The Complexity of Ramsey Arrowing: A Computational Approach for Hardness Proofs

Zohair Raza Hassan
Rochester Institute of Technology,NY, USA

Abstract

In graph Ramsey theory, the arrowing operator is used to describe the appearance of unavoidable substructures within colored graphs; for graphs G, F, and H, we say $G \to (F, H)$ (read, G *arrows* F, H) if every red/blue coloring of G's edges contains a red F or a blue H. For fixed F and H, the (F, H)-Arrowing problem asks whether $G \to (F, H)$ for some given graph G. (F, H)-Arrowing has been shown to be in P or coNP-complete for different pairs (F, H). However, categorizing the complexity for all pairs still remains wide open.

In general, categorizing the complexity of problems whose nature depends on some underlying graph – or, in our case, pair of graphs – is a daunting task, and (F, H)-Arrowing is no exception. Hardness proofs typically rely on ad-hoc, laborious constructions of special graphs known as "gadgets." In this work, we present a simple, computational approach to find these gadgets for small (F, H)-Arrowing problems and show how these can be extended to other (F, H)-Arrowing problems.

Our main focus is on the simplest case for which the complexity remains uncategorized: $F = P_3$. We showcase the efficacy of our computational approach by presenting hardness proofs for (P_3, H)-Arrowing problems previously not known to be coNP-hard. Moreover, we show how to generalize hardness to other (P_3, H)-Arrowing problems by either: (1) carefully inspecting and modifying our found gadgets, or (2) coming up with intuitive constructions to reduce (P_3, H')-Arrowing to (P_3, H)-Arrowing, where H' is a subgraph of H. We also discuss how our methodology can be extended to work for other (F, H)-Arrowing problems by showing new results for $F = P_4$ and $K_{1,3}$.

We see our work as an important step towards categorizing the complexity of (F, H)-Arrowing for all pairs (F, H). (F, H)-Arrowing is thought to be hard when (F', H')-Arrowing is hard where F' and H' are subgraphs of F and H, respectively. Under this assumption, our results narrow down the only uncategorized family of problems for which the problem may lie in P. Beyond the complexity of (F, H)-Arrowing, we believe that the broader impact of our work is showing how computational methodologies can be adopted for proving hardness, and we hope our approach will be adopted to find gadgets for other open graph problems as well.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness

Keywords and phrases Graph Arrowing, Ramsey theory, Hardness reductions, Computational proofs

Digital Object Identifier 10.4230/LIPIcs.WG.2026.25

Supplementary Material *Software (Code)*: `https://github.com/zohair-raza/arrowing-gadget-hunting`, archived at `swh:1:dir:e407d828c2df99adf8a7f5fd3599bee24bb95018`

Funding This work was supported by NSF grant CCF-2421977.

Acknowledgements I want to thank my advisors Edith Hemaspaandra and Stanisław Radziszowski.

1 Introduction and Related Work

Since its inception in 1930, Ramsey theory has played an inimitable role in mathematics and computer science. Beyond its applications in fields such as number theory, topology, and cryptography, the pursuit for Ramsey numbers and their bounds has brought forth new ideas in mathematical proofs and combinatorial computing [20]. Yet, despite its long history and intrigue, the computational complexity of the arrowing operator, around which graph

Ramsey theory is based, remains an enigma. For graphs G, F, and H, we say $G \to (F, H)$ (read, G *arrows* F, H) if every red/blue coloring of G's edges contains a red F or a blue H. The corresponding decision problem is defined as follows.

▶ **Problem 1** ((F, H)-Arrowing). *Let F and H be fixed graphs. Given a graph G, does $G \to (F, H)$?*

The problem is clearly in coNP, as a coloring of G without any red F's and blue H's can be verified in polynomial time since F and H are fixed graphs[1]. For various pairs of graphs (F, H), (F, H)-Arrowing is known to be in P or coNP-complete. It is believed that a dichotomy theorem between these classes holds for (F, H)-Arrowing, but categorizing the complexity for all pairs remains an open problem. We note that such theorems have been proven in the past for many other graph problems, e.g., [1, 5, 9, 11, 18]. The existing literature on the coNP-hardness of (F, H)-Arrowing problems focuses on proving hardness for specific cases or graph families. For example, the earliest hardness result in the literature is on the hardness of (P_4, P_4)-Arrowing [21] – where P_n is the path on n vertices, shortly followed by Burr [3] who showed hardness for the case where both F and H are 3-connected – a graph is 3-connected if it remains connected after the deletion of any two vertices, e.g., K_5. More recently, dichotomy theorems (P or coNP-complete) were proven for the case where F and H are paths [8], and the case where F is a star and H is a 2-connected graph [7] – a graph that remains connected after the deletion of any one vertex.

A common theme among hardness results for (F, H)-Arrowing is the use of special graphs known as "gadgets" to reduce NP-complete variants of CNF-SAT to (F, H)-Nonarrowing, the complement of (F, H)-Arrowing that asks if a given graph has a (F, H)-good coloring, i.e., a red/blue coloring of its edges with no red F's and no blue H's. These gadgets are then combined to construct a graph G_ϕ such that G has an (F, H)-good coloring if and only if the CNF-SAT formula ϕ is satisfiable. Gadgets are used to force desired behavior in G_ϕ. For example, Burr uses gadgets known as "senders" that force two disjoint edges to have the same color in all (F, H)-good colorings [3, 4]. In Burr's reduction, this corresponds to ensuring two instances of a variable have the same truth value across the formula. The more recent results by Hassan et al. use variable and clause gadgets that mimic the behavior of variables and clauses in ϕ, respectively [7, 8]. Intuitively, these graphs simulate sending and receiving "truth signals" from variables to clauses, where clause gadgets simulate clause-like behavior by only allowing (F, H)-good colorings if and only if at least one received truth signal corresponds to true. This is explained more in depth in Section 3, where we discuss generalizing this approach.

The construction of gadgets and the proof of these reductions work by exploiting properties of "well-structured" graphs. For instance, both Burr [3] and Hassan [7] are able to exploit the fact that multiple copies of 2-connected graphs can be combined by identifying vertices without constructing new copies of said graph, allowing for easier analysis. Similarly, while this is not true for paths, Hassan, Hemaspaandra, and Radziszowski are able to use the structure of paths to allow for a simpler analysis of (F, H)-good colorings in their constructions [8]. Unfortunately, extending this approach to prove hardness for more graphs has two major challenges: (1) not all graphs/graph families have nice, exploitable structures that are easily amenable to gadget construction, and (2) discovering properties of graph families to construct gadgets and prove hardness is a difficult and laborious task. In this work, we propose a simple computational approach that finds variables and clause gadgets, defined similarly to

[1] Deciding $G \to (F, H)$ is Π_2^p-complete when F and H are also part of the input [22].

the gadgets by Hassan [7], to obtain new hardness results for (F, H)-Arrowing[2], allowing us to avoid having to find ad-hoc constructions for every case. We focus on the case where $F = P_3$ because of the lower computational overhead (since P_3 is a small graph), the fact that P_3 is the smallest F for which the complexity of (F, H)-Arrowing is an open problem, and because (P_3, H)-Arrowing has an interesting connection to matching removal problems, the complexity of which have also sparked interest in the past [8, 12, 13, 14]. However, we note that this approach can be generalized to other trees and demonstrate this by showing results for the case where $F = P_4$ and $F = K_{1,3}$.

A shared – but unproven – belief in the literature of (F, H)-Arrowing is that the complexity of the problem does not decrease for "larger" graphs. We state this formally below:

▶ **Conjecture 2.** *If (F', H')-Arrowing is* coNP*-complete, then so is (F, H)-Arrowing, where F' and H' are subgraphs of F and H, respectively.*

In light of this conjecture, we focus on proving the hardness for cases that are "close" to (F, H)-Arrowing problems that are known to be in P, allowing us to ascertain which open cases we can try to find polynomial-time algorithms for. We quickly summarize the cases where (F, H)-Arrowing is known to be in P: when F and H are stars [3], F is a collection of disjoint edges [3], and the cases where $(F, H) = (P_3, P_3)$, (P_3, P_4) [8], (P_3, K_3) [7], or (P_3, TK_3) [7] where TK_3 is a triangle with a leaf edge (also known as a "paw"). By focusing on cases where H is "a few edges away" from a star or a triangle, we were able narrow down the only family of (P_3, H)-Arrowing problems whose complexity is unknown under Conjecture 2 and may lie in P. Additionally, our results on (P_4, H)-Arrowing show that said family is the only uncategorized family of (F, H)-Arrowing problems that may lie in P.

Beyond the complexity of cases of (P_3, H)-Arrowing for specific H, we show how our results can be generalized to families of H by carefully inspecting and modifying the gadgets obtained by our methodology. Finally, we extend an idea by Hassan who, after using gadgets to prove the hardness of (P_3, H)-Arrowing, used constructions to extend these results to $(K_{1,n}, H)$-Arrowing for $n \geq 3$ where $K_{1,n}$ is the star graph on $n + 1$ vertices [7]. In a similar fashion, we show how to generalize hardness results by presenting reductions that reduce (P_3, H')-Arrowing to (P_3, H)-Arrowing, where H' is a subgraph of H, using simple and intuitive constructions. We summarize the contributions of our work below:

1. We propose a novel, yet simple, computational approach to find gadgets to prove the hardness of (F, H)-Arrowing, a problem that has eluded a complete complexity categorization for decades. Our approach is focused on the case where $F = P_3$ but can be generalized to any F where F is a tree. We illustrate this by our results for $F = P_4$ and $F = K_{1,3}$.
2. We showcase the efficacy of our approach by presenting four new hardness results for (F, H)-Arrowing using gadgets found computationally.
3. We show how the results obtained by our computational approach can be extended to prove the hardness for other (F, H)-Arrowing problems, by showing results for an infinite class of (P_3, H)-Arrowing problems.
4. Under the widely-believed assumption that (F, H)-Arrowing does not decrease in complexity as F and H become "larger," we narrow down the only open case for which the (F, H)-Arrowing problem may lie in P.

[2] This is certainly not the first time computational techniques have proven useful in Ramsey theory, as many results for Ramsey numbers are the product of computational graph generation. See [19].

The rest of our paper is organized as follows. Important notation and terminology is introduced in Section 2. We discuss the hardness reduction which we generalize in Section 3, and the computational approach to find gadgets for this reduction in Section 4. In Section 5 we discuss our new hardness results. We conclude in Section 6.

2 Notation and Terminology

All graphs discussed in this work are simple and undirected. $V(G)$ and $E(G)$ denote the vertex and edge set of a graph G, respectively. We denote an edge in $E(G)$ between $u, v \in V(G)$ as (u, v). The path, cycle, and complete graphs on n vertices are denoted as P_n, C_n, and K_n, respectively. $K_{1,n}$ is the star graph on $n+1$ vertices. Vertex identification is the process of replacing two vertices u and v with a new vertex w such that w is adjacent to all remaining neighbors $N(u) \cup N(v)$. A bistar $B_{m,n}$ is the graph on $n+m+2$ vertices obtained by connecting the center vertices of $K_{1,n}$ and $K_{1,m}$ by an edge. This edge is referred to as the center edge of the bistar. A multitailed triangle MTT_n is the graph on $n+3$ vertices obtained by identifying the center vertex of a $K_{1,n}$ with a vertex of a K_3.

A coloring of a graph will always refer to a red/blue coloring of its edges. We say that $G \to (F, H)$ if every coloring of G contains a red F or blue H. An (F, H)-good coloring of a graph G is a coloring of where the red subgraph is F-free, and the blue subgraph is H-free. When the context is clear, we will omit (F, H) and refer to the coloring as a good coloring. We say that G is (F, H)-good if it has at least one (F, H)-good coloring. Note that in this work we only consider the cases where F and H are connected graphs.

3 Hardness Reduction

In this section, we first present a coNP-hardness proof for (P_3, P_5)-Arrowing; we will reduce an NP-complete variant of SAT to (P_3, P_5)-Nonarrowing. Note that this problem was already proven to be hard by using gadgets discovered by-hand to reduce from a different SAT variant [8]. However, we believe it serves as a good pedagogical example to demonstrate the idea behind gadget-based hardness reductions for (F, H)-Arrowing before we generalize this approach to prove the hardness of other (P_3, H)-Arrowing problems. The gadgets presented were found using the methodology described in Section 4. We reduce from the following:

▶ **Problem 3** ((2,1)-≤3SAT [15]). *Let ϕ be a CNF formula where: (1) each clause contains at most three literals, (2) the variables in each clause are distinct, and (3) each variable appears exactly three times: twice unnegated and once negated. Does there exist a satisfying assignment for ϕ?*[3]

▶ **Theorem 4.** *(P_3, P_5)-Arrowing is* coNP*-complete.*

Proof. Let ϕ be an instance of $(2, 1)$-≤3SAT. We will construct a graph, G_ϕ, such that G_ϕ is (P_3, P_5)-good if and only if ϕ is satisfiable. To better explain our reduction, we find it easier to imagine the formula ϕ as a circuit, wherein truth signals are being sent from variables to clauses. We depict this visualization in Figure 4(a), wherein we also show an example of G_ϕ.

[3] It is important to note that clauses have "at most three" literals instead of exactly three literals; if clauses have exactly three literals and each variable occurs three times the formula ϕ is always satisfiable [23].

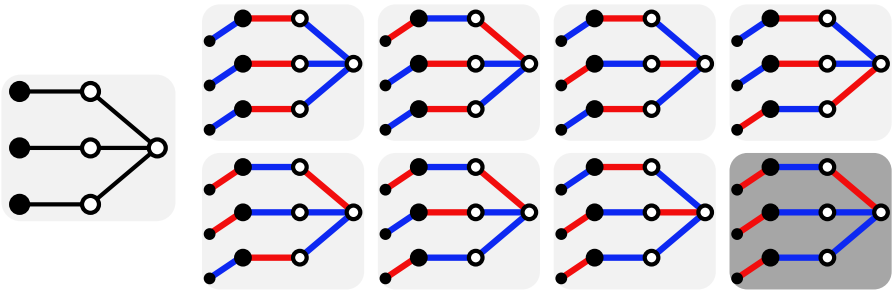

Figure 1 On the left, we show the clause gadget, CG, for (P_3, P_5)-Arrowing. The filled vertices are referred to as input vertices. On the right, we show that when the input vertices are adjacent to external edges, a (P_3, P_5)-good coloring is always possible unless all external edges are red.

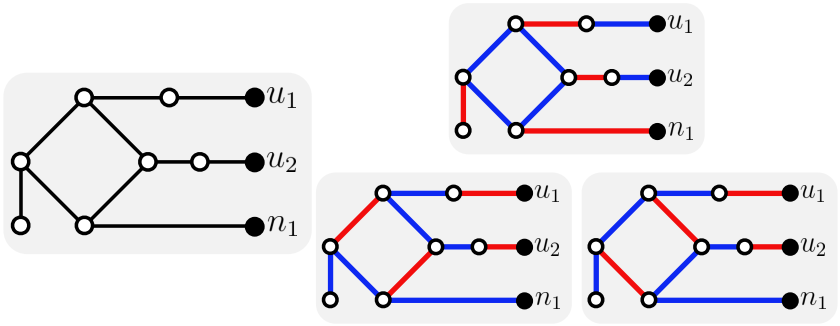

Figure 2 The variable gadget, VG, for (P_3, P_5)-Arrowing is shown on the left. The filled vertices are referred to as output vertices. Vertices marked u_1 and u_2 represent output vertices corresponding to unnegated output signals, and n_1 represents an output vertex corresponding to a negated output signal. On the right, we show all (P_3, P_5)-good colorings of VG. On the top, we have a (P_3, P_5)-good coloring of VG corresponding to the variable being set to true. Such colorings are referred to as true colorings. On the bottom, we have (P_3, P_5)-good colorings of VG corresponding to the variable being set to false. Such colorings are referred to as false colorings.

Let CG be the clause gadget shown in Figure 1. In Figure 1, we observe how CG has a good coloring if and only if the external edges that the three input vertices are adjacent to have at least one blue edge. This is reminiscent of an OR-gate which takes as input three "truth signals." Thus, in our reduction, a red edge is a false signal, and blue is true. Thus, just as a clause in ϕ is satisfied if and only if at least one of its input signals is true, a clause gadget in G_ϕ has a good coloring if and only if at least one of the incoming edges is blue.

Our variable gadget is a graph that simulates sending these truth signals to the clauses it belongs to. Let VG be the variable gadget shown in Figure 2. VG has three output vertices that emulate the role of sending a signal from a variable to a clause. The vertices labeled u_1 and u_2 correspond to unnegated signals, while n_1 corresponds to a negated signal. VG has two types of good colorings, that we refer to as "true colorings" and "false colorings," that correspond to whether the variable is set to true or false .

Finally, let EN be the enforcer gadget shown in Figure 3. Observe that the vertex labeled v is always the endpoint of a red edge. In our construction of G_ϕ, enforcers will be used to mimic false signals into clauses which use fewer than three literals. However, we later use enforcers for other purposes as well.

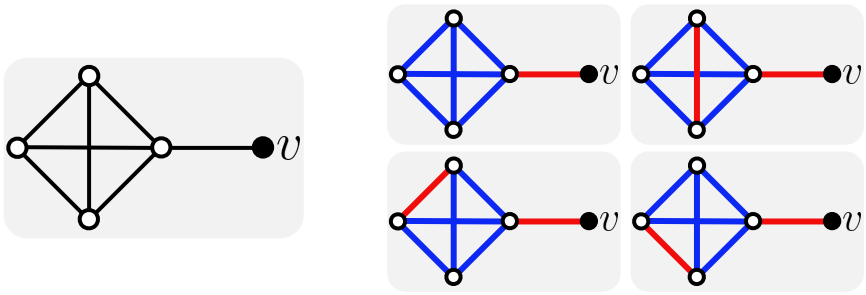

Figure 3 The enforcer gadget, EN, for (P_3, P_5)-Nonarrowing is shown on the left. The filled vertex, referred to as a signal vertex, is always the adjacent to a red edge in all (P_3, P_5)-good colorings, all of which are shown on the right.

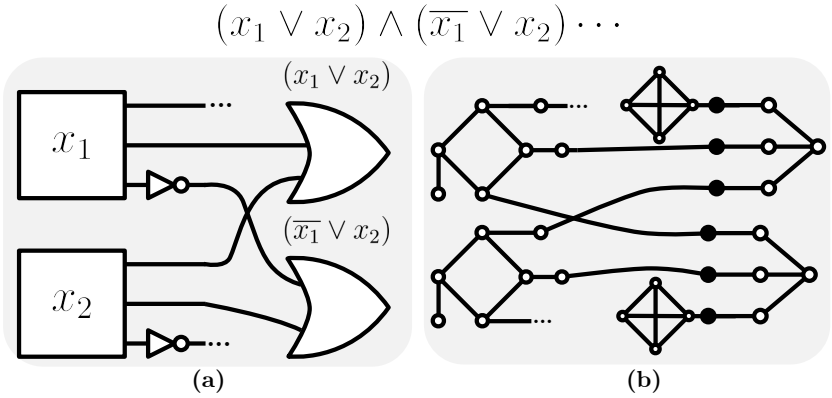

Figure 4 On the top, we show part of an instance ϕ of $(2,1)$-≤3SAT. In (a), we show a circuit view of ϕ. In (b), we show what G_ϕ looks like corresponding to the part of ϕ shown above. We use ellipses to signify that variable (resp., variable gadget) outputs are going elsewhere in the rest of the circuit (resp., graph), since every variable occurs exactly three times.

We now discuss how to construct G_ϕ. This is also shown in Figure 4(b). For each clause in ϕ, we add a copy of CG (Fig. 1) to G_ϕ. For each variable in ϕ, we add a copy of VG (Fig. 2) to G_ϕ. Suppose a variable x appears unnegated in clauses CL_1 and CL_2 and negated in clause CL_3. Then, u_1 (resp., u_2) from the VG that corresponds to x is identified with a previously unidentified input vertex of the CG that corresponds to CL_1 (resp., CL_2). n_1 is similarly identified with an input vertex of the CG that corresponds to CL_3. The correspondence between satisfying assignments of ϕ and good colorings of G_ϕ is clear. ◀

It is easy to generalize the reduction idea shown above for other (P_3, H)-Arrowing problems. For example, if we wanted to show the hardness of (P_3, C_5)-Nonarrowing via a reduction from $(2,1)$-≤3SAT, we would need a (P_3, C_5)-good variable gadget with two types of colorings: (1) true colorings, where unnegated output vertices are adjacent to red edges while the negated one is not, and (2) false colorings, where the negated output vertex is adjacent to a red edge while the unnegated ones are not. Similarly, we would need a clause gadget that is (P_3, C_5)-good unless the input vertices are adjacent to three red external edges. Finally, we need an enforcer gadget that has a vertex adjacent to a red edge in every (P_3, C_5)-good coloring. This idea extends easily because of the restrictive nature of red-P_3-free colorings. Before we move on, there is one important caveat we must mention

that we glossed over in our hardness reduction above: while it is easy to see how red P_3's are avoided in G_ϕ by coloring the gadgets according to a satisfying assignment, it is not as obvious that blue H's are avoided; there might exist a blue H in G_ϕ that goes across multiple gadgets. So, a complete reduction proof would also need to show that these "rogue" blue copies of H don't appear in G_ϕ when the gadgets are colored according to a satisfying assignment. This is seen easily in our proof for (P_3, P_5)-Nonarrowing above since whenever a blue edge is input to a clause gadget the adjacent edge within the clause gadget can be colored red to obtain a good coloring, thus completely avoiding the construction of a blue P_5 going from one gadget to another. However, for another H, say $H = C_5$, we would have to prove that there are no rogue blue copies of C_5 going across the gadgets in a coloring corresponding to a satisfying assignment.

Gadget Definitions and Proof Recipe

We now provide the formal definitions from the existing literature for the gadgets discussed above. Note that our definition for variable gadgets is slightly different from the one by Hassan [7] since he reduces from a different SAT variant; Hassan's variable gadget has four output vertices (with two negated outputs) whereas ours has three output vertices (with only one negated output). This was deliberate since it allows for faster computation of gadgets. Moreover, the variable gadget defined by Hassan is also a variable gadget that fits our definition below (Definition 6); we can just ignore one of the negated output vertices in Hassan's gadgets. This gives us a larger search space for our gadget search.

▶ **Definition 5** ([7]). *For a fixed H, a **clause gadget** is a (P_3, H)-good graph CG containing vertices i_1, i_2, and i_3 – referred to as **input vertices**, such that if vertices outside the gadget, o_1, o_2 and o_3, are connected to i_1, i_2, and i_3, respectively, then each of the eight possible combinations of (o_j, i_j)'s colors should allow a (P_3, H)-good coloring for CG, except the coloring where all (o_j, i_j)'s are red, which should not allow a good coloring of CG.*

▶ **Definition 6** ([7]). *For a fixed H, a **variable gadget** is a (P_3, H)-good graph VG containing three **output vertices**: two **unnegated output vertices**, u_1 and u_2, and one **negated output vertex**, n_1, such that:*

1. *In each (P_3, H)-good coloring of VG:*
   a. *If u_1 or u_2 is not adjacent to a red edge, then n_1 must be adjacent to a red edge.*
   b. *If n_1 is not adjacent to a red edge, then u_1 and u_2 must be adjacent to red edges.*
2. *There exists at least one (P_3, H)-good coloring of VG where both u_1 and u_2 are not adjacent to red edges. Such colorings will be referred to as "true colorings."*
3. *There exists at least one (P_3, H)-good coloring of VG where n_1 is not adjacent to a red edge. Such colorings will be referred to as "false colorings."*

Note that the definition of variables gadgets allows for colorings where u_1, u_2, and n_1 are all adjacent to red edges, but this does not affect the correctness of our reduction as such a coloring would correspond to a variable that does not satisfy any clause in the formula. It is also worth noting that unlike the reduction for proving the hardness of (P_3, P_5)-Arrowing, it is not necessary for output vertices to be leaf vertices. However, our reductions will often have output vertices that are leafs wherever applicable as they allow for simple proofs.

▶ **Definition 7** ([22]). *A graph G is called an **(F, H)-enforcer** with **signal vertex v** if it is (F, H)-good and the graph obtained from G by attaching a new edge to v has the property that this edge is colored blue in all (F, H)-good colorings.*

For a graph G, we will use the phrase **append an enforcer to $u \in V(G)$** to mean we will add an (F, H)-enforcer to G and identify its signal vertex with u. With the gadgets defined and the reduction idea presented above, we can formally state the following proposition, which we will refer to in our hardness proofs:

▶ **Proposition 8.** *(P_3, H)-Arrowing is* coNP*-hard if there exist variable, clause, and enforcer gadgets and every satisfying assignment of ϕ corresponds to a good coloring of G_ϕ.*

We prove the existence of gadgets using our computational methodology. Recall from our discussion after the proof of Theorem 4 that showing the existence of gadgets shows that any good coloring of G_ϕ corresponds to a satisfying assignment of ϕ, but it does not immediately give us the converse: that every satisfying assignment of ϕ has a corresponding good coloring. This is because we have to deal with the issue of rogue blue H's. Thus, showing that a good coloring exists for each satisfying assignment is proven by hand for each case, by focusing on showing that rogue blue H's do not appear when coloring the gadgets according to a satisfying assignment.

4 Finding Gadgets

In this section we present our computational methodology to find gadgets and provide details of our implementation. The code for our work is available in the supplementary material.

Let H be a fixed graph and $\mathcal{G}_H^n$ be the set of all connected (P_3, H)-good graphs on n vertices. We briefly explain how enforcers are found, and then present a more detailed algorithm to find variable and clause gadgets.

To find enforcers, we first generate $\mathcal{G}_H^n$ up to some fixed n (usually 10), using a simple iterative methodology, i.e., we generate $\mathcal{G}_H^n$ using $\mathcal{G}_H^{n-1}$. This generation process is explained in more detail below. Then, we check each graph generated to see if it has a vertex that is adjacent to a red edge in all of its good colorings. A similar approach is used to find variable and clause gadgets, but we also use enforcers to increase our search space for possible variable and clause gadgets. In Section 4.1 we explain how good colorings of graphs were obtained. In Section 4.2, we discuss how we generated good graphs iteratively. Then, finding variable and clause gadgets is discussed in Section 4.3. Our implementation details are provided in Section 4.4.

4.1 Obtaining Good Colorings

To test whether a graph is (P_3, H)-good we used a SAT solver: for a graph G, we made a formula ψ_G over the variables $\{r_e \mid e \in E(G)\}$:

$$\psi_G = \bigvee_{e_1, e_2 \in E(G) \text{ form a } P_3} (\overline{r_{e_1}} \vee \overline{r_{e_2}}) \wedge \bigvee_{e_1, e_2, \ldots, e_k \in E(G) \text{ form } H} (r_{e_1} \vee r_{e_2} \vee \cdots \vee r_{e_k})$$

Using the correspondence that r_e is true if and only if e is red, it is easy to see that any satisfying assignment of ψ_G forces at least one edge in every P_3 to be blue and at least one edge in every H to be red. We opt for SAT solvers since the same approach can be easily used to find colorings for other (F, H)-Arrowing problems in the future.

4.2 Generating Good Graphs

Note that for any H on at least four vertices, we have $\mathcal{G}_H^3 = \{P_3, K_3\}$. Thus, we iteratively generated (P_3, H)-good graphs starting from $n = 3$ vertices. For a graph G on $n - 1$ vertices, let X_G be the set of one-vertex-extensions of G, i.e., connected graphs on n vertices that have

Algorithm 1 Generate $\mathcal{G}_H^n$ given $\mathcal{G}_H^{n-1}$.

```
Let Z = ∅
Generate ALLX_G = ⋃_{G∈G_H^{n-1}} X_G
for G ∈ ALLX_G do
    if G has a connected subgraph G' on n − 1 vertices and G' ∉ G_H^{n-1} then
        continue
    end if
    Construct and solve ψ_G
    if ψ_G is satisfiable then
        Add G to Z
    end if
end for
return Z
```

G as a subgraph. Given the graphs in $\mathcal{G}_H^{n-1}$, we generate the graphs in $\mathcal{G}_H^n$ by generating the set $\bigcup_{G\in\mathcal{G}_H^{n-1}} X_G$ and removing any graph from it that is not (P_3, H)-good. We used nauty, a tool that canonically labels graphs and encodes them as strings [17], to generate the set of one-vertex-extensions and remove any isomorphs. To speed up the process for rejecting graphs on n vertices that arrow (P_3, H), we used the graphs in $\mathcal{G}_H^{n-1}$. For a graph G, we know that $G \to (F, H)$ if some subgraph $G' \to (F, H)$. Thus, we can reject any graph G that has connected subgraph on $n-1$ vertices G' such that $G' \notin \mathcal{G}_H^{n-1}$. We were able to implement this efficiently by using nauty's canonical string labeling and a dictionary (also known as a hash table): we built a dictionary for graphs in $\mathcal{G}_H^{n-1}$ using nauty's strings, and filtered out a graph if it had a connected subgraph on $n-1$ vertices whose canonical string was not found in said dictionary. This is summarized in Algorithm 1.

4.3 Searching for Variable and Clause Gadgets

Let G be a graph we are checking to see if it is a variable or clause gadget. We can proceed like so. First, obtain all good colorings of G by using a SAT solver to obtain all satisfying assignments to ψ_G. Then, select each 3-tuple of vertices and check if the colorings adhere to Definition 6 or Definition 5 above. However, we take an extra step to increase our search space. Recall that by appending a (P_3, H)-enforcer to a vertex v of a graph G, we force all edges in $E(G)$ adjacent to v to be blue. Let G be a graph we are checking to see if it is a variable or clause gadget. Suppose we choose to append k enforcers to k vertices of $V_k \subseteq V(G)$. We can restrict ourselves to good colorings of G where each vertex in V_k is adjacent only to blue edges and check if there exists some 3-tuple of vertices in $V(G) - V_k$ adhering to the definitions of a variable or clause gadget. We repeated this process for every graph and for every k-combination of vertices. Note that this increases our search space vastly, as it allows us to look at graphs that would have otherwise been much larger, on account of the overhead gained in the number of vertices by including these enforcers. Our approach to finding a gadget is summarized in Algorithm 2, wherein we return all possible combinations of the input/out vertices and enforced vertices if they exist.

We note that if multiple gadgets were found, we chose the one we believed was the easiest to work with for our proofs. For example, the variable gadget selected in Figure 2 was selected because the output vertices were always endpoints of red/blue P_2's, making it easier to show that no rogue blue P_5's were formed when combining gadgets.

Algorithm 2 Obtain all combinations, if any, for which G is a variable or clause gadget.

```
Let Z = ∅
Construct ψ_G and generate all satisfying assignments
Generate all (P_3, H)-good colorings of G, COL using ψ_G.
for k ∈ {0, 1, 2, ..., |V(G)| − 3} do
    for each k-combination of vertices V_k ⊆ V(G) do
        Let NEWCOL be all colorings in COL where every vertex in V_k is adjacent only
        to blue edges
        for each 3-tuple a, b, c ∈ V(G) − V_k do
            Using the colorings in NEWCOL, check if G is a variable (resp., clause) gadget
            with output (resp., input) vertices a, b, c, and add (G, V_k, a, b, c) to Z accordingly
        end for
    end for
end for
return Z
```

4.4 Implementation Details

Our methodology was implemented in Python, where we used we used PySAT's [10] implementation for the Glucose SAT solver [2], PyNauty[4] to use nauty's string labeling in Python, NetworkX [6] to interact with graphs, and Grand-Iso for finding subgraphs [16]. Our gadgets were computed on a desktop computer (Intel Xeon W2145 3.70GHz, 32GB RAM).

5 Results

In this section we discuss our new hardness results for (F, H)-Arrowing.

5.1 New Results for Small H

Our focus was to prove the hardness of (P_3, H)-Arrowing problems where H does not belong to a "well-structured" family of graphs (e.g., paths, complete graphs, 2-connected graphs), since this lack of structure makes them more difficult to work with. Furthermore, we focused on problems that are "close" to known polynomial-time cases – e.g, (P_3, P_4)-, (P_3, K_3)-Arrowing, to get a better understanding of the line that separates (F, H)-Arrowing problems that are in P from those that are coNP-complete. We used our methodology to find gadgets for (P_3, H)-Arrowing for $H = B_{2,2}$ and $H = MTT_2$, which we present below. Note that we are able to find gadgets for many other cases, but present only these two cases for brevity, and because they help us narrow down the cases that may lie in P under Conjecture 2. Our proofs follow a similar recipe: we present the variable, clause, and enforcer gadgets in a figure and then argue that every satisfying assignment of ϕ corresponds to a good coloring of G_ϕ. In our figures, when a (P_3, H)-enforcer is used in a variable or clause gadget, it is depicted as a circle labeled EN that has edges going to the vertex in VG or CG that the enforcer is appended to. As per the definition of an enforcer, one of these edges is always red, and this is illustrated in our figures. Note that in most of our proofs the enforcer's signal vertex is a leaf vertex, so, often, there is only one edge going from EN to VG or CG.

[4] `https://github.com/pdobsan/pynauty`

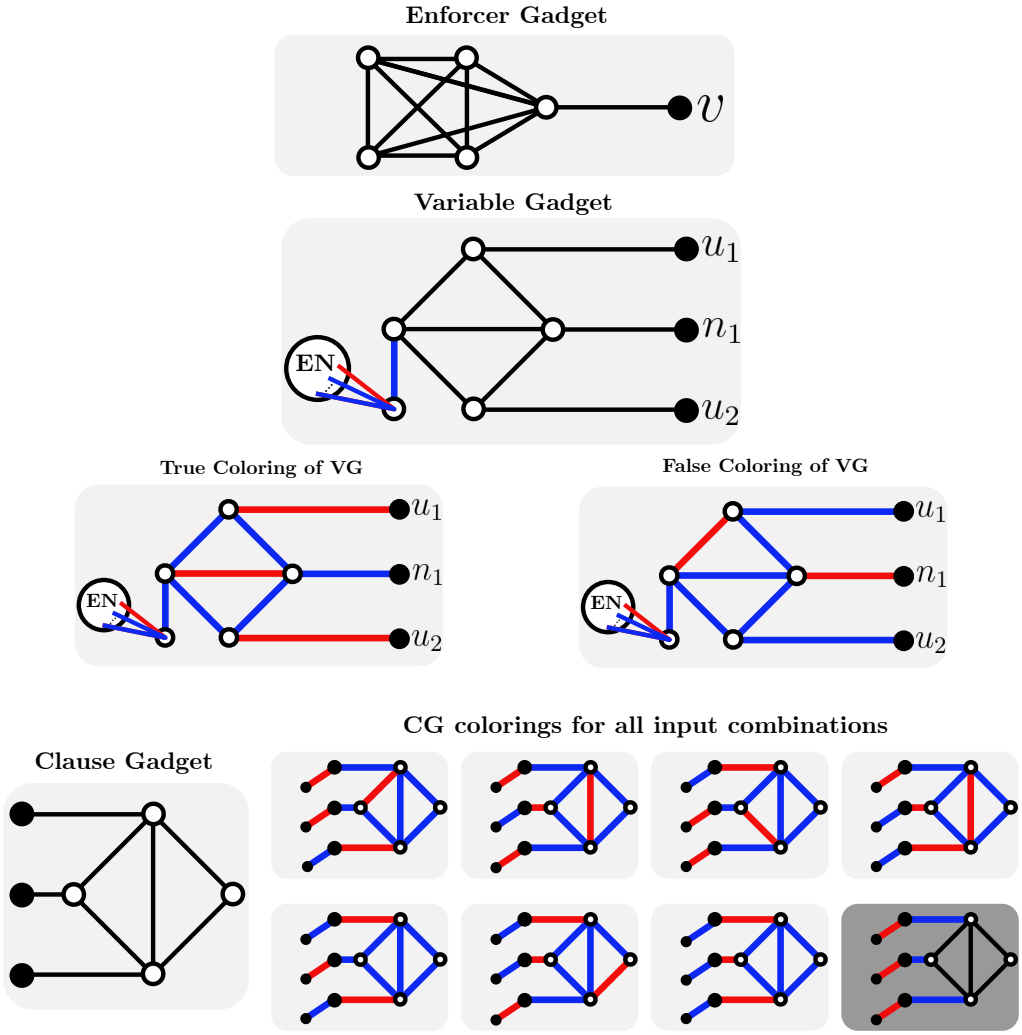

Figure 5 The enforcer, variable, and clause gadgets for $(P_3, B_{2,2})$-Arrowing are shown. The variable gadget we found has an enforcer gadget appended to a vertex that is depicted using a circle labeled EN, and the edge that it is adjacent to is already colored blue. We show a true and false coloring for VG. Note that there are more colorings, but all of them adhere to the constraints presented in Definition 6. Next to the clause gadget, we show that each possible input combination to CG permits a good coloring, except the case where the input is three red edges.

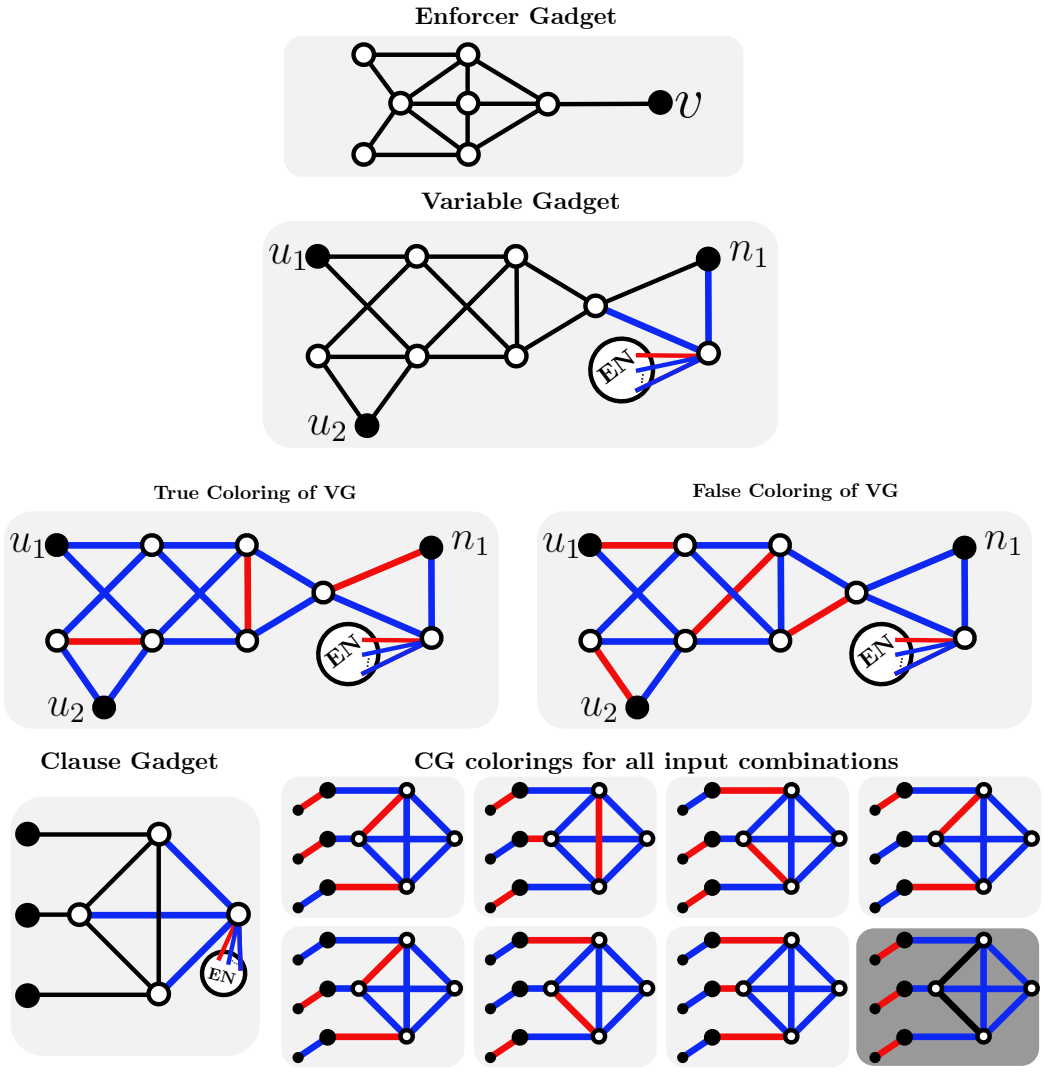

Figure 6 The enforcer, variable, and clause gadgets for (P_3, MTT_2)-Arrowing are shown. Enforcer gadgets connected to variable and clause gadgets are depicted as in Figure 5. We show a true and false coloring for VG. Note that there are more colorings, but all of them adhere to the constraints presented in Definition 6. Next to the clause gadget, we show that each possible input combination to CG permits a good coloring, except the case where the input is three red edges. To avoid clutter, the enforcers are not shown in these colorings.

▶ **Theorem 9.** $(P_3, B_{2,2})$*-Arrowing is* coNP*-complete.*

Proof. Using Proposition 8, it is enough to show the existence of gadgets (see Figure 5) and a proof that every satisfying assignment has a corresponding good coloring. It is clear that no red P_3's are formed when the gadgets are colored according to a satisfying assignment. We now show that there are no rogue blue $B_{2,2}$'s when the variable gadgets are colored according to the true and false colorings shown in Figure 5, and the clause gadgets are colored according to the good colorings shown in Figure 5. There are no blue $B_{2,2}$'s formed between the enforcers and variable/clause gadgets since the signal vertex is a leaf vertex and the edge going from EN to VG or CG is red in all good colorings. To see that no blue $B_{2,2}$'s are formed between variable and clause gadgets, observe from the colorings of CG in our figure that whenever a blue edge goes from a VG to a CG the edge adjacent to the input vertex of the CG can always be colored red, allowing us to avoid having any blue $B_{2,2}$'s. ◀

▶ **Theorem 10.** (P_3, MTT_2)*-Arrowing is* coNP*-complete.*

Proof. Using Proposition 8, it is enough to show the existence of gadgets (see Figure 6) and a proof that every satisfying assignment has a corresponding good coloring. It is clear that no red P_3's are formed when the gadgets are colored according to a satisfying assignment. We now show that there are no rogue blue MTT_2's when the variable gadgets are colored according to the true and false colorings, and the clause gadgets are colored according to the good colorings shown in Figure 6. Observe that when we identify the vertices of two gadgets, no new K_3 can be formed. Thus, any new MTT_2 formed while constructing G_ϕ must either: (1) have the K_3 in one gadget, and the two tails in the other gadget, or (2) have the K_3 and one tail in one gadget, and the other tail in the other gadget. Thus, this is only possible when the identified vertex has degree at least four. Observe that this is only possible when identifying the enforcer gadget and clause gadget. However, since the edge from the enforcer gadget is always red, this MTT_2 is not blue. ◀

5.2 Generalizing Results

In this section, we present generalized results for the hardness of (P_3, H)-Arrowing by modifying the gadgets obtained from our computational methodology and presenting constructions that reduce problems (P_3, H')-Arrowing to (P_3, H)-Arrowing, where H' is a subgraph of H.

5.2.1 Modifying Gadgets

After exploring gadgets for $(P_3, B_{2,3})$-Arrowing and comparing them to the ones found for $(P_3, B_{2,2})$-Arrowing, we were able to find a pattern we could generalize to construct gadgets for $(P_3, B_{2,m})$-Arrowing for any $m \geq 3$. We first state the following lemma for all $B_{n,m}$'s, which will also be useful later. The proof of this lemma is deferred to the appendix.

▶ **Lemma 11.** *For $n, m \geq 2$, there exists a $(P_3, B_{n,m})$-enforcer where the signal vertex is a leaf vertex.*

▶ **Theorem 12.** $(P_3, B_{2,m})$*-Arrowing is* coNP*-complete for all $m \geq 3$.*

Proof. Using Proposition 8, it is enough to show the existence of gadgets (see Figure 7 for variable and clause gadgets, and Lemma 11 for the enforcer gadget) and a proof that every satisfying assignment has a corresponding good coloring. We first show that the variable gadget adheres to the definition given in Definition 6. It is easy to verify that the true and false colorings shown are $(P_3, B_{2,m})$-good. However, we must also show that if u_1 or u_2 is

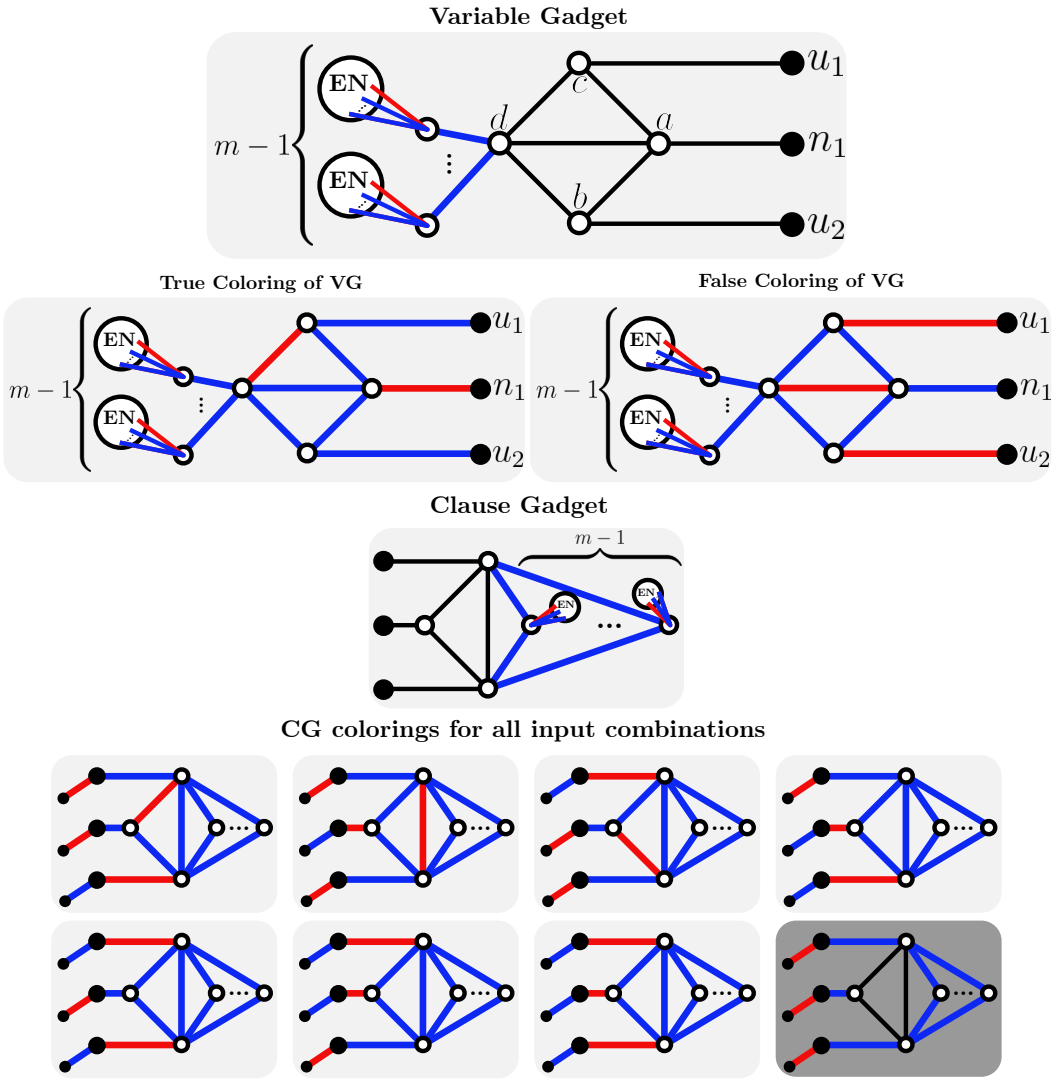

Figure 7 The variable, and clause gadgets for $(P_3, B_{2,m})$-Arrowing are shown. Enforcer gadgets attached to variable and clause gadgets are depicted using a circle labeled EN, and the edges they are adjacent to are already colored blue. We show a true and false coloring for VG. Below the clause gadget, we show that each possible input combination to CG, except the case where the input is three red edges, a good coloring of CG is possible. To avoid clutter, the enforcers are not shown in these colorings.

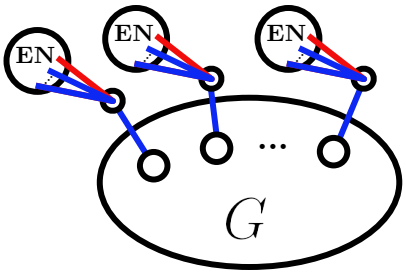

Figure 8 The graph G' discussed in the proof of Theorem 13.

blue, n_1 must be false, and vice verse – note this step was handled in the earlier proofs via computation. Due to symmetry, it is sufficient to show that no $(P_3, B_{2,m})$-good coloring of VG exists when (u_1, c) and (n_1, a) are both blue. We prove via contradiction. Suppose (u_1, c) and (n_1, a) are blue in a $(P_3, B_{2,m})$-good coloring of VG. Note that coloring (a, d) red forces (c, d), (a, c), and (b, d) to be blue, forming a blue $B_{2,m}$ with center edge (c, d). Thus, (a, d) must be blue in any good coloring. We can similarly show that (a, c) and (b, d) must be blue, but this gives us a blue $B_{2,m}$ with (a, d) as a center edge. Thus, the graph is a valid variable gadget. For the clause gadget, we can verify that the colorings shown for each valid input combination are good colorings. To see that the clause gadget does not have a good coloring when three red edges are input, observe that at least two of the edges in the uncolored K_3 must be blue, and this always leads to a blue $B_{2,m}$.

To show that there are no rogue blue $B_{2,m}$'s going across gadgets, a similar argument as the one presented in the proof for Theorem 9 holds. There are no blue $B_{2,m}$'s formed between the enforcers and variable/clause gadgets since the signal vertex is a leaf vertex and the edge going from EN to VG or CG is red in all good colorings. No blue $B_{2,2}$'s are formed between variable and clause gadgets since whenever a blue edge goes from a VG to a CG the edge adjacent to the input vertex of the CG can always be colored red. ◀

5.2.2 Extending the Hardness of Arrowing via Construction

We propose an approach similar to the one used to extend the hardness of (P_3, H)-Arrowing to $(K_{1,n}, H)$-Arrowing [7] to prove the hardness of $(P_3, B_{n,m})$-Arrowing by extending the hardness results from $(P_3, B_{2,m})$-Arrowing.

▶ **Theorem 13.** *$(P_3, B_{n,m})$-Arrowing is* coNP*-complete for all $n, m \geq 3$.*

Proof. Recall that $(P_3, B_{2,m})$-Arrowing is coNP-complete for all $m \geq 2$. We reduce $(P_3, B_{n,m})$-Arrowing to $(P_3, B_{n+1,m+1})$-Arrowing like so. Given a graph G, we construct G' by attaching a leaf edge to each vertex of G and appending a $(P_3, B_{n+1,m+1})$-enforcer to each new leaf vertex. This is depicted in Figure 8. It is clear that $G \to (P_3, B_{n,m})$ if and only if $G' \to (P_3, B_{n+1,m+1})$. ◀

5.3 Beyond P_3

Our reduction idea makes use of the fact that in any (P_3, H)-good coloring, vertices adjacent to red edges must be blue. Thus, when the output vertices of our variable gadgets are adjacent to a red edge within itself, they send a "false signal" to clause gadgets by forcing edges in the clause gadget to be colored blue. This can be extended to any tree by modifying our variable gadget definition so that false signals correspond to forcing external edges blue.

For example, when $F = P_4$, we must have that the output vertices are endpoints of red P_3's when they are supposed to be sending false signals. To demonstrate this, we implement our methodology above for $F = P_4$ and $F = K_{1,3}$, *mutatis mutandis*, and present the following two results in the appendix:

▶ **Theorem 14.** *(P_4, K_3)-Arrowing is* coNP*-complete.*

▶ **Theorem 15.** *$(K_{1,3}, P_4)$-Arrowing is* coNP*-complete.*

Note that with these results we can make the following claim, the proof of which we defer to the appendix:

▶ **Proposition 16.** *$(P_3, B_{1,m})$-Arrowing for $m \geq 2$ is the only family of (F, H)-Arrowing problems whose complexity is unknown assuming that Conjecture 2 is true.*

We note that we were unable to find variable gadgets for $(P_3, B_{1,m})$-Arrowing for any $m \in \{2, 3, 4\}$ up to 10 vertices.

6 Conclusion and Future Work

In this work we presented a novel computational framework to generate hardness gadgets for (F, H)-Arrowing and demonstrated its efficacy for $F \in \{P_3, P_4, K_{1,3}\}$. We presented four hardness proofs using these computationally generated gadgets, and even showed how these can be used to prove more generalized hardness results.

Our ultimate goal is to categorize the complexity of all (F, H)-Arrowing problems. Moving forward, we plan on focusing on $(P_3, B_{1,n})$-Arrowing, since it has eluded all attempts at hardness reductions or polynomial-time algorithms so far. For hardness of the remaining cases, we plan on identifying more constructions to reduce (F', H')-Arrowing to (F, H)-Arrowing where F' and H' are subgraphs of F and H. We believe that enough of these constructions will guide us towards a set of "base cases" that we can target to prove the hardness of, and the remaining hardness results will follow from these base cases. We also plan on implementing a similar methodology to find senders to prove the hardness of cases where F and H are not trees, in a similar fashion as in [3].

References

1. D. Achlioptas. The Complexity of G-Free Colorability. *Discrete Math.*, 165-166:21–30, 1997. doi:10.1016/S0012-365X(97)84217-3.
2. G. Audemard and L. Simon. On the Glucose SAT Solver. *Int. J. Artif. Intell. Tools*, 27(1):1840001:1–1840001:25, 2018. doi:10.1142/S0218213018400018.
3. S. A. Burr. On the Computational Complexity of Ramsey-Type Problems. *Math. Ramsey Theory Algorithms Combin.*, 5:46–52, 1990.
4. S. A. Burr, P. Erdős, and L. Lovász. On Graphs of Ramsey Type. *Ars Comb.*, 1(1):167–190, 1976.
5. S. Fortune, J. Hopcroft, and J. Wyllie. The Directed Subgraph Homeomorphism Problem. *Theor. Comput. Sci.*, 10:111–121, 1980. doi:10.1016/0304-3975(80)90009-2.
6. A. A. Hagberg, D. A. Schult, and P. J. Swart. Exploring Network Structure, Dynamics, and Function using NetworkX. In *SciPy 2008*, pages 11–15, 2008.
7. Z. R. Hassan. The Complexity of (P_3, H)-Arrowing and Beyond. In *MFCS 2024*, volume 306, pages 59:1–59:16, 2024.
8. Z. R. Hassan, E. Hemaspaandra, and S. Radziszowski. The Complexity of (P_k, P_ℓ)-Arrowing. *J. Comput. Syst. Sci*, 156:103726, 2026.

9 P. Hell and J. Nešetřil. On the Complexity of H-Coloring. *J. Comb. Theory, B*, 48:92–110, 1990. doi:10.1016/0095-8956(90)90132-J.

10 A. Ignatiev, Z. L. Tan, and C. Karamanos. Towards Universally Accessible SAT Technology. In *SAT*, pages 4:1–4:11, 2024.

11 H. Le and V. Le. Complexity of the Cluster Vertex Deletion Problem on H-Free Graphs. In *MFCS 2022*, volume 241, pages 68:1–68:10, 2022. doi:10.4230/LIPIcs.MFCS.2022.68.

12 C.V.G.C. Lima, D. Rautenbach, U.S. Souza, and J.L. Szwarcfiter. Decycling with a Matching. *Inf. Process. Lett.*, 124:26–29, 2017. doi:10.1016/J.IPL.2017.04.003.

13 C.V.G.C. Lima, D. Rautenbach, U.S. Souza, and J.L. Szwarcfiter. Bipartizing with a Matching. In *COCOA 2018*, pages 198–213, 2018.

14 C.V.G.C. Lima, D. Rautenbach, U.S. Souza, and J.L. Szwarcfiter. On the Computational Complexity of the Bipartizing Matching Problem. *Ann. Oper. Res.*, 316(2):1235–1256, 2022. doi:10.1007/S10479-021-03966-9.

15 J. Manuch and D. R. Gaur. Fitting Protein Chains to Cubic Lattice is NP-Complete. *J. Bioinform. Comput. Biol.*, 6(1):93–106, 2008. doi:10.1142/S0219720008003308.

16 J. K. Matelsky, E. P. Reilly, E. C. Johnson, J. Stiso, D. S. Bassett, B. A. Wester, and W. Gray-Roncal. DotMotif: an open-source tool for connectome subgraph isomorphism search and graph queries. *Sci. Rep.*, 11(1), June 2021.

17 B. D. McKay and A. Piperno. Practical Graph Isomorphism, II. *J. Symb. Comput.*, 60:94–112, 2014. doi:10.1016/J.JSC.2013.09.003.

18 J. J. Oostveen, D. Paulusma, and E. J. van Leeuwen. The Complexity of Diameter on H-Free Graphs. *SIAM J. Discret. Math.*, 39(2):1213–1245, 2025.

19 S. Radziszowski. Small Ramsey Numbers. *Electron. J. Comb.*, DS1:1–116, June 2024. URL: https://www.combinatorics.org/.

20 V. Rosta. Ramsey Theory Applications. *Electron. J. Comb.*, DS13:1–43, December 2004. URL: https://www.combinatorics.org/.

21 V. Rutenburg. Complexity of Generalized Graph Coloring. In *MFCS 1986*, volume 233, pages 573–581. Springer, 1986. doi:10.1007/BFB0016284.

22 M. Schaefer. Graph Ramsey Theory and the Polynomial Hierarchy. *J. Comput. Syst. Sci*, 62:290–322, 2001. doi:10.1006/JCSS.2000.1729.

23 C. A. Tovey. A simplified NP-complete satisfiability problem. *Discret. Appl. Math.*, 8(1):85–89, 1984. doi:10.1016/0166-218X(84)90081-7.

A Proof of Lemma 11

▶ **Lemma 11.** *For $n, m \geq 2$, there exists a $(P_3, B_{n,m})$-enforcer where the signal vertex is a leaf vertex.*

Proof. Note that in any $(P_3, B_{n,m})$-good coloring of a bistar $B_{n+1,m+1}$, the center edge must be red. Thus, all other edges are blue. This gives us a graph that with a vertex that is the endpoint of a blue $K_{1,n}$ in every $(P_3, B_{n,m})$-good coloring. We can combine these blue $K_{1,n}$'s as shown in Figure 9. Observe that if (u, v) was blue, it would form bistar with center edge (u, w_i) for some w_i, so it must be red in all good colorings. ◀

B Proof of Proposition 16

Recall Conjecture 2:

▶ **Conjecture 2.** *If (F', H')-Arrowing is* coNP*-complete, then so is (F, H)-Arrowing, where F' and H' are subgraphs of F and H, respectively.*

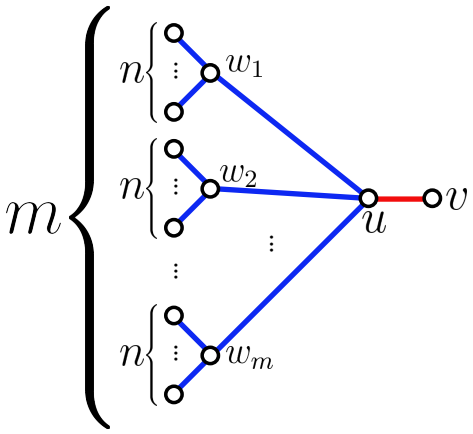

Figure 9 Construction for $(P_3, B_{n,m})$-enforcers.

▶ **Proposition 16.** *$(P_3, B_{1,m})$-Arrowing for $m \geq 2$ is the only family of (F, H)-Arrowing problems whose complexity is unknown assuming that Conjecture 2 is true.*

Proof. We first show for any F larger than P_3, we have a complete categorization under Conjecture 2. It is sufficient to consider the cases $F \in \{P_4, K_3, K_{1,3}\}$. Consider the case where $F = P_4$. We know that (F, H)-Arrowing is coNP-complete when $H = K_{1,3}$, $H = K_3$, and $H = P_4$ [21, 8]. The only connected graphs that are $K_{1,3}$-free, K_3-free, and P_4-free, are P_3 and P_2, and (P_4, P_3)-Arrowing is known to be in P [8] and (P_4, P_2)-Arrowing is trivially in P. When $F = K_3$, we know that (F, H)-Arrowing is coNP-complete when $H = P_4$, $H = K_{1,3}$ [7], and $H = K_3$ [3]. The only connected graphs that are $K_{1,3}$-free, K_3-free, and P_4-free, are P_3 and P_2, and (K_3, P_3)-Arrowing is known to be in P [7] and (K_3, P_2)-Arrowing is trivially in P. Now, consider the case where $F = K_{1,3}$. We know that (F, H)-Arrowing is coNP-complete when $H = P_4$, $H = K_3$ [7]. The only connected graphs that are K_3-free and P_4-free, are the stars $K_{1,n}$, and $(K_{1,3}, K_{1,n})$-Arrowing is known to be in P for all $n \geq 1$ [3]. Finally, consider the case where $F = P_3$. We know from our results above that (F, H)-Arrowing is coNP-complete when $H = B_{2,2}$, $H = MTT_2$, and $H = P_5$. We also know that the problem is hard when $H = C_4$ [7]. The only graphs that are free of all these H are the stars $K_{1,n}$, bistars $B_{1,m}$, and $MTT_1 = TK_3$. We know that (P_3, H)-Arrowing is in P for all $H = K_{1,n}$ [3] and $H = MTT_1$ [7]. Thus, $(P_3, B_{1,m})$-Arrowing is the only family remaining. ◀

C Proofs of Theorems 14 and 15

▶ **Theorem 14.** *(P_4, K_3)-Arrowing is* coNP*-complete.*

Proof. It is enough to show the existence of gadgets (see Figure 10) and a proof that every satisfying assignment has a corresponding good coloring. It is clear that no red P_4's are formed when the gadgets are colored according to a satisfying assignment due to the colorings shown in the figure. No rogue blue K_3's are formed because identifying vertices to construct G_ϕ can not form any new K_3's. ◀

▶ **Theorem 15.** *$(K_{1,3}, P_4)$-Arrowing is* coNP*-complete.*

Proof. It is enough to show the existence of gadgets (see Figure 11) and a proof that every satisfying assignment has a corresponding good coloring. It is clear that no red P_4's are formed when the gadgets are colored according to a satisfying assignment due to the colorings shown in the figure. No rogue blue K_3's are formed because identifying vertices to construct G_ϕ can not form any new K_3's. ◀

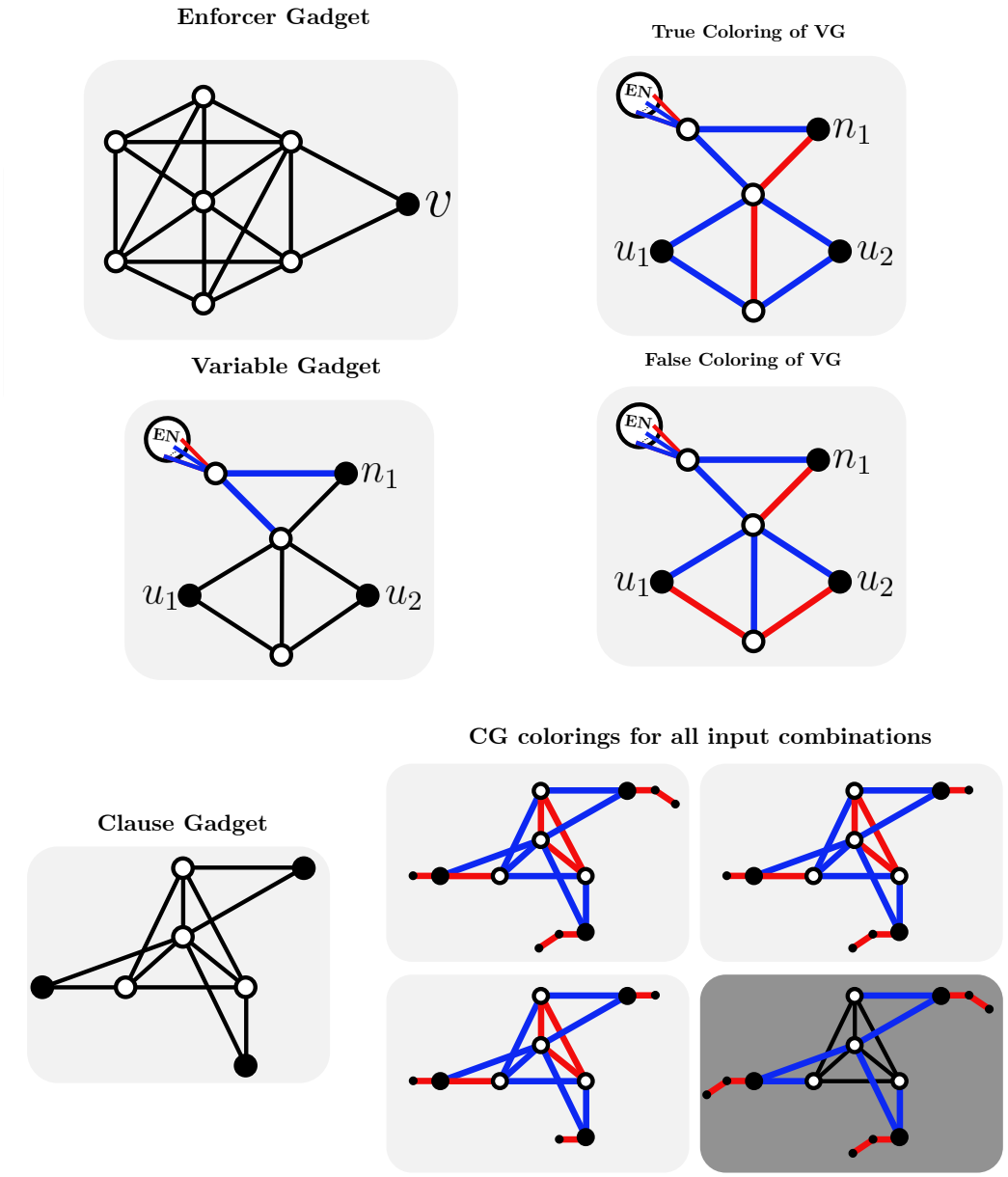

Figure 10 The enforcer, variable, and clause gadgets for (P_4, K_3)-Arrowing are shown. We show a true and false coloring for VG. Since the output vertices of the variable gadget can be the endpoints of red P_2's or red P_3's, we depict the inputs to the clause gadget as such. Next to the clause gadget, we show that each possible input combination to CG, except the case where the input is three red P_3's, a good coloring of CG is possible. Due to symmetry, it is enough to only consider the four clause gadget colorings shown.

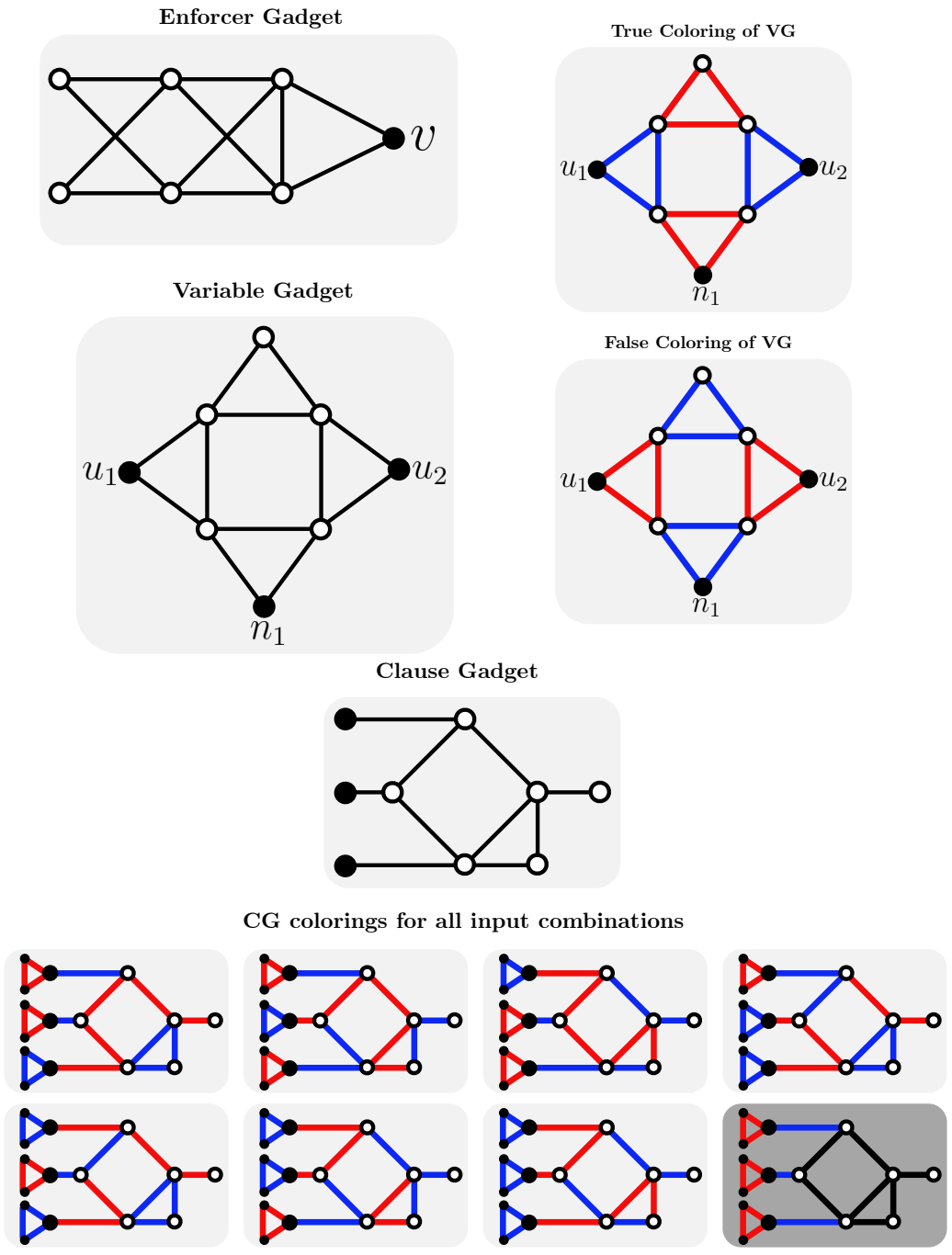

Figure 11 The enforcer, variable, and clause gadgets for $(K_{1,3}, P_4)$-Arrowing are shown. We show a true and false coloring for VG. Since the output vertices of the variable gadget can be the endpoints of red K_3's or blue K_3's, we depict the inputs to the clause gadget as such. Below the clause gadget, we show that each possible input combination to CG, except the case where the input is three red K_3's, a good coloring of CG is possible.