Trial and Error for Graph Properties: Final Report

Aditya Saraf and Thomas Schneider

Summer of 2018

1 Introduction

The goal of this paper is to make progress towards answering the question: *What is the minimum amount of information needed to determine if an n-vertex graph has a monotone graph property?*

One approach to answering this question uses the decision-tree model and focuses on determining how many edge queries are needed, in the worst case, to determine whether a graph has some monotone graph property. This approach forms the study of evasiveness, a survey of which can be found in (Miller, 2013). Our approach differs in that we're more interested in the type of queries needed than the number. In Chapter 7 of her PhD thesis, Sundaram proposes studying this question by using the trial-and-error framework with a Lex-first revealing oracle (Sundaram, 2017). As the question suggests, we keep the number of vertices, n, fixed. Only the *edges* in the graph are unknown. In the trial-and-error framework, an algorithm is able to propose certificates for some monotone graph property (the trial) and an oracle provides feedback about some edge that is present in the certificate but not in the graph (the error). If there are multiple errors, the Lex-first revealing oracle always returns the one that is lexicographically first in its internal list of edges. It's important to note that the algorithm is only allowed to propose *valid* certificates—these are certificates which verify the property in at least one possible n-vertex graph.

The oracle we consider is adapted from a more general class of constraint satisfaction oracles. These oracles maintain an internal list of constraints, which correspond in this case to edges not in the graph. Given this framework, there are many possibilities for the type of information this oracle may reveal. It has been shown that an oracle that only reveals the index of an arbitrary violated constraint is too weak, whereas an oracle that reveals the content of some violated constraint is too powerful (Sundaram, 2017). Hence, we focus on a Lex-first revealing oracle, an oracle that keeps some internal list of edges not in the graph, the constraints, and reveals the index of the lexicographically first violated constraint in this list.

2 Hidden Graph Properties

Definition 1. A *graph property* $\mathcal{P}$ is an isomorphism-closed set of graphs. Moreover, for all graphs $G = (V, E)$ and $H = (V, E')$ such that $E \subseteq E'$ we say that a property $\mathcal{P}$ is *monotone increasing* if $G \in \mathcal{P}$ implies that $H \in \mathcal{P}$.

In this work, since we are concerned only with monotone increasing properties we will use the terms "monotone" and "monotone increasing" interchangeably. Note that the number of vertices is fixed—super graphs can be formed only by adding edges. A monotone increasing graph property is completely characterized by its *certificates*, minimal graphs that have the property. For example, if $\mathcal{P}$ is connectivity, the certificates would be spanning trees. For a fixed n, one may think of the set of valid certificates for a property, $C(\mathcal{P})$, as the certificates of $\mathcal{P}$ in the complete graph on n vertices, K_n. This notation is convenient for hidden graphs, where for any nontrivial monotone $\mathcal{P}$ the complete graph must have the property. For each monotone graph property, we associate multiple constraint satisfaction problems, the first of which is discussed extensively in Sundaram (Sundaram, 2017).

2.1 Hidden CSPs for Monotone Graph Properties

To phrase a graph property as a constraint satisfaction problem, we express a property $\mathcal{P}$ as a series of Boolean functions on $\binom{n}{2}$ variables. Each variable corresponds to the presence of an edge in the complete n-vertex graph. The original definition for an instance of the CSP associated with $\mathcal{P}$, $\mathrm{S}_{\mathcal{P}}$, uses the relation set $\mathcal{R} = \{\mathrm{Neg}\}$ to indicate which edges are not present in the graph. Note that there are at most $\binom{n}{2}$ unique constraints in an instance of $\mathrm{S}_{\mathcal{P}}$. Additionally, we restrict the set of possible assignments $W_{\mathcal{P}}$ to the set of certificates for $\mathcal{P}$. The hidden version of this problem is denoted H-$\mathrm{S}_{\mathcal{P}}$.

We will generalize $\mathrm{S}_{\mathcal{P}}$ to $\mathrm{S}_{\mathcal{P}}^{k-\mathrm{Id}}$, to indicate that we're allowing k occurrences of the Id relation. We have that $\mathrm{S}_{\mathcal{P}}^{k-\mathrm{Id}}$ uses the relation set $\mathcal{R}_{\mathrm{Id}} = \{\mathrm{Neg}, \mathrm{Id}\}$, but instances of $\mathrm{S}_{\mathcal{P}}^{k-\mathrm{Id}}$ are further restricted to containing exactly k constraints of type Id which must before the negation constraints. (Recall that constraints of type Id correspond to edges which must be present in valid certificates.) Note that there can still be at most $\binom{n}{2}$ unique constraints in an instance of this problem; k of them show the presence of some edge and there can at most $\binom{n}{2} - k$ remaining constraints that identify the absence of some edge.

In summary, we're concerned with the following possibilities for CSPs based on monotone graph properties. The former has been studied earlier, and the latter is our own addition.

1. $\mathrm{Neg}, \cdots, \mathrm{Neg}$
2. $\overbrace{\mathrm{Id}, \cdots, \mathrm{Id}}^{k}, \mathrm{Neg}, \cdots, \mathrm{Neg}$

The hidden versions of these problems that have access to a Lex-first oracle will be denoted as H-$\mathcal{P}$ and H-$\mathcal{P}$-k-Id. We also define the following classes of problems.

Definition 2 (Hidden Graph Properties)**.** H is the set of all H-$\mathcal{P}$ for any monotone graph property $\mathcal{P}$.
Definition 3 (Fixed Id Hidden Graph Properties)**.** H-k-Id is the set of all H-$\mathcal{P}$-k-Id for any monotone graph property $\mathcal{P}$.

In practical terms, an algorithm that solves $H - \mathcal{P}$-k-Id will be of the form $\mathcal{A}(E')$, where $|E'| \leq k$. The algorithm, when given access to a consistent oracle that hides graph G, will return a valid certificate, $\mathbf{T}$ for $\mathcal{P}$ if one exists in G that extends E' (e.g. $E' \subseteq \mathbf{T}$). Otherwise, the algorithm will return UNSAT.

2.2 Relevant Problems About Monotone Graph Properties

This section introduces several problems related to monotone graph properties. Though these problems are divorced from the hidden setting, we'll show in the next section their relationship to Hidden CSPs.

Definition 4 (The Certificate Finding Problem)**.** Given a monotone increasing graph property $\mathcal{P}$, the $\mathcal{P}$-CF problem asks: Given a graph $G = (V, E)$, return a certificate $\mathbf{T}$ for $\mathcal{P}$ such that $\mathbf{T} \subseteq E$.

This problem is what one would expect of a certificate finding problem—the goal is to simply find a certificate for a specified property in a given graph.

Definition 5 (The Certificate Extending Problem)**.** Given a monotone increasing graph property $\mathcal{P}$, the $\mathcal{P}$-CE problem asks: Given a graph $G = (V, E)$, and a set of edges $E' \subseteq E$, return a certificate $\mathbf{T}$ for $\mathcal{P}$ such that $E' \subseteq \mathbf{T} \subseteq E$.

The problem is a generalization of the previous one, since we can always take E' to be the empty set. Given a partial assignment, E', the goal of this problem is to determine whether this partial assignment can be extended to a valid certificate of the property in the given graph. For parity with our hidden problem definition, we also define a related problem.

Definition 6 (Fixed Size Certificate Extending Problem)**.** Given a monotone increasing graph property $\mathcal{P}$, the $\mathcal{P}$-k-CE problem asks: Given a graph $G = (V, E)$, a positive integer k, and a set of edges $E' \subseteq E$ where $|E'| \leq k$, return a certificate $\mathbf{T}$ for $\mathcal{P}$ such that $E' \subseteq \mathbf{T} \subseteq E$.

Note that k need not be constant with regard to the input - k can be a factor of n.

Though $\mathcal{P}$-CF, $\mathcal{P}$-CE and $\mathcal{P}$-k-CE are defined in terms of a specific monotone graph property, we can also think about them as classes of problems, which give the following characterizations:

Definition 7 (Certificate Finding)**.** CF is the set of all $\mathcal{P}$-CF for any monotone graph property $\mathcal{P}$.
Definition 8 (Certificate Extending)**.** CE is the set of all $\mathcal{P}$-CE for any monotone graph property $\mathcal{P}$.
Definition 9 (Fixed Size Certificate Extending)**.** k-CE is the set of all $\mathcal{P}$-k-CE for any monotone graph property $\mathcal{P}$.

Note that for $k = 0$, k-CE = CF.

3 The Difficulty of Hidden Problems

In earlier work, Sundaram proved that H-$\mathcal{P} \leq_p \mathcal{P}$-1-CE (Sundaram, 2017). Our initial goal was to provide a polynomial equivalence between graph problems in the hidden setting and some class of problems in the non-hidden setting. We refer to this as a transfer theorem. As a first step towards this goal, we bound the complexity of hidden graph problems between two related problems in the classical setting. We aim to prove the following theorem.

Theorem 3.1. *For any $k \in \mathbb{N}$, k-CE $\leq_p$ H-k-Id $\leq_p$ $(k+1)$-CE.*

We prove the two parts of this theorem separately.

Lemma 3.2. *For any $k \in \mathbb{N}$, k-CE $\leq_p$ H-k-Id.*

Proof. This direction is fairly straightforward to see from the definitions. Given a property, an algorithm that can find (through oracle access) a certificate with (at most) k fixed edges in a *hidden* graph can surely be used to find a certificate with some set of at most k edges in a *known* graph. The only difference between these two classes of problems is the hidden graph; so clearly with more information, the problem becomes easier. To prove this, we simply simulate an oracle for an algorithm that solves the hidden problem.

Let $\mathcal{P}$ be an arbitrary monotone graph property and let $(G = (V, E), E' = \{e_1^*, \ldots, e_k^*\})$ be the input to the $\mathcal{P}$-k-CE problem. Let $\mathcal{A}(E')$ be an algorithm that solves the $H - \mathcal{P}$-k-Id hidden CSP and let $e_1, \ldots, e_m$ be an arbitrary ordering of the edges missing from G (i.e edges in $\mathcal{K}_n \setminus G$). We define a CSP with $m + k$ constraints where $C_1 = e_1^*, \ldots, C_k = e_k^*$ and $C_{k+1} = e_1, \ldots, C_{k+m} = e_m$. Our algorithm works as such: Simulate a Lex-first oracle for $\mathcal{A}(E')$ by returning the index of the first violated constraint and output what $\mathcal{A}$ outputs. Because $\mathcal{A}$ solves the hidden CSP $H - \mathcal{P}$-k-Id, it must eventually propose a certificate $\mathbf{T}$ such that $e_i^* \in \mathbf{T}$ for all $i \in [k]$ and $e_j \notin \mathbf{T}$ for all $j \in [k+1, \ldots, k+m]$. If $\mathcal{A}$ rejects the instance, then no certificate can exist in the original graph. Thus, we solve the $\mathcal{P}$-k-CE problem with only an $O(|\overline{E}|)$ (where $|\overline{E}| = m$, as $\overline{E}$ is the set of edges not in G) running time overhead to construct the list of constraints. ☐

Lemma 3.3. *For any $k \in \mathbb{N}$, H-k-Id $\leq_p$ $(k+1)$-CE.*

We prove this for an arbitrary monotone graph property $\mathcal{P}$ by designing an algorithm similar to Sundaram's Lex-first algorithm (Sundaram, 2017). Our new algorithm has two general phases. First, we learn as much as we can about the Id constraints by narrowing down an initially exhaustive set of candidates. Then, we use a slightly modified version of the Lex-first algorithm that ensures that the Id constraints are present in all considered certificates.

The original algorithm keeps track of a set of "critical edges" and ensures that it never removes these edges when modifying a trial. This set is defined as:

$$\mathbf{Crit} := \{e \in \mathcal{G} | \mathcal{G} \setminus \{e\} \text{ does not have a certificate for } \mathcal{P}\}$$

where $\mathcal{G}$ the current guess for the graph. Since we want the algorithm to treat the edges required by the identity constraint as critical (in the sense that they are never removed), we must propose a set of queries to deduce as much as possible about the Id constraints $C_1, \dots, C_k$. The complicating factor is that we can only propose valid certificates to the oracle—we cannot simply propose each edge one at a time. This complication also means that we cannot guarantee that we learn the Id constraints. Suppose that, for some property $\mathcal{P}$, and some $i < k$, $C_i = (1, 2)$ (where (i, j) denotes the edge from vertex i to j) and every certificate for $\mathcal{P}$ that contains (1,2) also contains (3,4) and (5,6). Then, if C_i was one of these edges, no certificate proposed to the oracle could differentiate between the three edges. The key idea is that in such a case, edges (3,4) and (5,6) are de-facto required—they are also critical edges. We will have to treat these critical edges (denoted $\mathbf{Crit}_i^*$ for the critical set corresponding to the ith Id constraint) differently; it is possible to construct a certificate without these edges (but such a certificate wouldn't necessarily contain the Id-required edges), so we must ensure that every certificate we examine contains all edges in all $\mathbf{Crit}_i^*$'s. So we learn $\mathbf{Crit}_\mathbf{1}^*, \dots, \mathbf{Crit}_\mathbf{k}^*$ corresponding to hidden ID constraints $C_1, \dots, C_k$. Then, we can use the $\mathcal{P}$-$(k+1)$-CE solver to ensure that all certificates examined in Phase 2 contain all $\mathbf{Crit}_i^*$.

Now that the key ideas have been highlighted, we present the algorithm. We use solver $\mathcal{A}$, which takes a graph and an edge subset of size at most $k + 1$, producing a certificate for $\mathcal{P}$ that contains the subset.

Algorithm 1 A generalized lex-first algorithm for $H - \mathcal{P}$-k-Id

```
1. Initialize global variables, setting $\mathcal{G} \leftarrow \mathcal{K}_n$, $\mathcal{I} \leftarrow \emptyset$, $\mathcal{I}_e \leftarrow \emptyset$, $\textbf{Crit} \leftarrow \emptyset$, and $\forall j : [1, k+m]$, $\hat{C}_j \leftarrow \{e_1, \ldots, e_{\binom{n}{2}}\}, \textbf{Crit}_j^* \leftarrow \emptyset$.
2. for $i \leftarrow 1$ to $k$ do
  2.1. Let $R$ be a set that includes one arbitrarily selected edge from each of $\textbf{Crit}_1^* \ldots \textbf{Crit}_{i-1}^*$.
  2.2. Choose $(e, e')$ from $\hat{C}_i'$. Find a certificate, $\mathbf{T} = \mathcal{A}(\mathcal{K}_n \setminus \{e\}, \{e'\} \cup R)$. Get a violation $j$. If $\mathbf{T}$ cannot be created for any $(e, e') \in \hat{C}_i'$, go to Step 2.3.
      • If $j = i$: $\hat{C}_i' \leftarrow \hat{C}_i' \cap \overline{\mathbf{T}}$.
      • If $j > i$: $\hat{C}_i' \leftarrow \hat{C}_i' \cap \mathbf{T}$.
      • Repeat from Step 2.2.
  2.3. $\textbf{Crit}_i^* \leftarrow \hat{C}_i'$.
3. end for
4. $\textbf{Crit} = \bigcup_{i=1}^{k} \textbf{Crit}_i^*, \mathcal{I} \leftarrow \{1, \ldots, k\}$.
5. Let $R$ be a set that includes one arbitrarily selected edge from each of $\textbf{Crit}_1^* \ldots \textbf{Crit}_k^*$. If no certificate can be returned from $\mathcal{A}(\mathcal{G}, R)$, abort and output UNSAT.
6. Find a certificate for $\mathcal{P}$ in $\mathcal{G}$, $\mathbf{T}_1 = \mathcal{A}(\mathcal{G}, R)$, get a violation $j_1$ and set $\hat{C}_{j_1} \leftarrow \hat{C}_{j_1} \cap \mathbf{T}_1$.
7. If $\hat{C}_{j_1} \subseteq \textbf{Crit}$, return UNSAT and abort. If $\hat{C}_{j_1} = \{e”\}$, then $C_{j_1} = (\bar{e}”)$, set $j \leftarrow j_1$ and go to Step 9. Otherwise, to create the next trial:
    • Pick an edge, $e \in \hat{C}_{j_1} \cap \mathbf{T}_1 \cap \overline{\textbf{Crit}}$
    • Find a certificate $\mathbf{T}_2 = \mathcal{A}(\mathcal{G} \setminus \{e\}, R \cup \{e'\})$ by trying all $e' \in \hat{C}_j \setminus \{e'\}$.
    • Set $A \leftarrow \mathbf{T}_1 \setminus \mathbf{T}_2$ and $B \leftarrow \mathbf{T}_2 \setminus \mathbf{T}_1$.
    • If no $\mathbf{T}_2$ can be created for any choice of $e \in \hat{C}_{j_1} \cap \mathbf{T}_1 \cap \overline{\textbf{Crit}}$, $j \leftarrow j_1$ and go to Step 9.
8. Use $\mathbf{T}_2$ as the next trial, get the violation $j_2$ and compare $j_1, j_2$:
    • If $j_1 = j_2$, $\hat{C}_{j_1} \leftarrow \hat{C}_{j_1} \setminus A$.
    • If $j_1 \lneq j_2$, $\hat{C}_{j_1} \leftarrow \hat{C}_{j_1} \cap A$.
    • If $j_2 \lneq j_1$, $\hat{C}_{j_2} \leftarrow \hat{C}_{j_2} \cap B$, $j_1 \leftarrow j_2$ and $\mathbf{T}_1 \leftarrow \mathbf{T}_2$.
    • $j \leftarrow \min\{j_1, j_2\}$ and if $|\hat{C}_j| > 1$, repeat from Step 7.
9. Update $\mathcal{I}_e \leftarrow \mathcal{I}_e \cup \hat{C}_j$, $\mathcal{G} \leftarrow \mathcal{K}_n \setminus \mathcal{I}_e$, $\mathcal{I} \leftarrow \{j' | \hat{C}_{j'} \subseteq \mathcal{I}_e\}$, $\textbf{Crit} \leftarrow \{e \in \mathcal{G} | \mathcal{G} \setminus \{e\}$ does not satisfy $\mathcal{P}\}$.
10. Repeat from Step 5 till the oracle returns YES.
```

Phase 1 of the algorithm is from step 1 to step 4. Phase 2 of the algorithm is from step 5 to step 10.

Our algorithm takes an iterative approach to the k-Id problem by discovering as much as possible about each Id constraint, one at a time. The intuition behind the set R is that we need some way of ensuring that we're making progress — that every iteration of Step 2.2 removes an edge from the current $\hat{C}_i'$ or progresses to the next constraint: $\hat{C}_{i+1}'$. In other words, once the $\textbf{Crit}_i^*$'s are fixed in Step 2.3, they never need to be modified. By requiring R in our certificates, we ensure that at iteration i (in Phase 1), we will not violate any constraint C_j for $j < i$, which achieves the progress we just discussed. We now need to prove an important property about the $\textbf{Crit}_i^*$'s.

Lemma 3.4. *If* $\mathbf{T}$ *has* $e \in \textbf{Crit}_i^*$*, and further* $\mathbf{T}$ *contains* $\textbf{Crit}_1^*, \ldots, \textbf{Crit}_{i-1}^*$*,* $\mathbf{T}$ *contains* $\textbf{Crit}_i^*$*. Alternatively,* $\forall e \in \textbf{Crit}_i^* : \textbf{Crit}_1^* \cup \ldots \cup \textbf{Crit}_{i-1}^* \cup \{e\} \subseteq \mathbf{T} \implies \textbf{Crit}_i^* \subseteq \mathbf{T}$

Proof. Suppose, for the sake of contradiction, for arbitrary i and for some certificate $\mathbf{T}$ that contains $\textbf{Crit}_1^*, \ldots, \textbf{Crit}_{i-1}^*$, there exists $e, e' \in \textbf{Crit}_i^*$ such that $e \in \mathbf{T}$, but $e' \notin \mathbf{T}$. This means that $\mathbf{T}$ can be returned from $\mathcal{A}(\mathcal{K}_n \setminus \{e'\}, \{e\} \cup R)$, as R is comprised of one edge from each of $\textbf{Crit}_1^*, \ldots, \textbf{Crit}_{i-1}^*$. Note that when $\mathcal{A}(\mathcal{K}_n \setminus \{e'\}, \{e\} \cup R)$ is first called, it need not return $\mathbf{T}$. But since we only proceed to Step 2.3 when no more certificates can be found, $\mathbf{T}$ must eventually be returned from some call to $\mathcal{A}(\mathcal{K}_n \setminus \{e'\}, \{e\} \cup R)$. Thus, either e or e' would've been removed from $\hat{C}'_i$ (depending on the violation index, j). Since $\mathbf{Crit}^*_i$ contains the edges in $\hat{C}_i$ after Step 2.2, both e, e' cannot exist in $\mathbf{Crit}^*_i$. Thus, we have reached a contradiction. □

Since we always include R in our certificate, by Lemma 2.4, our certificate will never violate any $C_i : i < j$. Note also that $|R| \leq k - 1$, so $\mathcal{A}$, which can solve for subsets of size at most $k + 1$, will always be able to solve for $\{e\} \cup R$.

For our proof, let $C_1 = e^*_1, \ldots, C_k = e^*_k$.

Proof of Lemma 3.3. To prove this theorem, we first prove that every iteration of Step 2.2 decreases $|\hat{C}_i|$ while ensuring $C_i \subseteq \hat{C}_i$, for arbitrary $i \in [1, k]$. We present an inductive argument (inducting over the number of iterations): at first, $\hat{C}_i$ contains all edges, so $C_i \subseteq \hat{C}_i$. For the inductive step, there are two cases. If $j = i$, then $\mathbf{T}$ does not contain e^*_i, which means that $C_i \subseteq \hat{C}_i \cap \overline{\mathbf{T}}$. Further, since $e' \in \hat{C}_i \cap \mathbf{T}$, $|\hat{C}_i|$ decreases. If $j > i$, then $\mathbf{T}$ does contain e^*_i, which means that $C_i \subseteq \hat{C}_i \cap \mathbf{T}$. Further, since $e \in \hat{C}_i \cap \overline{\mathbf{T}}$, $|\hat{C}_i|$ decreases. As we showed earlier, j will never be less than i. So Step 2 decreases $|\hat{C}_i|$ while retaining the invariant. Thus, in Step 2.3 for some iteration i, $C_i \subseteq \hat{C}_i = \mathbf{Crit}^*_i$.

Now, we prove that, $\forall i \in [1, k]$, $e^*_1, \ldots, e^*_k$ will be present in every certificate in Phase 2 and that we don't exclude from consideration any certificate that contains $e^*_1, \ldots, e^*_k$. In Phase 2, we require all certificates to have a nonempty intersection with R. This, combined with Lemma 2.4, ensures that every certificate contains all edges in $\mathbf{Crit}^*_1, \ldots, \mathbf{Crit}^*_k$. Since $C_i \subseteq \mathbf{Crit}^*_i$, this ensures that every certificate contains $e^*_1, \ldots, e^*_k$.

Further, by Lemma 2.4, any certificate that contains $e^*_1, \ldots, e^*_k$ must also contain $\mathbf{Crit}^*_1, \ldots, \mathbf{Crit}^*_k$. Thus, by requiring certificates to contain R, we are not excluding any certificate that contains $e^*_1, \ldots, e^*_k$.

Now we address the small change made to Phase 2 of the algorithm to switch from the $\mathcal{P}^\cap$ model to the $\mathcal{P}$-CE model. In Step 7, in order to find a certificate that intersects $\hat{C}_j \setminus \{e\}$ with our $\mathcal{P}$-$(k+1)$-CE solver, we simply try to include every edge in $\hat{C}_j$, one at a time. Only if no such edge exists do we proceed to Step 9.

Finally, we prove the complexity of our algorithm. Let T be the runtime of $\mathcal{A}$. Note that Step 2.2 will always remove at least one edge from $\hat{C}_i$, so it will take at most $O(n^2)$ iterations of Step 2.2 to move to Step 2.3. Each iteration of Step 2.2 takes time $O(T)$. So each iteration of Step 2 takes time $O(Tn^2)$, for total time $O(Tkn^2)$ spent in Step 2. Finally, Step 4 takes at most $O(k)$ time, so the total time complexity for Phase 1 is $O(Tkn^2)$. Finally, our modification to Phase 2 adds only polynomially more work, as each $\hat{C}_i$ can have at most $O(n^2)$ edges. So, our algorithm is in poly(n)-time when $T = \text{poly}(n)$. □

Now we'll combine the above results into a transfer theorem.

Theorem 3.5. H $=_p$ $\mathcal{P}$-CE

Proof. Note that this theorem follows directly from Theorem 3.1. Any hidden problem must have some number, k, of Id constraints. And for any k, there is a corresponding problem in $(k+1)$-CE that the hidden problem reduces to. Since this problem in contained in CE, we have that H $\leq_p$ CE. The proof for the other direction is symmetrical. □

This transfer theorem allows us the consider the complexity of hidden graph properties in terms of the certification extension problems on those properties. So, to determine which hidden graph properties are easily solved, we explore the difficulty of certificate extensions. We first show that certificate extension gets harder as we increase the size of the initial set (E').

4 Fixed Size Certificate Extension

In this section, we'll prove strict inclusion between Fixed Size Certificate Extension classes of varying size.

Theorem 4.1. *For every $k \in \mathbb{N}$, k-CE $\lneq_p$ $(k+1)$-CE.*

To show this, we must only show this for a particular graph property, $\mathcal{P}$-$(k+1)$-CE is hard but $\mathcal{P}$-k-CE is easy. We first show that 0-CE $\lneq_p$ 1-CE. Note that 0-CE is simply CF.

Lemma 4.2. *Let $\mathcal{P}$ be the property that a directed graph has a simple path from s to t for some fixed vertices s and t. Then the $\mathcal{P}$-CF problem is in **P** while the $\mathcal{P}$-1-CE problem is **NP**-Hard.*

Proof. In order to find a simple path from s to t in polynomial-time (if such a path exists), we can use breath-first search.

To show that the $\mathcal{P}$-1-CE problem is **NP**-Hard, we will reduce it to the *directed path-through-a-vertex problem*, i.e., the subgraph homeomorphism problem where the pattern graph is a path of length two. The directed path-through-a-vertex problem is shown to be **NP**-Hard in (Fortune et al., 1980).

Let G be a graph and fix the distinct vertices s, t, and u, and let's also assume we have access to an oracle for the $\mathcal{P}$-1-CE problem. We wish to determine if there exists a path from s to t that passes through the vertex u. To do this, we loop over all possible edges (u, v) and query our oracle for a path from s to t that uses the edge (u, v). If the oracle gives us a path, then this path must contain the vertex u. Moreover, any path from s to t that passes through the vertex $u \neq t$ must involve the edge (u, v) for some v. Thus, the $\mathcal{P}$-1-CE problem is **NP**-Hard. □

4.1 The Size of the Incomplete Certificates Matters

We've shown in the last section that there exists a property for which $\mathcal{P}$-0-CE is in **P** while $\mathcal{P}$-1-CE is **NP**-Complete (for convenience, we'll say that we've found a property that *separates* $\mathcal{P}$-0-CE from $\mathcal{P}$-1-CE). Next, we will show that for any natural number k there exists a property that separates $\mathcal{P}$-k-CE from $\mathcal{P}$-$(k+1)$-CE.

To do this, we'll generalize the directed simple path problem. Let $\mathcal{P}_k$ be the property that a graph G has a simple path from some fixed s to some fixed t that leaves at least k edges in G unused. Certificates for this property will be paths from s to t along k edges in G. Checking whether a edge set T is a certificate can be done in polynomial time using breath-first search to find a path of length $|T| - k$ in the graph made by the edges in T.

Theorem 4.3. *The $\mathcal{P}_k$-k-CE problem is in **P**.*

Proof. First, we will give a polynomial-time algorithm that solves the $\mathcal{P}_k$-k-CE problem. Assume we're given a graph $G = (V, E)$, an edge set E' of size at most k, and two vertices s and t. The algorithm works in three parts:

1. Set T to be the edges in the shortest path from s to t. If no path exists, abort. If $|T| > |E| - k$, abort.
2. Let T' be $T \cup E'$.
3. Keep adding edges in from $E \setminus T'$ to T' until $|T'| = k + |T|$. Output T'.

Clearly, T' will be a certificate for $\mathcal{P}_k$ as well as $T' \subseteq E$. We'll show that if this algorithm fails to find a certificate for $\mathcal{P}_k$ from G then none exist. If the algorithm is unable to find a shortest-path from s to t then there is no certificate for $\mathcal{P}_k$ because such a certificate would need to contain a path from s to t. Now assume that $|T| > |E| - k$ but there exists some certificate S for $\mathcal{P}_k$ from G. Then S can be decomposed into some path P from s to t and a set of k other edges. So, $|S| = |P| + k$. Since $|S| \leq |E|$, we get that:

$$0 \leq |E| - |S| < |T| - |P|.$$

But this would imply that $|P| < |T|$, contradicting the fact that T is a shortest path from s to t.

To prove the correctness of the third step, observe that $|T| \leq |P|$ and $|P| + k \leq |E|$ imply that $|T| + k \leq |E|$. Adding $|T \cap E'|$ to both sides and rearranging gets:

$$|T \cap E'| \leq |E| + |T \cap E'| - |T| - k = |E| + (|T \cap E'| - |T| - |E'|) = |E| - |T \cup E'| = |E \setminus T'|.$$

This means that there are enough edges in $E \setminus T'$ to extend $T \cup E'$ into a full certificate. Thus, our algorithm will always provide an output if a certificate exists for $\mathcal{P}_k$ from G that uses all of the edges in E'. □

The proof for Theorem 4.3 relies heavily on the fact that $|E'| = k$, the number of edges required in a certificate outside of the path from s to t. To prove Theorem 4.4, we will reduce the $\mathcal{P}_1$-0-CE problem to the $\mathcal{P}_{(k+1)}$-k-CE problem.

Theorem 4.4. *The $\mathcal{P}_{(k+1)}$-k-CE problem is **NP**-Complete.*

Proof. Assume that we have a solver for $\mathcal{P}_{(k+1)}$-k-CE problems. Let $G = (V, E)$ and $E' = \{e^*\}$ be an instance of the $\mathcal{P}_1$-0-CE problem we wish to solve. We will modify G by adding $2k$ vertices and k edges as such:

$$\begin{aligned} G_{\text{mod}} = \{&V \cup \{u_1, v_1, u_2, v_2, \ldots u_k, v_k\}, \\ &E \cup \{(u_1, v_1), (u_2, v_2), \ldots, (u_k, v_k)\}\} \end{aligned}$$

where u_i and v_i are all new vertices. Now we will show that we can solve the $\mathcal{P}_1$-0-CE problem by posing G_{mod} and $E'_{\text{mod}} = \{e^*, (u_1, v_1), (u_2, v_2), \ldots, (u_k, v_k)\}$ to our solver for $\mathcal{P}_{(k+1)}$-k-CE problems. We will show that the G, E' instance of the $\mathcal{P}_1$-0-CE problem has a solution if and only if the $G_{\text{mod}}, E'_{\text{mod}}$ instance of the $\mathcal{P}_{(k+1)}$-k-CE problem has a solution.

First, let's say that T is a solution to the G, E' instance of the $\mathcal{P}_1$-0-CE problem. Then $T_{\text{mod}} = T \cup \{(u_1, v_1), (u_2, v_2), \ldots, (u_k, v_k)\}$ is a solution to the $G_{\text{mod}}, E'_{\text{mod}}$ instance of the $\mathcal{P}_{(k+1)}$-k-CE problem. This is because $E'_{\text{mod}} = \{e^*, (u_1, v_1), (u_2, v_2), \ldots, (u_k, v_k)\} \subseteq T$ since $e^* \in T$ and T_{mod} and T_{mod} contains a path from s to t since T does.

Now assume that we have that our $\mathcal{P}_{(k+1)}$-k-CE problem-solver gives us T_{mod} as a solution. Since T_{mod} must necessarily contain $\{(u_1, v_1), (u_2, v_2), \ldots, (u_k, v_k)\}$, we can create $T = T_{\text{mod}} \setminus \{(u_1, v_1), (u_2, v_2), \ldots, (u_k, v_k)\}$, which will only have one edge in it, e^*. Since none of the vertices in $\{u_1, v_1, u_2, v_2, \ldots u_k, v_k\}$ are connected to s or t, they can't appear in a path from s to t. Thus, T contains a path from s to t. Moreover, T contains e^* since T_{mod} contains e^* and it wasn't removed. So T is a solution to the $\mathcal{P}_1$-0-CE problem. Thus, our reduction is complete. □

5 When is Certificate Extension Efficient?

We've shown in the previous sections that we're not always able to efficiently extend certificates for efficiently-certifiable graph properties. We'll now look at some monotone graph properties, and determine if their corresponding $\mathcal{P}$-CE problems are solvable in polynomial time. For brevity, we'll say that a graph property is *extensible* if the related $\mathcal{P}$-CE problem is in **P** and we'll say that a property is *k-extensible* if the related $\mathcal{P}$-k-CE problem is in **P**. Extensible properties are of particular interest because they imply that the corresponding hidden problem is easy to solve.

In each of the following sections, $G = (V, E)$ will be a (un)directed graph for which we wish to find a certificate. The partial certificate we wish to extend will be E'.

5.1 Connectivity and Spanning Trees (Undirected Graphs)

Let $\mathcal{P}_{\text{Con}}$ be the property that a graph is connected. In this section, we'll show that $\mathcal{P}_{\text{Con}}$ is extensible.

Notice that the set of certificates $\mathcal{C}_{\text{Con}}$ for $\mathcal{P}_{\text{Con}}$ is the set of all spanning trees. While one characterization of spanning trees is that they are minimal connected graphs, spanning trees can be equivalently characterized as being maximal acyclic graphs. This duality forms the basis for why partial-certificates connectivity are easily expendable.

Given a set of edges E', determining whether E' is acyclic can be done in polynomial time. Extending E' is simple: we can simply keep adding edges from $E \setminus E'$ to E' such that E' is always a forest. If E' ever contains $n-1$ edges, it must be a spanning tree, and we're done.

The reason this greedy algorithm works is because the $\mathcal{C}_{\text{Con}}$ is the set of bases for a matroid—in particular, they form the maximal independent sets in the graphic matroid of a complete graph.

5.2 Perfect Matchings (Undirected Graphs)

Let $\mathcal{P}_{\text{PM}}$ be the property that a graph has a perfect matching (see 22.4 in (Cormen et al., 2009)). In this section, we'll show that $\mathcal{P}_{\text{PM}}$ is extensible.

Then the set of certificates $\mathcal{C}_{\text{PM}}$ for $\mathcal{P}_{\text{PM}}$ will be perfect matchings. Given a set of edges E', we first determine whether E' is a matching by determining whether there is some vertex in V that is adjacent to multiple edges in E'. Now we want to determine whether E' is able to be extended to a perfect matching—a matching of size $n/2$. We start by defining $V' = \{v' \in V \ : \ (u', v') \in E' \text{ for some } u' \in V\}$, all the vertices involved in the matching. Clearly, E' is a perfect matching for the subgraph of G restricted to vertices in V'. Now consider the modified graph H, which is the subgraph induced when the vertices V' are removed from G. We claim that H has a perfect matching if and only if G has a perfect matching if and only if G has a perfect matching that is an extension of E'.

Thus, we've reduced the $\mathcal{P}_{\text{PM}}$-CE problem to the $\mathcal{P}_{\text{PM}}$ problem, which is known to be in **P** by (Edmonds (1965); Micali and Vazirani (1980)).

5.3 Vertex-disjoint Cycle Cover (Undirected Graphs)

Let $\mathcal{P}_{\text{CC}}$ be the property that a graph has a cycle cover.

Then the set of certificates $\mathcal{C}_{\text{CC}}$ for $\mathcal{P}_{\text{CC}}$ will be cycle covers.

A vertex-disjoint cycle cover can also be found in polynomial time due to a common reduction to bipartite perfect matching, which we include for completeness. Given a graph $G = (V, E)$, we construct bipartite graph $G' = (V', E')$ with $V' = L \cup R$ by duplicating every vertex $v \in V$ into $v_l \in L$ and $v_r inR$. Then for edge $(u, v) \in E$ we add (u_l, v_r) to E'. Thus a perfect matching will choose exactly two edges for every vertex to create cycles that span the graph.

We're able to reduce the $\mathcal{P}_{\text{CC}}$-CE problem to the $\mathcal{P}_{\text{CC}}$ problem as such by performing the following operation. Let H be the graph obtained by replacing every edge $(u, v) \in E'$ with two edges (u, x) and (x, v) where x is a new vertex. Since a cycle cover must cover every vertex, we must have that both (u, x) and (x, v) are included in a cycle cover, since they're the only edges adjacent to x. Given a cycle cover for H, we can produce a cycle cover for G containing all the edges in E' by collapsing all the x's we've introduced.

5.4 Undirected Path

Let $\mathcal{P}_{\text{UP}}$ be the property that an undirected graph has a path from some fixed s to some fixed t. We will show that $\mathcal{P}_{\text{UP}}$ is k-extensible for every constant k but is not extensible in general. First, to show that $\mathcal{P}_{\text{UP}}$ is k-extensible for every constant k, we reduce $\mathcal{P}_{\text{UP}}$-k-CE to the Disjoint Paths problem, which is in **P** for constant k. (An $O(n^3)$ solution to the decision version of this problem was originally found by (Robertson and Seymour, 1995), but (Kawarabayashi et al., 2012) improves this to an $O(n^2)$ solution to the search problem, which we use),

Definition 10 (The Disjoint Paths Problem)**.** Given a graph G and pairs $(s_1, t_1), \dots, (s_k, t_k)$ of vertices of G, find vertex-disjoint paths $P_1, \dots, P_k$ such that P_i connects s_i and t_i.

Note that while (Robertson and Seymour, 1995) provides an $O(n^3)$ solution to the decision version of this problem, (Kawarabayashi et al., 2012) improves upon this by providing an $O(n^2)$ solution to the search version, which we use.

Given a set $E' = \{(u_1, v_1), \dots, (u_k, v_k)\}$, consider the various potential paths from s to t that intersect these edges. In particular, consider the possible orderings of vertices $u_1, v_1, \dots, u_k, v_k$ along the path. There are $k!$ permutations of intermediate edges, and for each e_i, we could either start at u_i or v_i. So, there are $k! * 2^k$ possible orderings of the vertices—since k is constant, this number of also constant. Given an ordering, say $(s, x_1, x_2, \dots, x_{2k}, t)$, where $x_1, \dots, x_{2k}$ is the ordering of the specified vertices, we can reduce to the problem of finding vertex disjoint paths between pairs $(s, x_1), (x_1, x_2), \dots, (x_2k, t)$. Note that the number of pairs is polynomial in k, and thus also constant. And since the number of orderings is also constant, $\mathcal{P}_{\text{UP}}$ is k-extensible for constant k.

Now we will show that $\mathcal{P}_{\text{UP}}$-CE is **NP**-hard via reduction from Hamiltonian Path, or HAMPATH. HAMPATH is the problem of determining if a given graph has a path that visits every vertex exactly once. We provide a poly-time function, f, that converts instances $\{G_1 = (V_1, E_1)\}$ of HAMPATH to instances $\{G_2 = (V_2, E_2), E' \subseteq E_2, s, t\}$ of $\mathcal{P}_{\text{UP}}$-CE. We then prove that $x \in$ HAMPATH if and only if $f(x) \in \mathcal{P}_{\text{UP}}$-CE. (We're being slightly imprecise here: we denote $f(x) \in$ iff $\mathcal{P}_{\text{UP}}$-CE$(f(x))$ returns some **T**).

f modifies the graph as follows. First, we add s, t to V_2. Then, for every vertex $v \in V_1$, we construct two vertices in V_2: v^a, v^b. We add the following edges to E_2: $(v^a, v^b), (s, v^a), (v^b, t)$. Then, for every $(u, v) \in E_1$, we add (u^b, v^a) to E_2. Finally, $E' = \{(v^a, v^b) : v \in V_1\}$. So the final instance for $\mathcal{P}_{\text{UP}}$-CE is $\{G_2 = (V_2, E_2), E', s, t\}$. This function is clearly computable in polynomial time.

Suppose $x \in$ HAMPATH, where $x = \{G_1 = (V_1, E_1)\}$. Let the Hamiltonian path be $v_{i_1}, \dots, v_{i_n}$, where $i_1, \dots, i_n$ is the order the vertices are visited in the path. Then, there is a path in G_2 between s and t that uses all vertices in E': $((s, v^a_{i_1}), (v^a_{i_1}, v^b_{i_1}), (v^b_{i_1}, v^a_{i_2}), \dots, (v^b_{i_n}, t))$. Note that there is an edge from s to any v^a_i and v^b_i to any t, so the first and last edges exist. Then, there is an edge from every v^a_i to v^b_i. Also, there is an edge from every $v^b_{i_j}$ to $v^a_{i_{j+1}}$, since $(v_{i_j}, v_{i_{j+1}}) \in E_1$. So, all edges in our path exist in E_2. Note also that since the Hamiltonian path doesn't repeat any vertices, this path doesn't repeat vertices. Finally, since every vertex is visited in the Hamiltonian path, all edges in E' are used in this path. So, $f(x) \in \mathcal{P}_{\text{UP}}$-CE.

Suppose $f(x) \in \mathcal{P}_{\text{UP}}$-CE, where $x = \{G_2 = (V_2, E_2), E', s, t\}$. This means that there exists a (simple) path between s, t that uses all (v^a_i, v^b_i). Note that such a path corresponds to a Hamiltonian path in the original graph: once any v^a_i, v^b_i are visited, they may not be visited again in the simple path. Further, since all (v^a_i, v^b_i) are in the path, all vertices are visited. Lastly, there exists $(v^b_i, v^a_j) \in E_2$ only when $(v_i, v_j) \in E_1$. Thus, $x \in$ HAMPATH.

Thus, HAMPATH $\leq_p \mathcal{P}_{\text{UP}}$-CE; and since HAMPATH is NP-Hard, so is $\mathcal{P}_{\text{UP}}$-CE.

6 Certificate Extensions and Matroids

Now that we've seen some examples of extendable properties, we try to form sufficient and necessary conditions for when a property is extendable. Since extending a certificate is at least as hard as finding a certificate, we can restrict our consideration to properties that are easily found. We use matroids as a means of classification.

We consider forming an independence system from $\mathcal{P}$ and input graph $G = (V, E)$ by setting the bases of the independence system to be the certificates of $\mathcal{P}$.
Theorem 6.1. *If $\mathcal{S}(\mathcal{P})$ is a matroid, then $\mathcal{P}$ is extensible.*

Proof. We are given as input $G = (V, E), E' \subseteq E$. We will also use $\mathcal{O}$ to denote the matroid's independence oracle. As we said earlier, we are considering properties $\mathcal{P}$ for which $\mathcal{P}$-CF is easy. So we first find any certificate $\mathbf{T}$ in G. Then, we loop over all elements e from $\mathbf{T} \setminus E'$ until $\mathcal{O}(E' + e)$ returns independent. Then, $E' \leftarrow E' + e$ and we repeat. Due to the exchange property, this process is guaranteed to terminate with E' equal to some basis of the matroid. By construction, the basis is a certificate. By the fact that we augmented E' using edges in T (which was in G), we have shown that $\mathcal{P}$ is extensible. $\square$

6.1 The Intersection of Two Matroids

Now consider the case where the certificate set $\mathcal{C}$ for a monotone graph property $\mathcal{P}$ corresponds to the maximum sets of a matroid intersection. Formally, we write $\mathcal{C} = \{T \in \mathcal{I}_1 \cap \mathcal{I}_2 \ : \ |T| = m\}$ where $(E, \mathcal{I}_1)$ and $(E, \mathcal{I}_2)$ are matroids and $m = \max\{|T'| \ : \ T' \in \mathcal{I}_1 \cap \mathcal{I}_2\}$. We'll show that if a property $\mathcal{P}$ is of this form, then it is extensible. As a subroutine, we will use the Weighted Matroid Intersection Algorithm

Chapter 13, Section 7 of (Korte and Vygen, 2007) gives us an algorithm for finding the largest independent set in a matroid intersection (where 'largest' is subject to a given weight function on elements of E). By choosing the weight function that weighs all elements in E' and $E \setminus E'$ equally relative to each other, but such that any element of E' is preferred over all elements in $E \setminus E'$, we're able to efficiently extend partial-certificates.

Theorem 6.2. *If $\mathcal{P}$ is a property whose certificate set is $\mathcal{C} = \{T \in \mathcal{I}_1 \cap \mathcal{I}_2 \ : \ |T| = m\}$ for some matroids $(E, \mathcal{I}_1)$ and $(E, \mathcal{I}_2)$ where $m = \max\{|T'| \ : \ T' \in \mathcal{I}_1 \cap \mathcal{I}_2\}$, then the $\mathcal{P}$-CE problem is in **P**.*

Proof. Suppose $\mathcal{P}$ is a property whose certificate set is $\mathcal{C} = \{T \in \mathcal{I}_1 \cap \mathcal{I}_2 \ : \ |T| = m\}$ where $(E, \mathcal{I}_1)$, $(E, \mathcal{I}_2)$, and $m = \max\{|T'| \ : \ T' \in \mathcal{I}_1 \cap \mathcal{I}_2\}$. Additionally, suppose we're given a graph $G = (V, E)$ and a set of edges E' and are asked to produce a certificate T such that $E' \subseteq T \subseteq E$ or to correctly assert that none exists. We'll produce such a T (if one exists) as follows.

Let $k = |E \setminus E'|$, $k' = |E'|$, and the weight function $w : E \to \mathbb{R}$ be defined as such:

$$w(e) = \begin{cases} 1 & e \in E \setminus E' \\ k+1 & e \in E' \end{cases}$$

Run the Weighted Matroid Intersection Algorithm on $(E, \mathcal{I}_1)$, $(E, \mathcal{I}_2)$, and w. Suppose it returns an edge set S. We calculate $\sum_{e \in S} w(e)$ and if $\sum_{e \in S} w(e) \geq k'(k+1)$ and $|S| = m$ we return S; otherwise, we assert that no such T exists.

Now we must show: 1) that if our algorithm returns a T then $E' \subseteq T \subseteq E$ and $T \in \mathcal{C}$ and 2) that if no such T exists then our does not return a T. Clearly T will be an element of $\mathcal{I}_1 \cap \mathcal{I}_2$, will have m elements, and will be subset of E. Now suppose that our algorithm returns T and E' is not a subset of T. Then

$$\sum_{e \in T} w(e) \leq (k'-1)(k+1) + k = k'k + k' - k - 1 + k = k'k + k' - 1 = k'(k+1) - 1$$

which contradicts the fact that our algorithm only returns sets of total weight at least $k'(k+1)$.

Now suppose that there exists some T such that $E' \subseteq T \subseteq E$, $T \in \mathcal{I}_1 \cap \mathcal{I}_2$, and $|T| = m$. Then $\sum_{e \in T} w(e) = k'(k+1) + m - k'$ so the Weighted Matroid Intersection Algorithm will find a S of total weight at least that of T and our algorithm is guaranteed to return something. □

7 Conclusion and Future Work

We started this work trying to understand which hidden graph properties are easily solvable. We first developed a transfer theorem, which led to the question of when certificate extension is easy. We tried to use matroids to characterize which properties are extensible. Through a combination of examples and theorems, we proved the following statements:

1. For all properties $\mathcal{P}$ if the independence system with bases corresponding to the certificates of $\mathcal{P}$ is a matroid, then $\mathcal{P}$ is extensible.
2. For all properties $\mathcal{P}$ if the certificates of $\mathcal{P}$ are the maximal independent sets of two matroids, then $\mathcal{P}$ is extensible. We call properties that satisfy this or the previous claim "matroidal" properties.
3. There exists a property, namely perfect patchings in general (non-bipartitite) graphs, that are extensible, yet not apparently matroidal. We know that bipartite perfect matching can be framed as the intersection of two matroids, but we conjecture that this isn't the case for general graphs. This illustrates the incomplete nature of our characterization.
4. There exists an apparently non-matroidal property (directed path) that is not extensible. Since we have no substantive necessary conditions, we aren't able to further characterize non-extensible properties.
5. There exists an apparently non-matroidal property (undirected path) that is not extensible in general, but is extensible for constant k. This is interesting because it is a concrete example of an non-contrived property that gets harder with larger k (where k is the size of the E' we seek to extend).

As these statements illustrate, our CE clasification is incomplete—we're missing additional sufficient conditions and any substantial necessary conditions. This is a clear area for future research.

Another area for future work is to explore more transfer theorems. We initially thought we could find a transfer theorem between the hidden problem with no Id constraints and the certificate extension with $k = 1$. One way of disproving this possibility would be to make the reductions in Theorem 3.1 tight (e.g. k-CE $\lneq_p$ H-k-Id $\lneq_p$ $(k+1)$-CE).

Lastly, our theorems in Section 6 relied on independence oracles for our given matroids. The natural question this poses is for which matroidal properties is the independence oracle easily to construct. For example, for connectivity, the independence oracle simply determines if a given set of edges contains a cycle. For all the matroidal properties we've considered, the independence oracle is easy to construct—so one natural line of research would be to prove this claim, or to classify which properties have simple independence oracles. This classification would be closely related to classifying which properties are extensible.

References

T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, Third Edition.* The MIT Press, 3rd edition, 2009. ISBN 0262033844, 9780262033848.

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

S. Fortune, J. Hopcroft, and J. Wyllie. The directed subgraph homeomorphism problem. *Theoretical Computer Science*, 10(2):111 – 121, 1980. ISSN 0304-3975. doi: https://doi.org/10.1016/0304-3975(80)90009-2. URL http://www.sciencedirect.com/science/article/pii/0304397580900092.

K. Kawarabayashi, Y. Kobayashi, and B. Reed. The disjoint paths problem in quadratic time. *Journal of Combinatorial Theory, Series B*, 102(2):424 – 435, 2012. ISSN 0095-8956. doi: https://doi.org/10.1016/j.jctb.2011.07.004. URL http://www.sciencedirect.com/science/article/pii/S0095895611000712.

B. Korte and J. Vygen. *Combinatorial Optimization: Theory and Algorithms*. Springer Publishing Company, Incorporated, 4th edition, 2007. ISBN 3540718435, 9783540718437.

S. Micali and V. V. Vazirani. An $\mathcal{O}(\sqrt{|V|}|e|)$ algoithm for finding maximum matching in general graphs. In *Foundations of Computer Science, 1980., 21st Annual Symposium on*, pages 17–27. IEEE, 1980.

C. A. Miller. Evasiveness of graph properties and topological fixed-point theorems. 2013. doi: 10.1561/0400000055.

N. Robertson and P. Seymour. Graph minors .xiii. the disjoint paths problem. *Journal of Combinatorial Theory, Series B*, 63(1):65 – 110, 1995. ISSN 0095-8956. doi: https://doi.org/10.1006/jctb.1995.1006. URL http://www.sciencedirect.com/science/article/pii/S0095895685710064.

A. Sundaram. *On Classical and Quantum Constraint Satisfaction Problems in the Trial and Error Model*. PhD thesis, National University of Singapore, 2017.