT. Let $C = C_1 \wedge ... \wedge C_n$ be an instance of M1-3SAT. Let H be a characteristic graph for Π with least maximum degree. We construct a graph Γ and forbidden vertex pairs F to act as input to the problem.

First, if any component of H is a subdivision of a star of higher degree than 3, replace the star with a widget that has the star as a minor but reduces the degree to 3. This is demonstrated in **Figure 1**.

Now, for each edge $uv \in E_H$ perform the following:

(1) Delete the edge.

(2) Let $(\mathcal{G}, \mathcal{F})$ be a fresh instance of the **SAT Widget** for C.

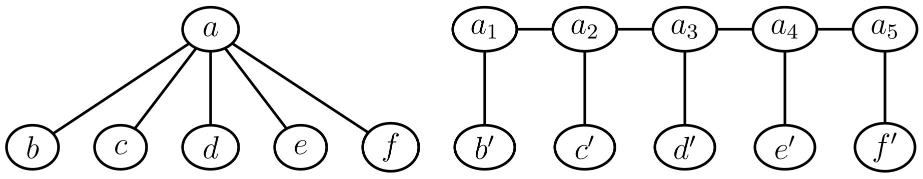

FIGURE 1. The star on the left can be contracted from the graph on the right.

(3) Let $v_{i,j}$ be the vertices in $\mathcal{G}$.

(4) Add all pairs in $\mathcal{F}$ to F.

(5) Add edges $uv_{1,1}, uv_{1,2}, uv_{1,3}, wv_{n,1}, wv_{n,2}, wv_{n,3}$.

CLAIM 1. *If C is satisfiable then there is a graph $\Gamma' \subseteq \Gamma$ that respects F with H as a minor.*

PROOF. By **Lemma 4** there is a path through each $\mathcal{G}$. Take the subgraph $\Gamma' \subseteq \Gamma$ that selects each of these paths and all of the original vertices from H. H is a minor of Γ', since each widget inserted between vertices u and v is a path that can be contracted into a single edge from u to w. After contracting each $\mathcal{G}$, the resulting graph is H. □

CLAIM 2. *If C is not satisfiable then no subgraph of Γ respecting F can have any characteristic graph for Π as a minor.*

PROOF. Let Γ' be any subgraph of Γ. By **Lemma 5**, in Γ' there is no path between any two of the original vertices from H. Additionally, by **Lemma 4**, the remains of each widget in Γ' is a disjoint union of path graphs. We can have multiple paths connecting to original vertices from H, which means that Γ' consists of only paths and subdivisions of stars.

No characteristic graph for Π can be a minor of Γ' because each vertex from G is now a star of lesser degree (by construction) than any stars in the characteristic graphs. $\square$

We proved that there is a subgraph of Γ that respects F and has H as a minor if and only if C is satisfiable. Additionally, the construction can clearly be completed in polynomial time. Therefore, the proof of the theorem is complete. $\square$

THEOREM 5. FVP_Π *is in* $\boldsymbol{P}$ *if a characteristic graph for* Π *is a path.*

PROOF. Suppose that there is a characteristic graph for Π that is a path of length p. $\text{FVP}_\Pi(G, F)$ can be decided as follows.

First, check to see if there is a path of length p in G. This can be done by merely checking every set of p vertices to see if they form a path and are not pairwise forbidden in any way. If there is such a path, we have decided the problem affirmatively.

Next, for each other characteristic graph C, we will show that we can decide whether or not G contains C as a minor that respects the forbidden pairs in polynomial time.

We are going to need at least $|V_C|$ vertices from G to identify with the vertices in C. Once these are picked, they might not be connected in the same way that C is. Extra edges can be discarded, but missing edges would need to be formed through edge contraction of G.

There are up to $|V_C|^2$ missing edges. Each edge will take no more than p contractions. This is because if it took more than p contractions, the path along the vertices being contracted would be a subgraph isomorphic to the path characteristic graph, and the initial search for paths would have located this.

We have an upper bound of $p|V_C|^2$ contractions possible to form C from any subgraph of G. In the worst-case scenario, we try every way to pick $|V_C| + p|V_C|^2$ vertices from G and then try every way to contract.

There are at most $\dfrac{|V_G|!}{(|V_G| - p|V_C|^2)!} \leq |V_G|^{p|V_C|^2}$ ways to pick sufficient vertices, try the appropriate contractions, and check that each resulting graph is isomorphic to C and respects F. This is polynomial in the size of the input graph G. $\square$

CHAPTER 5

Additional Results

In this chapter, we show a few additional results concerning forbidden pairs problems.

1. Hereditary Properties are Easy

THEOREM 6. *If* Π *is edge-, vertex-, or minor-hereditary, then* FVP_{Π} *and* FEP_{Π} *can be decided in polynomial (in fact,* $O(1)$*) time.*

PROOF. If Π is trivial, then there are only a finite number of graphs that do or do not satisfy Π. Thus, FVP_{Π} and FEP_{Π} could be decided in constant time by checking for isomorphism with the input graph.

If Π is nontrivial, then K_1 (a lone vertex) is in Π. Since K_1 is a subgraph, induced subgraph, and minor of every other graph (except the trivial graph), if it were not in Π, then Π could only have finitely many (two) graphs and thus would be trivial.

Pick K_1 as the witness. It has no pairs of vertices or edges so it cannot possibly violate them. We have successfully found a subgraph, induced subgraph, or minor of our input that has the property and does not violate the forbidden pairs. It was found in time that is constant regardless of the size of the input to FVP_{Π}. $\square$

2. Finitely-Defined Edge- and Vertex-Ancestral Properties

THEOREM 7. *Suppose that* Π *is an edge- or vertex-ancestral property that can be characterized by a finite number of graphs. Then one can decide* FEP_Π *and* FVP_Π *in polynomial time.*

PROOF. Let k be the minimum number of characteristic (induced if vertex-ancestral) subgraphs required to define Π, with the largest having l vertices.

We can check for the existence of one of these for a graph $G = (V, E)$ in time on the order of $|V|^l$, or polynomial in the size of the input graph. For each characteristic subgraph $H = (V', E')$, check if any tuple of distinct vertices from $V^{|V'|}$ is isomorphic to H. We spend on the order of $|V|^l$ time on each of the k subgraphs, yielding a time complexity of $O(k|V|^l)$. □

3. Hereditary and Ancestral Optimization

LEMMA 6. *If* Π *is nontrivial and edge-hereditary then all empty graphs satisfy* Π.

PROOF. An empty graph with k vertices is a subgraph of every other graph with at least k vertices. Since Π is hereditary, if there is a graph in Π with at least k vertices, then the empty graph with k vertices satisfies Π. Π is nontrivial, so there are graphs of arbitrary sizes. Thus, for each empty graph, there is a larger graph with the property, so every empty graph satisfies Π. □

THEOREM 8. *Let* Π *be nontrivial, infinitely characterized, and edge-hereditary or edge-ancestral. Then* $\text{FVP}_\Pi(G, F, k)$ *is* $\boldsymbol{NP}$*-hard. Furthermore, if* Π *can be recognized in polynomial time, then the problem is* $\boldsymbol{NP}$*-complete.*

PROOF. If Π can be recognized in polynomial time, then we can show that the optimization variant on FVP_Π is in **NP**. If G is a certificate to FVP_Π then it is given that we can verify that it satisfies Π in polynomial time.

Let $C = C_1 \wedge ... \wedge C_n$ be an instance of M1-3SAT.

First, consider the case where Π is edge-hereditary. We must now show that the optimization variant is **NP**-hard. Our graph and forbidden pairs are exactly those specified in the **SAT Widget** for C. By **Lemma 6**, all empty graphs satisfy Π. **Lemmas 2** and **3** ensure that a subgraph of the widget with n vertices exists if and only if C is satisfiable. We can discard all edges from $\mathcal{G}$ to ensure that any subgraph we choose has the property. Therefore, we can answer M1-3SAT by asking $\text{FVP}_\Pi(\mathcal{G}, \mathcal{F}, n)$.

Finally, we consider the case where Π is edge-ancestral. Let G be the smallest graph that satisfies Π. We can find this in constant time. To this, append a copy of the **SAT Widget** for C to produce a combined graph H and adopt the widget's forbidden pairs. Notice that any subgraph of H that includes all of G is in the property. By **Lemmas 3, 1** and **2**, we can get at most n vertices from the widget, and n if and only if C is satisfiable.

Thus, $\text{FVP}(H, \mathcal{F}, |V_G| + n)$ is true if and only if the widget can be completed, which implies that C must be satisfiable. $\square$

4. Fixed Parameter Tractable

A *parameter* is a function from instances of a problem to $\mathbb{N}$. A problem is *fixed parameter tractable* (FPT) in parameter k if there is an algorithm that runs in time $f(k) * n^{O(1)}$, where f is a function independent of the size of the input problem, n. In other words, holding k constant means that the complexity of the problem grows polynomially with the remainder of the input.

THEOREM 9. *If Π can be recognized as a subgraph in polynomial time, then* FVP_Π *and* FEP_Π *are FPT on the number of forbidden pairs.*

PROOF. Suppose that any input to FVP_Π will have no more than k forbidden pairs. From the k pairs, we can derive at most 2^k different ways to pick one vertex from each pair. For each way, delete the vertices not picked and check the resulting graph for the property. If one of these has it, then answer affirmatively, otherwise reject.

This algorithms takes up to a constant 2^k iterations of a polynomial-time check for the presence of a subgraph with a property. Lower constant overheads have not been investigated. $\square$

CHAPTER 6

Conclusions & Future Work

This thesis seeks to shine light on the computational complexity of an interesting family of problems. Most of the questions in the domain remain open or have yet to be investigated; the flexibility of the problem means it can be transformed in any number of ways.

Of particular interest are the complete bounds on minor-ancestral properties. The following claim will be investigated imminently:

CONJECTURE 1. *Let* Π *be minor-ancestral.* FVP_Π *is in* ***P*** *if and only if there exists a clawful characteristic graph* C*, and for each other characteristic graph for* Π*, one can subdivide any one edge some number of times and find* C *as a minor. Otherwise,* FVP_Π *is* ***NP****-complete.*

Future work could investigate edge- or vertex-ancestral properties, as the forbidden pairs problems for many individual ancestral properties are hard. For many properties, the FVP and FEP problems have the same complexity; further research could determine if the problems are precisely equally hard for every property.

Bibliography

[1] Noga Alon, Asaf Shapira, and Benny Sudakov. Additive approximation for edge-deletion problems. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '05, pages 419–428, Washington, DC, USA, 2005. IEEE Computer Society.

[2] Takao Asano and Tomio Hirata. Edge-deletion and edge-contraction problems. In *Proceedings of the fourteenth annual ACM symposium on Theory of computing*, STOC '82, pages 245–254, New York, NY, USA, 1982. ACM.

[3] John Adrian Bondy. *Graph Theory With Applications*. Elsevier Science Ltd, 1976.

[4] Arthur Brady, Kyle Maxwell, Noah Daniels, and Lenore J. Cowen. Fault tolerance in protein interaction networks: Stable bipartite subgraphs and redundant pathways. *PLoS ONE*, 4(4):e5364, 04 2009.

[5] A. Cobham. The intrinsic computational difficulty of functions. In *Proceedings of the 1964 Congress on Logic, Mathematics and the Methodology of Science*, pages 24–30, 1964.

[6] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, STOC '71, pages 151–158, New York, NY, USA, 1971. ACM.

[7] J. Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17(3):449–467, 1965.

[8] Michael R. Fellows and Michael A. Langston. Nonconstructive tools for proving polynomial-time decidability. *J. ACM*, 35:727–739, June 1988.

[9] H N Gabow, S N Maheshwari, and L J Osterweil. On Two Problems in the Generation of Program Test Paths. *IEEE Transactions on Software Engineering*, SE-2(3):227–231, 1976.

[10] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[11] R. Karp. Reducibility among combinatorial problems. In R. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[12] Petr Kolman and Ondřej Pangrác. On the complexity of paths avoiding forbidden pairs. *Discrete Applied Mathematics*, 157(13):2871–2876, jul 2009.

[13] K. W. Krause, M. A. Goodwin, and R. W. Smith. *Optimal software test planning through automated network analysis*. TRW Systems Group, Redondo Beach, Calif., 1973.

[14] M. S. Krishnamoorthy and Narsingh Deo. Node-deletion np-complete problems. *SIAM Journal on Computing*, 8(4):619–625, 1979.

[15] L. Levin. Universal search problems. *Problems of Information Transmission*, 9:265–266.

[16] John Lewis and Mihalis Yannakakis. The node-deletion problem for hereditary properties is NP-complete. *Journal of Computer and System Sciences*, 20(2):219–230, apr 1980.

[17] Christos Papadimitriou. *Computational Complexity*. Addison Wesley, Reading, MA, USA, 1993.

[18] N. Robertson and P. D. Seymour. Graph minors .xiii. the disjoint paths problem. *Journal of Combinatorial Theory, Series B*, 63(1):65 – 110, 1995.

[19] Neil Robertson and P.D. Seymour. Graph minors. i. excluding a forest. *Journal of Combinatorial Theory, Series B*, 35(1):39 – 61, 1983.

[20] B.A. Trakhtenbrot. A survey of Russian approaches to Perebor (brute-force searches) algorithms. *IEEE Annals of the History of Computing*, 6:384–400, 1984.

[21] Mihalis Yannakakis. Node-and edge-deletion np-complete problems. In *Proceedings of the tenth annual ACM symposium on Theory of computing*, STOC '78, pages 253–264, New York, NY, USA, 1978. ACM.

[22] Hananya Yinnone. On paths avoiding forbidden pairs of vertices in a graph. *Discrete Appl. Math.*, 74:85–92, April 1997.

APPENDIX A

Monotone 1-in-3 SAT

1-IN-3 SAT is the decision problem true of satisfying assignments that include exactly one true literal per clause. MONOTONE 1-IN-3 SAT is true of satisfying assignments that respect the one-in-three condition and have input that includes no negated literals.

First we prove that 1-IN-3 SAT is NP-complete, then we prove the monotone variation.

LEMMA 7. 1-IN-3 SAT *is NP-complete.*

PROOF. We shall use reduction from 3-SAT. Let $X = C_1 \wedge ... \wedge C_k$ be the conjunction of disjunctive clauses of an instance of 3-SAT. We will construct another 3-CNF formula Y that is true exactly when X is true and satisfies the one-in-three requirement.

If $C_i = (x \vee y \vee z)$, it corresponds to the clauses in Y with new variables a, b, c, d, e, f:

$$(x \vee a \vee d) \wedge (y \vee b \vee d) \wedge (a \vee b \vee e) \wedge (c \vee d \vee f) \wedge (z \vee c \vee \bot) \tag{1}$$

We must now prove that if $x \vee y \vee z$ is true, then we can assign truth values to the literals $a, b, c, d, e,$ and f such that one literal per clause disjunctive clause is true. We must also prove that when $x \vee y \vee z$ is false, no assignment of truth values to $a, b, c, d, e,$ and f will have exactly one true literal per clause and satisfy the entire expression.

For the first case, there are 8 possible assignments to x, y, z. Feasible assignments to the remainder of the variables are listed in each row:

x	y	z	a	b	c	d	e	f
$\top$	$\bot$	$\bot$	$\bot$	$\top$	$\top$	$\bot$	$\bot$	$\bot$
$\bot$	$\top$	$\bot$	$\top$	$\bot$	$\top$	$\bot$	$\bot$	$\bot$
$\bot$	$\bot$	$\top$	$\bot$	$\bot$	$\bot$	$\top$	$\bot$	$\bot$
$\top$	$\top$	$\bot$	$\bot$	$\bot$	$\top$	$\bot$	$\top$	$\bot$
$\top$	$\bot$	$\top$	$\bot$	$\top$	$\bot$	$\bot$	$\bot$	$\top$
$\bot$	$\top$	$\top$	$\top$	$\bot$	$\bot$	$\bot$	$\bot$	$\top$
$\top$	$\top$	$\top$	$\bot$	$\bot$	$\bot$	$\bot$	$\top$	$\top$

As each row is a valid truth assignment that satisfies formula 2.1, this proves the first case. Now suppose $x = y = z = \bot$. We quickly see that the clauses must be false:

$x = y = z = \bot$	
$\Longrightarrow c = \top$	from last clause
$\Longrightarrow d = f = \bot$	from penultimate clause
$\Longrightarrow a = b = \top$	from first two clauses
$\Rightarrow\Leftarrow$	in the third clause

This proves that without $x \vee y \vee z$ we cannot satisfy formula 2.1. Thus, 1-IN-3 SAT is just as difficult to solve as 3-SAT. □

PROOF OF THEOREM 1. Now, we prove the monotone variation. Suppose we have a clause of the form $\overline{x} \vee y \vee z$. We replace this with $(a \vee y \vee z) \wedge (a \vee x \vee \bot)$. Due to the one-in-three property, the second clause promises that exactly one of x and a are true. In other words, a must have an opposite truth value of x, just like $\overline{x}$.

Substitute all negated variables in an instance of 1-IN-3 SAT using this method and fresh variables for each. The size of the instance will at most double, and all of this can be performed in linear time. This proves that MONOTONE 1-IN-3 SAT is NP-complete. □