Recognizing Graphs Close to Bipartite Graphs*

Marthe Bonamy[1], Konrad K. Dabrowski[2], Carl Feghali[3], Matthew Johnson[4], and Daniël Paulusma[5]

1 CNRS, LaBRI, Université de Bordeaux, Bordeaux, France
marthe.bonamy@u-bordeaux.fr

2 Durham University, Durham, UK
konrad.dabrowski@durham.ac.uk

3 IRIF & Université Paris Diderot, Paris, France
feghali@irif.fr

4 Durham University, Durham, UK
matthew.johnson2@durham.ac.uk

5 Durham University, Durham, UK
daniel.paulusma@durham.ac.uk

Abstract

We continue research into a well-studied family of problems that ask if the vertices of a graph can be partitioned into sets A and B, where A is an independent set and B induces a graph from some specified graph class $\mathcal{G}$. We let $\mathcal{G}$ be the class of k-degenerate graphs. The problem is known to be polynomial-time solvable if $k = 0$ (bipartite graphs) and NP-complete if $k = 1$ (near-bipartite graphs) even for graphs of diameter 4, as shown by Yang and Yuan, who also proved polynomial-time solvability for graphs of diameter 2. We show that recognizing near-bipartite graphs of diameter 3 is NP-complete resolving their open problem. To answer another open problem, we consider graphs of maximum degree Δ on n vertices. We show how to find A and B in $O(n)$ time for $k = 1$ and $\Delta = 3$, and in $O(n^2)$ time for $k \geq 2$ and $\Delta \geq 4$. These results also provide an algorithmic version of a result of Catlin [1979] and enable us to complete the complexity classification of another problem: finding a path in the vertex colouring reconfiguration graph between two given k-colourings of a graph of bounded maximum degree.

1998 ACM Subject Classification G.2.2 Graph Theory

Keywords and phrases degenerate graphs, near-bipartite graphs, reconfiguration graphs

Digital Object Identifier 10.4230/LIPIcs.MFCS.2017.70

1 Introduction

We consider the problem of partitioning the vertex set of a graph into two sets A and B where A is an independent set and B belongs to a specified graph class $\mathcal{G}$. The classes $\mathcal{G}$ we consider are those of k-degenerate graphs (a graph is k-degenerate if every induced subgraph has a vertex of degree at most k). A 0-degenerate graph has no edges, so in this case the problem is simply that of recognizing bipartite graphs. A graph is 1-degenerate if and only if it is a forest; we say that a graph is *near-bipartite* if it can be decomposed into an independent set and a forest. The problem of deciding whether or not a graph is near-bipartite is NP-complete (see [6]). We consider this problem when the input is restricted, but our main focus is the problem of *finding* the decomposition into an independent set and a

* This paper received support from EPSRC (EP/K025090/1), London Mathematical Society (41536), the Leverhulme Trust (RPG-2016-258) and Fondation Sciences Mathématiques de Paris.

k-degenerate graph for graphs of bounded maximum degree. We also describe an application to a graph reconfiguration problem.

Related Work. As noted above, recognizing near-bipartite graphs is known to be an NP-complete problem. Yang and Yuan [29] proved that the problem remains NP-complete even for graphs of maximum degree 4 and for graphs of diameter 4. They also showed that it can be solved in polynomial time for graphs of maximum degree at most 3 and for graphs of diameter at most 2. They left open the problem of determining its computational complexity for graphs of diameter 3 (see also [4]). We note that the problem of recognizing near-bipartite graphs has also been studied from the perspective of parameterized complexity (in this setting the problem is known as Independent Feedback Vertex Set). With the size of A as the parameter, FPT algorithms have been found for general graphs [26]. The problem of finding a decomposition where the size of A is minimum has been shown to be NP-hard even for planar bipartite graphs of maximum degree 4, but linear-time solvable for graphs of bounded treewidth, chordal graphs and cographs [27] (it was already known that near-bipartite cographs can be recognized in linear time [4]). Recently, we extended the result of cographs to P_5-free graphs, and we also proved that recognizing near-bipartite line graphs of maximum degree 4 is NP-complete [2]. The variant where the maximum degree of the forest induced by B is bounded by some constant has also been studied, in particular from a structural point of view (see, for instance, [12]).

The Induced Forest 2-Partition problem is closely related to the problem of recognizing near-bipartite graphs. It asks whether the vertex set of a given graph can be decomposed into two disjoint sets A and B, where both A and B induce forests. Wu, Yuan and Zhao [28] proved that Induced Forest 2-Partition is NP-complete for graphs of maximum degree 5 and polynomial-time solvable for graphs of maximum degree at most 4. Brandstädt et al. [6] proved NP-completeness of another closely related problem, namely that of deciding whether the vertex set of a given graph can be decomposed into an independent set and a tree. Note that both this problem and that of recognizing near-bipartite graphs can be seen as restricted variants of the classical 3-Colouring problem,[1] which is well known to be NP-complete [20]. In fact, for a fixed graph class $\mathcal{G}$ (that is, $\mathcal{G}$ is not part of the input), Brandstädt et al. [6] considered the following more general problem:

Stable($\mathcal{G}$)

Instance:	A graph $G = (V, E)$.
Question:	Can V be decomposed into two disjoint sets A and B, where A is an independent set and B induces a graph in $\mathcal{G}$?

Note that Stable($\mathcal{G}$) is equivalent to 3-Colouring if we choose $\mathcal{G}$ to be the class of bipartite graphs. For a positive integer k, we denote the class of k-degenerate graphs by $\mathcal{D}_k$. Then Stable($\mathcal{D}_1$) is the problem of finding near-bipartite graphs. If $\mathcal{G}$ is the class of complete graphs, Stable($\mathcal{G}$) is the problem of recognizing split graphs, which can be solved in polynomial time. Brandstädt et al. [6] proved that Stable($\mathcal{G}$) is NP-complete when $\mathcal{G}$ is the class of trees or the class of trivially perfect graphs, and polynomial-time solvable when $\mathcal{G}$ is the class of co-bipartite graphs, the class of split graphs, or the class of threshold graphs. Moreover, Stable($\mathcal{G}$) has also been shown to be NP-complete when $\mathcal{G}$ is

[1] For a fixed integer $k \geq 1$, the k-Colouring problem asks whether a given graph is *k-colourable*, that is, whether its vertices can be coloured with at most k colours such that no two adjacent vertices are coloured alike. As trees and forests are 2-colourable, the observation follows.

the class of triangle-free graphs [7], the class of cographs [17], the class of graphs of maximum degree 1 [22], or, more generally, the class of graphs that have any additive hereditary property not equal to or divisible by the property of being edgeless [19], whereas it is also polynomial-time solvable if $\mathcal{G}$ is the class of complete bipartite graphs [13] (see [5] for a faster algorithm). The Stable($\mathcal{G}$) problem has also been studied for hereditary graph classes $\mathcal{G}$ with subfactorial or factorial speed [11, 21] (the speed of a graph class is the function that given an integer n returns the number of labelled graphs on n vertices in the class).

To obtain a more general problem than Stable($\mathcal{G}$), we can relax the condition on the set A being independent. This leads to the $(\mathcal{G}_1, \mathcal{G}_2)$-Recognition problem, which asks whether the vertex set of a graph can be decomposed into disjoint sets A and B, such that A induces a graph in $\mathcal{G}_1$ and B induces a graph in $\mathcal{G}_2$. For instance, if $\mathcal{G}_1$ is the class of cliques and $\mathcal{G}_2$ is the class of disjoint unions of cliques, the $(\mathcal{G}_1, \mathcal{G}_2$-Recognition problem is equivalent to recognizing unipolar graphs (see [24] for a quadratic algorithm). Generalizing Stable($\mathcal{G}$) also leads to a family of transversal problems, such as Feedback Vertex Set. However, such problems are beyond the scope of our paper.

As Stable($\mathcal{D}_k$) is NP-complete for every $k \geq 1$, its complexity has been determined for special graph classes. Recall the aforementioned hardness and tractability results of Yang and Yuan [29] on graph classes of bounded maximum degree and classes of bounded diameter when $k = 1$. Brandstädt, Brito, Klein, Nogueira and Protti [4] proved that Stable($\mathcal{D}_1$) is NP-complete on perfect graphs and, as stated earlier, that it is polynomial-time solvable for cographs. In particular, their hardness result for perfect graphs shows that the complexities of Stable($\mathcal{D}_1$) and 3-Colouring do not coincide, as k-Colouring is polynomial-time solvable for perfect graphs even when the number of colours k is part of the input [16].

Our Results. In Section 2, we consider the Stable($\mathcal{D}_1$) problem for graphs of bounded diameter. Recall that Yang and Yuan [29] left one missing case: they proved that recognizing near-bipartite graphs of diameter 4 is an NP-complete problem and that near-bipartite graphs of diameter 2 can be recognized in polynomial time. We show that Stable($\mathcal{D}_1$) is NP-complete for graphs of diameter 3, which resolves their open problem.

In Section 3, we consider near-bipartite decompositions of *subcubic* graphs (that is, graphs of maximum degree at most 3). By a result of Catlin and Lai [9], the only connected subcubic graph that is not near-bipartite is K_4 (see also Yang and Yuan [29]) and so near-bipartite subcubic graphs can be recognized in polynomial time. However, the proofs of [9, 29] do not lead to a linear-time algorithm for finding the desired partition (A, B); in particular, the result of [9] would require solving an NP-complete problem, namely Independent Set for cubic graphs [15]. We give a linear-time algorithm that finds such a partition.

We say a partition (A, B) of the vertex set of a graph is a *k-degenerate decomposition* if A is independent and B induces a $(k-2)$-degenerate graph (so Section 3 is concerned with 3-degenerate decompositions of graphs of maximum degree 3). In Section 4, we consider, more generally, Δ-degenerate decompositions of graphs of maximum degree at most Δ for any $\Delta \geq 3$. We give a quadratic-time algorithm to find such a decomposition in this general case (contrast with the linear-time algorithm for the special case of $\Delta = 3$). By a result of Matamala [23], the only connected graph with maximum degree at most Δ that does not have such a decomposition is $K_{\Delta+1}$. Matamala's result generalizes that of [9] and thus does not imply a polynomial-time algorithm for finding such a decomposition. We show that our algorithms for finding Δ-degenerate decompositions of graphs of maximum degree at most Δ also provide an algorithmic version of a result of Catlin [8]. In Section 5, we show how they can be applied to completely settle the complexity classification of a graph colouring reconfiguration problem considered in [14]. In Section 6 we give directions for future work.

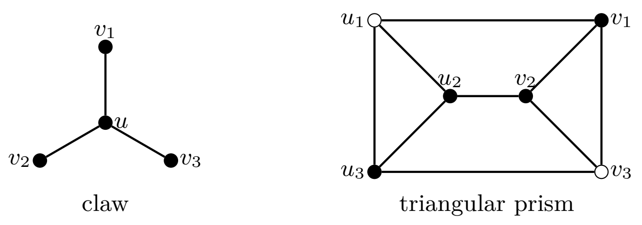

Figure 1 The claw and the triangular prism. A near-bipartite decomposition of the triangular prism is indicated: the white vertices form an independent set and the black vertices induce a forest.

2 Graphs of Diameter 3

We present a hardness result (proof omitted) for the problem of deciding whether a graph of diameter at most 3 has a near-bipartite decomposition. In [25], Mertzios and Spirakis proved a remarkable result: 3-Colouring is NP-complete for graphs of diameter 3. The outline of their proof is straightforward: a reduction from 3-Sat that constructs, for any instance ϕ, a graph H_ϕ that is 3-colourable if and only if ϕ is satisfiable. It is possible to reduce 3-Sat to Stable($\mathcal{D}_1$) for graphs of diameter 3 using the same construction, thus showing that Stable($\mathcal{D}_1$) is also NP-complete on this class. That is H_ϕ is near-bipartite if and only if ϕ is satisfiable. So Theorem 1 below is an observation about the proof of Mertzios and Spirakis.

▶ **Theorem 1.** Stable($\mathcal{D}_1$) *is* NP-*complete for graphs of diameter at most* 3.

3 Subcubic Graphs

In this section we prove the following result.

▶ **Theorem 2.** *Let G be a subcubic graph on n vertices, with no component isomorphic to K_4. Then a near-bipartite decomposition of G can be found in $O(n)$ time.*

Proof. We will repeatedly apply a set of *rules* to G. Each rule takes constant time to apply and after each application of a rule, the resulting graph contains fewer vertices. The rules are applied until the empty graph is obtained. We then reconstruct G from the empty graph by working through the rules applied in reverse order. As we rebuild G in this way, we find a near-bipartite decomposition of each obtained graph. We do this by describing how to extend, in constant time, a near-bipartite decomposition of a graph before some rule is undone to a near-bipartite decomposition of the resulting graph after that rule is undone. If we can do this then we say that the rule is *safe*. We conclude that the total running time of the algorithm is $O(n)$. It only remains to describe the rules, show that it takes constant time to do and undo each of them and prove that they are safe.

We need the following terminology. The *claw* is the graph with vertices u, v_1, v_2, v_3 and edges uv_1, uv_2, uv_3; the vertex u is the *centre* of the claw (see Figure 1). The *triangular prism* is the graph obtained from two triangles on vertices u_1, u_2, u_3 and v_1, v_2, v_3, respectively, by adding the edges u_iv_i for $i \in \{1, 2, 3\}$ (see Figure 1). Two vertices are *false twins* if they have the same neighbourhood (note that such vertices must be non-adjacent).

Let u be an arbitrary vertex of G. Our choice of u as an arbitrary vertex implies that u can be found in constant time. We then use the first of the following rules that is applicable.

Rule 1. If u has degree at most 2, then remove u.

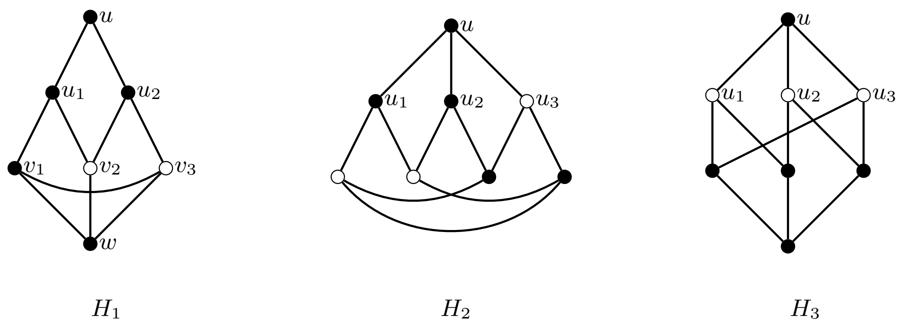

Figure 2 The graphs used in Rule 8. A near-bipartite decomposition of each is indicated: the white vertices form an independent set and the black vertices induce a forest.

Rule 2. If there is a vertex v of degree at most 2 that is at distance at most 3 from u, then remove v.

Rule 3. If G contains an induced diamond D whose vertices are at distance at most 3 from u, then remove the vertices of D.

Rule 4. If there is a pair of false twins u_1, u_2 each at distance at most 2 from u, then remove u_1, u_2 and their common neighbours (note that $u \in \{u_1, u_2\}$ is possible).

Rule 5. If u is in a connected component that is a triangular prism P, then remove the vertices of P.

Rule 6. If Rules 1–5 do not apply but u is in a triangle T, then the neighbours of the vertices in T that are outside T are pairwise distinct (since there is no induced diamond) and at least two them, which we denote by x', y', are non-adjacent (otherwise u belongs to a triangular prism). Remove the vertices of T and add an edge between x' and y'.

Rule 7. If u is the centre of an induced claw but has a neighbour v that belongs to a triangle, then apply one of the Rules 1–6 on v.

Rule 8. If the graph induced by the vertices at distance at most 3 from u contains the graph H_1, H_2 or H_3, depicted in Figure 2, with the vertex u in the position shown in the figure, then remove the vertices of this graph H_i.

Rule 9. If Rules 1–8 do not apply but u is the centre of an induced claw and its three neighbours u_1, u_2, u_3 are also centres of induced claws, then remove u, u_1, u_2, u_3 and for $i \in \{1, 2, 3\}$ add an edge joining the two neighbours of u_i distinct from u and denote it by e_i; we say that such an edge is *new* (note that such neighbours of two distinct u_i and u_j may overlap).

Let us show that at least one of the rules is always applicable. Suppose that, on the contrary, there is a vertex u of a subcubic graph for which no rule applies. Then u and its neighbours each have degree 3 (Rules 1 and 2) and so each either belongs to a triangle or is the centre of an induced claw. By Rule 6, u must be the centre of an induced claw and therefore, by Rule 7, the same is also true for each neighbour of u. This implies that Rule 9 applies, a contradiction.

Because G is subcubic, each of these rules takes constant time to verify and process. In particular, in some rules we need to detect some induced subgraph of constant size that contains u or replace u by some other vertex v. In all such cases we need to explore a set of vertices of distance at most 4 from u. As G is subcubic, this set has size at most $1 + 3 + 3^2 + 3^3 + 3^4 = 121$, so we can indeed do this in constant time.

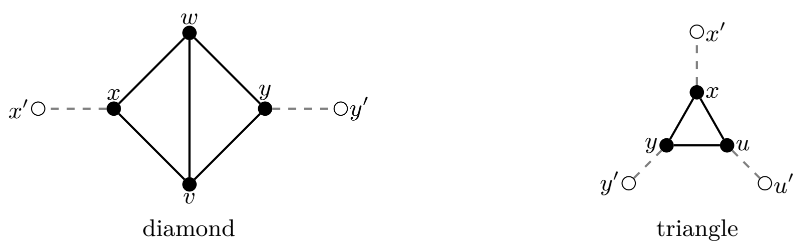

Figure 3 The diamond and triangle (solid edges and vertices) together with their neighbourhoods in a cubic graph.

It is clear that, as claimed, the application of a rule reduces the number of vertices and that if we repeatedly choose an arbitrary vertex u and apply a rule, we eventually obtain the empty graph. We now consider undoing the applied rules in reverse order to rebuild G. As this is done, we will irrevocably *colour* vertices with colour 1 or 2 in such a way that the vertices coloured 1 will form an independent set and the vertices coloured 2 will induce a forest. Thus a rule is safe if this colouring can be extended whenever that rule is undone. When we reach G, the final colouring will correspond to the required near-bipartite decomposition.

We must prove each rule is safe. At each step of reconstructing G, we refer to the graph before a rule is undone as the *prior graph* and to the graph after that rule is undone as the *subsequent graph*. Note that the application of any of the Rules 1–9 again yields a subcubic graph. By the result of Yang and Yuan [29], every connected subcubic graph is near-bipartite, apart from K_4. So we need to ensure that an application of a rule does not create a K_4. This cannot happen when we remove vertices, but we will need to consider it for Rules 6 and 9.

Claim 1. *Rules 1–5 are safe.*
In Rules 1–5 we only delete vertices. Rule 1 is safe since if both neighbours of u are coloured 2, then u can be coloured 1; otherwise u can be coloured 2. Similarly, we see that Rule 2 is safe. To see that Rule 3 is safe, let D be the diamond with vertex labels as illustrated in Figure 3, where u is one of v, w, x, y. If x' and y' are coloured 2, we colour x and y with 1 and v and w with 2. Otherwise we colour v with 1 and x, y and w with 2. We now show that Rule 4 is safe. Let u_1 and u_2 be false twins (at distance at most 2 from u). As G is subcubic, every vertex in $N(u_1)$ with a neighbour in $N(u_1)$ has no neighbours outside $N(u_1) \cup \{u_1, u_2\}$ and every vertex in $N(u_1)$ with no neighbour in $N(u_1)$ has at most one neighbour not equal to u_1 or u_2. Moreover, as G is subcubic, $N(u_1)$ contains no cycle. Hence we can always colour u_1, u_2 with 1 and the vertices of $N(u_1)$ with 2 regardless of the colours of vertices outside $N(u_1) \cup \{u_1, u_2\}$. Indeed, every vertex of $N(u_1)$ will have at most one neighbour that is not coloured 1, so cannot be in a cycle of vertices coloured 2 in the subsequent graph. Rule 5 is also safe since P is 3-regular and hence would be a component of our subsequent graph, so we can colour its vertices by assigning colour 1 to exactly one vertex from each of the two triangles and colour 2 to its other vertices (see Figure 1). This completes the proof of Claim 1.

Claim 2. *Rules 6 and 7 are safe.*
First, let us demonstrate that Rule 6 is safe. If x' and y' are contained in a K_4 of the prior graph, then the subsequent graph contains a diamond whose vertices are at distance at most 3 from u. This contradicts Rule 3. Let T be the triangle with vertex labels as illustrated in Figure 3. Suppose x', y' and u' are coloured 2. Then we colour u with 1 and x and y with 2. The vertices in the subsequent graph with colour 2 still induce a forest, as we have replaced

an edge in the forest by a path on four vertices. Suppose x' and y' are coloured 2 and u' is coloured 1. Then we colour x with 1, and y and u with 2. Otherwise, since x' and y' are joined by an edge in the prior graph, we may assume that x' has colour 1 and y' has colour 2. In this case we can colour y with 1, and x and u with 2. This completes the proof that Rule 6 is safe. Since Rules 1–6 are safe, it follows that Rule 7 is also safe. This completes the proof of Claim 2.

Claim 3. *Rule 8 is safe.*
We now show that Rule 8 is safe. Suppose u is contained in H_1. We use the vertex labels from Figure 2. As Rule 1 could not be applied, we find that u has a third neighbour u_3 distinct from u_1 and u_2. Regardless of whether u_3 is coloured 1 or 2, we colour u, u_1, u_2, v_1, w with 2 and v_2, v_3 with 1 to obtain a near-bipartite decomposition of G. We can also readily colour the vertices of H_2 or H_3 should u be contained in one of them (note that since H_2 and H_3 are 3-regular, these graphs can only appear as components in our subsequent graph). This completes the proof of Claim 3.

Claim 4. *Rule 9 is safe.*
Suppose that the prior graph contains fewer than three new edges. Then we may assume without loss of generality that $e_1 = e_2$. Then u_1 and u_2 are false twins at distance 1 from u and we can apply Rule 4, a contradiction. So we may assume that the prior graph contains exactly three new edges.

We claim that the application of Rule 9 does not yield a K_4. For contradiction, suppose it does. Let K be the created K_4. Then at least one new edge is contained in K. If exactly one new edge e is contained in K, then $K - e$ is a diamond in the subsequent graph. Then we could have applied Rule 3, a contradiction. If all three new edges are in K, then they must induce either a path on four vertices or a triangle in the subsequent graph. In the first case the subsequent graph is H_2 and in the second case the subsequent graph is H_3. In both cases we would have applied Rule 8, a contradiction. Finally, suppose that K contains exactly two new edges, say e_1 and e_2. If e_1 and e_2 do not share a vertex, then they cover the vertices of K. Hence the end-vertices of e_1 are false twins (at distance 2 from u) in the subsequent graph, since they are both adjacent to u_1 and to each end-vertex of e_2. Then we could have applied Rule 4, a contradiction. If $e_1 = v_1v_2$ and $e_2 = v_3v_4$ share a vertex, say $v_2 = v_4$, then v_1 and v_3 are adjacent in the subsequent graph and the vertex $w \in K \setminus \{v_1, v_2, v_3\}$ is adjacent only to v_1, v_2 and v_3. Therefore Rule 8 could have been applied, a contradiction.

Thus an application of Rule 9 does not yield a K_4, and we may colour u_1, u_2, u_3 with 2 and u with 1. Indeed note that since if two end-vertices of a new edge are coloured 2, then in the subsequent graph the vertices coloured 2 will still induce a forest, in which such a new edge is replaced by a path of length 2. This completes the proof of Claim 4 and therefore completes the proof of Theorem 2. ◀

4 Graphs of Bounded Maximum Degree

Let $k \geq 3$ be an integer. Recall that a graph G has a k-degenerate decomposition if its vertex set can be decomposed into sets A and B where A is an independent set and B induces a $(k-2)$-degenerate graph. Note that 3-degenerate decompositions are near-bipartite decompositions. We give an $O(n^2)$ algorithm for finding a Δ-degenerate decomposition of a graph on n vertices of maximum degree at most Δ for every $\Delta \geq 3$ (note that for $\Delta = 3$ we can also use Theorem 2).

For $k \geq 1$, we say that an order $v_1, v_2, \dots, v_n$ of the vertices of a graph G is *k-degenerate* if for all $i \geq 2$, the vertex v_i has at most k neighbours in $\{v_1, \dots, v_{i-1}\}$. It is clear that a graph is k-degenerate if and only if it has a k-degenerate order. If $\mathcal{O}$ is a k-degenerate order for G and W is a subset of the vertex set of G, then we let $\mathcal{O}|_W$ be the restriction of $\mathcal{O}$ to W, and let $G[W]$ be the subgraph of G induced by W. For a set of vertices C, we denote the *neighbourhood* of C by $N(C) = \bigcup\{N(u) \mid u \in C\} \setminus C$.

We need the following lemma, which is a refinement of Lemma 8 in [14] with the same proof.

▶ **Lemma 3.** *Let $k \geq 2$. Let G be a $(k-1)$-degenerate graph on n vertices. If a $(k-1)$-degenerate order $\mathcal{O}$ of G is given as input, a k-degenerate decomposition (A, B) of G can be found in $O(kn)$ time. In addition, we can ensure that $\mathcal{O}|_B$ is a $(k-2)$-degenerate order of $G[B]$, the set A is a maximal independent set and the first vertex in $\mathcal{O}$ belongs to A.*

A pair of non-adjacent vertices $\{u, v\}$ in a graph G is *strong* if u and v have a common neighbour in each component of the graph $G \setminus \{u, v\}$. In particular, note that if u and v have a common neighbour and $G \setminus \{u, v\}$ is connected, then $\{u, v\}$ is a strong pair. We need the following two lemmas (we omit the proof of the first one).

▶ **Lemma 4.** *Let $k \geq 3$. Let G be a connected k-regular graph on n vertices that contains a strong pair $\{u, v\}$. If $\{u, v\}$ is given as input, a k-degenerate decomposition of G can be found in $O(kn)$ time.*

▶ **Lemma 5.** *Let $k \geq 3$. Let G be a k-regular connected graph on n vertices, which contains a set C of $k+1$ vertices that induce a clique minus an edge uv. If C, u and v are given as input, then a k-degenerate decomposition of G can be found in $O(kn)$ time.*

Proof. Let x be a vertex in C distinct from u and v. Let G' be the graph obtained from G by deleting C. Each of u and v has exactly one neighbour that does not belong to C and all other vertices of C have no neighbours outside C. Let t be the neighbour of u not in C, and let w be the neighbour of v not in C. We may assume that t is distinct from w, otherwise we are done by Lemma 4. We can find a $(k-1)$-degenerate order $\mathcal{O}$ of G' in $O(kn)$ time by taking the vertices in the reverse of the order they are found in a breadth-first search from t, and then, if t and w do not belong to the same component of G', appending the vertices in the reverse of the order they are found in a breadth-first search from w. By Lemma 3, we can compute a k-degenerate decomposition (A, B) of G' in $O(kn)$ time such that $\mathcal{O}|_B$ is a $(k-2)$-degenerate order of B. If both t and w belong to B, let $A' = A \cup \{u, v\}$ and $B' = B \cup (C \setminus \{u, v\})$ and, since $(C \setminus \{u, v\})$ is a clique on $k-1$ vertices with no edge joining it to B, if follows that (A', B') is a k-degenerate decomposition of G. Assume now without loss of generality that $t \in A$ (we make no assumption about whether w is also in A). Then let $A' = A \cup \{x\}$ and $B' = B \cup (C \setminus \{x\})$. Then A' is an independent set. Recall that $\mathcal{O}|_B$ is a $(k-2)$-degenerate order of B. We show that we can append the vertices of $C \setminus \{x\}$ to obtain a $(k-2)$-degenerate order of B'. First add v, then the vertices of $C \setminus \{u, v, x\}$ and finally u. It is clear that no vertex has more than $k-2$ neighbours earlier in the order. ◀

▶ **Lemma 6.** *Let $k \geq 3$. Let G be a k-regular connected graph on n vertices containing a clique C on k vertices whose neighbourhood is of size 2. If C is given as input, a k-degenerate decomposition of G can be found in $O(kn)$ time.*

Proof. Let u and v be vertices not in C such that for each vertex in C, its unique neighbour not in C is either u or v. Neither u nor v can be adjacent to every vertex in C (as then the other would be adjacent to none, contradicting the premise that the neighbourhood has size 2). Since $k \geq 3$, one of u and v has at least two neighbours in C. Consider the graph G' obtained from G by removing C and adding the edge uv (if it does not already exist). Note that u and v each have degree at most k in G' and at least one of them, say u, has degree less than k. Therefore, we can find a $(k-1)$-degenerate order $\mathcal{O}$ of G' in $O(kn)$ time by taking the vertices in the order they are found in a breadth-first search from u. Thus we can obtain a k-degenerate decomposition (A, B) of G' by Lemma 3, such that $\mathcal{O}|_B$ is a $(k-2)$-degenerate order of $G[B]$ and A is a maximal independent set. At least one of u and v must belong to B. Assume without loss of generality that either $v \in A$, $u \in B$ or both u and v belong to B, and, in the latter case, assume that u has at least two neighbours in C. Consider a neighbour t of u in C. We set $A' = A \cup \{t\}$ and $B' = B \cup (C \setminus \{t\})$, and claim that (A', B') is a k-degenerate decomposition of G. It is clear that A' is an independent set. Recall that $\mathcal{O}|_B$ is a $(k-2)$-degenerate order for $G[B]$. We must amend it to find a $(k-2)$-degenerate order for $G[B']$ that also includes the vertices of $C \setminus \{t\}$. We consider two cases.

First suppose $v \in A$. Then append to $\mathcal{O}|_B$ first the neighbours of u in $C \setminus \{t\}$ and then the neighbours of v. As the vertices of $C \setminus \{t\}$ are adjacent to t the only one that could have more than $k-2$ vertices before it in the order is the one that appears last, but this is also adjacent to v so we do indeed have a $(k-2)$-degenerate order.

Now suppose $v \in B$. Then u has a neighbour in G' that belongs to A (as A is a maximal independent set). Hence u has at most $k-2$ neighbours in B'. Append to $\mathcal{O}|_B$ the vertices of $C \setminus \{t\}$, ending with a neighbour of u (we know there is at least one), then move u to be the final vertex in the order. Again the only vertex of $C \setminus \{t\}$ that could have more than $k-2$ neighbours before it in the order is the one that appears last, and by choosing it to be a neighbour of u and putting u later in the order we ensure that a $(k-2)$-degenerate order is obtained. ◀

Given a graph G, five of its vertices t, u, v, w, x and a set of vertices C, we say that C induces a (u, v)-*lock with special vertices* $(t, \{w, x\})$ if $t, w, x \in C$ and $N(C) = \{u, v\}$, and both u and v are adjacent to t, each vertex in $\{w, x\}$ is adjacent to precisely one vertex in $\{u, v\}$, and $G[C]$ contains all possible edges except for wt and xt. We say that C is a *lock* if it is a (u, v)-*lock with special vertices* $(t, \{w, x\})$ for some choice of t, u, v, w, x.

▶ **Lemma 7.** *Let $k \geq 3$. Let G be a k-regular connected graph on n vertices containing a (u, v)-lock C with special vertices $(t, \{w, x\})$. If C and u, v, t, w, x are given as input, then a k-degenerate decomposition of G can be found in $O(kn)$ time.*

Proof. Since t has two neighbours outside C, it follows that $C \setminus \{t\}$ is a clique on k vertices. If w and x have the same neighbour in $\{u, v\}$, say u, then $N(C \setminus \{t\}) = \{t, u\}$ and so we are done by Lemma 6. We may therefore assume that w and x have distinct neighbours in $\{u, v\}$. Let G' be the graph obtained from G by deleting $C \setminus \{t\}$ and note that G' is connected since t is adjacent to both u and v. Note that both u and v have degree $k-1$ in G'. We can therefore find a $(k-1)$-degenerate order $\mathcal{O}$ of G' in $O(kn)$ time by taking the vertices in the reverse of the order they are found in a breadth-first search from u. Furthermore, since the only neighbours of t in G' are u and v, both of which have degree $k-1$, by moving t to the start of the order $\mathcal{O}$, we obtain a another $(k-1)$-degenerate order $\mathcal{O}'$. By Lemma 3, we can therefore find a k-degenerate decomposition (A, B) of G' such that $\mathcal{O}'|_B$ is a $(k-2)$-degenerate order on B and $t \in A$. Thus both u and v belong to B. We let $A' = A \cup \{w\}$ and $B' = B \cup (C \setminus \{w, t\})$, and claim that (A', B') is a k-degenerate decomposition of G. It is clear that A' is an independent set. We have the $(k-2)$-degenerate order $\mathcal{O}'|_B$ on B. We obtain a $(k-2)$-degenerate order on B' by appending to $\mathcal{O}'|_B$ the vertices of $C \setminus \{w, t\}$ beginning with x. Indeed, the only neighbour of x that is earlier in the order is its single neighbour in $\{u, v\}$ (and note that $1 \leq k-2$ since $k \geq 3$). Furthermore, since $w, t \in A'$, every vertex in $C \setminus \{w, t, x\}$ has only $k-2$ neighbours in B'. ◀

A pair of non-adjacent vertices u, v in a graph is a *good* pair if u and v have a common neighbour. Note that if a good pair u, v is not strong, then $G \setminus \{u, v\}$ must be disconnected. We are now ready to state and prove the following result.

▶ **Theorem 8.** *Let $\Delta \geq 3$ and G be a graph on n vertices with maximum degree at most Δ. If no component of G is isomorphic to $K_{\Delta+1}$, then a Δ-degenerate decomposition of G can be found in $O(\Delta n^2)$ time.*

Proof. We may assume that G is connected, otherwise it can be considered componentwise. If G is not Δ-regular, then it has a vertex u of degree at most $\Delta - 1$, so we can find a $(\Delta-1)$-degenerate order $\mathcal{O}$ of G in $O(kn)$ time by taking the vertices in the reverse of the order they are found in a breadth-first search from u. In this case, we are done by Lemma 3. For Δ-regular graphs, we use the procedure shown in Algorithm 1. (Whenever we apply one of the lemmas of this section, we set $k = \Delta$. We note in passing that if we could assume that G is biconnected, then we would immediately have, by [1, Lemma 3], that there is always a good pair u, v such that $G \setminus \{u, v\}$ is connected. As we cannot make this assumption however, we are not able to make use of this result.)

Algorithm 1: Finding a k-degenerate decomposition for connected Δ-regular graphs.

```
Input  : A connected $\Delta$-regular graph $G$
Output : A $k$-degenerate decomposition of $G$
1  find a good pair $u$, $v$ and let $C$ be a component of $G \setminus \{u, v\}$;
2  if $u$, $v$ is a strong pair then apply Lemma 4;
3  else if the union of $C$ and one or both of $u$ and $v$ is a clique on $\Delta + 1$ vertices minus
     an edge then apply Lemma 5;
4  else if $C$ is a clique on $\Delta$ vertices whose neighbourhood is $\{u, v\}$ then apply Lemma 6;
5  else if $C$ is a $(u, v)$-lock then apply Lemma 7;
6  else
7      find a good pair $u', v' \in C$ such that either $C' = G \setminus \{u', v'\}$ is connected or
         $G \setminus \{u', v'\}$ has a component $C'$ that is strictly contained in $C$;
8      set $u \leftarrow u'$, $v \leftarrow v'$, $C \leftarrow C'$;
9      go to Line 2
10 end
```

Let us make a few comments on this procedure. As G is regular, connected and not complete, we can initially choose any vertex as u and find another vertex v to form a good pair in $O(\Delta^2) = O(\Delta n)$ time. If we perform a breadth-first search (which takes $O(n+m) = O(\Delta n)$ time) from a neighbour of u that retreats from u or v whenever they are encountered, we discover a component of $G \setminus \{u, v\}$. If the component contains a common neighbour of u and v but is not equal to $G \setminus \{u, v\}$, we repeat starting from a neighbour of u or v that was not discovered. Thus we discover in $O(\Delta n)$ time that either u, v is a strong pair (if we find all components of $G \setminus \{u, v\}$ and they each contain a common neighbour of u and v), or that it is not. We set C to be one of the components of $G \setminus \{u, v\}$ arbitrarily. By Lemma 4, we therefore conclude that Lines 1 and 2 take $O(\Delta n)$ time. It is easy to check in $O(\Delta n)$ time whether we apply Lemmas 5–7 on Lines 3–5 and applying these lemmas takes $O(\Delta n)$ in each case. Now suppose that we do not apply any of these lemmas, in which case we reach Line 7. We will show that we can find u', v' and, if necessary, C' in $O(\Delta n)$ time. If we find u', v' such that $C' = G \setminus \{u', v'\}$ is connected, then u', v' is a strong pair, so after executing Line 9, the algorithm will stop on Line 2. In all other cases C' will be strictly smaller than C. This means that we apply Line 9 at most $O(n)$ times, implying that we execute Lines 2–9 at most $O(n)$ times. This will give an overall running time of $O(\Delta n^2)$. It remains to show that if execution reaches Line 7 then we can find the required good pair u', v' and the component C' in $O(\Delta n)$ time.

Let us first show that C contains good pairs — that is, that it is not a clique. If C is a clique, then it contains either $\Delta - 1$ or Δ vertices (as each vertex has degree Δ in G and the only other possible neighbours of vertices in C are u and v). If C has $\Delta - 1$ vertices, then each vertex of C must be adjacent to both u and v and we would have applied Lemma 5 on Line 3, a contradiction. If C has Δ vertices, then each vertex of C is adjacent to exactly one of u and v (and neither u nor v can be adjacent to every vertex in C, as this would form a $K_{\Delta+1}$, contradicting the fact that G is connected), in which case we would have applied Lemma 6 on Line 4, a contradiction. Therefore we may assume that C is not a clique.

We need describe how to choose a good pair u', v' in C. If we can show that u and v are in the same component of $G \setminus \{u', v'\}$ (which must necessarily contain all of $G \setminus C$), then we are done as either $G \setminus \{u', v'\}$ is connected or there is another component C' of $G \setminus \{u', v'\}$ which must be contained in C (and note that in this case C' can be found in $O(\Delta n)$ time using breadth-first search).

If u and v have a common neighbour outside C or at least three common neighbours in C, then any good pair in C can be chosen as u', v' (as u and v will then be in the same component of $G \setminus \{u', v'\}$). If u and v have exactly two common neighbours t_1, t_2 that both belong to C, then any good pair other than t_1, t_2 can be chosen as u', v'. If t_1, t_2 is the only good pair in C (so all other vertices in C are adjacent), then C is a clique minus an edge and must contain Δ vertices (t_1, t_2 and the $\Delta - 2$ neighbours of t_1 that are not in $\{u, v\}$). Considering degree, any vertex in C other than t_1 or t_2 must be adjacent to exactly one of u and v. If every vertex in $C \setminus \{t_1, t_2\}$ is adjacent to, say u, then $C \cup \{u\}$ is a clique on $\Delta + 1$ vertices minus an edge and we would have applied Lemma 5 on Line 3, a contradiction. We may therefore assume that at least one vertex in $C \setminus \{t_1, t_2\}$ is adjacent to u and at least one is adjacent to v, so there is a path from u to v avoiding t_1 and t_2, and $G \setminus \{t_1, t_2\}$ is connected, so we are done.

Finally, suppose that u and v have exactly one common neighbour t that belongs to C. Then any good pair not including t can be chosen as u', v', as then u and v will be in the same component of $G \setminus \{u', v'\}$. Suppose, for contradiction, that no such pair exists. Then $C \setminus \{t\}$ is a clique. The vertex t has $\Delta - 2$ neighbours in $C \setminus \{t\}$. Since $\Delta \geq 3$, let z be one of those neighbours. Since t is the only common neighbour of u and v, we have that z can only be adjacent to at most one of u and v. Therefore, t has a neighbour non-adjacent to z, so z must have a neighbour non-adjacent to t, which we denote w. As w is also adjacent to at most one of u and v, it also has a neighbour x that is a non-neighbour of t (and cannot be t itself). So $C \setminus \{t\}$ contains at least Δ vertices: the $\Delta - 2$ neighbours of t plus w and x. As $C \setminus \{t\}$ induces a clique, it must have exactly Δ vertices, since G cannot contain a $K_{\Delta+1}$. Thus the set C forms a lock, and so we would have applied Lemma 7 on Line 5. This contradiction completes the proof. ◀

We note that Theorems 2 and 8 concern decompositions (A, B) of the vertex set of a graph where A is independent and B induces a $(\Delta - 2)$-degenerate graph. As B therefore cannot be a clique on Δ vertices, we have the following corollary.

▶ **Corollary 9.** *Let $\Delta \geq 3$ and G be a graph on n vertices with maximum degree at most Δ. If the clique number of G is at most Δ, a decomposition of the vertices of G into sets A and B, where A is an independent set and B induces a graph that has clique number at most $\Delta - 1$, can be found in $O(n)$ time if $\Delta = 3$ and $O(\Delta n^2)$ time for all Δ.*

Catlin [8] proved that such decompositions exist, but his result did not imply the existence of a polynomial-time algorithm to find the decomposition. We note that his result contains the additional claim that A is maximum; we cannot hope for an algorithmic version, as this would solve the NP-hard problem of finding a maximum independent set in a cubic graph [15].

5 Reconfigurations of Vertex Colourings

Our interest in Stable($\mathcal{D}_k$) when $k \geq 2$ case stems from an open problem in the area of graph reconfigurations. For a graph G and integer $k \geq 1$, the k-colouring reconfiguration graph $R_k(G)$ has the vertex set consisting all possible k-colourings of G and two vertices of $R_k(G)$ are adjacent if and only if the corresponding k-colourings differ on exactly one vertex. A central problem in the area of reconfiguration is the Reachability problem. In the context of k-colourings, this is the problem of finding a path (if one exists) in $R_k(G)$ between two given k-colourings α and β of a graph G. The complexity of this problem has been studied for graphs G with maximum degree Δ and any positive integer k. This problem is PSPACE-hard for $k \geq 4$, $\Delta \geq k$ [3], solvable in $O(n+m)$ time on (general) graphs with n vertices and m edges for $k \leq 3$ [18] and solvable in $O(n^2)$ time for $k \geq 4$, $0 \leq \Delta \leq k-2$ [10]. This leaves us with the case $k \geq 4$, $\Delta = k-1$. In [14, Theorem 6], three of the authors of the current paper showed that this case can be solved in polynomial time as long as the input graph G is not Δ-regular, that is, if not all vertices in G have maximum degree Δ. The case where G is Δ-regular was left as an open problem, but using Theorem 8 we can now resolve it by giving an $O(n^2)$-time algorithm (assuming Δ is fixed and not part of the input).

▶ **Theorem 10.** *Fix $\Delta \geq 0$. Let G be a connected graph on n vertices with maximum degree Δ. The problem of finding a path (if one exists) between two k-colourings α and β in $R_k(G)$ is*

- *$O(n)$-time solvable if $1 \leq k \leq 3$;*
- *$O(n^2)$-time solvable if $k \geq 4$ and $0 \leq \Delta \leq k-1$;*
- PSPACE-*hard if $k \geq 4$ and $\Delta \geq k$.*

6 Future Work

As well as showing that the problem of recognizing near-bipartite graphs of diameter 3 is NP-complete, we have proven that for every integer $\Delta \geq 3$, the Stable($\mathcal{D}_{\Delta-2}$) problem is polynomial-time solvable on graphs of maximum degree Δ. Is Stable($\mathcal{D}_{\Delta-2}$) NP-complete for graphs of maximum degree $\Delta+1$? Recall that this is known to be the case for $\Delta = 3$, as proven by Yang and Yuan [29]. Their proof can easily be adapted to prove that Stable($\mathcal{D}_{\Delta-2}$) is NP-complete for graphs of maximum degree $2\Delta - 2$ for every $\Delta \geq 4$.

References

1 Bradley Baetz and David R. Wood. Brooks' vertex-colouring theorem in linear time. *CoRR*, abs/1401.8023, 2014.

2 Marthe Bonamy, Konrad K. Dabrowski, Carl Feghali, Matthew Johnson, and Daniël Paulusma. Independent feedback vertex set for P_5-free graphs. Manuscript, 2017.

3 Paul S. Bonsma and Luis Cereceda. Finding paths between graph colourings: PSPACE-completeness and superpolynomial distances. *Theoretical Computer Science*, 410(50):5215–5226, 2009.

4 Andreas Brandstädt, Synara Brito, Sulamita Klein, Loana Tito Nogueira, and Fábio Protti. Cycle transversals in perfect graphs and cographs. *Theoretical Computer Science*, 469:15–23, 2013.

5 Andreas Brandstädt, Peter L. Hammer, Van Bang Le, and Vadim V. Lozin. Bisplit graphs. *Discrete Mathematics*, 299(1–3):11–32, 2005.

6 Andreas Brandstädt, Van Bang Le, and Thomas Szymczak. The complexity of some problems related to graph 3-colorability. *Discrete Applied Mathematics*, 89(1–3):59–73, 1998.

7 Leizhen Cai and Derek G. Corneil. A generalization of perfect graphs – i-perfect graphs. *Journal of Graph Theory*, 23(1):87–103, 1996.

8 Paul A. Catlin. Brooks' graph-coloring theorem and the independence number. *Journal of Combinatorial Theory, Series B*, 27(1):42–48, 1979.

9 Paul A. Catlin and Hong-Jian Lai. Vertex arboricity and maximum degree. *Discrete Mathematics*, 141(1–3):37–46, 1995.

10 Luis Cereceda. *Mixing graph colourings.* PhD thesis, London School of Economics, 2007.

11 Konrad K. Dabrowski, Vadim V. Lozin, and Juraj Stacho. Stable-Π partitions of graphs. *Discrete Applied Mathematics*, 182:104–114, 2015.

12 François Dross, Mickaël Montassier, and Alexandre Pinlou. Partitioning sparse graphs into an independent set and a forest of bounded degree. *CoRR*, abs/1606.04394, 2016.

13 Tomás Feder, Pavol Hell, Sulamita Klein, and Rajeev Motwani. List partitions. *SIAM Journal on Discrete Mathematics*, 16(3):449–478, 2003.

14 Carl Feghali, Matthew Johnson, and Daniël Paulusma. A reconfigurations analogue of Brooks' Theorem and its consequences. *Journal of Graph Theory*, 83(4):340–358, 2016.

15 Michael Randolph Garey, David S. Johnson, and Larry J. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1(3):237–267, 1976.

16 Martin Grötschel, László Lovász, and Alexander Schrijver. Polynomial algorithms for perfect graphs. *Annals of Discrete Mathematics*, 21:325–356, 1984.

17 Chính T. Hoàng and Van Bang Le. On P_4-transversals of perfect graphs. *Discrete Mathematics*, 216(1–3):195–210, 2000.

18 Matthew Johnson, Dieter Kratsch, Stefan Kratsch, Viresh Patel, and Daniël Paulusma. Finding shortest paths between graph colourings. *Algorithmica*, 75(2):295–321, 2016.

19 Jan Kratochvíl and Ingo Schiermeyer. On the computational complexity of $(\mathcal{O}, \mathcal{P})$-partition problems. *Discussiones Mathematicae Graph Theory*, 17(2):253–258, 1997.

20 László Lovász. Coverings and coloring of hypergraphs. *Congressus Numerantium*, VIII:3–12, 1973.

21 Vadim V. Lozin. Between 2- and 3-colorability. *Information Processing Letters*, 94(4):179–182, 2005.

22 Nadimpalli V. R. Mahadev and Uri N. Peled. *Threshold graphs and related topics*, volume 56 of *Annals of Discrete Mathematics.* North-Holland, Amsterdam, 1995.

23 Martín Matamala. Vertex partitions and maximum degenerate subgraphs. *Journal of Graph Theory*, 55(3):227–232, 2007.

24 Colin McDiarmid and Nikola Yolov. Recognition of unipolar and generalised split graphs. *Algorithms*, 8(1):46–59, 2015.

25 George B. Mertzios and Paul G. Spirakis. Algorithms and almost tight results for 3-colorability of small diameter graphs. *Algorithmica*, 74(1):385–414, 2016.

26 Neeldhara Misra, Geevarghese Philip, Venkatesh Raman, and Saket Saurabh. On parameterized independent feedback vertex set. *Theoretical Computer Science*, 461:65–75, 2012.

27 Yuma Tamura, Takehiro Ito, and Xiao Zhou. Algorithms for the independent feedback vertex set problem. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E98-A(6):1179–1188, 2015.

28 Yongqi Wu, Jinjiang Yuan, and Yongcheng Zhao. Partition a graph into two induced forests. *Journal of Mathematical Study*, 29:1–6, 1996.

29 Aifeng Yang and Jinjiang Yuan. Partition the vertices of a graph into one independent set and one acyclic set. *Discrete Mathematics*, 306(12):1207–1216, 2006.