The Complexity of Edge-Induced Greedy Subgraph Building Algorithms Within P

Zohair Raza Hassan
Rochester Institute of Technology, NY, USA

Edith Hemaspaandra
Rochester Institute of Technology, NY, USA

Abstract

A common approach used to efficiently solve problems is to develop sequential greedy algorithms. Such algorithms are easily implemented and provide polynomial-time solutions. A natural next step towards building more efficient algorithms is to develop parallel algorithms. However, sequential greedy algorithms seldom lead to parallel algorithms; computing the output of sequential greedy algorithms is often shown to be P-complete and thus "inherently sequential" under the commonly believed assumption that P $\neq$ NC, where NC is the class of efficiently parallelizable problems.

Greedy edge-induced (resp., vertex-induced) subgraph building algorithms for a property π operate like so. For given graph G, a subgraph of G is built by adding edges (resp., vertices) in a given order unless the inclusion of said edge (resp., vertex) would contradict property π within the subgraph. For vertex-induced greedy subgraph building algorithms, Miyano (1989) provided a comprehensive result: computing the subgraph output by such algorithms is typically P-complete. In contrast, little is known about its edge-induced counterpart. In this work, we analyze the complexity of the Lexicographically First Maximal H-free edge-induced subgraph problem, which is concerned with computing the output of greedy edge-induced subgraph building algorithms where the property π is that the subgraph is H-free. This gives us insight into the largely overlooked edge-induced versions of greedy subgraph building algorithms and into how graph structure influences the complexity of such algorithms.

Our primary contribution is a trichotomy theorem for the cases where H is a tree: we show that the problem is either P-complete, CC-complete, or in L, where CC is the class of problems solvable using comparator circuits – or, equivalently, problems reducible to the lexicographically first maximal matching problem. In contrast, the vertex-induced version is either P-complete or in L, and such dichotomy theorems are much more common. Our additional technical contributions include: (1) an iterative approach to hardness proofs by focusing on a set of "smaller" problems and extending hardness via simple constructions, and (2) expanding on the scarce set of problems known to be CC-complete.

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

Keywords and phrases P-completeness, parallelizability, lexicographically first edge problems

Digital Object Identifier 10.4230/LIPIcs.MFCS.2026.92

Funding Supported in part by NSF grant CCF-2421977 and a Renewed Research Stay grant from the Alexander von Humboldt Foundation.

Acknowledgements Work done in part while visiting Heinrich-Heine-Universität Düsseldorf. We thank Joanna Kaczmarek for helpful discussions. We thank the reviewers for their helpful comments.

1 Introduction and related work

Sequential greedy algorithms are ubiquitous in computer science and are frequently used to obtain solutions or approximate solutions for many natural problems such as certain graph coloring problems [21], computing committee elections [18], and scheduling [19]. In their textbook on P-completeness, Greenlaw, Hoover, and Ruzzo say [14], "Greedy algorithms

often lead to easily implemented efficient sequential solutions to problems. Unfortunately, it also seems to be that sequential greedy algorithms frequently lead to solutions that are inherently sequential." For instance, for the examples provided earlier, it has been shown that computing the solutions computed by greedy algorithms is P-complete (e.g., simple variants of graph coloring [14, 21], computing committee elections [6, 7, 8, 11], and scheduling [9, 29]) and thereby "inherently sequential" under the widely-believed assumption that P $\neq$ NC, where NC is the set of efficiently parallelizable problems. In this work, we explore how graph structure influences the complexity of greedy subgraph building algorithms within P. In particular, we explore the complexity of the following problem.

▶ **Problem 1** (The LFM H-free problem)**.** *Let H be a fixed graph. Given a graph G, an ordering of the edges in $E(G)$, and a designated edge $e \in E(G)$, is e in the* ***lexicographically first maximal H-free edge-induced subgraph*** *(LFM H-free subgraph) of G? The LFM H-free subgraph of G is the graph formed by processing each edge in $E(G)$ in order and adding it to the subgraph being built if it does not form a copy of H.*

The problem is clearly in P for any H. Miyano showed that the problem is P-complete when H is a cycle [23], but completeness results for P or complexity classes below P for all H remains wide open. Other examples of edge-induced subgraph problems are the P-complete Edge Maximal Acyclic Subgraph problem on directed graphs [13], the CC-complete Lexicographically First Maximal Matching problem [22] (CC $\subseteq$ P is the class of problems solvable using comparator circuits), and the minimum spanning tree problem which is in NC. To the best of our knowledge, there are not many subgraph building problems with an ordering on the edges whose complexity below P has been explored. In contrast, for vertex-induced greedy subgraph building (i.e., when building the subgraph by adding vertices instead of edges), a comprehensive result is known: for any nontrivial graph property π that is hereditary on vertex-induced subgraphs and computable in polynomial time, computing the lexicographically first maximal vertex-induced subgraph satisfying π is P-complete [23]. In general, it seems as though proving hardness results for edge-based graph problems is more difficult than for their vertex-induced counterparts. Examples include coloring graphs while avoiding monochromatic subgraphs on vertices [27] vs. edges [15], and decomposing/factoring graphs into vertices [17] vs. edges [10].

In this work, we focus on the complexity of the lexicographically first maximal H-free edge-induced subgraph problem where H is a tree and prove the following trichotomy theorem.

▶ **Theorem 2.** *Let H be a tree. The LFM H-free problem is: (1)* P*-complete if H contains P_4, (2)* CC*-complete if H contains P_3 and is P_4-free, or (3) in* L *if H is P_3-free.*

We note that such theorems have been proven in the past for other graph problems as well [16, 20, 24], but seldom for complexity classes within P. We also note the contrast between the edge-induced and vertex-induced versions of the LFM H-free problem; the vertex-induced version is P-complete for any H except $H = K_1$, where it is in easily seen to be in L. We note that CC-hardness is used to show that problems are inherently sequential as well [3]; CC is believed to be incomparable to NC, and thus CC-hard problems are also believed to be inherently sequential [5, 28]. As such, our results establish that the LFM H-free problem is inherently sequential under the assumptions that P $\neq$ NC and CC $\neq$ NC. We summarize our additional technical contributions below.

1. We expand on the scarce set of problems known to be complete for CC, an esoteric and obscure subclass of P; the LFM $K_{1,b+1}$-free problem is equivalent to the lexicographically first maximal b-matching problem and we prove its CC-completeness in Section 3.

2. We note that there are limited resources available on the class CC and proving that problems belong to CC is a challenging task, but we believe our presentation in Section 3 provides an accessible introduction to the topic.
3. We define a simple graph operation, trim(H), that removes pendant vertices of H to obtain smaller graphs. Using this operation and a simple construction, we show how to extend hardness results for "smaller" H to "larger" H. Hardness proofs typically rely on explicit constructions of gadgets, but our approach allows us to focus our attention on constructing gadgets only for a restricted family of H and extending this hardness result to larger H via a simple and intuitive construction.

The rest of the paper is organized as follows. The necessary preliminaries are provided in Section 2. The CC-completeness and P-completeness results of Theorem 2 are proven in Sections 3 and 4, respectively. We note that these sections are self-contained and do not refer to each other. We conclude and discuss future directions of our work in Section 5.

2 Preliminaries

2.1 Graphs

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 graph on n vertices is denoted as P_n. The star graph on $n+1$ vertices is denoted as $K_{1,n}$. A tree is a graph that is connected and does not contain any cycle. 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 $N(u) \cup N(v)$. We say that a graph G is H-free if G does not contain any, not necessarily induced, copy of H. A pendant vertex is a vertex of degree one. For a graph G with vertex v, we often use the phrase "attach a pendant vertex to v" to mean adding a new vertex to G and connecting it to $v \in V(G)$. A matching of a graph G is a set $M \subseteq E(G)$ of edges where no two edges share a vertex.

2.2 Complexity

The complexity classes relevant to our paper are P, CC, and L. P is the class of decision problems solvable in polynomial time. CC is the class of decision problems solvable using comparator circuits – a formal definition is deferred to Section 3. L is the class of decision problems solvable using logarithmic space (logspace). Since we are looking at complexity classes within P, we must use reductions with less computational power. As such, all reductions presented in this work are logspace reductions.

We know that NL $\subseteq$ CC $\subseteq$ P [22], and NC and CC are believed to be incomparable; Cook et al. give oracles relative to which CC and NC are incomparable [5]. There are not many problems known to be CC-complete, but notable examples include the Lexicographically First Maximal Matching problem [22], the Stable Marriage problem [22, 25], and the Telephone Connection problem [26].

3 CC-completeness

In this section, we will show that the lexicographically first maximal H-free edge-induced subgraph problem for stars is CC-complete, since P_4-free trees containing P_3 are the star graphs $K_{1,n}$ for $n \geq 2$. As mentioned previously, LFMM (lexicographically first maximal matching) is CC-complete [22]. Note that LFMM = the LFM P_3-free problem = the LFM

$K_{1,2}$-free problem. Also note that for all $b \geq 1$ the lexicographically first b-matching (a matching in which each vertex is matched at most b times) is the same as the lexicographically first $K_{1,b+1}$-free edge-induced subgraph. We will think of our problems in terms of matchings, since CC is particularly suited for matchings.

▶ **Problem 3** (LFMbM). *Given a graph G, an ordering of the edges in $E(G)$, and a designated edge $e \in E(G)$, is e in the lexicographically first maximal b-matching of G?*

▶ **Theorem 4.** *LFMbM is* CC*-complete for every $b \geq 1$.*

Not only does this theorem prove the CC-complete part of the trichotomy theorem 2, it also adds new natural CC-complete problems, which are quite rare. We present our CC-hardness proof in Section 3.1. We show that LFMbM is in CC in Section 3.2.

3.1 LFMbM is CC-hard

As mentioned before, LFMM (= LFM1M) is CC-complete. The hardness of LFMbM follows easily from the hardness of LFMM.

▶ **Lemma 5.** *LFMM reduces to LFMbM for every $b \geq 1$.*

Proof. Let G be a graph with an ordering of the edges in $E(G)$, and let e be the designated edge $e \in E(G)$. Construct a graph G' by attaching $b-1$ pendant vertices to each $v \in V(G)$. Order the edges such that all the newly added edges appear before the edges in $E(G)$. When constructing the lexicographically first maximal b-matching of G', all newly added edges will be selected first, i.e., every vertex in $V(G)$ is matched $b-1$ times and every edge incident with a new vertex has been added. Thus, the edges added from $E(G)$ will correspond exactly to edges selected in the lexicographically first maximal matching of G, and e is selected in the lexicographically first maximal matching of G if and only if it is selected in the lexicographically first maximal b-matching of G'. ◀

3.2 LFMbM is in CC

As to inclusion in CC, it makes sense to try and directly reduce LFMbM to LFMM. That would avoid having to delve into the mechanics of the rather obscure class CC.[1] It is easy to see how to reduce the perfect b-matching problem to the perfect matching problem using the construction from [30] (see, for example, [4, Chapter 8]), but this reduction does not seem to be adaptable to reduce LFMbM to LFMM.

So instead of reducing to LFMM we will adapt the proof that LFMM is in CC. The original proof from [22, Section 6.2], there attributed to Cook, personal communication, is a little tricky. The more recent paper by Cook et al. [5] has a more accessible proof, but their definition of LFMM is restricted to bipartite graphs and the order is on the vertices instead of the edges. Below, we show that LFMbM is in CC by first reducing the general case to the bipartite case, and then adapting their proof. We note in passing that our approach also gives an accessible reduction for LFMM in the general case.

▶ **Lemma 6.** *LFMbM reduces to LFMbM restricted to bipartite graphs for every $b \geq 1$.*

[1] Aaronson [1] calls it "a class that's obscure enough not to be in the Complexity Zoo (!)," though that is no longer the case.

Proof. Let G be a graph with an ordering of the edges in $E(G)$, and let e be the designated edge. For each $v \in V(G)$, add vertex v' and replace each edge $f = (v, w)$ in the order of edges by (v, w'), (w, v'). The resulting graph G' is bipartite (unprimed vertices in one set; primed vertices in the other). It is easy to see that running the LFMbM algorithm on G' closely follows running the algorithm on G, since for every edge $f = (v, w)$ that we encounter in the order of $E(G)$, if f is put in the b-matching of G then (v, w') and (w, v') are put in the b-matching of G' and if f is not put in the b-matching of G then neither (v, w') nor (w, v') are put in b-matching of G' (note that at any point in the algorithm, every vertex v is matched in G exactly as many times as v and v' are matched in G'). To finish the reduction, let the designated edge of G' be (v, w') where (v, w) is the designated edge of G.[2] ◀

In Section 3.2.1, we introduce comparator circuits and how they can be used to compute lexicographically first maximal matchings in bipartite graphs, as shown by Cook [5], and we show how this approach can be generalized to compute b-matchings in bipartite graphs. In Section 3.2.2, we provide the general construction that shows LFMbM is in CC using the ideas presented in Section 3.2.1.

3.2.1 Comparator circuits for b-matchings

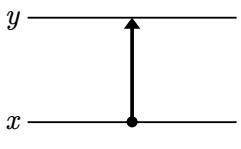
A **comparator circuit** is a circuit that maps n bits to n bits. The only allowed gate is the comparator gate, which sorts two input bits (i.e., input (p, q) gives output $(min(p, q), max(p, q))$. The **comparator circuit value problem (CCV)** is the problem of determining whether the designated output bit of a comparator circuit with specified input bits is 1.

▶ **Definition 7** ([28]). **Comparator Circuits (CC)** *is the class of problems logspace many-one reducible to CCV.*

The following useful observation follows immediately from [5].[3] We note that closure under complementation was already pointed out in [22].

▶ **Observation 8.** CC *is closed under union, intersection, and complementation.*

An appealing way to visualize a comparator circuit is as a sorting network [2], with n horizontal wires, each carrying one bit, and a sequence of comparator gates [5]. The comparator gate that sorts bits x and y is denoted by an arrow from x to y, where the arrow points to the larger bit (see Figure 1).



IN		OUT	
x	y	x	y
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

Figure 1 Wires x and y with comparator gate $x \to y$ and corresponding truth table.

[2] Note that this construction does not give a reduction from maximum b-matching to maximum b-matching for bipartite graphs. For example, the triangle $(v, w), (w, x), (x, v)$ has a maximum matching of size 1, but $(v, w'), (w, v'), (w, x'), (x, w'), (v, x'), (x, v')$ has perfect matching $(v, w'), (w, x'), (x, v')$. The reason that our reduction works is that for every edge $f = (v, w)$ in G, (v, w') and (w, v') are both in or both not in the lexicographically first maximal b-matching of G'.

[3] Cook et al. [5] showed that CC $= (\mathrm{AC}^0)^{\mathrm{CCV}}$, which is closed under union, intersection, and complementation.

Note that sorting bits x and y corresponds to moving a bit from x to y. Note that $x \to y$ changes values only when the input to x is 1 and the input to y is 0. In this case we say that $x \to y$ "fires." When using comparator circuits to compute a matching in a bipartite graph, moving a bit from x (with input 1) to y (with input 0) (i.e., $x \to y$ firing) will correspond to matching vertex x and vertex y. See Figure 2 for an example.

In a b-matching, each vertex can be matched b times. b-matchings for bipartite graphs can be computed in a similar way by using modified comparator circuits where each wire is allowed to carry a value between 0 and b. Now, when using a circuit to compute a b-matching in a bipartite graph, moving one unit from x to y (i.e., $x \to y$ firing) will correspond to matching vertex x and vertex y. See Figure 3 for an example with $b = 2$.

We will now see how to simulate modified comparator circuits using standard comparator circuits. The general construction will follow, but we will first look at the case for $b = 3$ as an example. Every wire x in the modified comparator circuit will be represented using three wires x_1, x_2, and x_3 in the standard comparator circuit. The values 0, 1, 2, and 3 of x will be represented by 000, 001, 011, and 111 of $x_1x_2x_3$. Each gate $x \to y$ in the modified comparator circuit is simulated in the standard comparator circuit using the following sequence of gates: $x_3 \to y_1$ (to transfer one unit), $x_2 \to x_3$ and $x_1 \to x_2$ (to "renormalize" $x_1x_2x_3$), $y_1 \to y_2$ and $y_2 \to y_3$ (to renormalize $y_1y_2y_3$) (see Figure 4).

3.2.2 General construction

Let $b \geq 1$. Let G be a bipartite graph with bipartition (V, V'), an ordering of the edges in $E(G)$, and a designated edge $e \in E(G)$. We need to determine whether e is in the lexicographically first maximal b-matching of G. In our examples above, we intuitively explained how a b-matching corresponds to a comparator circuit computation. However, CC is a class of decision problems, namely, the class of problems reducible to CCV (Definition 7), where CCV is the problem of determining whether the designated output bit of a comparator circuit with specified input bits is 1. We define a related problem, LFMbM-helper, and show that it is in CC via a reduction to CCV. LFMbM's inclusion in CC follows from CC's closure properties (Observation 8).

▶ **Problem 9** (LFMbM-helper). *Given a nonempty bipartite graph G with bipartition (V, V'), an ordering of the edges in $E(G)$ with last element $(v, w') \in V \times V'$, and an integer k, is w' matched at least k times in the lexicographically first maximal b-matching.*

Below we will show that LFMbM-helper is in CC. First, we will argue that this implies that LFMbM is in CC. Let $e = (v, w') \in V \times V'$ be the designated edge in G. Let G' be G with all edges after e removed. Note that e is in the lexicographically first maximal b-matching of G if and only if e is in the lexicographically first maximal b-matching of G', since edges after e do not influence whether e is in the lexicographically first maximal b-matching. Also note that $e = (v, w')$ is in the lexicographically first maximal b-matching of G' if and only if w' is matched k times in G' and w' is matched $k - 1$ times in $G' - e$, for some k, $1 \leq k \leq b$. This is equivalent to w' being matched $\geq k$ times in G' and w' not being matched $\geq k$ times in $G' - e$, for some for some k, $1 \leq k \leq b$. Since CC is closed under union, intersection, and complementation by Observation 8, and we will show that LFMbM-helper is in CC, this proves that LFMbM is in CC as required. It remains to show the following lemma.

▶ **Lemma 10.** *LFMbM-helper is in* CC.

Proof. Let $b \geq 1$, let G be a nonempty bipartite graph with bipartition (V, V'), an ordering of the edges in $E(G)$ with last element $(v, w') \in V \times V'$, and let $k \leq b$ be an integer. We need to determine whether w' is matched at least k times in the lexicographically first maximal b-matching.

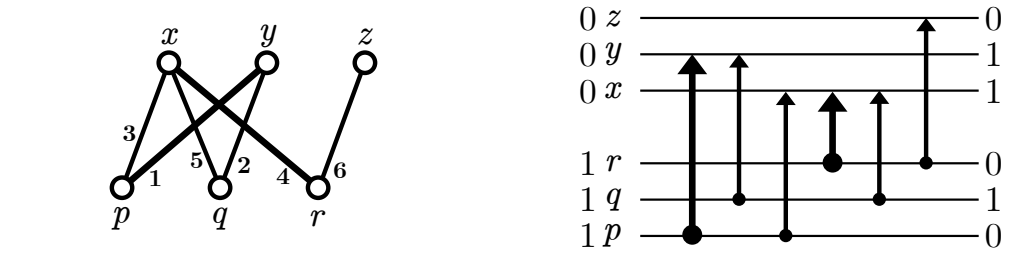


Figure 2 Example of correspondence between the lexicographically first maximal matching in a bipartite graph (with matched edges bolded) on the left and comparator circuit computation (with firing comparator gates bolded) on the right. Note that the output value of a vertex in $\{x, y, z\}$ is 1 if and only if that vertex is matched.

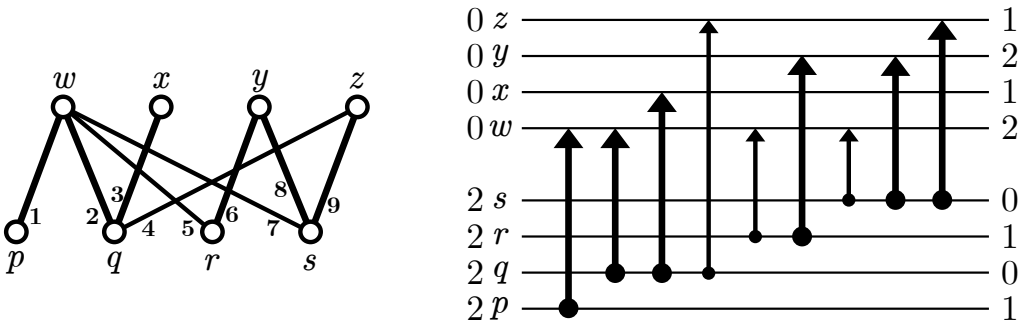


Figure 3 Example of correspondence between the lexicographically first maximal 2-matching in a bipartite graph (with matched edges bolded) on the left and a modified comparator circuit computation (with firing comparator gates bolded) on the right. Note that the output value of a vertex in $\{w, x, y, z\}$ is the number of times that vertex is matched.

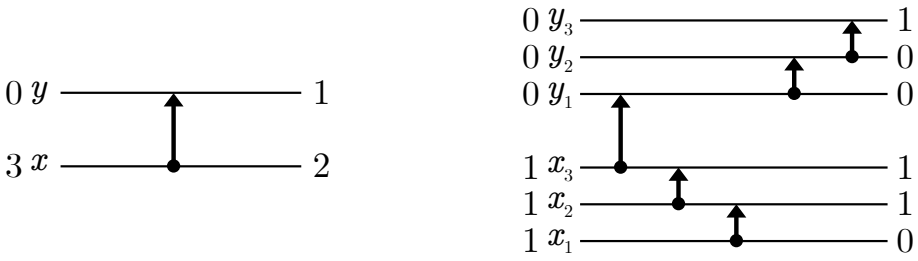


Figure 4 A gate in a modified comparator circuit with $b = 3$ (left) and its corresponding standard comparator circuit (right).

The following is a straightforward generalization of the $b = 3$ case described above and pictured in Figure 4. Each vertex x will correspond to b wires, $x_1, \ldots, x_b$ and values $0, 1, 2, \ldots, b$ will be represented as 0^b, $0^{b-1}1$, $0^{b-2}11$, $\ldots$, 1^b. Wires corresponding to vertices in V will be initialized to 1^b (they have b units to pass on, i.e., they can be matched with b vertices in V') and wires corresponding to vertices in V' are initialized to 0^b (they can receive b units, i.e., they can be matched with b vertices in V). An edge $(p, q') \in V \times V'$ is represented by the following sequence of comparator gates:

- $p_b \to q'_1$ to transfer one unit from p to q' (if the value of $p > 0$ and the value of $q' < b$).
- $p_{b-1} \to p_b$, $p_{b-2} \to p_{b-1}$, …, and $p_1 \to p_2$ to renormalize $p_1 p_2 \cdots p_b$ to a representation in 0^*1^*.
- $q'_1 \to q'_2$, $q'_2 \to q'_3$, …, and $q'_{b-1} \to q'_b$ to renormalize $q'_1 q'_2 \cdots q'_b$.

It is easy to see that (v, w') is put in the b-matching if and only if $v_b \to w'_1$ fires. Moreover, at the end of the comparator circuit computation, the value of w' is the number of times w' is matched. w' is matched at least k times in the lexicographically first maximal b-matching if and only if the output bit of w'_{b-k+1} is 1. Thus, this corresponds to a reduction to CCV with designated output bit w'_{b-k+1}. ◀

4 P-hardness

In this section, we discuss our P-hardness proofs for the LFM H-free problem where H is a tree that contains a P_4, the path graph on four vertices. We adopt a two-tier approach:

1. We define a restricted set of trees, Γ_S, and construct gadgets for each $H \in \Gamma_S$ so that we can reduce from a restricted version of the circuit value problem (CVP).
2. For trees H containing a P_4 that do not belong to Γ_S, we reduce from "smaller" LFM H-free problems. We define a graph operation, trim$(\cdot)$, and show how to reduce the LFM trim(H)-free problem to the LFM H-free problem. By showing that repeated applications of trim$(\cdot)$ give us a graph in Γ_S, we obtain a hardness proof for every tree containing a P_4 without having to construct gadgets.

Essentially, we do all the hard work for a restricted set of trees, and the hardness of all other trees follows easily using an intuitive construction. For a more palatable presentation, we first, in Section 4.1, present a hardness proof for the LFM P_4-free problem and show an intuitive construction that then gives a hardness proof for the LFM P_6-free problem. Then, in Section 4.2, we discuss how this idea generalizes to other trees, while deferring the technical details to the appendix.

4.1 Hardness for $H = P_4$ and $H = P_6$

We first define the P-complete problem needed for our hardness proofs. Loosely, the circuit value problem is to simulate a given boolean circuit on a given input and determine whether the circuit outputs a value of 1. For their simplicity, we opt for the definitions presented by Miyano [23]. A **boolean circuit** is a sequence $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ where each α_i is either a truth value (1 or 0) or a gate which takes values from previous α_j's (e.g., an AND gate would be $\alpha_i = \alpha_j \wedge \alpha_k$ for $j < k < i$). value(α_i) is the truth value of α_i after it has been evaluated.

▶ **Problem 11** (Planar CVP [12, 23]). *Suppose we are given a circuit $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ where each α_i is either a true value ($\alpha_i = 1$), a NOT gate ($\alpha_i = \neg\alpha_j$ for $j < i$) or an AND gate ($\alpha_i = \alpha_j \wedge \alpha_k$ for $j < k < i$). α is planar, i.e., the graph representation of α where each vertex is an α_i and two vertices α_i and α_j are adjacent if α_i takes input from α_j is planar. Is* value$(\alpha_n) = 1$*?*

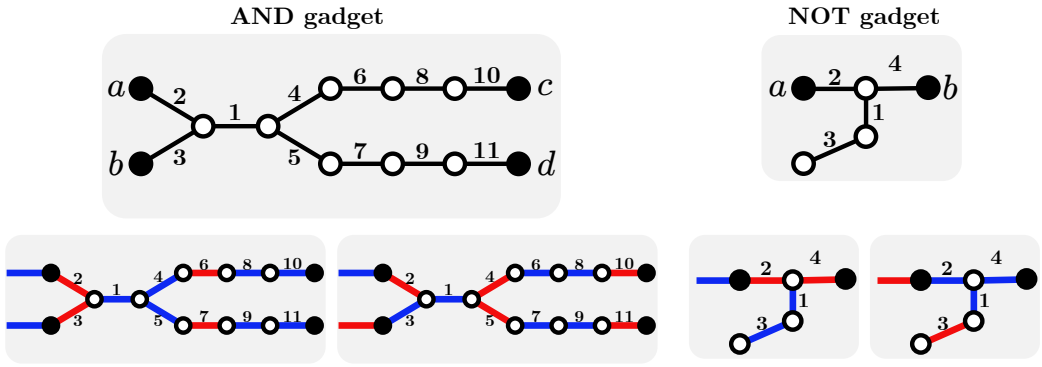


Figure 5 On the top left, we show the AND gadget used in our reduction from Planar CVP to the LFM P_4-free problem to simulate an AND gate. The vertices labeled a and b (resp., c and d) are referred to as input (resp., output) vertices. The numbers next to the edges depict their order. Below the gadget, we show which edges of the gadget are selected (depicted as blue) or not selected (depicted as red) to be in the LFM P_4-free subgraph depending on whether or not external edges adjacent to the input vertices are selected – "blue is true." These external edges appear earlier in the edge order than any edge in the gadget. We show two of the four possible input combinations. The other two are very similar. The NOT gadget is presented similarly on the right, where a (resp., b) is the input (resp., output) vertex.

Miyano [23] notes that Planar CVP is P-complete even when each α_i has fanout 2, i.e., is input to two α_j's. We further note the following observation to make our reduction easier.[4]

▶ **Observation 12.** *Planar CVP is* P*-complete even when each true value ($\alpha_i = 1$) has fanout 1, each NOT gate has fanout 1, and each AND gate has fanout at most 2.*

A question of interest for most graph problems is whether the problem remains hard even when the input graph's structure is restricted. Using the planarity of this problem and a careful construction of gadgets, we are able to show that the LFM H-free problems we discuss are P-complete even when the input graph is simultaneously planar and bipartite.

▶ **Lemma 13.** *The LFM P_4-free problem is* P*-complete, even when the input graph is simultaneously planar and bipartite.*

Proof. Given an instance $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ of Planar CVP where each true value and NOT gate has fanout 1 and each AND gate has fanout at most 2, we construct a graph G_α with an ordering on the edges and a designated edge $e \in E(G_\alpha)$ such that value$(\alpha_n) = 1$ if and only if e is in the LFM P_4-free subgraph of G_α. G_α is constructed by combining gadgets that simulate each gate. For our reduction, a gadget refers to a graph and an ordering of its edges. In Figure 5, we show a gadget that simulates an AND gate: when both of the external edges adjacent to the input vertices are selected to be in the LFM P_4-free subgraph, the edges adjacent to the output vertices are also selected. If at least one of the edges adjacent to the input vertices is not selected, then the edges adjacent to the output vertices are not selected. It is easy to see the correspondence between selected edges and true values. Figure 5 also shows a gadget that simulates a NOT gate. The gadget for true values is an edge labeled 1

[4] This holds because: (1) whenever a true value α_i is input to α_{j_1} and α_{j_2}, we can construct true values α_{i_1} and α_{i_2} that take α_i's place in α_{j_1} and α_{j_2}, respectively, and (2) whenever a NOT gate α_i is input to α_{j_1} and α_{j_2}, we can construct $\alpha_i' = \alpha_i \wedge \alpha_i''$ and replace α_i in α_{j_1} and α_{j_2} with α_i', where $\alpha_i'' = 1$. This can be done in logspace and preserves planarity.

where one vertex is an output vertex. G_α is constructed as follows. For every α_i we add a copy of the corresponding gadget. We refer to this copy as G_{α_i}. The edges of G_{α_i} are labeled so that they follow the order shown in the figures, but are offset so that each edge in G_α has a unique label and all edges of G_{α_j} appear before all edges of G_{α_i} when $j < i$. If α_i takes input from α_j then we identify an input vertex of α_i with an output vertex of α_j. It is clear that the edge adjacent to an output vertex of G_{α_n} is selected to be in the LFM P_4-free subgraph of G_α if and only if $\alpha_n = 1$. Thus, said edge is our designated edge e. If there are multiple output vertices (i.e., α_n is an AND gate), we arbitrarily select the lexicographically first of the edges adjacent to the output vertices.

G_α's planarity follows from the fact that each gadget is planar and the original circuit α is planar. Note that the length of any path between any pair of input or output vertices is even, so no odd cycles can be made while constructing G_α. Thus, G_α must be bipartite. We note that it is possible to come up with simpler gadget constructions either by modifying the gadgets shown, or by considering variants of CVP that use different gates. We chose these gadgets since they allowed us to show hardness under both planarity and bipartiteness. ◀

Unfortunately, as is the case with many problems whose complexity is based on some underlying graph (e.g., [15, 16]), the P-hardness result of the LFM P_4-free problem does not automatically provide a P-hardness result for all LFM H-free problems where H is a tree containing P_4; this needs to be proven explicitly. One way to prove this would be to present a general gadget construction for any tree containing P_4, but this seems very difficult. Furthermore, even after discovering such a gadget construction, one would have to argue its correctness, which is not as straightforward as it is for P_4. To illustrate this point, let us take a look at the reduction above. Note that although a truth value is represented by a single selected edge, the selected edge going into a gadget (i.e., the external edge adjacent to the input vertex) may be part of a larger selected subgraph from the previous gadget. Thus, when generalizing this reduction to some H using different gadgets, it is necessary to show that when the gadgets are combined (i.e., their vertices are identified during the construction of G_α) with selected edges going across them, they do not form a copy of H within the selected subgraph going across the gadgets. It is easily observed in the reduction above that such copies of P_4 do not exist since whenever an external edge going into a gadget is selected, the edge it is adjacent to within the gadget is not selected. This subtlety will be relevant to the gadget constructions provided for $H \in \Gamma_S$ (see the proof of Lemma 19). Fortunately, due to the restricted nature of the trees in Γ_S, the validity of these gadgets will be easy to see.

Another, more intuitive, approach to showing hardness is to use the hardness of the LFM P_4-free problem to show the hardness of the LFM H-free problem where P_4 is a subgraph of H. We now describe how this hardness result can be extended to $H = P_6$ without having to construct new gadgets. Consider the following construction:

▶ **Definition 14.** *For a graph G, let $G^\star$ be the graph constructed by attaching a pendant vertex (i.e., a vertex of degree 1) to every vertex in G. The new edges added are labeled so that they appear before the edges in $E(G)$ in lexicographic order.*

Observe that when constructing the LFM P_6-free subgraph of $G^\star$, the edges in $E(G^\star) - E(G)$ will be selected to be in the LFM P_6-free subgraph first. Then we start selecting edges from $E(G)$. Note that when selecting edges from $E(G)$, we will avoid selecting edges that form a P_4, since a P_4 and two edges from $E(G^\star) - E(G)$ form a P_6. It is easy to see that for any $e \in E(G)$, e is in the LFM P_4-free subgraph of G if and only if e is in the LFM P_6-free subgraph of $G^\star$. We discuss how to generalize this idea in the next section.

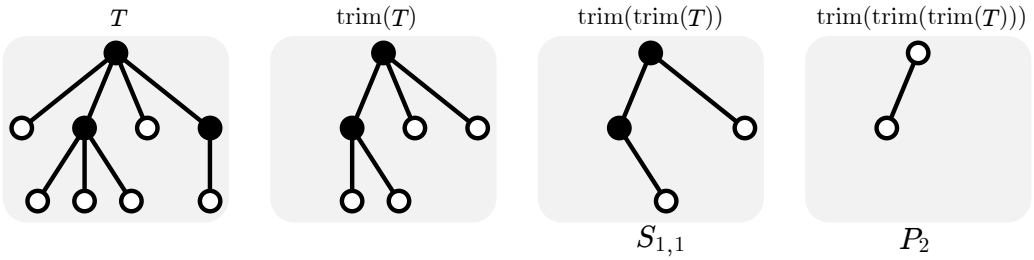


Figure 6 A tree T and the trees produced by repeated applications of the trim$(\cdot)$ operation on T. The edges are ordered from left to right. The vertices of X (as defined in Definition 15) for each tree are filled. Note how repeated applications of trim$(\cdot)$ gives us a graph in Γ_S, as stated in Lemma 18.

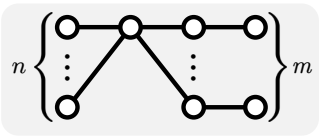


Figure 7 The generalized star graph $S_{n,m}$.

4.2 Hardness for all trees containing P_4

In this section, we discuss the main idea behind our P-hardness proof for all trees containing P_4. The proofs for the lemmas presented in this section are deferred to the appendix. Observe that P_4 can be obtained from P_6 by deleting pendant vertices from P_6, a process that we describe as "trimming." We generalize this below.

▶ **Definition 15.** *Suppose H is a tree on at least three vertices and $X \subseteq V(H)$ are the vertices of H adjacent to at least one pendant vertex (note that the pendant vertices are exactly the leaf vertices of H). We define the trimmed subgraph of H as follows. For $v \in X$ adjacent to pendant vertices $L_v \subseteq N(v)$, let (v, ℓ_v) be the lexicographically first edge in $\{(v, w) \mid w \in L_v\}$. The graph* $\mathbf{trim}(\boldsymbol{H})$ *is the graph obtained by removing the vertices in $\{\ell_v \mid v \in X\}$ from H.*

Note that trim$(P_6) = P_4$. An example is provided in Figure 6. We can reduce the LFM trim(H)-free problem to the LFM H-free problem using the construction from Definition 14 by making use of the following lemmas.

▶ **Lemma 16.** *Let H be a tree on at least three vertices and let G be a graph. G contains a copy of* trim(H) *if and only if $G^\star$ contains a copy of H.*

▶ **Lemma 17.** *For a fixed tree H on at least three vertices, if the LFM* trim(H)*-free problem is* P*-complete, then the LFM H-free problem is also* P*-complete.*

Note that the construction of $G^\star$ preserves the planarity and bipartiteness of G. By analyzing the sequence of trees obtained using repeated applications of trim$(\cdot)$, we are able to identify a restrictive family of trees. We define a "generalized star" graph, $S_{n,m}$. Let $S_{0,1} = P_3$. For all other $n \geq 0$ and $m \geq 1$, let $S_{n,m}$ be the graph constructed by attaching a pendant vertex to m pendant vertices of a $K_{1,n+m}$. $S_{n,m}$ is illustrated in Figure 7. Let $\Gamma_S = \{S_{n,m} \mid m \geq 2 \text{ or } (m = 1 \text{ and } n \geq 1)\}$.

▶ **Lemma 18.** *Suppose T is a tree that contains a P_4. Let $T_1, T_2, T_3, \ldots$ be the sequence of trees generated by repeatedly trimming T, i.e., $T_1 = T$ and $T_i =$* trim(T_{i-1}) *for $i \geq 2$. There exists a tree in this sequence that belongs to Γ_S.*

Due to the restrictive nature of $S_{n,m}$, we can build hardness gadgets to reduce Planar CVP to the LFM $S_{n,m}$-free problem by exhaustively considering the cases for n and m. As before, we show hardness even when the input graph is simultaneously planar and bipartite.

▶ **Lemma 19.** *For any $H \in \Gamma_S$, the LFM H-free problem is* P*-complete, even when the input graph is simultaneously planar and bipartite.*

The proof for Lemma 19 (in the appendix) follows the same recipe as the proof for Lemma 13. Using Lemmas 17, 18, and 19, we can finally show the following:

▶ **Theorem 20.** *Let H be a tree containing a P_4. The LFM H-free problem is* P*-complete, even when the input graph is simultaneously planar and bipartite.*

Proof. If $H \in \Gamma_S$, hardness follows from Lemma 19. If $H \notin \Gamma_S$, then by Lemma 18 we know that repeated applications of $\text{trim}(\cdot)$ will give us a graph in Γ_S. Thereby, hardness follows from repeated applications of Lemma 17. ◀

5 Conclusion

In this work, we presented results on the complexity of the LFM H-free problem where H is a tree. We discussed that for any H containing a P_4 the problem is P-complete in Section 4 and that for any P_4-free H containing a P_3 the problem is CC-complete. The only P_3-free tree is a P_2, and it is easy to see that the LFM P_2-free subgraph of a graph G contains no edges. Thus, it can be decided using logarithmic space whether a designated edge e is part of the LFM P_2-free subgraph. This completes the proof of our trichotomy theorem.

Our P-completeness results utilize an iterative approach that allows for simple extension of hardness results from a set of "base" P-hard problems, and our CC-completeness results expand on the otherwise scarce set of problems known to be CC-complete.

Our planned future work is to prove a trichotomy theorem for all H – not just trees, and look into the parallel complexity of the problem when the structure of the input graph is restricted. Our ultimate goal is to provide a comprehensive complexity result for edge-induced greedy subgraph building problems for any nontrivial graph property π, similar to the one by Miyano [23] about vertex-induced subgraph building problems.

References

1 S. Aaronson. The power of the Digi–Comp II. `https://scottaaronson.blog/?p=1902`. Posted on Friday, July 4th, 2014 at 10:16 AM.

2 K. E. Batcher. Sorting networks and their applications. In *AFIPS 1968*, pages 307–314. Association for Computing Machinery, 1968.

3 M. Bateni, L. Dhulipala, K. N. Gowda, D. Ellis Hershkowitz, R. Jayaram, and J. Lacki. It's Hard to HAC Average Linkage! In *ICALP 2024*, 2024.

4 C. Berge. *Graphs*. Amsterdam, 1985.

5 S. Cook, Y. Filmus, and D. Lê. The complexity of the comparator circuit value problem. *ACM Trans. Comput. Theory*, 6(4):15:1–15:44, 2014. `doi:10.1145/2635822`.

6 T. Csar, M. Lackner, and R. Pichler. Computing the Schulze Method for Large-Scale Preference Data Sets. In *IJCAI 2018*, pages 180–187, 2018.

7 T. Csar, M. Lackner, and R. Pichler. Computing the Schulze Method for Large-Scale Preference Data Sets. Technical Report arXiv:2505.12976 [cs.GT], arXiv.org, May 2025.

8 T. Csar, M. Lackner, R. Pichler, and E. Sallinger. Winner Determination in Huge Elections with MapReduce. In *AAAI 2017*, pages 451–458, 2017.

9 D. Dolev, E. Upfal, and M. K. Warmuth. The Parallel Complexity of Scheduling with Precedence Constraints. *J. Parallel Distributed Comput.*, 3(4):553–576, 1986. doi:10.1016/0743-7315(86)90014-6.

10 D. Dor and M. Tarsi. Graph Decomposition Is NPC-A Complete Proof of Holyer's Conjecture. In *STOC 1992*, pages 252–263. ACM, 1992. doi:10.1145/129712.129737.

11 Z. Fitzsimmons, Z. R. Hassan, and E. Hemaspaandra. On the Parallelizability of Approval-Based Committee Rules. In *ECAI 2025*. IOS Press, 2025.

12 L. M. Goldschlager. The monotone and planar circuit value problems are log space complete for P. *SIGACT News*, 9(2):25–29, 1977. doi:10.1145/1008354.1008356.

13 R. Greenlaw. The Parallel Complexity of Approximation Algorithms for the Maximum Acyclic Subgraph Problem. *Math. Syst. Theory*, 25(3):161–175, 1992. doi:10.1007/BF01374523.

14 R. Greenlaw, H. Hoover, and W. Ruzzo. *Limits to Parallel Computation: P-Completeness Theory*. Oxford University Press, 1995.

15 Z. R. Hassan. The Complexity of (P_3, H)-Arrowing and Beyond. In *MFCS 2024*, volume 306, pages 59:1–59:16, 2024.

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

17 D. G. Kirkpatrick and P. Hell. On the Completeness of a Generalized Matching Problem. In *STOC 1978*, pages 240–245. ACM, 1978. doi:10.1145/800133.804353.

18 M. Lackner and P. Skowron. *Multi-Winner Voting with Approval Preferences*. Springer Nature, 2023.

19 E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys. Sequencing and scheduling: Algorithms and complexity. In S. C. Graves, A. H. G. Rinnooy Kan, and P. H. Zipkin, editors, *Logistics of Production and Inventory*, chapter 9. North-Holland, 1993.

20 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.

21 M. Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM Journal on Computing*, 15(4):1036–1053, 1986. doi:10.1137/0215074.

22 E. Mayr and A. Subramanian. The complexity of circuit value and network stability. *Journal of Computer and System Sciences*, 44(2):302–323, 1992. doi:10.1016/0022-0000(92)90024-D.

23 S. Miyano. The Lexicographically First Maximal Subgraph Problems: P-Completeness and NC Algorithms. *Math. Syst. Theory*, 22(1):47–73, 1989. doi:10.1007/BF02088292.

24 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.

25 G. Pólya, R. E. Tarjan, and D. R. Woods. *Notes on Introductory Combinatorics*, volume 4 of *Progress in Computer Science*. Birkhäuser, 1983.

26 Vijaya Ramachandran and Li-Chung Wang. Parallel Algorithm and Complexity Results for Telephone Link Simulation. In *IPDPS 1991*, pages 378–385. IEEE, 1991. doi:10.1109/SPDP.1991.218216.

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

28 A. Subramanian. *The computational complexity of the circuit value and network stability problems*. PhD thesis, Stanford, 1990.

29 S. Sunder and X. He. Scheduling Interval Ordered Tasks in Parallel. *J. Algorithms*, 26(1):34–47, 1998. doi:10.1006/JAGM.1997.0895.

30 W. Tutte. A short proof of the factor theorem for finite graphs. *Canadian Journal of Mathematics*, 6:347–352, 1954.

A Proofs for Section 4

The proofs for the lemmas in Section 4 are provided below. We introduce the following terminology: the vertices in trim(H) from which pendant vertices were removed are referred to as **trimmed vertices**.

Proof of Lemma 16. The forward direction is easy to see. We now show the reverse. Let Z be a copy of H in $G^\star$. Using our construction for $\text{trim}(Z)$, we will show via contradiction that $\text{trim}(Z)$ must be in G.

If $\text{trim}(Z)$ is not in G, then there must exist a vertex v of $\text{trim}(Z)$ in $V(G^\star) - V(G)$. Note that v cannot be a trimmed vertex of $\text{trim}(Z)$: every vertex in $V(G^\star) - V(G)$ is a vertex of degree one, and trimmed vertices must have degree at least two in Z. Thus, v must be a pendant vertex in $\text{trim}(Z)$. Let $u \in V(G)$ be the neighbor of v. Note that u must be a trimmed vertex in $\text{trim}(Z)$. Recall from Definition 15 that $\text{trim}(Z)$ is constructed by removing the vertex adjacent to the lexicographically first edge going from u to one if its pendant vertices. Since the edges of $E(G^\star) - E(G)$ appear before the edges in $E(G)$, v must have been removed from Z to obtain $\text{trim}(Z)$, a contradiction. ◀

Proof of Lemma 17. Given an instance of the LFM $\text{trim}(H)$-free problem (i.e., a graph G with an ordering on the edges and a designated edge e), construct $G^\star$ (Definition 14). When constructing the LFM H-free subgraph of $G^\star$, the edges in $E(G^\star) - E(G)$ are added before the edges in $E(G)$ since they appear earlier in the ordering. By Lemma 16, it follows that the edges in $E(G) \subseteq E(G^\star)$ are added to the LFM H-free subgraph of $G^\star$ if and only if they are added to the LFM $\text{trim}(H)$-free subgraph of G. Thus, this corresponds to a reduction from the LFM $\text{trim}(H)$-free problem to the LFM H-free problem with designated edge e. ◀

Proof of Lemma 18. Let ℓ be the smallest index in the sequence such that $T_{\ell+1}$ is a star and T_ℓ is not – we know such an ℓ exists because repeated applications of $\text{trim}(\cdot)$ must eventually give a P_2, which is the star $K_{1,1}$. We show that T_ℓ is a member of Γ_S. Let $T_{\ell+1} = K_{1,k}$ for some $k \geq 1$. Since T_ℓ is not a star graph, it must contain a P_4. If $T_{\ell+1} = K_{1,1}$ then it is easy to see that $T_\ell = P_4$ is the only possibility and the statement holds because $P_4 = S_{1,1} \in \Gamma_S$.

Suppose $T_{\ell+1} = K_{1,k}$ for $k \geq 2$. Since T_ℓ contains a P_4, at least one of the pendant vertices in $T_{\ell+1}$ is a trimmed vertex. If exactly one is a trimmed vertex, we have that $T_\ell = S_{k,1}$ or $T_\ell = S_{k+1,1}$, depending on whether or not the root of $T_{\ell+1}$ is trimmed. In either case, we have $T_\ell \in \Gamma_S$. If $x \geq 2$ of these vertices are trimmed we have $T_\ell = S_{k+1-x,x}$ or $T_\ell = S_{k+2-x,x}$, and both of these belong to Γ_S. ◀

Proof of Lemma 19. It is easy to see how to adapt the hardness proof presented for LFM P_4-free problem presented in Lemma 13 to a hardness proof for any tree if we can show gadgets that behave similarly. As before, a selected edge represents a truth value of 1, so the gadget for $\alpha_i = 1$ is the same as in Lemma 13. We consider a total of five cases for $S_{n,m}$ based on the values of n and m and construct AND and NOT gadgets for every case.

Note that although a truth value is represented by a single selected edge, the selected edge going into a gadget (i.e., the external edge adjacent to the input vertex) may be part of a larger selected subgraph from the previous gadget. Thus, for proving the validity of our gadgets, it is necessary to show that when the gadgets are combined (i.e., their vertices are identified during the construction of G_α) with selected edges going across them, they do not form a copy of $S_{n,m}$ within the selected subgraph – we refer to such a copy as a **rogue** copy of $S_{n,m}$. It is easily observed in the gadgets for the LFM P_4-free problem presented in Lemma 13 that no rogue copies of P_4 exist since whenever an external edge going into a gadget is selected, the edge it is adjacent to within the gadget is not selected. We present gadgets for different cases of $S_{n,m}$. It is easy to observe that no rogue copies of $S_{n,m}$ are constructed by considering all cases of identified input/output vertices across both gadgets. The simplest case ($H = P_5$) is illustrated first, and the other figures will have a similar format.

- **Case 1:** $m = 1$. If $n = 1$, we have $H = S_{1,1} = P_4$, whose hardness is proven in Lemma 13 using the gadgets in Figure 5. For $n \geq 2$, we provide gadgets in Figure 9.
- **Case 2:** $m = 2$. If $n = 0$, we have $H = S_{0,2} = P_5$. The gadgets for this case are shown in Figure 8. For $n \geq 1$, we provide gadgets in Figure 10.
- **Case 3:** $m \geq 3$. Gadgets are shown in Figure 11.

Note that as in the proof for Lemma 13, G_α's planarity follows from the fact that each gadget is planar and the original circuit α is planar. G_α's bipartiteness follows from the fact that the length of any path between any pair of input or output vertices is even, so no odd cycles can be made while constructing G_α. ◀

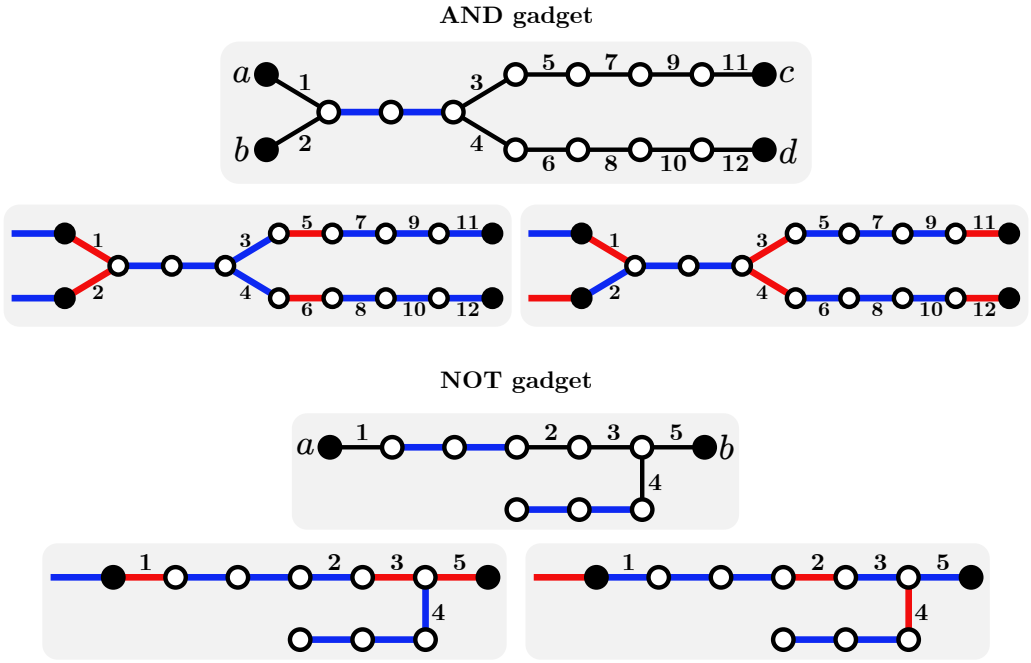


Figure 8 On the top, we show the AND gadget used in our reduction from Planar CVP to the LFM P_5-free problem to simulate an AND gate. The vertices labeled a and b (resp., c and d) are referred to as input (resp., output) vertices. The numbers next to the edges depict their order. Edges that are already colored blue are used to represent edges that appear earlier in the order (i.e., before the edge labeled 1). This is done to avoid clutter. Next to the gadget, we show which edges of the gadget are selected (depicted as blue) or not selected (depicted as red) to be in the LFM P_5-free subgraph depending on whether or not external edges adjacent to the input vertices are selected. These external edges appear earlier in the edge order than any edge in the gadget. We show two of the four possible input combinations. The NOT gadget is presented similarly on the bottom, where a (resp., b) is the input (resp., output) vertex.

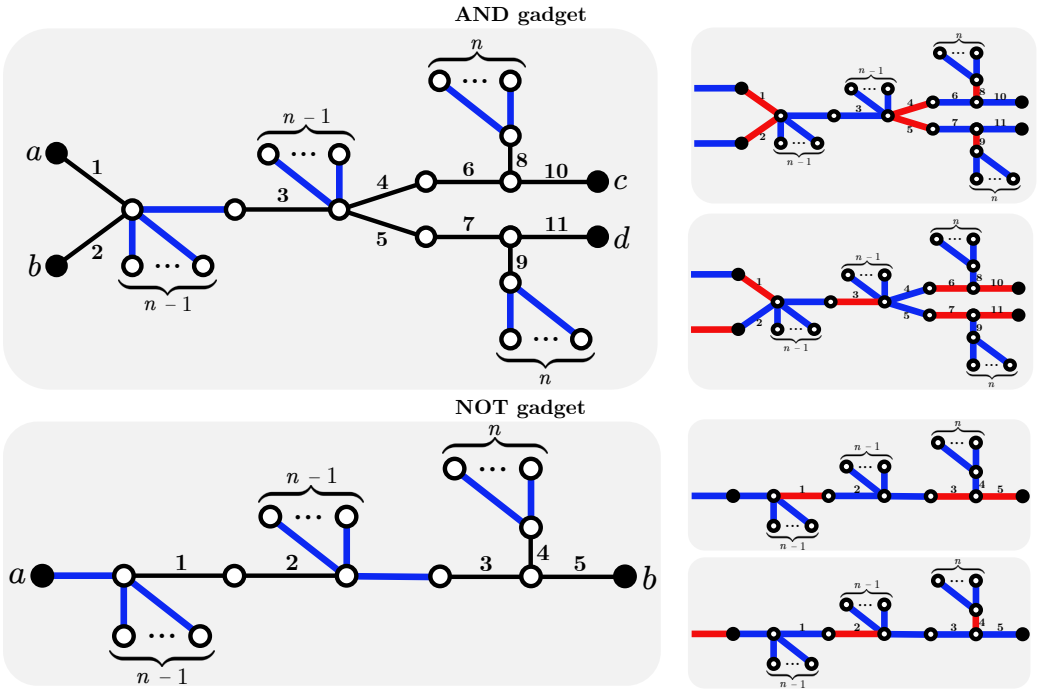


Figure 9 The AND and NOT gadgets used to reduce Planar CVP to the LFM $S_{n,1}$-free problem (where $n \geq 2$). The format is similar to Figure 8.

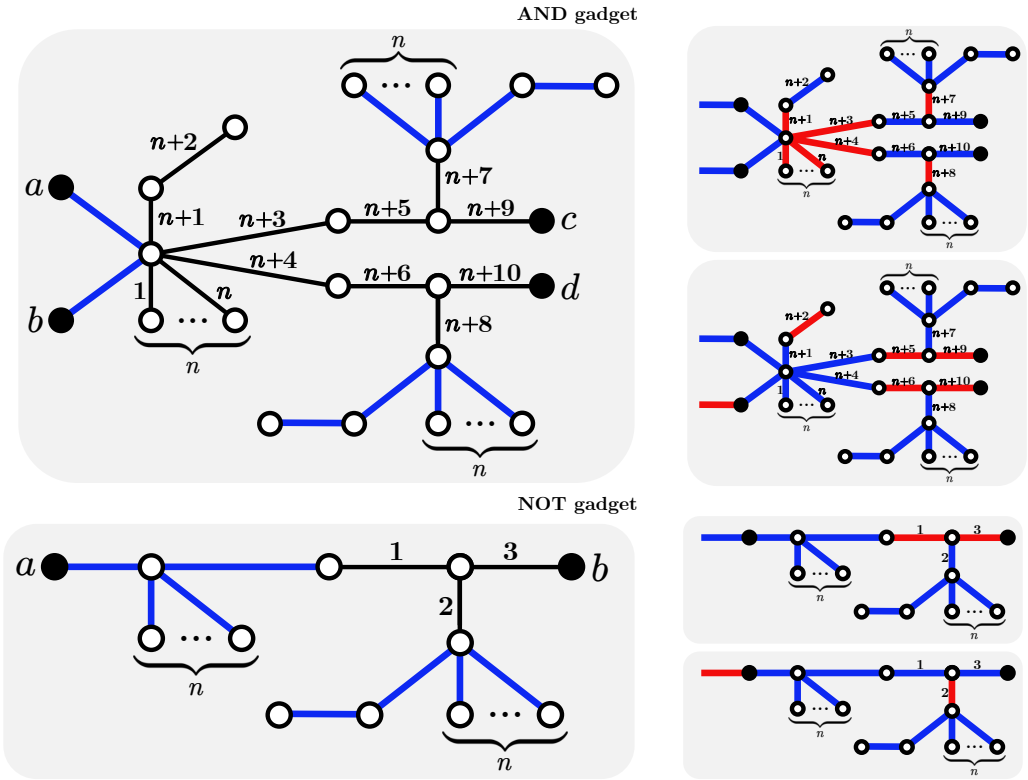


Figure 10 The AND and NOT gadgets used to reduce Planar CVP to the LFM $S_{n,2}$-free problem (where $n \geq 1$). The format is similar to Figure 8.

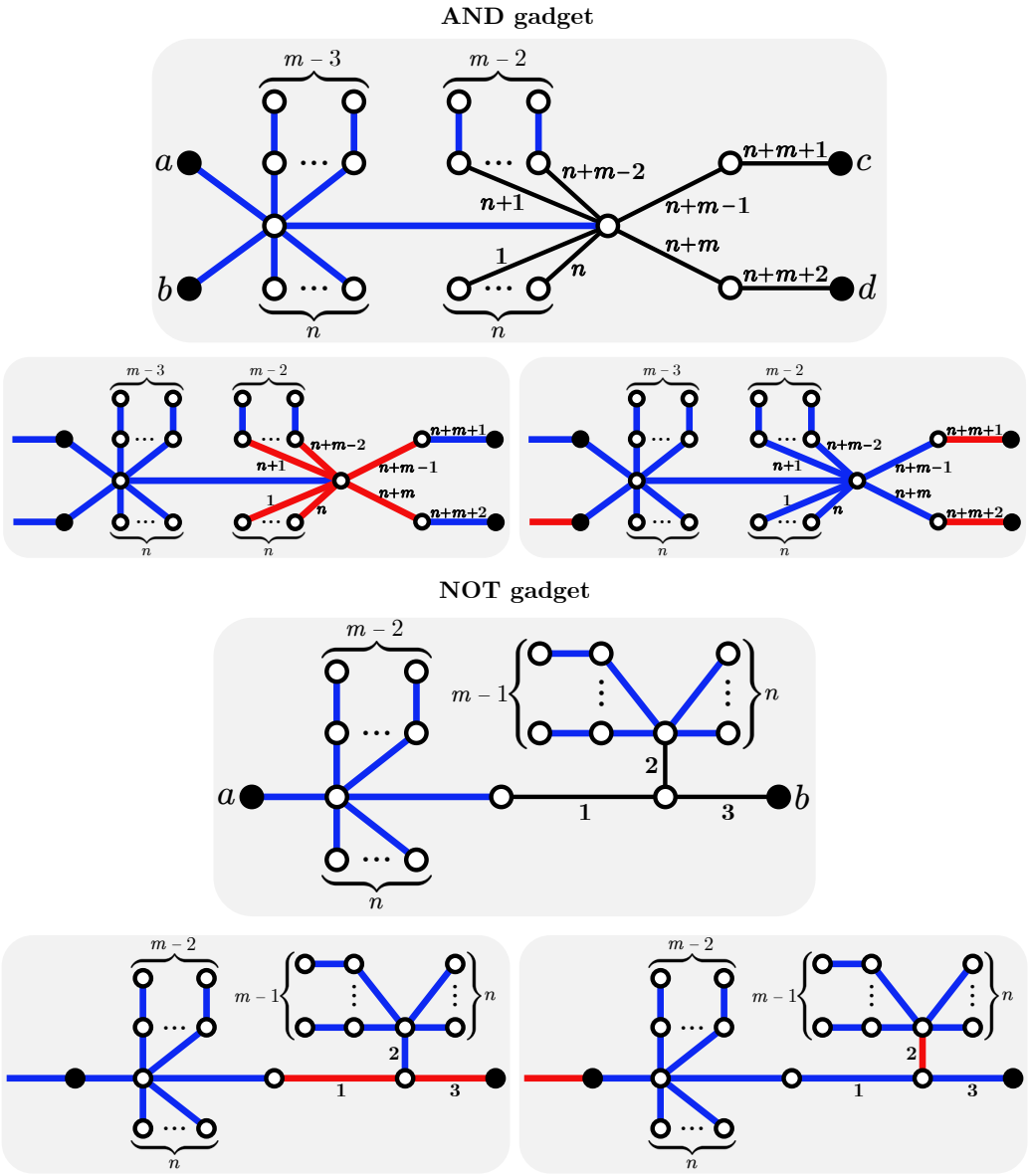


Figure 11 The AND and NOT gadgets used to reduce Planar CVP to the LFM $S_{n,m}$-free problem (where $m \geq 3$). The format is similar to Figure 8.