A note on k-colorability of P_5-free graphs

Chính T. Hoàng* Marcin Kamiński† Vadim Lozin‡ J. Sawada§ X. Shu¶

Abstract

A polynomial time algorithm that determines whether or not, for a fixed k, a P_5-free graph can be k-colored is presented in this paper. If such a coloring exists, the algorithm will produce a valid k-coloring.

Keywords: P_5-free, graph coloring, dominating clique

1 Introduction

Graph coloring is among the most important and applicable graph problems. The *k-colorability problem* is the question of whether or not the vertices of a graph can be colored with one of k colors so that no two adjacent vertices are assigned the same color. In general, the k-colorability problem is NP-complete [10]. Even for planar graphs with no vertex degree exceeding 4, the problem is NP-complete [5]. However, for other classes of graphs, like perfect graphs [8], the problem is polynomial-time solvable. For the following special class of perfect graphs, there are efficient polynomial time algorithms for finding optimal colorings: chordal graphs [6], weakly chordal graphs [9], and comparability graphs [4]. For more information on perfect graphs, see [1], [3], and [7].

Another interesting class of graphs are those that are P_t-free, that is, graphs with no chordless paths $v_1, v_2, \ldots, v_t$ of length $t-1$ as induced subgraph for some fixed t. If $t = 3$ or $t = 4$, then there exists efficient algorithms to answer the k-colorability question (see [3]). However, it is known that CHROMATIC NUMBER for P_5-free graphs is NP-complete [11]. Thus, it is of some interest to consider the problem of k-coloring a P_t-free graph for some fixed $k \geq 3$ and $t \geq 5$. Taking this parameterization

*Physics and Computer Science, Wilfrid Laurier University, Canada. Research supported by NSERC. E-mail: choang@wlu.ca

†RUTCOR, Rutgers University, 640 Bartholomew Road, Piscataway, NJ 08854, USA. E-mail: mkaminski@rutcor.rutgers.edu

‡Mathematics Institute, University of Warwick, Coventry CV4 7AL UK. E-mail: V.Lozin@warwick.ac.uk

§Computing and Information Science, University of Guelph, Canada. Research supported by NSERC. E-mail: sawada@cis.uoguelph.ca

¶Computing and Information Science, University of Guelph, Canada. E-mail: xshu@uoguelph.ca

$k\backslash t$	3	4	5	6	7	8	$\dots$	12	$\dots$
3	$O(m)$	$O(m)$	$O(n^{\alpha})$	$O(mn^{\alpha})$	?	?	?	?	$\dots$
4	$O(m)$	$O(m)$	**??**	?	?	?	?	NP_c	$\dots$
5	$O(m)$	$O(m)$	**??**	?	?	NP_c	NP_c	NP_c	$\dots$
6	$O(m)$	$O(m)$	**??**	?	?	NP_c	NP_c	NP_c	$\dots$
7	$O(m)$	$O(m)$	**??**	?	?	NP_c	NP_c	NP_c	$\dots$
$\dots$	$\dots$	$\dots$	**...**	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$

Table 1: Known complexities for k-colorability of P_t-free graphs

into account, a snapshot of the known complexities for the k-colorability problem of P_t-free graphs is given in Table 1. From this chart we can see that there is a polynomial algorithm for the 3-colorability of P_6-free graphs [12].

In this paper we focus on P_5-free graphs. Notice that when $k = 3$, the colorability question for P_5-free graphs can be answered in polynomial time (see [13]). We obtain a theorem (Theorem 2) on the structure of P_5-free graphs and use it to design a polynomial-time algorithm that determines whether a P_5-free graph can be k-colored. If such a coloring exists, then the algorithm will yield a valid k-coloring.

The remainder of the paper is presented as follows. In Section 2 we present relevant definitions, concepts, and notations. Then in Section 3, we present our recursive polynomial-time algorithm that answers the k-colorability question for P_5-free graphs.

2 Background and Definitions

In this section we provide the necessary background and definitions used in the rest of the paper. For starters, we assume that $G = (V, E)$ is a simple undirected graph where $|V| = n$ and $|E| = m$. If A is a subset of V, then we let $G(A)$ denote the subgraph of G induced by A.

DEFINITION 1 *A set of vertices A is said to* dominate *another set B, if every vertex in B is adjacent to at least one vertex in A.*

The following structural result about P_5-free graphs is from Bacsó and Tuza [2]:

THEOREM 1 *Every connected P_5-free graph has either a dominating clique or a dominating P_3.*

DEFINITION 2 *Given a graph G, an integer k and for each vertex v, a list $l(v)$ of k colors, the k-*list coloring problem *asks whether or not there is a coloring of the vertices of G such that each vertex receives a color from its list.*

DEFINITION 3 *The* restricted k-list coloring problem *is the k-list coloring problem in which the lists $l(v)$ of colors are subsets of* $\{1, 2, \ldots, k\}$.

Our general approach is to take an instance of a specific coloring problem Φ for a given graph and replace it with a polynomial number of instances $\phi_1, \phi_2, \phi_3, \ldots$ such that the answer to Φ is "yes" if and only if there is some instance ϕ_k that also answers "yes".

For example, consider a graph with a dominating vertex u where each vertex has color list $\{1, 2, 3, 4, 5\}$. This listing corresponds to our initial instance Φ. Now, by considering different ways to color u, the following set of four instances will be equivalent to Φ:

ϕ_1: $l(u) = \{1\}$ and the remaining vertices have color lists $\{2, 3, 4, 5\}$,

ϕ_2: $l(u) = \{2\}$ and the remaining vertices have color lists $\{1, 3, 4, 5\}$,

ϕ_3: $l(u) = \{3\}$ and the remaining vertices have color lists $\{1, 2, 4, 5\}$,

ϕ_4: $l(u) = \{4, 5\}$ and the remaining vertices have color lists $\{1, 2, 3, 4, 5\}$.

In general, if we recursively apply such an approach we would end up with an exponential number of equivalent coloring instances to Φ.

3 The Algorithm

Let G be a connected P_5-free graph. This section describes a polynomial time algorithm that decides whether or not G is k-colorable. The algorithm is outlined in 3 steps. Step 2 requires some extra structural analysis and is presented in more detail in the following subsection.

1. Identify and color a maximal dominating clique or a P_3 if no such clique exists (Theorem 1). This partitions the vertices into **fixed sets** indexed by available colors. For example, if a P_5-free graph has a dominating K_3 (and no dominating K_4) colored with $\{1, 2, 3\}$ and $k = 4$, then the fixed sets would be given by: $S_{124}, S_{134}, S_{234}, S_{14}, S_{24}, S_{34}$. For an illustration, see Figure 1. Note that all the vertices in S_{124} are adjacent to the vertex colored 3 and thus have color lists $\{1, 2, 4\}$. This gives rise to our original restricted list-coloring instance Φ. Although the illustration in Figure 1 does not show it, it is possible for there to be edges between any two fixed sets.

2. Two vertices are *dependent* if there is an edge between them and the intersection of their color lists is non-empty. In this step, we remove all dependencies between each pair of fixed sets. This process, detailed in the following subsection, will create a polynomial number of coloring instances $\{\phi_1, \phi_2, \phi_3, \ldots\}$ equivalent to Φ.

3. For each instance ϕ_i from Step 2 the dependencies between each pair of fixed sets has been removed which means that the vertices within each fixed set can be colored independently. Thus,

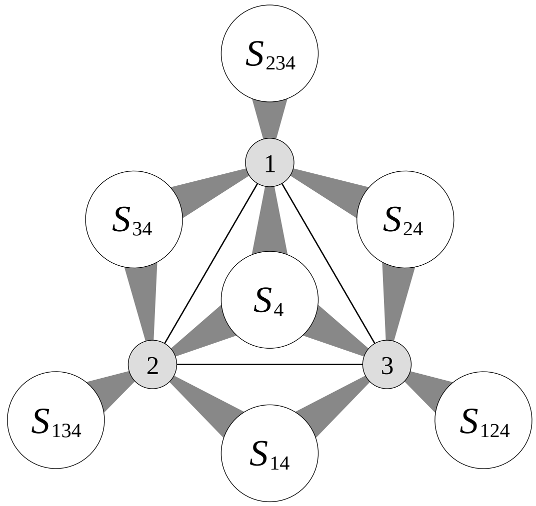

Figure 1: The fixed sets in a P_5-free graph with a dominating K_3 where $k = 4$.

for each instance ϕ_i we recursively see if each fixed set can be colored with the corresponding restricted color lists (the base case is when the color lists are a single color). If *one* such instance provides a valid k-coloring then return the coloring. Otherwise, the graph is not k-colorable.

As mentioned, the difficult part is reducing the dependencies between each pair of fixed sets (Step 2).

3.1 Removing the Dependencies Between Two Fixed Sets

Let S_{list} denote a fixed set of vertices with color list given by $list$. We partition each such fixed set into **dynamic sets** that each represent a unique subset of the colors in $list$. For example: $S_{123} = P_{123} \cup P_{12} \cup P_{13} \cup P_{23} \cup P_1 \cup P_2 \cup P_3$. Initially, $S_{123} = P_{123}$ and the remaining sets in the partition are empty. However, as we start removing dependencies, these sets will dynamically change. For example, if a vertex u is initially in P_{123} and one of its neighbors gets colored 2, then u will be removed from P_{123} and added to P_{13}.

Recall that our goal is to remove the dependencies between two fixed sets S_p and S_q. To do this, we remove the dependencies between each pair (P, Q) where P is a dynamic subset of S_p and Q is a dynamic subset of S_q. Let $col(P)$ and $col(Q)$ denote the color lists for the vertices in P and Q respectively. By visiting these pairs in order from largest to smallest with respect to $|col(P)|$ and then $|col(Q)|$, we ensure that we only need to consider each pair once. Applying this approach, the crux of the reduction process is to remove the dependencies between a pair (P, Q) by creating at most a polynomial number of equivalent colorings.

Now, observe that there exists a vertex v from the dominating set found in Step 1 of the algorithm that dominates every vertex in one set, but is not adjacent to any vertex in the other. This is because P and Q are subsets of different fixed sets. WLOG assume that v dominates Q.

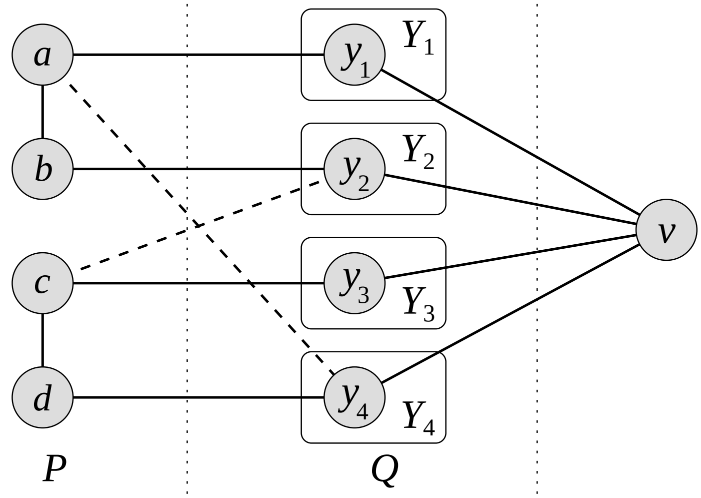

Figure 2: Illustration for proof of Theorem 2

THEOREM 2 *Let H be a P_5-free graph partitioned into three sets P, Q and $\{v\}$ where v is adjacent to every vertex in Q but not adjacent to any vertex in P. If we let Q' denote all components of $H(Q)$ that are adjacent to some vertex in P then one of the following must hold.*

1. *There exists exactly one* special *component C in $G(P)$ that contains two vertices a and b such that a is adjacent to some component $Y_1 \in G(Q)$ but not adjacent to another component $Y_2 \in G(Q)$ while b is adjacent to Y_2 but not Y_1.*
2. *There is a vertex x that dominates every component in Q', except at most one (call it T).*

PROOF: Suppose that there are two unique components $X_1, X_2 \in G(P)$ with $a, b \in X_1$ and $c, d \in X_2$ and components $Y_1 \neq Y_2$ and $Y_3 \neq Y_4$ from $G(Q)$ such that:

- a is adjacent to Y_1 but not adjacent to Y_2,
- b is adjacent to Y_2 but not adjacent to Y_1,
- c is adjacent to Y_3 but not adjacent to Y_4,
- d is adjacent to Y_4 but not adjacent to Y_3.

Let y_1 (respectively, y_2, y_3, y_4) be a vertex in Y_1 (respectively, Y_2, Y_3, Y_4) that is adjacent to a (respectively, b, c, d) and not b (respectively, a, d, c). Since H is P_5-free, there must be edges (a, b) and (c, d), otherwise a, y_1, v, y_2, b or c, y_3, v, y_4, d would be P_5s. An illustration of these vertices and components is given in Figure 2.

Suppose $Y_2 = Y_3$. Then b is not adjacent to y_3, for otherwise there exists a P_5 a, b, y_3, c, d. Now, there exists a P_5 y_1, a, b, y_2, y_3 (if y_2 is adjacent to y_3) or a P_5 a, b, y_2, v, y_3 (if y_2 is not adjacent to y_3). Thus, Y_2 and Y_3 must be unique components. Similarly, we have $Y_2 \neq Y_4$. Now since b, y_2, v, y_3, c cannot be

Procedure RemoveDependencies(P', Q, φ)

if no dependencies between P' and Q
  then output φ
  else find x, T from Theorem 2
    for each $c \in col(P) \cap col(Q)$ **do**
      output ReduceComponent(T, φ with x colored c)
    RemoveDependencies($P' - \{x\}$, Q, φ with $l(x) = col(P) - col(C)$)

Figure 3: Algorithm to remove dependencies between two dynamic sets P' and Q (with no special component C) by creating an equivalent set of coloring instances with the dependences removed.

a P_5, either b is adjacent to y_3 or c must be adjacent to y_2. WLOG, suppose the latter. Now a, b, y_2, v, y_4 implies that either a or b is adjacent to y_4. If y_4 is adjacent to b but not a, then a, b, y_4, d, c would be a P_5 which implies that a must be adjacent to y_4 anyway. Thus, we end up with a P_5 a, y_4, v, y_2, c which is a contradiction to the graph being P_5-free. Thus there must be at most one special component C.

Now suppose that there is no special component C. Let Q' denote all components in Q that are adjacent to some vertex in P. Let x be a vertex in P that is adjacent to the largest number of components in Q'. Suppose that x is not adjacent to a component T of Q' . Thus there is some other vertex $x' \in P$ adjacent to T. The maximality of x implies there is a component S of Q such that x is adjacent to S but x' is not. If x is not adjacent to x', then there is a P_5 with x, s, v, r, x' with some vertices $s \in S, r \in T$. Thus x and x' belongs to a special component C of P - a contradiction. Thus, x must be adjacent to all components of Q'.

If there are two components A, B of Q' that are not dominated by x, then there are adjacent vertices $a, b \in A$, adjacent vertices $c, d \in B$ such that x is adjacent to a, c but not to b, d; but now the five vertices b, a, x, c, d form a P_5. $\square$

Given a list-coloring instance ϕ of a P_5-free graph, we will at some points need to reduce the color lists for a given connected component C. This can be done by considering all possible ways to color C's dominating clique or P_3 (Theorem 1). Since there are a constant number of vertices in such a dominating set, we obtain a constant number of new instances that together are equivalent to ϕ. For future reference, we call this function that returns this set of equivalent instances ReduceComponent(C, ϕ). If C is empty, the function simply returns ϕ.

Using this procedure along with Theorem 2, we can remove the dependencies between two dynamic sets P and Q for a given list-coloring instance ϕ. First, we find the special component C if it exists, and set $C = \emptyset$ otherwise. Then we call ReduceComponent(C, ϕ) which will effectively remove all vertices in C from P as their color lists change. Then, for each resulting coloring instance φ we remove the remaining dependencies between $P' = P - C$ and Q by applying procedure RemoveDe-pencencies(P', Q, φ) shown in Figure 3.1. In this procedure we find a vertex x and component T from Theorem 2, since we know that the special component C has already been handled. If T does not exist, then we set $T = \emptyset$. Now by considering each color in $col(P') \cap col(Q)$ along with the list $col(P') - col(Q)$ we can create a set of equivalent instances to φ (as described in Section 2). If we modify φ by assigning x a color from $col(P') \cap col(Q)$, then all vertices in Q adjacent to x will have their color list reduced by the color of x. Thus, only the vertices in T may still have dependencies with the original set P - but these dependencies can be removed by a single call to ReduceComponent. In the single remaining instance where we modify φ by assigning x the color list $col(P) - col(Q)$, we simply repeat this process (at most $|P|$ times) by setting $P' = P' - \{x\}$ until there are no remaining dependencies between P and Q. Thus, each iteration of RemoveDependencies produces a constant number of instances with no dependencies between P and Q and one instance in which the size of P' is reduced by one at least one.

The output of this step is $O(n)$ list-coloring instances (that are obtained in polynomial time), with no dependencies between P and Q, that together are equivalent to the original instance ϕ. Since there are a constant number of pairs of dynamic sets for each pair of fixed sets, and since there are a constant number of pairs of fixed sets, this proves the following theorem:

THEOREM 3 *Determining whether or not a P_5-free graph can be colored with k-colors can be decided in polynomial time.*

4 Summary

In this paper, we obtain a theorem (Theorem 2) on the structure of P_5-free graphs and use it to design a polynomial-time algorithm that determines whether a P_5-free graph can be k-colored. The algorithm recursively uses list coloring techniques and thus its complexity is high even though it is polynomial, as is the case with all list coloring algorithms. In a related paper (in preparation), we will give a slightly faster algorithm also based on list coloring techniques, however this algorithm provides less insight into the structure of P_5-free graphs. It would be of interest to find a polynomial-time algorithm to k-color a P_5-free graph without using list coloring techniques.

Continuing with this vein of research, the following open problems are perhaps the next interesting avenues for future research:

- Does there exist a polynomial time algorithm to determine whether or not a P_7-free graph can 3-colored?
- Does there exist a polynomial time algorithm to determine whether or not a P_6-free graph can 4-colored?
- Is the problem of k-coloring a P_7-free graph NP-complete for any $k \geq 3$?

Two other related open problems are to determine the complexities of the MAXIMUM INDEPENDENT SET and MINIMUM INDEPENDENT DOMINATING SET problems on P_5-free graphs.

References

[1] Jorge L. Ramirez Alfonsin, Bruce A. Reed, *Perfect Graphs*, John Wiley & Sons, LTD, 2001

[2] G. Bacsó and Z. Tuza, Dominating cliques in P_5-free graphs, Period. Math. Hungar. Vol. 21 No. 4 (1990) 303-308.

[3] C. Berge and V. Chvátal (eds.), Topics on perfect graphs, North-Holland, Amsterdam, 1984.

[4] S. Even, A. Pnueli and A. Lempel, Permutation graphs and transitive graphs, J. Assoc. Comput. Mach. 19 (1972) 400-410.

[5] M. R. Garey, D. S. Johnson and L. Stockmeyer, Some simplified NP-complete problems, Theo. Comput. Sci. 1, (1976) 237-267.

[6] F. Gavril, Algorithms for minimum coloring, maximum clique, minimum coloring by cliques, and maximum independent set of a chordal graph, SIAM J. Comput. 1, (1972) 180-187.

[7] M. C. Golumbic, *Algorithmic graph theory and perfect graphs*, Academic Press, New York, 1980.

[8] M. Grötschel, L. Lovász and A. Schrjver, The ellipsoid method and its consequences in combinatorial optimization, Combinatorica 1, (1981) 169-197.

[9] R. Hayward, C. T. Hoàng and F. Maffray, Optimizing weakly triangulated graphs, Graphs and Combinatorics 5 (1989) 339-349.

[10] R. M. Karp, Reducibility among combinatorial problems. In: R. E. Miller and J. W. Thatcher (eds), *Complexity of Computer Computations*, Plenum Press, New York, (1972) 85-103.

[11] D. Kral, J. Kratochvil, Z. Tuza and G. J. Woeginger, Complexity of coloring graphs without forbidden induced subgraphs, in: WG 2001, LNCS 2204, (2001) 254-262.

[12] B. Randerath and I. Schiermeyer, *3-Colorability* $\in P$ *for* P_6*-free graphs*, Discrete Applied Mathematics, Vol. 136 No. 2-3 (2004) 299-313.

[13] J. Sgall, G.J. Woeginger, The complexity of coloring graphs without long induced paths, Acta Cybernet. Vol. 15 No. 1 (2001) 107-117.