Contracting to a Longest Path in H-Free Graphs

Walter Kern
Department of Applied Mathematics, University of Twente, The Netherlands
w.kern@twente.nl

Daniël Paulusma
Department of Computer Science, Durham University, UK
daniel.paulusma@durham.ac.uk

Abstract

The Path Contraction problem has as input a graph G and an integer k and is to decide if G can be modified to the k-vertex path P_k by a sequence of edge contractions. A graph G is H-free for some graph H if G does not contain H as an induced subgraph. The Path Contraction problem restricted to H-free graphs is known to be NP-complete if H = claw or $H = P_6$ and polynomial-time solvable if $H = P_5$. We first settle the complexity of Path Contraction on H-free graphs for every H by developing a common technique. We then compare our classification with a (new) classification of the complexity of the problem Long Induced Path, which is to decide for a given integer k, if a given graph can be modified to P_k by a sequence of vertex deletions. Finally, we prove that the complexity classifications of Path Contraction and Cycle Contraction for H-free graphs do not coincide. The latter problem, which has not been fully classified for H-free graphs yet, is to decide if for some given integer k, a given graph contains the k-vertex cycle C_k as a contraction.

2012 ACM Subject Classification Mathematics of computing → Graph theory

Keywords and phrases dichotomy, edge contraction, path, cycle, H-free graph

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2020.22

Funding *Daniël Paulusma*: supported by The Leverhulme Trust (RPG-2016-258).

1 Introduction

The goal in graph modification is to determine if a graph can be quickly modified to some specific family of graphs using some specified set of graph operations. For instance, the Hamiltonian Path problem is that of deciding if a graph can be modified into a path by using only edge deletions. A more general variant of this problem is that of determining the length of a longest path in a graph. Its decision version Long Path is equivalent to deciding if a given graph can be modified into the k-vertex path P_k for some given integer k by a sequence of vertex and edge deletions. As Hamiltonian Path is NP-complete (see [20]), Long Path is NP-complete as well. The same holds for the problem Long Induced Path [20]. The latter problem is to decide if a given graph G contains an induced path on at least k vertices for some given integer k, that is, if G can be modified into P_k by using only vertex deletions.

We mainly consider the variant of the above two problems corresponding to another central graph operation: the *contraction* of an edge uv of a graph G deletes the vertices u and v and replaces them by a new vertex made adjacent to precisely those vertices that were adjacent to u or v in G (without introducing self-loops or parallel edges). A graph G contains a graph F as a *contraction* if G can be modified into F by a sequence of edge contractions.

Contractions to specified graphs play an important role in graph modification problems, e.g. Hamiltonian Path [32, 33], but are also intensively studied in their own right; see, for example, [1, 2, 3, 4, 5, 9, 17, 23, 24, 28, 30, 43, 44, 45, 51] for a number of classical and parameterized complexity results on deciding if a graph G can be modified into a graph F from some specified family $\mathcal{F}$ by at most ℓ edge contractions for some given $\ell \geq 0$. Many of

31st International Symposium on Algorithms and Computation (ISAAC 2020).
Editors: Yixin Cao, Siu-Wing Cheng, and Minming Li; Article No. 22; pp. 22:1–22:18
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

these papers are recent and not only involve rich graph families $\mathcal{F}$, such as bipartite graphs and planar graphs, but also more basic graph families $\mathcal{F}$, such as complete graphs, complete bipartite graphs, cycles, stars, trees, and paths. For example, if $\mathcal{F}$ is the class of complete graphs, then the modification problem becomes the Hadwiger Number problem, which is NP-complete [15]. To give another example, if $\mathcal{F}$ is the class of stars, then we obtain the Connected Vertex Cover problem (see [39]), which is also NP-complete [19].

If $\mathcal{F}$ is the class of paths, which is our focus, we obtain the Path Contraction problem. An equivalent formulation (when considering the classical complexity of the problem) is to set $k = n - \ell$ and ask if the n-vertex input graph G has a graph $F \in \mathcal{F}$ with $|V(F)| \geq k$ as a contraction. This will be the formulation we use:

Path Contraction
Instance: a connected graph G and a positive integer k.
Question: does G contain P_k as a contraction?

The Path Contraction problem is NP-complete as well [8]. Recently, Agrawal et al. [1] gave an exact algorithm faster than $O^*(2^n)$ for it. Due to the computational hardness of Long Path, Long Induced Path and Path Contraction it is natural to restrict the input to special graph classes in order to increase our understanding of the computational hardness of these three path-pattern problems.[1]

Most of the studied graph classes are *hereditary*, that is, closed under vertex deletion. As such, they can be characterized by a family of forbidden induced subgraphs. For a graph H, a graph G is *H-free* if G does not H as an induced subgraph. Hereditary graph classes defined by a small family of forbidden induced subgraphs are well studied, as they enable a *systematic* study into the computational complexity of a graph problem. This is evidenced by extensive studies on (algorithmic and structural) decomposition theorems, e.g., for bull-free graphs [10] or claw-free graphs [11, 31], and surveys for graph problems or parameters, e.g., for Colouring [22, 49] or clique-width [13].

All known NP-hardness results for Hamiltonian Path (see, e.g. [6, 18, 48]) carry over to Long Path. There is a limited number of hereditary graph classes for which the Long Path problem is known to be polynomial-time solvable [25, 35, 36, 46, 47, 52, 53]. The few graph classes for which the Long Induced Path problem is known to be polynomial-time solvable include the classes of k-chordal graphs [21, 37], AT-free graphs [40], graphs of bounded clique-width [12] (see also [40]) and graphs of bounded mim-width (provided a branch decomposition of constant mim-width is given or can be "quickly" computed) [38].

Unlike the Long Path and Long Induced Path problems, Path Contraction is NP-complete even if k is *fixed* (that is, k is not part of the input). To explain this, let F-Contractibility be the problem of deciding if a graph G contains some fixed graph F as a contraction. The complexity classification of F-Contractibility is still open (see [8, 41, 42, 54]), but Brouwer and Veldman [8] showed that already P_4-Contractibility and C_4-Contractibility are NP-complete (where C_k denotes the k-vertex cycle). In fact, P_4-Contractibility problem is NP-complete even for P_6-free graphs [55], whereas Heggernes et al. [29] showed that P_6-Contractibility is NP-complete for bipartite graphs, which was later improved to $k = 5$ in [14]. Moreover, P_7-Contractibility is NP-complete for line graphs [16] and thus also for its superclass of claw-free graphs. Hence, Path Contraction

[1] These three problems are, with respect to basic graph operations, the most natural problems to consider, as the problems of asking for a long (induced) path as an (induced) minor or topological (induced) minor are all equivalent to Long (Induced) Path; we omit the proof details.

is NP-complete for all these graph classes as well. THe Path Contraction problem is polynomial-time solvable for chordal graphs [29]. For hereditary graph classes defined by only one forbidden subgraph, the only known positive result is for P_5-free graphs [55].

Our Results. We first give a dichotomy for Long Induced Path for H-free graphs. Using [55] as a starting point, we then prove our main result: a complete dichotomy for Path Contraction for H-free graphs. In both theorems, H is not part of the input. The run-time of the tractable cases, where H may have arbitrarily large size, is $n^{O(|V(H)|)}$ and $n^{O(|V(H)|^2)}$, respectively. Let $G_1 + G_2 = (V(G_1) \cup V(G_2), E(G_1) \cup E(G_2))$ be the disjoint union of two vertex-disjoint graphs G_1 and G_2, and sG the disjoint union of s copies of G. A *linear forest* is the disjoint union of one or more paths.

▶ **Theorem 1.** *Let H be a graph. If H is a linear forest, then* Long Induced Path *restricted to H-free graphs is polynomial-time solvable; otherwise it is* NP*-complete.*

▶ **Theorem 2.** *Let H be a graph. If H is an induced subgraph of $P_2 + P_4$, $P_1 + P_2 + P_3$, $P_1 + P_5$ or $sP_1 + P_4$ for some $s \geq 0$, then* Path Contraction *restricted to H-free graphs is polynomial-time solvable; otherwise it is* NP*-complete.*

Comparison. Theorem 2 shows that Path Contraction is polynomial-time solvable for H-free graphs for an infinite family of well-structured linear forests H. This is in contrast to the situation for Long Induced Path. Nevertheless, Theorem 1 also gives us an infinite family of polynomial-time solvable cases.

Methodology. We prove Theorem 1 in Section 3 by combining NP-completeness proof for Long Induced Path for graphs of high girt and line graphs with the observation that the length of a longest induced path in a H-fee graph is bounded by a constant c_H.

To extend the aforementioned results from [14, 16, 29, 55] for Path Contraction to the full classification given in Theorem 2 significant more work is required. First, in Section 4, we prove the four new polynomial-time solvable cases of Theorem 2. In each of these cases H is a linear forest, and proving these cases is where our main technical contribution lies. Every linear forest H is P_r-free for some suitable value of r and P_r-free graphs do not contain P_r as a contraction. Hence, it suffices to prove that for each $1 \leq k \leq r-1$, the P_k-Contractibility problem is polynomial-time solvable for H-free graphs for each of the four linear forests listed in Theorem 2. In fact, as P_3-Contractibility is trivial (see also [8]), we only have to consider the cases where $4 \leq k \leq r-1$. Our general technique for doing this is:

*Change an instance of P_k-*Contractibility *for $k \geq 5$ into a polynomial number of instances of P_{k-1}-*Contractibility *until $k = 4$ and solve P_4-*Contractibility *in polynomial time.*

For $k = 4$ we cannot reduce to P_3-Contractibility, as the case $k = 4$ is closely related to the 2-Disjoint Connected Subgraphs problem. This problem takes as input a triple (G, Z_1, Z_2), where G is a graph with two disjoint subsets Z_1 and Z_2 of $V(G)$. It asks if $V(G) \setminus (Z_1 \cup Z_2)$ has a partition (S_1, S_2), such that $Z_1 \cup S_1$ and $Z_2 \cup S_2$ induce connected subgraphs of G. Robertson and Seymour [50] proved that the more general problem k-Disjoint Connected Subgraphs (for k subsets Z_i), a central problem in their project, is polynomial-time solvable as long as the union of the sets Z_i has constant size.[2] However, in our context, Z_1 and Z_2 may have arbitrarily large size. In that case, 2-Disjoint Connected Subgraphs is NP-complete even if $|Z_1| = 2$ (and only Z_2 is large) [55].

[2] If every Z_i has size 2, then we obtain the well-known k-Disjoint Paths problem.

To work around this obstacle, we use the fact that the two outer vertices of the P_4, to which the input graph G must be contracted, may correspond to single vertices u and v of G [55], which we call *P_4-suitable.* We then "guess" u and v so:

We modify, in polynomial time, an instance graph G of P_4-Contractibility *into $O(n^2)$ instances $(G - \{u, v\}, N(u), N(v))$ of* 2-Disjoint Subgraphs.

That is, for each guess (u, v), we seek for a partition (S_u, S_v) of $(V(G)\setminus\{u, v\})\setminus(N(u)\cup N(v))$, such that $N(u) \cup S_u$ and $N(v) \cup S_v$ induce connected subgraphs of G. Then we can contract these two sets to single vertices corresponding to the two middle vertices of the P_4. We also say that we solve the P_4-Suitability problem on instance (G, u, v). In particular, we do not remove u and v from G but exploit their presence in the graph, together with the H-freeness of G, for an extensive analysis of the structure of S_u and S_v of a potential solution (S_u, S_v).

We first show how to check in polynomial time for solutions (S_u, S_v) where either the part of S_u that ensures the connectivity of $N(u) \cup S_u$, or the part of S_v that does this for $N(v) \cup S_v$ has bounded size. We call such solutions *constant.* If we do not find a constant solution, then we exploit their absence. This enables us to branch to a polynomial number of instances of Bipartite Matching; the connection between contractibility and the problem of finding a maximum matching in a bipartite graph is a new (and unexpected) discovery.

In Section 5 we prove the new NP-completeness results. In particular, we prove that P_k-Contractibility, for some suitable value of k, is NP-complete for bipartite graphs of large girth, strengthening the known result for bipartite graphs of [29]. Combining our new results with the NP-completeness results for $K_{1,3}$-free graphs [16] and P_6-free graphs [55] yields Theorem 2.

In Section 6 we pose some open problems. We give the state-of-art of the complexity classification of Long Path for H-free graphs, which is still incomplete. We also discuss the Cycle Contraction problem [7, 26, 27], which is to decide if a given graph contains C_k as a contraction for some given integer k. We show that its (incomplete) complexity classification of Cycle Contraction for H-free graphs differs from the classification of Path Contraction for H-free graphs (Theorem 2).

2 Preliminaries

Throughout the paper we consider finite, undirected graphs with no self-loops.

Let $G = (V, E)$ be a graph. For $S \subseteq V$, let $G[S] = (S, \{uv \in E \mid u, v \in S\})$ be the subgraph of G *induced* by S; then S is *connected* if $G[S]$ is connected. The *neighbourhood* of $v \in V$ is the set $N(v) = \{u \mid uv \in E\}$ and the *closed neighbourhood* is $N[v] = N(v) \cup \{v\}$. The *length* of a path P is its number of edges. The *distance* $\text{dist}_G(u, v)$ between vertices u and v is the length of a shortest path between them. Two disjoint sets $S, T \subset V$ are *adjacent* if there is at least one edge between them; S and T are *(anti)complete* to each other if every vertex of S is (non)adjacent to every vertex of T. The set S *dominates* T if every vertex of T has a neighbour in S. The *subdivision* of an edge $e = uv$ in G replaces e by a new vertex w and two new edges uw and wv.

For a set $H_1, \ldots, H_p$ of graphs, G is *$(H_1, \ldots, H_p)$-free* if G is H_i-free for $i = 1, \ldots, p$. A graph is *complete bipartite* if it has only one vertex or its vertex set can be partitioned into two independent sets A and B that are complete to each other. The *claw* $K_{1,3}$ is the complete bipartite graph with $|A| = 1$ and $|B| = 3$. The graph K_n is the complete graph on n vertices. The *line graph* $L(G)$ of G has the edges of G as vertices and there is an edge between two vertices e_1 and e_2 of G if and only if e_1 and e_2 have a common end-vertex in G. Every line graph is $K_{1,3}$-free.

The *girth* of a graph G that is not a forest is the number of vertices in a shortest induced cycle of G. A subgraph F of a graph G is *spanning* if $V(F) = V(G)$. The next lemma is well known (and we omit its proof).

▶ **Lemma 3.** *Every connected P_4-free graph on at least two vertices has a spanning complete bipartite subgraph, which can be found in polynomial time.*

3 The Proof of Theorem 1

We start with the following lemma.

▶ **Lemma 4.** *Let $p \geq 3$ be some constant. Then* Long Induced Path *is* NP*-complete for graphs of girth at least p.*

Proof. We reduce from Hamiltonian Path. Let G be a graph on n vertices. We subdivide each edge e of G exactly once and denote the set of new vertices v_e by V'. We denote the resulting graph by G' and note that G' is bipartite with partition classes V and V'. We claim that G has a Hamiltonian path if and only if G' has an induced path of length $2n-2$.

First suppose that G has a Hamiltonian path $u_1u_2\cdots u_n$. Then the path on vertices $u_1, v_{u_1u_2}, u_2, \ldots, v_{u_{n-1}u_n}, u_n$ is an induced path of length $2n-2$ in G'. Now suppose that G' has an induced path P' of length $2n-2$. Then either P' starts and finishes with a vertex of V, or P' starts and finishes with a vertex of V'.

In the first case P' contains n vertices of G, so P contains all vertices $u_1, \ldots, u_n$ of G, say we see the vertices of G in this order when we move from the first vertex to the last vertex of P. Then, by the construction of G', we find that $u_1u_2\cdots u_n$ is a Hamiltonian path of G.

In the second case P' contains $n-1$ vertices of V, say vertices $u_1, \ldots, u_{n-1}$ in that order. As P' is an induced path and vertices of V' are only adjacent to vertices of V, this means that the end-vertices of P' are both adjacent to u_n. Hence, we find that $u_1u_2\cdots u_n$ is a Hamiltonian path of G (and the same holds for $u_nu_1\cdots u_{n-1}$).

We note that the girth of G' is twice the girth of G. Now, to obtain the result, we apply this trick p times, that is, we subdivide each edge p times. Let G'' be the resulting graph. Then G'' has girth at least p times the girth of G. Hence, the girth of G'' is at least p, while $V(G'')$ is $|V(G)|+p|E(G)|$, which is polynomial in the size of G, as p is a constant. Moreover, just as we argued above, G has a Hamiltonian path if and only if G'' has an induced path of length $(p+1)(n-1)$. ◀

We also need the following lemma, which we prove by a reduction from Hamilton Path (proof details omitted).

▶ **Lemma 5.** *The* Long Induced Path *problem is* NP*-complete for line graphs.*

We are now ready to prove Theorem 1.

▶ **Theorem 1** (restated). *Let H be a graph. If H is a linear forest, then* Long Induced Path *restricted to H-free graphs is polynomial-time solvable; otherwise it is* NP*-complete.*

Proof. Let G be an H-free graph. If H is a linear forest, then there exists a constant k such that H is an induced subgraph of P_k. This means that the length of a longest induced path of G is at most $k-1$. Hence, we can determine a longest path in G in $O(n^{k-1})$ time by brute force. If H is not a linear forest, then the class of line graphs or the class of graphs of girth at least p for some suitable integer p forms a subclass of the class of H-free graphs. Hence, we can apply Lemma 4 or 5. ◀

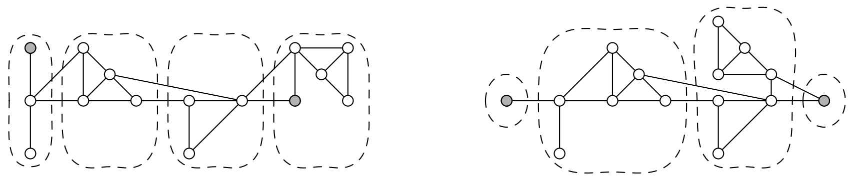

Figure 1 Two P_4-witness structures of a graph; the grey vertices form a P_4-suitable pair [55].

4 The Polynomial-Time Solvable Cases of Theorem 2

A graph G contains a graph F as a contraction if and only if for each $x \in V(F)$ there exists a nonempty subset $W(x) \subseteq V(G)$, such that (i) $W(x)$ is connected; (ii) the set $\mathcal{W} = \{W(x) \mid x \in V(F)\}$ is a partition of $V(G)$; and (iii) for every $x_i, x_j \in V(F)$, $W(x_i)$ and $W(x_j)$ are adjacent in G if and only if x_i and x_j are adjacent in F. By contracting, for each $x \in V(F)$, all edges of a spanning tree of $G[W(x)]$ we obtain the graph F (recall that self-loops or parallel edges are not introduced). The set $W(x)$ is called an *F-witness bag* of G for x. The set $\mathcal{W}$ is an *F-witness structure* of G (which does not have to be unique).

A pair of (non-adjacent) vertices (u, v) of a graph G is *P_k-suitable* for some integer $k \geq 3$ if and only if G has a P_k-witness structure $\mathcal{W}$ with $W(p_1) = \{u\}$ and $W(p_k) = \{v\}$, where $P_k = p_1 \dots p_k$; see Figure 1 for an example. The following known lemma shows why P_k-suitable pairs are of importance.

▶ **Lemma 6** ([55]). *For $k \geq 3$, a graph G contains P_k as a contraction if and only if G has a P_k-suitable pair.*

Lemma 6 leads to the following auxiliary problem, where $k \geq 3$ is a fixed integer, that is, k is not part of the input. See Figure 2 for an example.

P_k-Suitability	
Instance:	a connected graph G and two non-adjacent vertices u, v.
Question:	is (u, v) a P_k-suitable pair?

The next, known observation follows from the fact that P_k-Contractibility is trivial for $k \leq 2$, whereas for $k = 3$ we can use Lemma 6 combined with the triviality of P_3-Suitability.

▶ **Lemma 7** ([8]). *For $k \leq 3$, P_k-Contractibility can be solved in polynomial time.*

We denote the graph obtained from a graph G by contracting $e = uv$ by G/e. We may denote the resulting vertex by u (or v) again and say that we *contracted* e on u (or e on v). We need the following lemma (proof omitted).

▶ **Lemma 8.** *Let $k \geq 4$ and let (G, u, v) be an instance of P_k-Suitability with u and v at distance $d > k$. Let P be a shortest path from u to v. Then (G, u, v) can be reduced in polynomial time to $d - 2$ instances $(G/e, u, v)$, one for each edge $e \in E(P)$ that is not incident to u and v, with $\mathrm{dist}(u, v) = d - 1$, such that (G, u, v) is a yes-instance if and only if at least one of the new instances $(G/e, u, v)$ is a yes-instance of P_k-Suitability.*

In our polynomial-time algorithms for constructing P_k-witness structures we put vertices in certain sets, which we then try to extend to P_k-witness bags (possibly via branching) and we will often apply the following rule:

`Contraction Rule.` If two adjacent vertices s and t end up in the same bag of some potential P_k-witness structure, then contract the edge st.

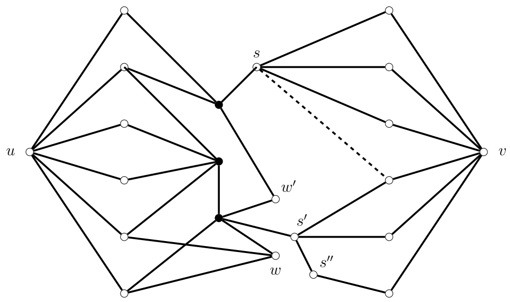

Figure 2 An example of an instance (G, u, v) of P_4-Suitability. Without the dotted line, (G, u, v) has no solution. With the dotted line, (G, u, v) has both a double-sided and single-sided solution but no independent solution. For example, let S'_u be the set of three black vertices. Then $S_u = S'_u \cup \{w, w'\}$ and $S_v = \{s, s', s''\}$ form a double-sided solution that is not only 5-constant (as $|S_u| = 5$) but even 3-constant (as $|S_v| = 3$). An alternative reason for the fact that (S_u, S_v) is 3-constant is that $S'_u \cup N(u)$ is connected and $|S'_u| = 3$. A single-sided 3-constant solution is formed by the independent set consisting of the two top black vertices and w and the set with the bottom black vertex and s, s', s'', w'.

For a graph G, we *apply the* `Contraction Rule` *on some set* $U \subseteq V(G)$ if we contract every edge in $G[U]$. This leads to a smaller instance and $G[U]$ becomes independent. We will exploit both properties in our algorithms. The following known lemma, which is readily seen, shows that applying the `Contraction Rule` preserves H-freeness as long as H is a linear forest.

▶ **Lemma 9.** *Let H be a linear forest and let G be an H-free graph. Then the graph obtained from G after contracting an edge is also H-free.*

We follow the same strategy (outlined in Section 1) for each case, so eventually we check if the input graph can be contracted to P_4 or not. This turns out to be the hardest situation to deal with in our proofs. Due to Lemma 6, we can solve it by checking for each pair of distinct vertices u, v with $N(u) \cap N(v) = \emptyset$ if (G, u, v) is a yes-instance of P_4-Suitability. Let (G, u, v) be an instance of P_4-Suitability. For every P_4-witness structure of G with $W(p_1) = \{u\}$ and $W(p_4) = \{v\}$ (if it exists), every neighbour of u belongs to $W(p_2)$ and every neighbour of v belongs to $W(p_3)$.

Throughout our proofs we let $T(u, v) = V(G) \setminus (N[u] \cup N[v])$ be the set of remaining vertices of G, which still need to be placed in either $W(p_2)$ or $W(p_3)$. We write $T = T(u, v)$ if no confusion is possible. A partition (S_u, S_v) of T is a *solution* for (G, u, v) if $N(u) \cup S_u$ and $N(v) \cup S_v$ are both connected. Hence, a solution (S_u, S_v) for (G, u, v) corresponds to a P_4-witness structure $\mathcal{W}$ of G, where $W(p_1) = \{u\}$, $W(p_2) = N(u) \cup S_u$, $W(p_3) = N(v) \cup S_v$ and $W(p_4) = \{v\}$. A solution (S_u, S_v) for (G, u, v) is *α-constant* for some constant $\alpha \geq 0$ if: either S_u contains a set S'_u of size $|S'_u| \leq \alpha$ such that $N(u) \cup S'_u$ is connected, or S_v contains a set S'_v of size $|S'_v| \leq \alpha$ such that $N(v) \cup S'_v$ is connected; see also Figure 2.

The following lemma is straightforward (we omit its proof) and shows that we can detect constant solutions in polynomial time.

▶ **Lemma 10.** *Let (G, u, v) be an instance of P_4-Suitability. For every constant $\alpha \geq 0$, it is possible to check in $O(n^{\alpha+2})$ time whether or not (G, u, v) has an α-constant solution.*

We need some additional terminology. Let (S_u, S_v) be a solution for an instance (G, u, v) of P_4-Suitability. If $G[S_u]$ and $G[S_v]$ each contain at least one edge, then (S_u, S_v) is *double-sided*. If exactly one of $G[S_u]$, $G[S_v]$ contains an edge, then (S_u, S_v) is *single-sided*. If both S_u and S_v are independent sets, then (S_u, S_v) is *independent*. We refer to Figure 2 for an illustration of these concepts.

We now show, via the auxiliary problem P_k-Suitability, that Path Contraction is polynomial-time solvable for $(P_2 + P_4)$-free graphs. We first give, in Lemma 11, a polynomial-time algorithm for P_4-Suitability for $(P_2 + P_4)$-free graphs. This is the most involved part of our algorithm, and we use it as a *showcase* for illustrating our techniques.

`Outline of the algorithm for` P_4`-Suitability on` $(P_2 + P_4)$`-free graphs (Lemma 11)`

Let (G, u, v) be an instance. Our aim is to reduce to a polynomial number of instances of Bipartite Matching. We may assume that u and v are of distance at least 3 (and thus $N(u) \cap N(v) = \emptyset$). Recall that $T = V(G) \setminus (N[u] \cup N[v])$. To get a handle on the adjacencies between T and $V(G) \setminus T$ we will apply a (constant) number of branching procedures. Each time we branch we obtain, in polynomial time, a polynomial number of new, smaller instances of P_4-Suitability satisfying additional helpful constraints, such that the original instance is a yes-instance if and only if at least one of the new instances is a yes-instance. We then consider each new instance separately until we solve the problem.

1. Exploit the structure of $G[T]$; in particular we prove that $G[T]$ may be assumed to be P_4-free.
2. Check if (G, u, v) has a 7-constant solution. If not, we prove that the absence of 7-constant solutions implies that (G, u, v) has no double-sided solution either. Then if we have not found a solution yet, it remains to test if (G, u, v) has a single-sided solution or an independent solution.
3. Check single-sidedness with respect to u and v independently. In both cases we show that this will lead either to a solution or to a polynomial number of smaller instances, for which we only need to check if they have an independent solution. This will enable us to branch in such a way that afterwards we may assume that T is an independent set and that the solution we are looking for is equivalent to finding a "star cover" of $N(u)$ and $N(v)$ with centers in T.
4 Reduce the "star cover" problem to Bipartite Matching, which we can solve in polynomial time by using the Hopcroft-Karp algorithm [34].

We are now ready to present the full algorithm. We sketch its correctness proof.

▶ **Lemma 11.** P_4-Suitability *can be solved in polynomial time for* $(P_2 + P_4)$*-free graphs.*

Proof. Let (G, u, v) be an instance of P_4-Suitability, where G is a connected $(P_2 + P_4)$-free graph. We may assume without loss of generality that u and v are of distance at least 3, that is, u and v are non-adjacent and $N(u) \cap N(v) = \emptyset$; otherwise (G, u, v) is a no-instance. Recall that $T = V(G) \setminus (N[u] \cup N[v])$ and that we are looking for a partition (S_u, S_v) of T that is a solution for (G, u, v), that is, both $N(u) \cup S_u$ and $N(v) \cup S_v$ must be connected. In order to do so we will construct partial solutions (S'_u, S'_v), which we try to extend to a solution (S_u, S_v) for (G, u, v). We use the `Contraction Rule` from Section 2 on $N(u) \cup S'_u$ and $N(v) \cup S'_v$, so that these two sets will become independent. By Lemma 9, the resulting graph will always be $(P_2 + P_4)$-free. For simplicity, we will denote the resulting instance by (G, u, v) again. After applying the `Contraction Rule` the size of the set T will be reduced if a vertex $t \in T$ was involved in an edge contraction with a vertex from $N(u)$ or $N(v)$. In that case we say that we *contracted* t *away*. We initialise by setting $S'_u = S'_v = \emptyset$ and apply the `Contraction Rule` on $N(u)$ and $N(v)$. Afterwards we can make the following claim.

▷ Claim 1. $N(u)$ and $N(v)$ are independent sets.

Phase 1: Exploiting the structure of G[T]

Suppose $G[T]$ contains an induced P_4 on vertices a_1, a_2, a_3, a_4. If there is a vertex $t \in N(u)$ not adjacent to any vertex of $\{a_1, a_2, a_3, a_4\}$, then $\{u, t\} \cup \{a_1, a_2, a_3, a_4\}$ induces a $P_2 + P_4$ in G, a contradiction. Hence, $\{a_1, a_2, a_3, a_4\}$ dominates $N(u)$. Similarly, $\{a_1, a_2, a_3, a_4\}$ must dominate $N(v)$. Suppose $G[T]$ has another induced P_4 on vertices $\{b_1, b_2, b_3, b_4\}$ such that $\{a_1, a_2, a_3, a_4\} \cap \{b_1, b_2, b_3, b_4\} = \emptyset$. By the same arguments, $\{b_1, b_2, b_3, b_4\}$ also dominates $N(u)$ and $N(v)$. Hence, $N(u) \cup \{a_1, a_2, a_3, a_4\}$ and $N(v) \cup \{b_1, b_2, b_3, b_4\}$ are both connected. We put each remaining vertex of T into either S_u or S_v (which is possible, as G is connected). This yields a (4-constant) solution for (G, u, v). From now on, assume that $G[T]$ contains no induced copy of P_4 that is vertex-disjoint from $a_1a_2a_3a_4$.

Branching I ($O(n^{16})$ branches)

We branch by considering every possibility for each a_i ($1 \leq i \leq 4$) to go into either S_u or S_v for some solution (S_u, S_v) of (G, u, v) (if it exists). We then branch into $O(n^{16})$ possibilities to ensure that we contracted each a_i away. We consider each resulting instance, which we denote by (G, u, v) again and for which the following claims hold. The first claim holds immediately. We omit the proof of the second claim.

▷ Claim 2. $G[T]$ is P_4-free.

▷ Claim 3. Let (S_u, S_v) be a solution for (G, u, v) that is not 7-constant. Let t, x_1, x_2 be three vertices of T with $tx_1 \notin E(G)$, $tx_2 \notin E(G)$ and $x_1x_2 \in E(G)$. If t, x_1, x_2 are in S_u, then every neighbour of t in $N(u)$ is adjacent to at least one of x_1, x_2. If t, x_1, x_2 are in S_v, then every neighbour of t in $N(v)$ is adjacent to at least one of x_1, x_2.

Phase 2: Excluding 7-constant solutions and double-sided solutions

By using Claim 3 we show the following claim on double-sided solutions (proof omitted).

▷ Claim 4. If (G, u, v) has a double-sided solution, then (G, u, v) has a 7-constant solution.

We now check in polynomial time if (G, u, v) has a 7-constant solution by using Lemma 10. If so, then we are done. From now on assume that (G, u, v) has no 7-constant solution. Then, by Claim 4 it follows that (G, u, v) has no double-sided solution. It remains to check if (G, u, v) has a single-sided solution or an independent solution. If (G, u, v) has a single-sided solution (S_u, S_v) that is not independent, then exactly one of S_u or S_v is independent. Our algorithm first looks for a solution (S_u, S_v) where S_u is independent. We say that it is doing a *u-feasibility check*. If afterwards we have not found such a solution, then our algorithm will perform a *v-feasibility check*, which is the same check but now performed with respect to v.

Phase 3: Doing a u-feasibility check

We start by exploring the structure of a solution (S_u, S_v) that is either single-sided or independent, and where S_u is an independent set. As S_u and $N(u)$ are both independent sets, $G[N(u) \cup S_u]$ is a connected bipartite graph. Hence, S_u contains a set S_u^*, such that S_u^* dominates $N(u)$. We assume that S_u^* has minimum size. For $s \in S_u^*$, let $Q(s)$ be the set that consists of all neighbours of s in $N(u)$ that are not adjacent to any vertex in $S_u^* \setminus \{s\}$. Then, for each $s \in S_u^*$, the set $Q(s)$ is nonempty, as otherwise we can remove s from S_u^*, contradicting our assumption that S_u^* has minimum size. We call the vertices of $Q(s)$ the *private* neighbours of s with respect to S_u^*. We can show that the following holds if (G, u, v) has a solution (S_u, S_v) in which S_u is an independent set (proof omitted):

(P) The set S_u contains a subset S_u^* of size at least 2 that dominates $N(u)$, such that each vertex in S_u^* has a nonempty set $Q(s)$ of private neighbours with respect to S_u^*, and moreover, the set $N(u) \setminus Q_u$, where $Q_u = \bigcup_{s \in S_u^*} Q(s)$, is nonempty and complete to S_u^*.

▶ Remark. We emphasize that S_u^* is unknown to the algorithm, as we constructed it from the unknown S_u, and consequently, our algorithm does not know (yet) the sets $Q(s)$. We also point out that the set S_u^* and thus the sets $Q(s)$ might not be unique. However, this is irrelevant and for us the existence of at least one set S_u^* suffices.

Phase 3a: Reducing $\mathbf{N(u) \setminus Q_u}$ to a single vertex $\mathbf{w_u}$

We will now branch into a polynomial number of smaller instances, in which $N(u) \setminus Q_u$ consists of just one single vertex w_u, which we can even identify.

Branching II ($O(n^4)$ branches)

We will determine exactly those vertices of $N(u)$ that belong to Q_u via some branching, under the assumption that (G, u, v) has a solution (S_u, S_v), where S_u is independent, that satisfies (P). By (P), S_u^* consists of at least two (non-adjacent) vertices s and s'. Let $w \in Q(s)$ and $w' \in Q(s')$. We branch by considering all possible choices of choosing these four vertices. This leads to $O(n^4)$ branches, which we each process in the way described below.

If we selected s and s' correctly, then s, s' belong to an independent set S_u that together with $S_v = T \setminus S_u$ forms a solution for (G, u, v) that is not 7-constant. This implies that $\{s, s'\}$ does not dominate $N(u)$. Hence, $N(u) \setminus \{w, w'\} \neq \emptyset$. For each $w^* \in N(u) \setminus \{w, w'\}$, we do as follows. If w^* is adjacent to both s and s', then w^* must belong to $N(u) \setminus Q_u$ due to property (P). In the other case, that is, if w^* is adjacent to at most one of s, s', then w^* must belong to Q_u, again by property (P). Hence, we have identified in polynomial time the (potential) sets Q_u and $N(u) \setminus Q_u$. Moreover, by applying the `Contraction Rule` on $N(u) \cup \{s, s'\}$ we can contract s and s' away. This also contracts all of $N(u) \setminus Q_u$ into a single vertex which, as we mentioned above, we denote by w_u. Thus w_u is complete to S_u^*. We denote the resulting instance by (G, u, v) again. We also let $T_1 = N(w_u) \cap T$ and $T_2 = T \setminus T_1$. Note that $S_u^* \subseteq T_1$. As S_u^* dominates $N(u)$ and every vertex of S_u^* is adjacent to w_u, we find that $N(u) \cup S_u^*$ is connected, and we can modify our instance (G, u, v) such that the following claim holds (proof omitted):

▷ Claim 5. T_2 is an independent set that is anticomplete to $N(v)$.

By definition, no vertex of T_2 is adjacent to w_u either. In a later stage we will modify T_2 and this property may no longer hold. However, we will always maintain Claim 5.

By Lemma 10 we check in polynomial time if (G, u, v) has a 7-constant solution. If so, then we are done. From now on suppose that (G, u, v) has no 7-constant solution. Recall that we are still looking for a single-sided or independent solution (S_u, S_v), where S_u is an independent set. We first show that we can modify G in polynomial time such that afterwards the following claim holds, while maintaining Claim 5 (we omit the proof of Claim 6).

▷ Claim 6. $G[T]$ is $(K_3 + P_1)$-free.

We will now do some further branching to obtain $O(n)$ smaller instances in which $G[T_1]$ is K_3-free, such that the following holds. If one of these new instances has a solution, then (G, u, v) has a solution. If none of these new instances has a solution, then (G, u, v) may still have a solution (S_u, S_v), but in that case S_u is not an independent set while S_v must be an independent set; this will be verified when we do the v-feasibility check.

Branching III ($O(n)$ branches)

We consider all possibilities of putting one vertex $t \in T_1$ in S_u. This leads to $O(n)$ branches. For each branch we do as follows. As t is adjacent to w_u (because $t \in T_1$), we contract t away using the `Contraction Rule` on $N(u) \cup \{t\}$. As S_u is independent, every neighbour t' of t in T_1 must go to S_v. If such a neighbour t' is adjacent to a vertex of $N(v)$, this means that we may contract t' away by using the `Contraction Rule` on $N(v) \cup \{t'\}$. If t' has no neighbour in $N(v)$, then we put t' in T_2. By the `Contraction Rule` we may contract all edges between t' and its neighbours in T_2, such that T_2 is an independent set again that is anticomplete to $N(v)$, so Claim 5 is still valid (but T_2 may now contain vertices adjacent to w_u). Denote the resulting instance by (G, u, v) again. As $G[T]$, and thus, $G[T_1]$ is $(K_3 + P_1)$-free due to Claim 6, we find afterwards that the following holds for each branch.

▷ Claim 7. $G[T_1]$ is K_3-free.

By Lemma 10 we check in polynomial time if (G, u, v) has a 7-constant solution. From now on assume not. Then (G, u, v) has no double-sided solution either, as then the original instance would have a double-sided solution, which we already ruled out (alternatively, apply Claim 4).

Phase 3b: Looking for independent solutions after branching
We will now branch to $O(n^5)$ smaller instances for which the goal is to find an independent solution. If one of the newly created instances has a solution, then (G, u, v) has a solution. If no new instance has a solution, then (G, u, v) may still have a solution (S_u, S_v). However, in that case S_u is not independent and S_v must be an independent set. This will be verified when doing the v-feasibility check. An instance (G, u, v) satisfies the $(*)$-property if:

$(*)$ *If* (G, u, v) *has a solution* (S_u, S_v) *where* S_u *is an independent set, then* (G, u, v) *has an independent solution.*

Let $D_1, \ldots, D_q$ be the connected components of $G[T]$ for some $q \geq 1$. If each D_i has size 1, then $G[T]$ is independent. Hence, any solution for (G, u, v) will be independent, and thus $(*)$ holds already. Now suppose at least one of $D_1, \ldots, D_q$, say D_1, has more than one vertex. We first consider the case where another D_i, say D_2, also has more than one vertex. We claim that $(*)$ is again satisfied already (proof omitted). So, from now on, assume that D_1 has more than one vertex and $D_2, \ldots, D_q$ each have a single vertex. Recall that T_2 is an independent set that is anticomplete to $N(v)$ due to Claim 5. Suppose $t \in T_2$ does not belong to D_1. Then t is an isolated vertex of $G[T]$ that is not adjacent to any vertex of $N(v)$. As G is connected, t is adjacent to at least one vertex of $N(u)$. We apply the `Contraction Rule` on $N(u) \cup \{t\}$ to contract t away. Afterwards, we find that every vertex of T_2 must belong to D_1. Let $B_1, \ldots, B_p$ be the connected components of $G[T_1 \cap V(D_1)]$ for some $p \geq 1$. By Claim 2, $G[T]$, and thus $G[T_1 \cap V(D_1)]$, is P_4-free (note that we only contracted edges during the branching and thus maintained P_4-freeness due to Lemma 9). As $G[T_1]$ is also K_3-free by Claim 7, each B_i is a complete bipartite graph on one or more vertices due to Lemma 3.

First suppose $p = 1$. Recall that T_2 is an independent set by Claim 5 that belongs to D_1.

Branching IV ($O(n^2)$ branches)
In this case we can branch into $O(n^2)$ new and smaller instances, such that (G, u, v) has a solution (S_u, S_v), in which S_u is an independent set, if and only if one of these new instances has such a solution. Moreover, we can show that each new instance, which we denote by (G, u, v) again, will either have the $(*)$ property or $p \geq 2$ holds (proof omitted).
So, if $(*)$ does not yet hold, (G, u, v) is an instance with $p \geq 2$. By Lemma 3 and because D_1 is connected and P_4-free, D_1 has a spanning complete bipartite graph B^*. As $p \geq 2$, all vertices of $V(B_1) \cup \cdots \cup V(B_p)$ belong to the same partition class of B^*. By definition, these vertices are in T_1. Hence, as T_2 is an independent set in D_1, all vertices of T_2 form the other bipartition class of B^*. Thus, T_2 is complete to $T_1 \cap V(D_1)$.

Branching V ($O(n)$ branches)
Every vertex of T_2 will belong to S_v in any solution (S_u, S_v) where S_u is an independent set, but without having any neighbours in $N(v)$ due to Claim 5. This means that S_v contains at least one vertex t of $V(D_1) \cap T_1$. We branch by considering all possibilities of choosing this vertex t. Indeed, as T_2 is complete to T_1, it suffices to check single vertices $t \in T_1$ that have a neighbour in $N(v)$. This leads to $O(n)$ branches. For each branch we do as follows. We contract the vertices of $T_2 \cup \{t\}$ away using the `Contraction Rule` on $N(v) \cup T_2 \cup \{t\}$. We denote the resulting instance by (G, u, v) and observe that $T_2 = \emptyset$, so $T = T_1$.
Note that $G[T] = G[T_1]$ now consists of connected components $B'_1, \ldots, B'_{p'}$ for some $p' \geq 1$, where each B'_i is complete bipartite. If each B'_i has size 1, then $G[T]$ is independent. Hence, any solution for (G, u, v) will be independent, and thus $(*)$ holds. Now suppose at least one of $B'_1, \ldots, B'_{p'}$, say B'_1, has more than one vertex. If another B'_i, also has more than one vertex, then $(*)$ holds: we can show this in the same way as when we proved this for the sets $D_1, \ldots, D_q$. From now on, assume that B'_1 has more than one vertex and $B'_2, \ldots, B'_{p'}$ have only one vertex. So, in particular, B'_1 is complete bipartite and has at least two vertices.

Branching VI ($O(n^2)$ branches)
We first consider each possibility of choosing one vertex $t \in B'_1$ to be placed in S_u, leading to $O(n)$ branches. Afterwards we perform $O(n)$ further branches. For each new instance, which we denote by (G, u, v) again, we can show that $T = T_1$ and T is independent, leading to $(*)$; we omit the proof of this claim.
If we did not yet find a solution, then by achieving $(*)$ we have further reduced the problem to $O(n^5)$ instances, for which we search for an independent solution. We consider these new instances one by one, and we denote the instance under consideration by (G, u, v) again.

Phase 3c: Searching for private solutions
In this phase we introduce a new type of independent solution after some branching.

Branching VII. ($O(n^4)$ branches)
First we process $N(v)$ via $O(n^4)$ branches in the same way as we did for $N(u)$ in Branching II. Hence, if (G, u, v) has a solution (S_u, S_v) in which S_u and S_v are independent sets, then:
(P1) S_u contains a subset S^*_u of size at least 2 that dominates $N(u)$, such that each $s \in S^*_u$ has a nonempty set $Q_u(s)$ of private neighbours with respect to S^*_u, and moreover, the set $N(u) \setminus Q_u$, where $Q_u = \bigcup Q_u(s)$, consists of a single vertex w_u that is complete to S^*_u.
(P2) S_v contains a subset S^*_v of size at least 2 that dominates $N(v)$, such that each $s \in S^*_v$ has a nonempty set $Q_v(s)$ of private neighbours with respect to S^*_v, and moreover, the set $N(v) \setminus Q_v$, where $Q_v = \bigcup Q_v(s)$, consists of a single vertex w_v that is complete to S^*_v.
We call an independent solution (S_u, S_v) satisfying (P1) and (P2) *private*.

By now all branches are guaranteed to have private solutions or no solutions at all. Thus we need only to search for private ones. While doing this we may modify the instance (G, u, v), but we will always ensure that private solutions are pertained. In particular, if we contract a vertex $t \in S^*_u$ to w_u using the `Contraction Rule` on $N(u) \cup \{t\}$, this leads to a private solution (S_u, S_v) with $t \notin S^*_u$. Then all private neighbours of t become adjacent to w_u and, by the `Contraction Rule`, they get contracted to w_u. However, if $t \notin S^*_u$, then contracting t to w_u will make the neighbours of t in $N(u)$ adjacent to w_u and the `Contraction Rule` contracts these to w_u. Consequently, some vertices in S^*_u may have no private neighbours in $N(u)$ and hence leave S^*_u. If this reduces $|S^*_u|$ to 1, we will notice this by checking, in polynomial time (Lemma 10), for 1-constant solutions. If we find a 1-constant solution, then we stop and conclude that our original instance is a yes-instance. Otherwise, we know that $|S^*_u| \geq 2$, and hence private solutions pertain (should such solutions exist at all). We will always perform this test implicitly whenever we apply the `Contraction Rule`.

We show the next two claims (proofs omitted).

▷ Claim 8. Every vertex of T is adjacent to both w_u and w_v.

▷ Claim 9. If (G, u, v) has a private solution, then $G[T]$ must be the disjoint union of one or more complete bipartite graphs.

By Claim 9 we may assume that $G[T]$ is the disjoint union of one or more complete bipartite graphs; otherwise we discard the branch (as we search for a private solution). We now prove that T can be changed into an independent set via some branching. Suppose T is not independent yet. Let $B_1, \ldots, B_r$, for some $r \geq 1$, denote the connected components of $G[T]$ with at least one edge (note that $G[T]$ may also contain some isolated vertices). By Claim 9, every B_i is complete bipartite. By Claim 10 (proof omitted) we may assume that $r \leq 3$; otherwise we discard the branch.

▷ Claim 10. If (G, u, v) has a private solution, then $r \leq 3$.

Branching VIII ($O(1)$ branches)
As $r \leq 3$ by Claim 10, we can branch to obtain $O(1)$ smaller instances, such that (G, u, v) has a private solution if and only if at least one of these new instances has a private solution. Moreover, for each new instance, which we denote by (G, u, v) again we can show that T is an independent set (proof omitted). As T is an independent set, the sets S_u and S_v of any solution (S_u, S_v) will be independent (should (G, u, v) have a solution). Recall that $\{w_u, w_v\}$ is complete to T by Claim 8. As we search for a private solution (S_u, S_v), the following two claims can be shown (proofs omitted).

▷ Claim 11. Let s and t be any two distinct vertices of T. Then we may assume without loss of generality that either $N(u) \cap N(s) \cap N(t) = \{w_u\}$; or $N(u) \cap N(s) = N(u) \cap N(t)$; or $\{s, t\}$ dominates $N(u)$. Similarly, we may assume without loss of generality that either $N(v) \cap N(s) \cap N(t) = \{w_v\}$; or $N(v) \cap N(s) = N(v) \cap N(t)$; or $\{s, t\}$ dominates $N(v)$.

▷ Claim 12. Let s and t be two distinct vertices in T such that $\{s, t\}$ dominates $N(u) \cup N(v)$. Then (G, u, v) has a 2-constant solution.

We continue as follows. By Lemma 10 we check in polynomial time if (G, u, v) has a 2-constant solution. If so, then we are done. Otherwise, we obtain the following claim.

▷ Claim 13. We may assume without loss of generality that every pair of (distinct) vertices $\{s, t\}$ in T does not dominate $N(u)$; hence, $\{s, t\}$ may only dominate $N(v)$.

We call a pair of vertices s, t of T a 2-*pair* if $\{s, t\}$ dominates $N(v)$. Let T_v be the set of vertices of T involved in a 2-pair. We show the following claim (proof omitted).

▷ Claim 14. $T_v = \emptyset$.

Phase 3d: Translating the problem into a bipartite matching problem
We now translate the instance (G, u, v) into an instance of Bipartite Matching. Recall that w_u and w_v are the vertices in $N(u)$ and $N(v)$ that are complete to T. By Claims 11 and 14 we can partition $N(u) \setminus \{w_u\}$ into sets $N_1(u) \cup \cdots \cup N_q(u)$ for some $q \geq 1$ such that two vertices of $N(u)$ have the same neighbours in T if and only if they both belong to $N_h(u)$ for some $h \in \{1, \ldots, q\}$. Similarly, we can partition $N(v) \setminus \{w_v\}$ into sets $N_1(v) \cup \cdots \cup N_r(v)$ for some $r \geq 1$ such that two vertices of $N(v)$ have the same set of neighbours in T if and only if they both belong to $N_i(v)$ for some $i \in \{1, \ldots, r\}$. We may remove all but one vertex of each $N_h(u)$ and $N_i(v)$ to obtain an equivalent instance, which we denote by (G, u, v) again.

Let G' be the graph obtained from G by removing u, v, w_u, w_v and all edges between $N(u)$ and $N(v)$. Then G' is bipartite with partition sets $(N(u) \setminus \{w_u\}) \cup (N(v) \setminus \{w_v\})$ and T. It remains to compute a maximum matching M in G', which can be done in polynomial time via the Hopcroft-Karp algorithm [34]. If $|M| = |N(u)| + |N(v)| - 2$, then each vertex in $(N(u) \setminus \{w_u\}) \cup (N(v) \setminus \{w_v\})$ is incident to an edge of M, and hence, we found a (private) solution for (G, u, v). If $|M| < |N(u)| + |N(v)| - 2$, then (G, u, v) has no (private) solution; we discard the branch. This concludes the u-feasibility check. If we found a solution, we translate it in polynomial time to a solution for the original instance; else we enter the last phase.

Phase 4: Doing a v-feasibility check

We repeat the same steps as in Phase 3 and this concludes the description of our algorithm. The correctness of our algorithm follows from the above description. To analyze its run-time, the branching (Branching I-VIII) yields a total of $O(n^{30})$ branches. As explained in each step above, processing each branch created in Branching I-VI until we start branching again takes polynomial time. Checking for 1-constant solutions to ensure survival of private solutions takes polynomial time as well. As processing the branches created in Branch VII-VIII takes polynomial time, we conclude that the total running time of our algorithm is polynomial. ◀

We use Lemma 8 to obtain Lemma 12, which we use for Lemma 13; we omit both proofs. Lemmas 6, 7, 11–13 together with the fact that a $(P_2 + P_4)$-graph has no P_7 as a contraction yields Theorem 14.

▶ **Lemma 12.** P_5-Suitability *can be solved in polynomial time for* $(P_2 + P_4)$*-free graphs.*

▶ **Lemma 13.** P_6-Suitability *can be solved in polynomial time for* $(P_2 + P_4)$*-free graphs.*

▶ **Theorem 14.** Path Contraction *is polynomial-time solvable for* $(P_2 + P_4)$*-free graphs.*

We prove the other polynomial-time cases, $H = P_1 + P_2 + P_3$, $H = P_1 + P_5$ and $H = sP_1 + P_4$ $(s \geq 1)$, by the same strategy but in a less involved way (we omit the details).

5 The NP-Complete Cases of Theorem 2

A *hypergraph* $\mathcal{H}$ is a pair $(Q, \mathcal{S})$, where $Q = \{q_1, \ldots, q_m\}$ is a set of m *elements* and $\mathcal{S} = \{S_1, \ldots, S_n\}$ is a set of n *hyperedges*, which are subsets of Q. A 2-*colouring* of $\mathcal{H}$ is a partition of Q into two (nonempty) sets Q_1 and Q_2 with $Q_1 \cap S_j \neq \emptyset$ and $Q_2 \cap S_j \neq \emptyset$ for each S_j. The Hypergraph 2-Colourability problem is to decide if a given hypergraph has a 2-colouring. This problem is NP-complete even for hypergraphs $\mathcal{H}$ with $S_i \neq \emptyset$ for $1 \leq i \leq n$ and $S_n = Q$. Brouwer and Veldman [8] proved that P_4-Contractibility is NP-complete by a reduction from Hypergraph 2-Colourability. That is, from a hypergraph $\mathcal{H}$ they built a graph $G_{\mathcal{H}}$, such that $\mathcal{H}$ has a 2-colouring if and only if $G_{\mathcal{H}}$ has P_4 as a contraction:

- Construct the *incidence graph* of $(Q, \mathcal{S})$, which is the bipartite graph with partition classes Q and $\mathcal{S}$ and an edge between two vertices q_i and S_j if and only if $q_i \in S_j$.
- Add a set $\mathcal{S}' = \{S'_1, \ldots, S'_n\}$ of n new vertices, where we call S'_j the *copy* of S_j.
- For $i = 1, \ldots m$ and $j = 1, \ldots, n$, add an edge between q_i and S'_j if and only if $q_i \in S_j$.
- For $j = 1, \ldots, n$ and $\ell = 1, \ldots, n$, add an edge between S_j and S'_ℓ, so the subgraph induced by $\mathcal{S} \cup \mathcal{S}'$ will be complete bipartite.
- For $h = 1, \ldots, m$ and $i = 1, \ldots, m$, add an edge between q_h and q_i, so Q will be a clique.
- Add two new vertices t_1 and t_2.
- For $j = 1, \ldots, n$, add an edge between t_1 and S_j, and between t_2 and S'_j.

As mentioned, Brouwer and Veldman [8] proved a hypergraph $\mathcal{H}$ has a 2-colouring if and only if $G_{\mathcal{H}}$ has P_4 as a contraction. The graph $G_{\mathcal{H}}$ is P_6-free [55]. We observe that $G_{\mathcal{H}}$ is also $(2P_1 + 2P_2, 3P_2, 2P_3)$-free. Hence, we obtain the following:

▶ **Lemma 15.** *P_4-Contractibility is NP-complete for $(2P_1+2P_2, 3P_2, 2P_3, P_6)$-free graphs.*

By modifying $G_{\mathcal{H}}$ we show Theorem 16 (proof details omitted).

▶ **Theorem 16.** *Let $p \geq 4$ be some constant. Then P_{2p}-Contractibility is NP-complete for bipartite graphs of girth at least p.*

Theorem 16 implies that Path Contraction is NP-complete for H-free graphs if H has a cycle. Combining Lemma 15 and Theorem 16 with the known NP-completeness result for $K_{1,3}$-free graphs [16] yields the NP-complete part of Theorem 2.

6 Conclusions

We completely classified the complexities of Long Induced Path and Path Contraction for H-free graphs. For Long Path, the classification is still incomplete. It is known that Hamiltonian Path, and thus Long Path, is NP-complete for chordal bipartite graphs and strongly chordal split graphs [48], and thus for H-free graphs if H has a cycle or an induced $2P_2$. Moreover, Hamiltonian Path is NP-complete for line graphs [6] and thus for H-free graphs if H is a forest of maximum degree at least 3. On the positive side, Long Path is polynomial-time solvable for cocomparability graphs [36, 47] and thus for P_4-free graphs. This leaves open, for both problems, the cases $H = sP_1 + P_r$ $(3 \leq r \leq 4$ and $s \geq 1)$, $H = sP_1 + P_2$ $(s \geq 2)$ and $H = sP_1$ $(s \geq 3)$.

The classification of Cycle Contraction for H-free graphs is also open. Hammack proved that this problem is NP-complete for general graphs [27] but polynomial-time solvable for planar graphs [26]. It is also NP-complete for $K_{1,3}$-free graphs [16]. The classifications of Cycle Contraction and Path Contraction do not coincide for H-free graphs. By Theorem 2, the former is polynomial for $(P_2 + P_4)$-free graphs. However, we can show the following result by inspecting the NP-hardness gadget of Brouwer and Veldman [8] for C_4-Contractibility (proof details omitted).

▶ **Theorem 17.** *C_4-Contractibility, and thus Cycle Contraction, is NP-complete for $(P_2 + P_4)$-free graphs.*

Preliminary research suggests that our techniques are applicable to other contractibility and connectivity problems as well, such as contracting to large (subdivided) stars or claws (the problem of contracting to a largest star is known as Connected Vertex Cover [39]).

Another natural question is how Path Contraction behaves on hereditary graph classes with more than one forbidden induced subgraph. Recall that Path Contraction is polynomial-time solvable for P_5-free graphs and NP-complete for P_6-free graphs. It would be interesting to determine the complexity of Path Contraction for $(K_{1,3}, P_t)$-free graphs for $t \geq 6$. Other problems, such as Graph Colouring, are also open for these graph classes and a better understanding of their structure is needed.

References

1 Akanksha Agrawal, Fedor Fomin, Daniel Lokshtanov, Saket Saurabh, and Prafullkumar Tale. Path contraction faster than 2^n. *Proc. ICALP 2019, LIPIcs*, 132:11:1–11:13, 2019.
2 Akanksha Agrawal, Lawqueen Kanesh, Saket Saurabh, and Prafullkumar Tale. Paths to trees and cacti. *Proc. CIAC 2017, LNCS*, 10236:31–42, 2017.
3 Akanksha Agrawal, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. Split contraction: The untold story. *ACM Transactions on Computation Theory*, 11:18:1–18:22, 2019.

4 Akanksha Agrawal, Saket Saurabh, and Prafullkumar Tale. On the parameterized complexity of contraction to generalization of trees. *Theory of Computing Systems*, 63:587–614, 2019.

5 Rémy Belmonte, Petr A. Golovach, Pim van 't Hof, and Daniël Paulusma. Parameterized complexity of three edge contraction problems with degree constraints. *Acta Informatica*, 51:473–497, 2014.

6 Alan A. Bertossi. The edge Hamiltonian path problem is NP-complete. *Information Processing Letters*, 13:157–159, 1981.

7 D. Blum. *Circularity of graphs.* Virginia Polytechnic Institute and State University, 1982.

8 A. E. Brouwer and Henk Jan Veldman. Contractibility and NP-completeness. *Journal of Graph Theory*, 11:71–79, 1987.

9 Leizhen Cai and Chengwei Guo. Contracting few edges to remove forbidden induced subgraphs. *Proc. IPEC 2013, LNCS*, 8246:97–109, 2013.

10 Maria Chudnovsky. The structure of bull-free graphs II and III - A summary. *Journal of Combinatorial Theory, Series B*, 102:252–282, 2012.

11 Maria Chudnovsky and Paul D. Seymour. The structure of claw-free graphs. *Surveys in Combinatorics, London Mathematical Society Lecture Note Series*, 327:153–171, 2005.

12 Bruno Courcelle, Johann A. Makowsky, and Udi Rotics. Linear time solvable optimization problems on graphs of bounded clique-width. *Theory of Computing Systems*, 33:125–150, 2000.

13 Konrad K. Dabrowski, Matthew Johnson, and Daniël Paulusma. Clique-width for hereditary graph classes. *Proc. BCC 2019, London Mathematical Society Lecture Note Series*, 456:1–56, 2019.

14 Konrad K. Dabrowski and Daniël Paulusma. Contracting bipartite graphs to paths and cycles. *Information Processing Letters*, 127:37–42, 2017.

15 David Eppstein. Finding large clique minors is hard. *Journal of Graph Algorithms and Applications*, 13:197–204, 2009.

16 Jirí Fiala, Marcin Kamiński, and Daniël Paulusma. A note on contracting claw-free graphs. *Discrete Mathematics & Theoretical Computer Science*, 15:223–232, 2013.

17 Fedor Fomin, Daniel Lokshtanov, Ivan Mihajlin, Saket Saurabh, and Meirav Zehavi. Computation of hadwiger number and related contraction problems: Tight lower bounds. *Proc. ICALP 2020, LIPCcs*, 168:49:1–49:18, 2020.

18 M. R. Garey, David S. Johnson, and Robert Endre Tarjan. The planar hamiltonian circuit problem is NP-complete. *SIAM Journal on Computing*, 5:704–714, 1976.

19 Michael R. Garey and David S. Johnson. The rectilinear Steiner tree problem is NP-complete. *SIAM Journal on Applied Mathematics*, 32:826–834, 1977.

20 Michael Randolph Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co., New York, NY, USA, 1979.

21 Fanica Gavril. Algorithms for maximum weight induced paths. *Information Processing Letters*, 81:203–208, 2002.

22 Petr A. Golovach, Matthew Johnson, Daniël Paulusma, and Jian Song. A survey on the computational complexity of coloring graphs with forbidden subgraphs. *Journal of Graph Theory*, 84:331–363, 2017.

23 Petr A. Golovach, Pim van 't Hof, and Daniël Paulusma. Obtaining planarity by contracting few edges. *Theoretical Computer Science*, 476:38–46, 2013.

24 Sylvain Guillemot and Dániel Marx. A faster FPT algorithm for bipartite contraction. *Information Processing Letters*, 113:906–912, 2013.

25 Yi-Lu Guo, Chin-Wen Ho, and Ming-Tat Ko. The longest path problem on distance-hereditary graphs. *Advances in Intelligent Systems and Applications*, 1:69–77, 2013.

26 Richard Hammack. Cyclicity of graphs. *Journal of Graph Theory*, 32:160–170, 1999.

27 Richard Hammack. A note on the complexity of computing cyclicity. *Ars Combinatoria*, 63, 2002.

28 Pinar Heggernes, Pim van 't Hof, Benjamin Lévêque, Daniel Lokshtanov, and Christophe Paul. Contracting graphs to paths and trees. *Algorithmica*, 68:109–132, 2014.

29 Pinar Heggernes, Pim van 't Hof, Benjamin Lévêque, and Christophe Paul. Contracting chordal graphs and bipartite graphs to paths and trees. *Discrete Applied Mathematics*, 164:444–449, 2014.

30 Pinar Heggernes, Pim van 't Hof, Daniel Lokshtanov, and Christophe Paul. Obtaining a bipartite graph by contracting few edges. *SIAM Journal on Discrete Mathematics*, 27:2143–2156, 2013.

31 Danny Hermelin, Matthias Mnich, Erik Jan van Leeuwen, and Gerhard J. Woeginger. Domination when the stars are out. *ACM Transactions on Algorithms*, 15:25:1–25:90, 2019.

32 Cornelis Hoede and Henk Jan Veldman. On characterization of hamiltonian graphs. *Journal of Combinatorial Theory, Series B*, 25:47–53, 1978.

33 Cornelis Hoede and Henk Jan Veldman. Contraction theorems in hamiltonian graph theory. *Discrete Mathematics*, 34:61–67, 1981.

34 John E. Hopcroft and Richard M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, 2:225–231, 1973.

35 Kyriaki Ioannidou, George B. Mertzios, and Stavros D. Nikolopoulos. The longest path problem has a polynomial solution on interval graphs. *Algorithmica*, 61:320–341, 2011.

36 Kyriaki Ioannidou and Stavros D. Nikolopoulos. The longest path problem is polynomial on cocomparability graphs. *Algorithmica*, 65:177–205, 2013.

37 Tetsuya Ishizeki, Yota Otachi, and Koichi Yamazaki. An improved algorithm for the longest induced path problem on k-chordal graphs. *Discrete Applied Mathematics*, 156:3057–3059, 2008.

38 Lars Jaffke, O-joung Kwon, and Jan Arne Telle. Mim-width I. Induced path problems. *Discrete Applied Mathematics*, 278:153–168, 2020.

39 Matthew Johnson, Giacomo Paesani, and Daniël Paulusma. Connected Vertex Cover for $(sP_1 + P_5)$-free graphs. *Algorithmica*, 82:20–40, 2020.

40 Dieter Kratsch, Haiko Müller, and Ioan Todinca. Feedback vertex set and longest induced path on AT-free graphs. *Proc. WG 2003, LNCS*, 2880:309–321, 2003.

41 Asaf Levin, Daniël Paulusma, and Gerhard J. Woeginger. The computational complexity of graph contractions I: polynomially solvable and NP-complete cases. *Networks*, 51:178–189, 2008.

42 Asaf Levin, Daniël Paulusma, and Gerhard J. Woeginger. The computational complexity of graph contractions II: two tough polynomially solvable cases. *Networks*, 52:32–56, 2008.

43 Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. On the hardness of eliminating small induced subgraphs by contracting edges. *Proc. IPEC 2013, LNCS*, 8246:243–254, 2013.

44 Barnaby Martin and Daniël Paulusma. The computational complexity of disconnected cut and $2K_2$-partition. *Journal of Combinatorial Theory, Series B*, 111:17–37, 2015.

45 Dániel Marx, Barry O'Sullivan, and Igor Razgon. Finding small separators in linear time via treewidth reduction. *ACM Transactions on Algorithms*, 9:30:1–30:35, 2013.

46 George B. Mertzios and Ivona Bezáková. Computing and counting longest paths on circular-arc graphs in polynomial time. *Discrete Applied Mathematics*, 164:383–399, 2014.

47 George B. Mertzios and Derek G. Corneil. A simple polynomial algorithm for the longest path problem on cocomparability graphs. *SIAM Journal on Discrete Mathematics*, 26:940–963, 2012.

48 Haiko Müller. Hamiltonian circuits in chordal bipartite graphs. *Discrete Mathematics*, 156:291–298, 1996.

49 Bert Randerath and Ingo Schiermeyer. Vertex colouring and forbidden subgraphs - A survey. *Graphs and Combinatorics*, 20:1–40, 2004.

50 Neil Robertson and Paul D. Seymour. Graph minors .xiii. the disjoint paths problem. *Journal of Combinatorial Theory, Series B*, 63:65–110, 1995.

51 Bin Sheng and Yuefang Sun. An improved linear kernel for the cycle contraction problem. *Information Processing Letters*, 149:14–18, 2019.

52 Ryuhei Uehara and Yushi Uno. On computing longest paths in small graph classes. *International Journal of Foundations of Computer Science*, 18:911–930, 2007.

53 Ryuhei Uehara and Gabriel Valiente. Linear structure of bipartite permutation graphs and the longest path problem. *Information Processing Letters*, 103:71–77, 2007.

54 Pim van 't Hof, Marcin Kamiński, Daniël Paulusma, Stefan Szeider, and Dimitrios M. Thilikos. On graph contractions and induced minors. *Discrete Applied Mathematics*, 160:799–809, 2012.

55 Pim van 't Hof, Daniël Paulusma, and Gerhard J. Woeginger. Partitioning graphs into connected parts. *Theoretical Computer Science*, 410:4834–4843, 2009.