Deciding k-colorability of P_5-free graphs in polynomial time

Chính T. Hoàng[*]   Marcin Kamiński[†]   Vadim Lozin[‡]   Joe Sawada[§]   Xiao Shu[¶]

April 16, 2008

Abstract

The problem of computing the chromatic number of a P_5-free graph (a graph which contains no path on 5 vertices as an induced subgraph) is known to be NP-hard. However, we show that for every fixed integer k, there exists a polynomial-time algorithm determining whether or not a P_5-free graph admits a k-coloring, and finding one, if it does.

Keywords: graph coloring, dominating clique, polynomial-time algorithm, P_5-free graph

1 Introduction

A k-*coloring* of a graph is an assignment of k colors to its vertices so that no two adjacent vertices receive the same color. The k-COLORABILITY problem is to determine whether or not a given graph G admits a k-coloring, and to output one, if it does. The smallest k for which exists a k-coloring of a graph is called its *chromatic number*. The algorithmic problem of computing it will be referred to as CHROMATIC NUMBER.

The k-COLORABILITY is one of the central problems of algorithmic graph theory with numerous applications [4]. It is also one of the most difficult problems: it is NP-complete in general [12] and remains difficult in many restricted graph families, for example in triangle-free graphs [17] or line graphs [11]. Moreover, even approximating the chromatic number of a graph is hard [13].

On the other hand, the problem can be solved in polynomial time when restricted to some families of graphs such as perfect graphs [8]. Efficient polynomial-time algorithms for coloring graphs in particular subclasses of perfect graphs (including chordal graphs [6], weakly chordal graphs [9], and comparability graphs [5]) are available.

All the families of graphs mentioned above have the property that together with any graph G they contain all induced subgraphs of G. Such classes are called *hereditary*. Every hereditary class of graphs can be described by its *forbidden induced subgraphs*, i.e. the unique set of minimal graphs which do not belong to the class. A

---

[*]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	9	10	11	12	...
3	$O(m)$	$O(m)$	$O(n^{\alpha})$	$O(mn^{\alpha})$	?	?	?	?	?	?	...
4	$O(m)$	$O(m)$	**??**	?	?	?	NP_c	NP_c	NP_c	NP_c	...
5	$O(m)$	$O(m)$	**??**	?	?	NP_c	NP_c	NP_c	NP_c	NP_c	...
6	$O(m)$	$O(m)$	**??**	?	?	NP_c	NP_c	NP_c	NP_c	NP_c	...
7	$O(m)$	$O(m)$	**??**	?	?	NP_c	NP_c	NP_c	NP_c	NP_c	...
...	...	...	**...**	...	...	...	...	...	...	...	...

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

comprehensive survey on coloring of graphs in hereditary classes can be found in [18]. An important line of research on colorability of graphs in hereditary classes deals with P_t-free graphs. The induced path on t vertices is called P_t, and a graph is called P_t-*free* if it does not contain P_t as an induced subgraph.

Sgall and Woeginger showed in [21] that 5-COLORABILITY is NP-complete for P_8-free graphs and 4-COLORABI-LITY is NP-complete for P_{12}-free graphs. The last result was improved in [16]; the authors claim that modifying the reduction from [21] 4-COLORABILITY can be shown to be NP-complete for P_9-free graphs. On the other hand, the k-COLORABILITY problem can be solved in polynomial time for P_4-free graphs (since they are perfect). For $t = 5, 6, 7$, the complexity of the problem is generally unknown, except for the case of 3-COLORABILITY of P_5-free [20, 21] and P_6-free graphs [19]. Known results on the k-COLORABILITY problem in P_t-free graphs are summarized in Table 1 (n is the number of the input graph, m the number of edges, and α is matrix multiplication exponent known to satisfy $2 \leq \alpha < 2.376$ [2]).

In this paper, we study the class of P_5-free graphs. This is the minimal class from Table 1 where the complexity of the k-COLORABILITY problem was not known. This class also proves resistant with respect to other graph problems. For instance, P_5-free graphs is the unique minimal class defined by a single forbidden induced subgraph with unknown complexity of the MAXIMUM INDEPENDENT SET and MINIMUM INDEPENDENT DOMINATING SET problems. Many algorithmic problems are known to be NP-hard in the class of P_5-free graphs, for example DOMINATING SET [14] and CHROMATIC NUMBER [15]. In contrast to the NP-hardness of finding the chromatic number of a P_5-free graph, we show that k-COLORABILITY can be solved in this class in polynomial time for every particular value of k. In the case of a positive answer, our algorithm returns a valid k-coloring. Along with the mentioned result on 3-COLORABILITY of P_5-free graphs, our solution generalizes several other previously studied special cases of the problem, such as 4-COLORABILITY of (P_5, C_5)-free graphs [16] and 4-COLORABILITY of P_5-free graphs containing a dominating clique on four vertices [10]. We also note the algorithm in [7] which colors a $(P_5, \overline{P}_5)$-free graph using at most the square of its chromatic number of colors.

2   Background and Definitions

Coloring vertices. Graphs considered in this paper are simple and undirected. We denote the number of vertices of a graph by n. If A is a subset of V, then $G[A]$ is the subgraph of G induced by A. An *independent set* is a set of mutually non-adjacent vertices. A set of vertices D is called a *dominating set* of the graph, if every vertex in V belongs to D or has a neighbor in D. For notions not defined here, we refer the reader to [3].

Recall that a k-*coloring* of a graph is an assignment of numbers from the set $\{1, \ldots, k\}$ (called *colors*) to the vertices of the graph in such a way that the endpoints of each edge receive different colors. For our purposes, it is more convenient to study a generalization of k-coloring. In this variant of coloring, each vertex v has its *palette*,

a subset of $\{1, \ldots, k\}$, denoted by $\ell(v)$. For a subset of vertices W, the union of palettes of the vertices in W will be denoted by $\ell(W)$.

The *k-restricted-coloring* of a graph with respect to the palettes of its vertices, is a k-coloring such that each vertex is assigned a color from its palette. Clearly, k-coloring is a special case of k-restricted-coloring, where all palettes are equal to $\{1, \ldots, k\}$. We say that an instance of k-restricted-coloring problem is *k-colorable* if there exists a k-restricted-coloring of this instance. (Below, "k-colorable" will always mean "k-colorable with respect to some instance of k-restricted-coloring", unless stated otherwise.)

Notice that once a vertex is assigned a color, this color can be removed from the palettes of all its neighbors. Also, when looking for a coloring of a graph, two adjacent vertices with disjoint palettes can be as well thought of as non-adjacent. Essential are only those pairs of adjacent vertices whose palettes are not disjoint. This observation motivates the following definition: two adjacent vertices are called *essential* if their palettes are not disjoint; otherwise they are called *non-essential*. Notice that the relation of being an essential (a non-essential) neighbor is symmetric. Assigning a color to a vertex does not change possible color choices for its non-essential neighbors. Two disjoint sets of vertices are said to be *separated* if no vertex in one of them has an essential neighbor in the other.

The main algorithmic problem studied in this paper is the k-RESTRICTED-COLORING problem. The input instance is a graph together with palettes for all its vertices. The problem is to decide if the input instance is k-colorable, and if so, return a k-restricted-coloring.

Dominating structure. Our algorithm is based on an interesting structural property of P_5-free graphs that has been described by Bacsó and Tuza in [1]. Following their terminology, we say that a graph H *is dominating in G* if G contains a dominating set that induces a graph isomorphic to H. In particular, a dominating clique in G is a dominating set which induces a complete graph. Similarly, a dominating P_3 is a dominating set which induces a path on 3 vertices.

THEOREM 1 (THEOREM 8 IN [1]) *Every connected P_5-free graph has a dominating clique or a dominating P_3.*

We will refer to a dominating set that induces a complete graph or P_3 as *a dominating structure*. To give an application of the theorem, let us consider the 3-RESTRICTED-COLORING problem in the class of P_5-free graphs. Notice that once a 3-coloring of the vertices in a dominating structure D is fixed, then the palettes of all vertices in $V - D$ can be truncated. For every fixed 3-coloring of D, the updated palettes of vertices contain at most 2 colors. The question whether the coloring of D can be extended to the whole graph, can be modeled as a 2-SAT instance and solved in polynomial time. Hence, considering all possible 3-colorings of D and checking extendability of each, we can obtain a polynomial-time algorithm for 3-RESTRICTED-COLORING in the class of P_5-free graphs. (See [20] for more details.)

Given a dominating structure D, let us fix an ordering $d_1, d_2, \ldots, d_{|D|}$ of vertices in D. Next, partition all vertices of $V - D$ into disjoint subsets depending on their neighborhood in D. Let F_1 be the neighborhood of d_1 in $V - D$, and for $i = 2, \ldots, |D|$ let F_i be the set of vertices in $V - D$ adjacent to d_i but not to any d_j with $j < i$. The sets F_i for $i = 1, \ldots, |D|$ will be referred to as *fixed sets*.

3   The Algorithm

We say that an instance G is *compatible* with a set of instances $\mathcal{G}$ if G is k-colorable if and only if at least one of the instances in $\mathcal{G}$ is k-colorable. Notice that if G is compatible with $\mathcal{G}$ and some $H \in \mathcal{G}$ is compatible with $\mathcal{H}$,

then G is compatible with $(\mathcal{G} - H) \cup \mathcal{H}$.

Our approach is based on a search tree technique. The nodes of the search tree are instances of the problem; the root corresponds to the input instance, and the children of a node form a set of instances compatible with the node. The leaves are instances which can be solved in an elementary way. In other words, an instance G is replaced be a set of instances compatible with G but (in some sense) simpler than G. We proceed recursively until all instances are easy to solve. Notice that if the degree of each node in the tree is polynomial (in n) and the depth of the tree is constant, then the tree has at most a polynomial number of nodes. (Hence, a polynomial number of leaves.)

The nature of our solution is also recursive in k. Designing the algorithm which solves the k-RESTRICTED-COLORING problem, we assume there exist polynomial-time algorithms for the same problem with smaller values of k. The problem can be easily solved for $k = 1, 2$, and as we have seen above, for $k = 3$; below we assume that k is at least 4.

For the purpose of clarity, we split the presentation of the algorithm into four blocks: finding a dominating structure, separating independent sets, separating fixed sets and the main algorithm.

Finding a dominating structure. First we find a dominating structure D of size at most k. (This can clearly be done in polynomial time.) If no such set exists, then the instance does not admit a k-restricted-coloring. Otherwise, we fix an ordering of the vertex set and consider the fixed sets with respect to this ordering. Even though we consider different instances below, the fixed sets remain the same for the runtime of the algorithm.

Separating two independent sets. Let X and Y be two independent sets of a P_5-free graph belonging to two different fixed sets. Also, let X' (respectively Y') be the set of vertices of X (respectively Y) that are essential neighbors of some vertex of Y (respectively X). Note that X' is non-empty if and only if Y' is non-empty. We also want to stress that the sets X and Y depend only on the graph (and not the palettes) but the sets X' and Y' depend on the instance (the palettes) and can change when a coloring of the graph is modified.

LEMMA 1 *If $X' \neq \emptyset$, there exists a vertex in X' which is adjacent to all vertices in Y'.*

Proof. Let x_1 be a vertex of X' with a maximal neighborhood in Y'. Assume there exists a vertex $y_2 \in Y'$ that is not adjacent to x_1. Then, there must exist a vertex $x_2 \in X'$ (different than x_1) adjacent to y_2. Also, by the choice of x_1, there must exist a vertex $y_1 \in Y'$ that is adjacent to x_1 but not x_2. Remember that X' (Y') are independent sets and so there is no edge between x_1 and x_2 (y_1 and y_2). Since X and Y belong to different fixed sets, there exists a vertex v in the dominating set such that either v is adjacent to x_1, x_2 but not y_1, y_2, or v is adjacent to y_1, y_2 but not x_1, x_2. But then $G[\{v, x_1, x_2, y_1, y_2\}]$ is an induced P_5; a contradiction. $\square$

LEMMA 2 *Let H be an instance of k-RESTRICTED-COLORING and X and Y be two independent sets of a graph belonging to two different fixed sets. There exists a polynomial-time algorithm which outputs a set of instances $\mathcal{H}$ compatible with H such that for every instance in $\mathcal{H}$ sets X and Y are separated.*

Proof. We split the proof of the lemma into two steps. First we describe a polynomial-time recursive procedure which outputs a set of instances $\mathcal{G}$ compatible with the input instance, such that for every instance in $\mathcal{G}$ the set X' is empty or the size of the palette of Y' is smaller than the size of the palette in the input instance.

Let us suppose that the procedure was called for an input instance G. Initialize $\mathcal{G}$ to be empty. If $X' \neq \emptyset$, then there is nothing to be done. Otherwise, there exists a vertex $v \in X'$ which dominates Y' (Lemma 1). For

every color $c \in \ell(v) \cap \ell(Y')$, create a new instance by coloring v with c and removing c from the color lists of the neighbors of v. Add all these instances to $\mathcal{G}$ and notice that for the new instances $|\ell(Y')|$ is strictly smaller than $|\ell(Y')|$ in the original instance. Additionally, create an instance in which the color list of v is truncated, $\ell(v) = \ell(v) - \ell(Y')$ (if non-empty), and run the procedure recursively for this instance. Add the result of the recursive call to $\mathcal{G}$ and return $\mathcal{G}$.

A run of the procedure can be seen as a search tree. At each level of recursion, we create at most k instances plus the ones coming from the recursive call. As we decrease the size of X' at each level, the depth of recursion is at most n. Hence, the number of instances created by the procedure is at most kn.

It follows easily from the search tree structure of the algorithm that at each level of recursion, the node is compatible with the set of its children. Hence, by the fact that an instance can be replaced with its compatible set (as mentioned in the beginning of this section) the input instance G is compatible with the output set $\mathcal{G}$.

The time spent at every node is clearly polynomial and the number of nodes of the search tree is also polynomial, thus the procedure runs in polynomial time.

Calling the procedure described above recursively until for all instances X' is empty builds another search tree. Every node of this search tree is an instance of the problem, the root corresponds to the input instance H and in the leaves sets X and Y are separated. Notice that the degree of every node is at most kn and the depth of the tree is also at most k (we decrease the palette of Y' at every step). Hence, the number of nodes is at most $(kn)^k$. The correctness follows from the correctness of the procedure presented above. $\square$

Separating fixed sets. Now we are ready to present an algorithm which produces a set of instances (compatible with the input instance) such that in all instances every pair of fixed sets is separated.

LEMMA 3 *Let H be an instance of k-RESTRICTED-COLORING. There exists a polynomial-time algorithm which outputs a set of instances $\mathcal{H}$ compatible with H such that for every instance in $\mathcal{H}$ all pairs of fixed sets are separated.*

Proof. First let us notice that if an instance admits a k-restricted-coloring, then the underlying graph is k-colorable in the usual sense. Since all vertices of every fixed set F_i are adjacent to the same vertex in D, then if H admits a k-restricted-coloring, then the graphs $H[F_i]$ must be $(k-1)$-colorable in the usual sense. Using our recursive assumption we can find in polynomial time a $(k-1)$-coloring of $H[F_i]$ for every fixed set F_i. Let us find these colorings for all fixed sets F_i. If some fixed set is not $(k-1)$-colorable (in the usual sense), then H does not admit a k-restricted-coloring. We will use $(k-1)$-colorings of the fixed sets only for the purpose of partitioning these sets and the colorings found have nothing to do with the final restricted colorings of each of the sets.

Now we will present a procedure which for a given input instance G and two distinct fixed sets F_i and F_j outputs a set of instances $\mathcal{G}$ such that for every instance in $\mathcal{G}$ the sets F_i and F_j are separated. Each color class is an independent set and for each pair of color classes – one from $G[F_i]$ and from $G[F_j]$ – we can run the algorithm separating two independent sets. Let us do it recursively until all pairs of color classes are separated, and hence sets F_i and F_j are separated. The procedure will create at most $((kn)^k)^{k^2} = (kn)^{k^3}$ instances compatible with G.

Now, we can run the procedure for the input instance H and then recursively until all pairs of fixed sets are separated. There is at most k^2 different pairs of fixed sets, so the number of output instances is at most $(kn)^{k^5}$. Each node requires a polynomial-time processing time, so the total running time of the algorithm is polynomial. The correctness follows recursively from the correctness of the procedure which separates two given fixed sets. $\square$

Main algorithm. Now we are ready to put all the components of the algorithm together.

THEOREM 2 *The k-RESTRICTED-COLORING problem can be solved in polynomial time in the class of P_5-free graphs.*

Proof. First find a dominating structure D. Choose an ordering of vertices of D and partition $V - D$ into fixed sets. This can be done in polynomial time. Consider all possible colorings of D (there is at most k^k of them) and for each coloring: separate all pairs of fixed sets. This, as shown in Lemma 3, can be done in polynomial time. Now all the instances have all pairs of fixed sets separated.

Let G be an instance in which all pairs of fixed sets are separated. Notice that G is k-colorable if and only if for every fixed set F_i the graph $G[F_i]$ induced by F_i is $(k - 1)$-colorable. Running the $(k - 1)$-coloring algorithm (which exists by our inductive assumption) for each $G[F_i]$, we obtain a polynomial-time algorithm for the k-RESTRICTED-COLORING problem. This step can be done in polynomial time, and hence k-RESTRICTED-COLORING can be solved in polynomial time in the class of P_5-free graphs. $\square$

4   Open problems

The algorithm presented in this paper narrows the gap in Table 1 between the cases solvable in NP-complete and polynomial time solvable cases. Here we would like to suggest some open problems which are in our opinion should be addressed next:

- Is 3-coloring of P_7-free graphs solvable in polynomial-time?

- Is 4-coloring of P_6-free graphs solvable in polynomial-time?

- Can k-coloring of P_5-free graphs be solved in FPT time?

- Is MAXIMUM INDEPENDENT SET (and MINIMUM INDEPENDENT DOMINATING SET) solvable in polynomial-time on P_5-free graphs?

References

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

[2] D. Coppersmith and S. Winograd, Matrix multiplication via arithmetic progressions. Journal of Symbolic Computation, Vol. 9 No. 3(1990) 251-280.

[3] Reinhard Diestel, Graph Theory, Electronic Edition 2005.

[4] D. de Werra and D. Kobler, Graph coloring: foundations and applications, RAIRO Oper. Res. 37 (2003) 29-66.

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

[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] V. Giakoumakis and I. Rusu, Weighted parameters in $(P_5, \overline{P}_5)$-free graphs, Discrete Applied Math. 80 (1997) 255-261.

[8] M. Grötschel, L. Lovász and A. Schrijver, Polynomial algorithms for perfect graphs, Ann. Discrete Math. 21 (1984) 325-356.

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

[10] C. T. Hoàng, J. Sawada and Z. Wang, Colorability of P_5-free graphs, manuscript, 2005.

[11] I. Holyer, The NP-completeness of edge-coloring, SIAM J. Computing, 10 (1981) 718-720.

[12] 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.

[13] S. Khanna, N. Linial and S. Safra, On the hardness of approximating the chromatic number, Combinatorica 20 (2000) 393-415.

[14] D.V. Korobitsyn, On the complexity of determining the domination number in monogenic classes of graphs, Diskret. Mat. 2, N 3 (1990), 90-96 in Russian, translation in Discrete Mathematics and Applications, 2 (1992), no. 2, 191-199).

[15] 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.

[16] V. Bang Le, B. Randerath, I. Schiermeyer, Two remarks on coloring graphs without long induced paths, in Report No. 7/2006 (Algorithmic Graph Theory), Mathematisches Forschungsinstitut Oberwolfach.

[17] F. Maffray and M. Preissmann, On the NP-completeness of the k-colorability problem for triangle-free graphs, Discrete Math. 162 (1996) 313-317.

[18] B. Randerath, I. Schiermeyer, Vertex coloring and forbidden subgraphs – a survey, Graphs and Combinatorics, 20(1) (2004) 1-40.

[19] B. Randerath, I. Schiermeyer, 3-colorability $\in \mathcal{P}$ for P_6-free graphs, Discrete Applied Mathematics 136 (2004) 299-313.

[20] B. Randerath, I. Schiermeyer, M. Tewes, Three-colorability and forbidden subgraphs. II: polynomial algorithms, Discrete Mathematics 251 (2002) 137-153.

[21] J. Sgall, G. J. Woeginger, The complexity of coloring graphs without long induced paths, Acta Cybernetica 15(1), (2001) 107-117.