

# Some Coloring and Walking Problems in Graphs

THÈSE N° 4090 (2008)

PRÉSENTÉE LE 5 JUIN 2008

À LA FACULTE DES SCIENCES DE BASE  
CHAIRE DE RECHERCHE OPÉRATIONNELLE ROSE  
PROGRAMME DOCTORAL EN MATHÉMATIQUES

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

**Benjamin LEROY-BEAULIEU**

ingénieur informaticien diplômé EPF  
de nationalité suisse et originaire de Oberburg (BE)

acceptée sur proposition du jury:

Prof. T. Mountford, président du jury  
Prof. D. de Werra, directeur de thèse  
Prof. J. Blazewicz, rapporteur  
Prof. M. Demange, rapporteur  
Prof. T. Lieblich, rapporteur



ÉCOLE POLYTECHNIQUE  
FÉDÉRALE DE LAUSANNE

Suisse  
2008



# Remerciements

Tout d'abord, merci Dominique. Merci pour tes encouragements, pour ton aide à l'immersion, non seulement dans la matière, mais aussi dans la société de la recherche opérationnelle, au travers de conférences, d'invitations, de voyages. Merci aussi, pour la bonne ambiance au sein de la rose, les nombreux soupers en toutes occasions, les marches, les verrées.

Merci Marc. Tu as été, officieusement, mon co-directeur de thèse. C'est toi qui m'a fait connaître la coloration online et qui m'as appris à l'étudier. Merci aussi pour les nombreux séjours à Paris, à Lausanne, à L'Aquila, qui ont chaque fois été un parfait mélange de travail et de sorties, de restos. À ce sujet, merci de ne pas m'avoir obligé à manger les crabes en mue. J'espère qu'on fera encore souvent des randonnées ensemble. Promis, l'année prochaine, je viens à la marche de la Bièvre.

Merci à tous ceux avec qui j'ai travaillé: Jacek Błażewicz, Marta Kasprzak, Gabriele di Stefano, Bernard Kouakou, Gaël Haab. Thank you for the nice and fruitful stays in Poznan and in L'Aquila. Merci pour les heures passées ensemble à plancher sur les graphes d'intervalles ou sur les overlap graphs.

Merci Jocelyne, pour ton énergie constante et pour ta participation à mon équipe Bike to Work. Merci David, pour ton théorème bien utile *Apéro cumulé = apéro annulé*. Merci Ivo, pour m'avoir fait découvrir l'ail des ours. Merci Tinaz, pour m'avoir fait découvrir Nâzım Hikmet. Merci Bernard, pour m'avoir fait découvrir les cuisses de canard confites. Merci Şivan, pour m'avoir fait découvrir l'extension Send Later. Merci Liliana pour m'avoir appris à jouer à Ticket to Ride. Merci Telma, pour m'avoir fait découvrir la mousse aux framboises. Merci Matthieu, pour m'avoir fait découvrir Battle for Wesnoth. Merci Sem, pour m'avoir fait découvrir la coinche. Merci Benjamin (l'autre) pour m'avoir fait découvrir Clans. Merci Daniela, Sacha, Nicolas, David C.

Merci à Thomas Liebling, pour m'avoir encouragé, il y a six ans, à faire une thèse et pour m'avoir aidé à trouver, il y a 3 ans et demi, un directeur de thèse.

Merci à ma famille, bien sûr, et tout particulièrement à mes parents, pour

m'avoir permis de faire des études.

Et puis ça n'a peut-être rien à voir avec cette thèse, mais merci quand-même à tous mes amis pour les bons moments passés ensemble, pour les sommets gravis, pour les cols franchis, pour les sorties à ski, pour les campagnes électorales où on a bien ri. Un merci tout particulier à Ali qui a relu cette thèse et sans qui il y aurait bien plus de fautes.

# Résumé

La théorie des graphes est un domaine important des mathématiques discrètes. C'est un domaine particulièrement intéressant parce qu'il a beaucoup d'applications. Deux des principaux problèmes de la théorie des graphes sont la coloration et la recherche de circuits hamiltoniens.

Le chapitre 1 donne des définitions de bases de la théorie des graphes, telles que la coloration, la coloration bornée par un entier  $b$  et les circuits et chemins hamiltoniens. Nous y présentons également les algorithmes online et en particulier la coloration online.

Le chapitre 2 commence par quelques remarques générales à propos des recouvrements online de graphes par des ensembles de tailles bornées (tels que la coloration bornée ou le recouvrement par des cliques de tailles bornées): nous y montrons une méthode simple pour transformer un algorithme de recouvrement online en un algorithme de recouvrement online borné, et pour calculer le rapport de performance du nouvel algorithme borné en fonction de celui de l'algorithme non borné. Nous montrons par la suite que cette transformation donne souvent lieu à des algorithmes online optimaux. De plus, nous présentons dans ce chapitre quelques résultats préliminaires sur le recouvrement borné: pour tout graphe, le rapport de performance est inférieur ou égal à  $\frac{1}{2} + \frac{b}{2}$  et pour  $b = 2$ , ce rapport est optimal.

Dans la deuxième partie de ce chapitre, nous nous intéressons à la coloration online de graphes de co-intervalles. En nous basant sur deux applications industrielles, nous étudions deux versions différentes de ce problème. Dans le cas où les intervalles sont présentés dans l'ordre croissant de leurs extrémités gauches, nous montrons que le rapport de performance est 1 dans le cas non borné et  $2 - \frac{1}{b}$  dans le cas borné. Dans le cas où les intervalles peuvent être présentés dans n'importe quel ordre, nous montrons que le rapport de performance est plus petit que 3 dans le cas borné.

Dans le chapitre 3, nous nous intéressons à la coloration online des graphes de permutations et des graphes de comparabilité. Tout d'abord, nous montrons que l'algorithme First-Fit a un rapport de performance de  $O(\sqrt{n})$  pour les graphes de permutations bipartis et que cette borne est atteinte même

pour certains ordres de présentation simples. Ensuite, nous montrons que, pour les deux classes de graphes, le rapport de performance est inférieur à  $\frac{\chi+1}{2}$  dans le cas non borné et que le rapport de performance de First-Fit est égal à  $\frac{1}{2} + \frac{b}{2}$  dans le cas borné.

Dans la deuxième partie de ce chapitre, nous nous intéressons à la coloration des graphes de permutations. Nous montrons que le rapport de performance est exactement  $\frac{n}{4} + \frac{1}{2}$  et nous donnons de meilleures bornes dans des cas particuliers.

Le chapitre 4 est consacré à une application de la coloration online: l'affectation de voies ferroviaires à des trains. Selon les hypothèses que l'on fait, ce problème peut être modélisé par de la coloration online de graphes de permutations ou de graphes de chevauchements d'intervalles.

Nous montrons que, si un graphe de permutations est présenté d'ouest en est sur un plan quadrillé, alors le rapport de performance est exactement  $2 - \frac{1}{\min\{b,k\}}$ , où  $k$  est la meilleure borne supérieure connue au nombre chromatique borné. Nous montrons également que si un graphe de permutations est présenté sur un plan quadrillé en commençant par l'origine, puis en progressant en direction de l'ouest et, indépendamment, en direction de l'est, alors le rapport de performance est exactement  $2 - \frac{1}{\chi}$ .

Dans le cas des graphes de chevauchements d'intervalles, nous montrons que le rapport de performance n'est pas borné par une constante, même si le graphe est biparti et est présenté dans l'ordre croissant des extrémités gauches des intervalles. Dans ce cas particulier, nous montrons que First-Fit a un rapport de performance de  $O(\sqrt{n})$ . Nous nous intéressons ensuite au cas où les longueurs des intervalles sont comprises entre 1 et un nombre  $M$ . Dans ce cas, nous montrons que le rapport de performance est borné supérieurement par  $2\sqrt{M}$  si  $M \leq M_0$ , et par  $\log M (\lceil \log M / \log \log M \rceil + 1)$  si  $M > M_0$ , où  $M_0$  est tel que  $2\sqrt{M_0} = 3 \log(M_0)$ . Si  $M$  est grand, alors le rapport de performance est  $O(\log^2 M / \log \log M)$ .

Dans le chapitre 5, nous nous intéressons à la coloration online d'arbres, de forêts et de graphes scindés. Nous montrons que pour les arbres et les forêts, le rapport de performance est exactement  $\frac{1}{2} \log_2(2n)$  dans le cas non borné et est inférieur ou égal à  $1 + \frac{\lfloor \log_2(b) \rfloor}{\chi_b}$  dans le cas borné. Pour les graphes scindés, nous montrons que le rapport de performance est exactement  $1 + \frac{1}{\chi}$  dans le cas non borné et est inférieur ou égal à  $2 + \frac{1}{\chi_b} - \frac{3}{b}$  dans le cas borné.

Dans le chapitre 6, nous présentons une classe de graphes orientés: les graphes quasi-adjoints. Ces graphes forment une super-classe des graphes adjoints ainsi que des graphes utilisés pour un algorithme de séquençage d'ADN dans (Blazewicz, Kasprzak, "Computational complexity of isothermic DNA sequencing by hybridization.", 2006). Nous y donnons un algorithme

polynomial en  $O(n^3)$  pour les reconnaître et un autre algorithme polynomial en  $O(n^2 + m^2)$  pour y trouver un circuit hamiltonien ou affirmer qu'il n'y en a pas. De plus, nous y étudions quelques problèmes liés, tel que celui de trouver un chemin eulérien tout en respectant certaines transitions interdites.

**Mots-clefs:** coloration online, couverture par cliques online, circuit hamiltonien, graphes de permutations, graphes de comparabilité, graphes de co-intervalles, graphes d'intervalles, graphes de chevauchement d'intervalles, arbres, forêts, graphes scindés, graphes quasi-adjoints





# Abstract

Graph theory is an important topic in discrete mathematics. It is particularly interesting because it has a wide range of applications. Among the main problems in graph theory, we shall mention the following ones: graph coloring and the Hamiltonian circuit problem.

Chapter 1 presents basic definitions of graph theory, such as graph coloring, graph coloring with color-classes of bounded size  $b$ , and Hamiltonian circuits and paths. We also presents online algorithms and online coloring.

Chapter 2 starts with some general remarks about online graph covering with sets of bounded sizes (such as online bounded coloring): we give a simple method for transforming an online covering algorithm into an online bounded covering algorithm, and to derive the performance ratio of the bounded algorithm from the performance ratio of the unbounded algorithm. As will be shown in later chapters, this method often leads to optimal results. Furthermore, some basic preliminary results on online graph covering with sets of bounded size are given: for every graph, the performance ratio is bounded above by  $\frac{1}{2} + \frac{b}{2}$  and for  $b = 2$ , this bound is optimal.

In the second part, online coloring of co-interval graphs is studied. Based on two industrial applications, two different versions of this problem are discussed. In the case where the intervals are presented in increasing order of their left ends, we show that the performance ratio is 1 in the unbounded case and  $2 - \frac{1}{b}$  in the bounded case. In the case where the intervals may be presented in any order, we show that the performance ratio is at most 3 in the bounded case.

Chapter 3 deals with online coloring of permutation and comparability graphs. First, we give a tight analysis of the First-Fit algorithm on bipartite permutation graphs and we show that its performance ratio is  $O(\sqrt{n})$ , even for some simple presentation orders. For both classes of graphs, we show that the performance ratio is bounded above by  $\frac{\chi+1}{2}$  in the unbounded case and that the performance ratio of First-Fit is equal to  $\frac{1}{2} + \frac{b}{2}$  in the bounded case.

In the second part of this chapter, we study cocoloring of permutation

graphs. We show that the performance ratio is  $\frac{n}{4} + \frac{1}{2}$  and we give better bounds in some more restricted cocoloring problems.

Chapter 4 deals with an application of online coloring: the online Track Assignment Problem. Depending on the assumptions that are made, the Track Assignment Problem can be reduced to coloring permutation or overlap graphs online.

We show that when a permutation graph is presented on a latticial plane, from west to east, then the performance ratio is exactly  $2 - \frac{1}{\min\{b,k\}}$ , where  $k$  is the best known upper bound on the bounded chromatic number. We also show that, when a permutation graph is presented on a latticial plan, starting from the origin and growing, simultaneously or not, towards west and east, then the performance ratio is exactly  $2 - \frac{1}{\chi}$ .

We also show that online coloring overlap graphs does not have a performance ratio bounded by a constant, even if the overlap graph is bipartite and presented in increasing order of the intervals left ends. In this special case, we show that First-Fit has a tight performance ratio of  $O(\sqrt{n})$ . We consider coloring overlap graphs online where the intervals have a bounded size between 1 and a given number  $M$ . In this case, we show that the performance ratio can be bounded above by  $2\sqrt{M}$  if  $M \leq M_0$ , and by  $\log M (\lceil \log M / \log \log M \rceil + 1)$  if  $M > M_0$ ,  $M_0$  being defined by the equation  $2\sqrt{M_0} = 3 \log(M_0)$ . For large values of  $M$ , the ratio is  $O(\log^2 M / \log \log M)$ .

Chapter 5 is about online coloring of trees, forests and split-graphs. For trees, we show that the performance ratio of online coloring is exactly  $\frac{1}{2} \log_2(2n)$  in the unbounded case and at most  $1 + \frac{\lfloor \log_2(b) \rfloor}{\chi_b}$  in the bounded case. For split-graphs, we show that the performance ratio of online coloring is exactly  $1 + \frac{1}{\chi}$  in the unbounded case and is at most  $2 + \frac{1}{\chi_b} - \frac{3}{b}$  in the bounded case.

In Chapter 6, we present a class of digraphs: the quasi-adjoint graphs. These are a super class of both the graphs used for a DNA sequencing algorithm in (Blazewicz, Kasprzak, "Computational complexity of isothermic DNA sequencing by hybridization.", 2006) and the adjoints. A polynomial recognition algorithm in  $O(n^3)$ , as well as a polynomial algorithm in  $O(n^2 + m^2)$  for finding a Hamiltonian circuit in quasi-adjoint graphs are given. Furthermore, some results about related problems such as finding a Eulerian circuit while respecting some forbidden transitions (a sequence of two consecutive arcs) are discussed.

**Keywords:** online coloring, online clique covering, Hamiltonian circuit, permutation graphs, comparability graphs, co-interval graphs, interval graphs, overlap graphs, trees, forests, split graphs, quasi-adjoint graphs

# Contents

<b>Introduction</b>	<b>11</b>
<b>1 Basic Definitions</b>	<b>13</b>
1.1 Graph Theory . . . . .	13
1.2 Online Coloring . . . . .	18
<b>2 General Remarks and Co-interval Graphs</b>	<b>21</b>
2.1 Introduction . . . . .	22
2.2 About Graph Covering with Sets of Bounded Size . . . . .	23
2.2.1 A Simple Transformation . . . . .	23
2.2.2 On the Bounded First-Fit Algorithm . . . . .	25
2.3 The Biology Research Center Problem . . . . .	26
2.4 Chemical Compounds Refrigeration . . . . .	30
2.4.1 Analysis of First-Fit . . . . .	30
2.4.2 A Lower Bound For All Algorithms . . . . .	32
2.5 Conclusion . . . . .	35
<b>3 Comparability and Permutation Graphs</b>	<b>37</b>
3.1 Definitions and Notations . . . . .	38
3.2 Coloring . . . . .	41
3.2.1 Preliminaries . . . . .	41
3.2.2 Competitive Analysis of First-Fit . . . . .	43
3.2.3 A Better Performance Ratio . . . . .	48
3.2.4 Bounded Coloring . . . . .	54
3.3 Cocoloring . . . . .	61
3.3.1 A Dramatic Bound . . . . .	61
3.3.2 Split Permutation Graphs in a Discrete Latticial Model	63
3.3.3 Delayed Cocoloring . . . . .	67
3.4 Conclusion . . . . .	68

<b>4</b>	<b>The Track Assignment Problem</b>	<b>71</b>
4.1	Introduction . . . . .	72
4.2	Permutation Graphs . . . . .	73
4.2.1	The Train Depot with the Midnight Condition . . . . .	73
4.2.2	The Small Train Station . . . . .	77
4.2.3	The Big Train Station . . . . .	81
4.3	Overlap Graphs . . . . .	83
4.3.1	Unbounded Coloring of Overlap Graphs . . . . .	84
4.3.2	Overlap Graphs With Intervals of Bounded Size . . . . .	89
4.4	Conclusion . . . . .	94
<b>5</b>	<b>Trees, Forests and Split Graphs</b>	<b>95</b>
5.1	Trees and Forests . . . . .	96
5.1.1	Online Coloring of Trees . . . . .	96
5.1.2	Online Bounded Coloring of Trees . . . . .	101
5.2	Split Graphs . . . . .	104
5.2.1	Online Coloring of Split Graphs . . . . .	104
5.2.2	Online Bounded Coloring of Split Graphs . . . . .	106
5.3	Conclusion . . . . .	108
<b>6</b>	<b>Finding Hamiltonian Circuits in Quasi-Adjoint Graphs</b>	<b>111</b>
6.1	Definitions and Characterization . . . . .	112
6.2	The Hamiltonian Circuit Problem in Quasi-Adjoint Graphs . . . . .	115
6.3	Generalizations of Quasi-Adjoint Graphs . . . . .	121
6.3.1	Removal of Arcs . . . . .	121
6.3.2	About Forbidden Transitions . . . . .	122
6.4	Conclusion . . . . .	123
	<b>Conclusion</b>	<b>125</b>
	<b>Bibliography</b>	<b>127</b>
	<b>Curriculum Vitæ</b>	<b>135</b>

# Introduction

*Combinatorial optimization* deals with problems which have a countable number of solutions. They are frequently used for modeling several real world problems occurring in various domains such as genetics, chemistry, telecommunication, transportation, robotics, inventory control or scheduling. Indeed, setting an appropriate model for a particular problem is a difficult task; but, often, solving it in an efficient way appears to be an even harder problem. This is why the combinatorial optimization problems are at the heart of many research topics in discrete mathematics.

Most combinatorial optimization problems of great practical relevance are shown to be extremely hard to solve; this means, with a very strong evidence, that there is no polynomial time algorithm to solve them (in an exact way). These problems are said to be *NP-hard* and the time required to obtain their optimal solutions grows very quickly beyond the time limits of human life. Of course, not all combinatorial optimization problems are hard to solve; for some of them, there exists algorithms that can produce an optimal solution in polynomial time.

Real world problems, before being analyzed as combinatorial optimization problems, have first to be modeled. A powerful tool to do this is *graph theory*. Its origin goes back to the year 1736 when Leonhard Euler published a paper on the seven bridges of Königsberg problem [Eul36]. Nowadays, graph theory is one of the fastest growing areas of modern mathematics.

One of the most famous problems in graph theory is probably the *graph coloring problem*, which consists in associating a color (or an integer) to each vertex of a graph in such a way that any two adjacent vertices (i.e. vertices which are linked by an edge) get different colors. Graph coloring with a minimum number of colors is known to be NP-hard for general graphs.

For real-world applications, such as those mentioned above, the classical graph coloring problem may sometimes be a too limiting model. These applications naturally provide the requirement to consider online versions of this problem, for which decisions must be taken while the complete data is not known yet. In an *online problem*, the instance is presented step by step.

Whenever a new part is presented, the solution dealing with this part is irrevocably decided by an *online algorithm*. In this context, the usual version of the problem, where the instance is fully known in advance, is referred to as the “offline” version. Most combinatorial problems can be studied in an online version [Alb03, BEY05]. In this dissertation, we will concentrate on online graph coloring, where the vertices of the graph are presented one by one, and must get a color as they are presented.

Online graph coloring has been widely studied in general graphs [HS94, KPT94, LST89]. The results obtained so far point out that only very loose bounds can be achieved. It is thus natural to look at particular classes of graphs; in particular, classes of graphs that can be used to model real world applications. Some classes have already been studied in online cases, such as interval graphs [BBE<sup>+</sup>03], triangle-free graphs [GKL99] or  $P_5$ -free graphs [KPT95]. This work studies co-interval graphs (Chapter 2), comparability graphs (Chapter 3), permutation graphs (Chapter 3 and Chapter 4), overlap graphs (Chapter 4), trees (Chapter 5), and split-graphs (Chapter 5).

Another famous problem in graph theory is the Hamiltonian circuit problem. A Hamiltonian path in a graph is a path which visits each vertex exactly once. A Hamiltonian cycle (or Hamiltonian circuit) in a graph is a cycle which visits each vertex exactly once and also returns to the starting vertex. Determining whether such paths and circuits exist in graphs is the Hamiltonian circuit problem, which is NP-complete.

The Hamiltonian circuit problem also has interesting applications in logistics. More interestingly, it has been shown [BHKdW99, BK06] that it can be used in order to solve a problem of DNA sequencing. For this, these papers exhibit classes of graphs in which the Hamiltonian circuit problem is solvable in polynomial time. Chapter 6 characterizes the quasi-adjoint graphs, which generalize the classes discussed in the two papers mentioned above, and shows how to solve the Hamiltonian circuit problem in polynomial time in these graphs.

# Chapter 1

## Basic Definitions

### 1.1 Graph Theory

In this section, we give some fairly standard definitions on graph theory. More specific notions will be defined in corresponding chapters.

Let us start with a definition that is not restricted to the domain of graph theory, but which we will use in this work, in particular in Chapter 6: *NP-complete* problems are a famous class of equivalent decision problems with open complexity status. In fact, it is unknown whether they can all be solved in time bounded by a polynomial in the size of the input or whether they are all intractable, although the majority believes the latter. Optimization problems that are at least as hard as NP-Complete problems are called *NP-hard*. Formal definitions on NP-completeness can be found in [GJ79] or in [PS82].

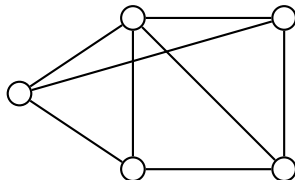
**Definition 1.1** (graph). *A graph  $G = (V, E)$  is defined by the sets  $V = \{v_1, \dots, v_n\}$  and  $E = \{v_i v_j : v_i, v_j \in V\}$ . The elements of  $V$  are called vertices and the elements of  $E$  are called edges. If the pairs in  $E$  are ordered, they are denoted by parenthesis  $(v_i, v_j)$  and are called arcs;  $G$  is then called a directed graph or a digraph.*

**Remark 1.2.** *Of course,  $(v_i, v_j) \neq (v_j, v_i)$  while  $v_i v_j = v_j v_i$ .*

In this work, graphs are simple (no multiple edges, no loops), unless otherwise stated. Figure 1.1 shows an example of graph where some pairs of vertices are linked by an edge.

The *size*  $n$  of a graph  $G = (V, E)$  is the cardinality of  $V$ . Furthermore,  $m$  is the cardinality of  $E$ .

We say that two vertices are *adjacent* if they are linked by an edge. A *neighbor* of a vertex  $v$  is a vertex which is adjacent to  $v$ .  $N(v)$  denotes the *neighborhood* of  $v$ , i.e., the set of all neighbors of  $v$ . The cardinality of  $N(v)$



**Figure 1.1:** A graph with 5 vertices and 8 edges.

is called the *degree* of  $v$  and is denoted by  $d(v)$ .  $\Delta(G)$  is the maximum degree of a vertex in  $G$ . The *non-neighborhood* of a vertex  $v$  is given by  $\overline{N}(v) = V \setminus (N(v) \cup \{v\})$ .

Two edges are said to be *adjacent* if they have a common vertex. Also, we say that an edge  $v_i v_j$  is *incident* to  $v_i$  and  $v_j$  (or equivalently  $v_i$  and  $v_j$  are *incident* to the edge  $v_i v_j$ ).

**Definition 1.3** (complement).  $\overline{G} = (V, \overline{E})$  is the complement of  $G = (V, E)$  if  $v_i v_j \in \overline{E} \Leftrightarrow v_i v_j \notin E$ .

**Remark 1.4.** If  $\overline{G}$  is the complement of  $G$ , then  $G$  is the complement of  $\overline{G}$ .

**Definition 1.5** (chain). A chain in a graph  $G = (V, E)$  is a sequence  $\langle v_1, \dots, v_k \rangle$  of distinct vertices from  $V$  such that, for all  $i \in \{1, \dots, (k-1)\}$ ,  $v_i v_{i+1} \in E$ . We say that the length of the chain is  $(k-1)$ .

**Definition 1.6** (cycle). Let  $G = (V, E)$  be a graph. Let  $\langle v_1, \dots, v_k \rangle$  be a sequence of distinct vertices such that  $v_i v_{i+1} \in E$  for all  $i = 1, \dots, k-1$  and  $v_1 v_k \in E$ . Then, it is a cycle of length  $k$ , denoted by  $C_k$ .

**Definition 1.7** (simple path). A simple path  $P$  in a digraph  $G = (V, E)$  is a sequence  $\langle v_1, \dots, v_k \rangle$  of distinct vertices from  $V$  such that  $(v_i, v_{i+1}) \in E$  for  $1 \leq i \leq k-1$ . Furthermore, if the number  $k$  of vertices of  $P$  is specified,  $P$  is denoted  $P_k$  and we say that the length of  $P_k$  is  $(k-1)$ .

**Definition 1.8** (circuit). Let  $G = (V, E)$  be a digraph. Let  $\langle v_1, \dots, v_k \rangle$  be a sequence of distinct vertices such that  $(v_i, v_{i+1}) \in E, i = 1, \dots, k-1$  and  $v_1 v_k \in E$ . Then, it is a circuit of length  $k$ .

**Definition 1.9** (Hamiltonian path). Let  $G = (V, E)$  be a digraph. A Hamiltonian path in  $G$  is a simple path that includes all the vertices of  $V$ .



**Definition 1.10** (Hamiltonian circuit). A Hamiltonian circuit in a digraph  $G = (V, E)$  is a circuit that includes all the vertices of  $V$ .

**Definition 1.11** (clique). Given a graph  $G$ , a clique is a set of pairwise adjacent vertices of  $G$ .

**Definition 1.12** (stable set). Given a graph  $G$ , a stable set is a set of pairwise non-adjacent vertices of  $G$ .

**Definition 1.13** (proper coloring). A proper coloring of a graph  $G = (V, E)$ , sometimes simply called a coloring, consists in a partitioning of  $V$  into stable sets. Each stable set is associated with a color.

**Definition 1.14** ( $k$ -coloring). A  $k$ -coloring is a coloring that uses at most  $k$  stable sets.

**Definition 1.15** ( $k$ -colorability).  $k$ -colorability is the problem of deciding whether a given graph admits a  $k$ -coloring.

**Definition 1.16** (chromatic number). The chromatic number  $\chi(G)$  of a graph  $G$  is the smallest number  $k$  such that  $G$  admits a  $k$ -coloring.

**Definition 1.17** (Min Coloring). Given a graph  $G$ , Min Coloring is the problem of finding a proper coloring of  $G$  with  $\chi(G)$  colors.

3-colorability is known to be NP-complete [GJS76], implying that Min Coloring is NP-hard in general.

**Definition 1.18** (clique covering). A clique covering of a graph  $G = (V, E)$  consists in a partition of  $V$  into cliques.

**Definition 1.19** ( $k$ -clique covering). A  $k$ -clique covering is a clique covering that uses at most  $k$  cliques.

**Definition 1.20** (clique cover number). The clique cover number  $\theta(G)$  of a graph  $G$  is the smallest number  $k$  such that  $G$  admits a  $k$ -clique covering.

Given a graph  $G$ ,  $\omega(G)$  and  $\alpha(G)$  denote respectively the size of a largest clique and the size of a largest stable set in  $G$ .  $\alpha(G)$  is also called the *stability number* of  $G$ . Clearly, for every graph  $G$ ,  $\chi(G) \geq \omega(G)$  and  $\theta(G) \geq \alpha(G)$ . Given a graph  $G$ , finding  $\omega(G)$  or  $\alpha(G)$  is NP-hard.

**Remark 1.21.** If  $\overline{G}$  is the complement of  $G$ , then  $\theta(\overline{G}) = \chi(G)$  and  $\alpha(\overline{G}) = \omega(G)$ .

**Definition 1.22** (Min Clique Covering). *Given a graph  $G = (V, E)$ , Min Clique Covering is the problem of partitioning  $V$  into  $\theta(G)$  cliques.*

Clearly, covering a graph  $G$  with cliques is equivalent to coloring its complement  $\overline{G}$ . Thus, Min Clique Covering is also NP-hard in general.

**Definition 1.23** (color-class). *Given a graph partitioning problem, a color-class is a set that is used to partition the graph.*

Typically, a color-class will be a stable set (in a coloring) or a clique (in a clique covering). In practical cases, it is sometimes convenient to consider that a color-class may not contain more than a given number of vertices. In this case, we talk about bounded coloring or bounded clique covering.

**Definition 1.24** ( $(k, b)$ -coloring). *A  $(k, b)$ -coloring is a coloring that uses at most  $k$  stable sets and where each set has a size of at most  $b$ . Such colorings are also called bounded colorings.*

**Definition 1.25** (bounded chromatic number). *The bounded chromatic number  $\chi_b(G)$  of a graph  $G$  is the smallest number  $k$  such that  $G$  admits a  $(k, b)$ -coloring for a given integer  $b$ .*

**Definition 1.26** (Min Bounded Coloring). *Given a graph  $G$ , Min Bounded Coloring is the problem of finding a  $(\chi_b(G), b)$ -coloring of  $G$  for a given integer  $b$ .*

Similarly, one can have the following definitions:

**Definition 1.27** ( $(k, b)$ -clique covering). *A  $(k, b)$ -clique covering is a clique covering that uses at most  $k$  cliques and where each clique has a size of at most  $b$  ( $b$  is given).*

**Definition 1.28** (bounded clique cover number). *For a given graph  $G$  and a given integer  $b$ , the bounded clique cover number  $\theta_b(G)$  is the smallest possible  $k$  for which a  $(k, b)$ -clique covering exists.*

**Definition 1.29** (Min Bounded Clique Covering). *Given a graph  $G$  and an integer  $b$ , Bounded Clique Covering is the problem of finding a  $(\theta_b(G), b)$ -clique covering of  $G$ .*

Min Coloring (respectively Min Clique Covering) can easily be reduced to Min Bounded Coloring (respectively Min Bounded Clique Covering) by setting  $b = n$ . Therefore, Min Bounded Coloring and Min Bounded Clique Covering are also NP-hard in general.

For the sake of simplicity, all numbers which are a function of a graph (for example  $\chi(G)$ ,  $\theta_b(G)$ , etc), are sometimes written without the “ $(G)$ ” when there is no possible confusion on the graph.

**Perfect graphs.** For a set  $V' \subset V$ ,  $G[V']$  denotes the subgraph of  $G$  induced by  $V'$ , meaning that  $G[V'] = (V', E')$  where  $E' = \{v_i v_j : v_i \in V', v_j \in V' \text{ and } v_i v_j \in E\}$ . Whereas a *partial* subgraph of  $G$  has a vertex set and an edge set, which are respectively subsets of the vertex set and the edge set of  $G$ . In what follows, a subgraph  $H$  of  $G$ , denoted by  $H \subseteq G$ , is always an induced subgraph of  $G$ , unless otherwise stated. Note that we sometimes denote  $G[V \setminus \{v\}]$  and  $G[V \setminus V']$  respectively by  $G \setminus v$  and  $G \setminus V'$ , for the sake of simplicity.

Let  $G$  and  $H$  be two graphs. We say that  $G$  is *H-free* if  $G$  does not contain  $H$  as an induced subgraph.

In the analysis of problems such as Min Coloring or Min Clique Covering, there is a particularly important class of graphs, called perfect graphs.

**Definition 1.30** (Perfect graphs). *A graph  $G$  is perfect if and only if for all  $H \subseteq G$ , we have  $\chi(H) = \omega(H)$ .*

In particular, for the class of perfect graphs, the stability number as well as the chromatic number can be determined in polynomial time (see [GLS84]). It was shown by Lovász in 1972 [Lov72] that a graph  $G$  is perfect if and only if its complement is perfect; this was known as the weak perfect graph conjecture (see [Ber76]). Therefore, a graph  $G$  is perfect if and only if for all  $H \subseteq G$ , we have  $\theta(H) = \alpha(H)$ . Let us also state the following theorem known as the Strong Perfect Graph Conjecture from 1961 (formulated by Berge [Ber61]) until it was proved by Chudnowsky, Robertson, Seymour and Thomas in 2002 [CRST06].

**Theorem 1.31** ([CRST06]). *A graph  $G$  is perfect if and only if it does not contain any odd hole (i.e.,  $C_{2k+1}, k \geq 2$ ), or odd antihole (i.e., the complement of  $C_{2k+1}, k \geq 2$ ) as an induced subgraph.*

Note that in this dissertation, we will be dealing mostly, but not always, with subclasses of perfect graphs.

A graph  $G$  is called *connected* if there is a path linking any pair of vertices in  $G$ . A *connected component* of  $G$  is then a maximal connected subgraph of  $G$ . A graph which is not connected is called *disconnected*.

A class of graphs  $\mathcal{G}$  is called *hereditary* if every subgraph of a graph in  $\mathcal{G}$  also belongs to  $\mathcal{G}$ . Note that in this dissertation, only hereditary classes of graphs are considered, unless otherwise stated.

See [Ber76] for all graph theoretical notions not given here.

## 1.2 Online Coloring

An online problem can be seen as a two players game involving an adversary and an algorithm. The adversary presents the instance and the algorithm gives the solution. The online problem is generally characterized by the underlying offline problem and two sets of rules that have to be respected by the adversary and the algorithm, respectively.

**Definition 1.32** (performance ratio). *Online algorithms are traditionally evaluated according to their performance ratio. Let  $A$  be an online graph-coloring algorithm. Then  $\chi_A(G)$  denotes the maximum number of colors  $A$  uses to color  $G$  over all online presentations of  $G$  respecting the given rules. An online algorithm is said to guarantee a performance ratio of  $\rho(G)$  if, for every graph  $G$ ,  $\chi_A(G) \leq \rho(G)\chi(G)$ . If there exists an instance for which  $\chi_A(G) = \rho(G)\chi(G)$ , we say that this bound is tight. For simplicity, the performance ratio is sometimes noted  $\rho$ .*

Some authors [KPT95] use an alternative way to characterize the performance of an online algorithm: a class of graphs  $\mathcal{G}$  is said to be  $\chi$ -bounded if the performance ratio for  $\mathcal{G}$  only depends on  $\chi$ . It means that there exists a function  $f$  such that for all  $G \in \mathcal{G}$ ,  $\chi_A(G) \leq f(\chi(G))$ .

**Definition 1.33** (Exact and optimal online algorithms). *An online algorithm is called exact (or solves the problem exactly) if it computes the optimal offline solution for any online instance (it has a performance ratio of 1). It is called optimal if its performance ratio cannot be improved by any other online algorithm.*

The performance ratio is sometimes called the *competitiveness ratio*. An algorithm which guarantees a performance ratio of  $\rho$  is then said to be  $\rho$ -competitive.

In this dissertation, unless otherwise stated, we consider an online model consisting of the following sets of rules:

- The adversary presents the vertices one by one, along with the arcs or edges connecting them to the vertices already presented. The adversary sees the decision taken by the algorithm and may adapt the instance it presents to these decisions.
- The algorithm must assign a color to a vertex as soon as it is presented and may not change its decision in the future.

In some cases, we will specify further rules to define a special case of the problem. In particular, we may specify that the adversary must present an instance in a given order.

A very common algorithm for coloring graphs is the greedy algorithm *First-Fit*, denoted by  $FF$ .

**Definition 1.34** (First-Fit). *First-Fit considers the vertices one after the other and puts each one in the first possible color-class.*

First-Fit is very popular for its simplicity and since, for some classes of graphs, it is easy to find an ordering which makes First-Fit exact. The class of graphs without induced  $P_4$ , called *cographs*, is known to be characterized by the fact that First-Fit will find an optimal coloring no matter the order in which it takes the vertices [Chv84]. The only minimal configuration for which First-Fit may find a non-optimal coloring is the graph  $P_4 = (\{a; b; c; d\}, \{(a, b); (b, c); (c, d)\})$ . More precisely, if the adversary presents the vertices in the order  $(a, d, b, c)$ , then, First-Fit uses three colors while two are sufficient.

**Definition 1.35** (perfect ordering). *Perfect orderings are orderings on the vertices of a graph  $G$  such that First-Fit solves Min Coloring on  $G$  and all its induced subgraphs exactly.*

In an online framework, First-Fit is a very natural algorithm, but its behavior depends on the order of presentation of the vertices. While it is exact for cographs, independently of the order of presentation of the vertices, it can be very bad for other classes of graphs. As will be shown in Chapter 3, it is simple to see that  $P_4$  is a permutation graph. Thus, permutation graphs do not have the nice property that any order is perfect.

**Remark 1.36.** *First-Fit can trivially be adapted to partitioning a graph into cliques by considering that the color-classes mentioned in Definition 1.34 are cliques. The algorithm is then denoted  $FF_k$ .*

**Remark 1.37.** *Similarly, First-Fit can be adapted for the problem of bounded coloring (respectively bounded clique covering): it puts a new vertex in the first possible color-class which contains less than  $b$  vertices. The algorithm is then denoted  $FF_b$  (respectively  $FF_{kb}$ ).*



# Chapter 2

## General Remarks and Co-interval Graphs

As mentioned in the definitions on graph theory (Section 1.1), coloring a graph is equivalent to covering its complement with cliques. Covering interval graphs with cliques has industrial applications (see for example [FJQS04], [Gol04] or [Jos06]), two of which will be presented here. Furthermore, it is very easy to represent an interval graph by drawing the corresponding intervals along an axis. For these two reasons, this chapter will approach the problem of online coloring a co-interval graph from its complementary side: online covering an interval graph with cliques.

**Related works.** The challenges of covering interval graphs with cliques have been the subject of many recent studies, in addition to the ones cited above. In [GJQ07], Gijswijt, Jost and Queyranne have studied clique partitioning of interval graphs with submodular costs on the cliques. Chan and Zarrabi-Zadeh have studied online partitioning into cliques in a generalization of interval graphs [CZZ07]. Zarrabi-Zadeh has also given interesting results in the particular case of unit-interval graphs in [ZZ07]. Jaromczyk, Pezarski and Slusarek have studied online covering of interval graphs with cliques in [JPS03] and have pointed out a performance ratio of 2 if there is no limit on the sizes of the intervals nor on the sizes of the cliques and if the intervals may be presented in arbitrary order.

**Main results.** First, we show that, for any class of graphs, if  $A$  is an online graph covering algorithm with performance ratio  $\rho$  and  $A_b$  is the simple transformation of  $A$  which cuts the color-classes given by  $A$  in chunks of sizes at most  $b$ , then the performance ratio  $\rho_{A_b}$  of  $A_b$  is at most  $(1 + \rho) - \frac{1}{\min\{\xi_b, b/\rho\}}$ , where  $\xi_b$  is the minimum number of color-classes of size at most  $b$  needed

to cover a given graph  $G$ . In particular, this holds for coloring and clique covering. Then, we generalize a result giving an upper bound of  $\frac{b}{2} + \frac{1}{2}$  on the performance ratio of  $FF_b$  on general graphs.

Furthermore, we show that if an interval graph is presented online in increasing order of the intervals left ends, it can be covered with cliques of bounded size  $b$  with a performance ratio of  $2 - \frac{1}{b}$  and that this result is optimal. Finally, if an interval graph is presented online in a random order, we give a nearly tight bound for the performance ratio  $\rho_{FF_{kb}}$  of  $FF_{kb}$ :  $3 - \frac{3}{\theta_b} \leq \rho_{FF_{kb}} \leq 3 - \frac{1}{\theta_b}$ , thus showing that the performance ratio  $\rho$  of this problem is  $2 \leq \rho \leq 3 - \frac{1}{\theta_b}$ .

## 2.1 Introduction

We consider various models of online clique covering of interval graphs, where the size of the cliques is upper-bounded by a given integer  $b$ . These models are motivated by two industrial applications which are described below.

The first application is the following: consider a large biology research center. Scientists use many different objects that need to be properly cleaned between two different uses, in order to avoid any infection of one organism by an other. To be cleaned, the objects are put in a big recipient, where they are sprayed with disinfectant and shortly heated up to kill bacterias. The scientists bring each object  $o_i$  at unpredictable time  $l_i$  to the cleaning department and specify a time  $r_i$  when they need the object back. Activating the cleaning machine costs energy and disinfectant and, therefore, the accountant of the research center wants to minimize the number of times the machine is activated.

This problem can be modeled this way: Let  $G$  be the interval graph with vertices  $o_1, o_2, \dots, o_n$  and connect two vertices  $o_i$  and  $o_j$  whenever the intersection of the intervals  $[l_i, r_i]$  and  $[l_j, r_j]$  is non-empty. By the Helly property [Gol04],  $\{o_{i_1}, o_{i_2}, \dots, o_{i_k}\}$  is a clique of  $G$ , then the intervals  $\{[l_{i_j}, r_{i_j}] : j = 1, 2, \dots, k\}$  will have a common point of intersection. Thus, a solution to the minimization problem will be obtained by solving Min Clique Covering on  $G$ . Since the objects are delivered in an unpredictable manner, this problem is an online problem.

The second application is described in [Gol04]. We rewrite it here for the sake of completeness.

Suppose  $c_1, c_2, \dots, c_n$  are chemical compounds which must be refrigerated under closely monitored conditions. If compound  $c_i$  must be kept at a *constant* temperature between  $l_i$  and  $r_i$  degrees Celsius, how many refrigerators will be needed to store all the compounds?



Let  $G$  be the interval graph with vertices  $c_1, c_2, \dots, c_n$  and connect two vertices whenever the temperature intervals of their corresponding compounds intersect. Again, by the Helly property, if  $\{c_{i_1}, c_{i_2}, \dots, c_{i_k}\}$  is a clique of  $G$ , then the intervals  $\{[l_{i_j}, r_{i_j}] : j = 1, 2, \dots, k\}$  will have a common point of intersection  $t$ . A refrigerator set at a temperature of  $t$  will be suitable for storing all of them. Thus also, a solution to the minimization problem will be obtained by finding a minimum clique cover of  $G$ .

There is an important difference between the two applications: in the Biology Research Center Problem, the intervals represent time-intervals. Therefore, they are presented in the increasing order of  $l_i$ . On the other hand, in the Chemical Compound Storage Problem, the compounds (and thus also the corresponding intervals) are presented in an arbitrary order.

In both cases, it is natural to consider that the size of a clique may have to be bounded. Indeed, a cleaning recipient may not be able to contain more than a given integer number  $b$  of objects. Similarly, a fridge may not be able to contain more than  $b$  compounds containers. This work will take a particular look at this.

This chapter is organized as follows: Section 2.2 contains some general remarks about graph covering with sets of bounded size. Section 2.3 deals with the Biology Research Center Problem and Section 2.4 considers the Chemical Compound Refrigeration Problem.

## 2.2 About Graph Covering with Sets of Bounded Size

### 2.2.1 A Simple Transformation

Consider an online algorithm  $A$  such that, if  $G'$  is a subgraph of  $G$  induced by a given set of color-classes used by  $A$  on  $G$ , then applying  $A$  on  $G'$  will not change the coloring of the vertices of  $G'$ . Suppose that  $A$  guarantees a performance ratio  $\rho$  on some online graph covering problem  $P$ . Let  $P_b$  be the same problem with the additional constraint that the sizes of the color-classes may not exceed a given bound  $b$ . Let  $A_b$  be the simple transformation of  $A$  which cuts the color-classes given by  $A$  in chunks of sizes at most  $b$ , without re-grouping any chunks.

Let  $\xi$  (respectively  $\xi_b$ ) be the minimum number of color-classes (respectively of bounded size  $b$ ) needed to cover a graph  $G$ . We can now formulate Theorem 2.1:

**Theorem 2.1.** *The competitive ratio  $\rho_{A_b}$  of  $A_b$  on  $P_b$  is bounded above by:*

$$\rho_{A_b} \leq (1 + \rho) - \frac{1}{l}$$

where  $l = \min \left\{ \xi_b, \frac{b}{\rho} \right\}$ .

*Proof.* Let  $\lambda_{A_b}(G)$  be the number of color-classes returned by  $A$  on the graph  $G$ . We call *saturated* the color-classes which contain exactly  $b$  vertices and *unsaturated* the color-classes which contain strictly less than  $b$  vertices. We denote by  $S$  the set of saturated color-classes and by  $\bar{S}$  the set of unsaturated color-classes. Furthermore, we define  $N_S = |S|$  (the number of saturated color-classes) and  $N_{\bar{S}} = |\bar{S}|$  (the number of unsaturated color-classes). Of course, the following equality always holds:

$$\lambda_{A_b}(G) = N_S + N_{\bar{S}} \quad (2.1)$$

Let  $G_{\bar{S}}$  be the subgraph induced by the color-classes in  $\bar{S}$ . Since the color-classes are unsaturated,  $A_b$  behaves just like  $A$  on  $G_{\bar{S}}$ . Therefore,

$$N_{\bar{S}} = \lambda_A(G_{\bar{S}}) \leq \rho \xi(G_{\bar{S}}) \leq \rho \xi(G) \leq \rho \xi_b(G) \quad (2.2)$$

Since color-classes in  $N_{\bar{S}}$  are non-empty, we have:

$$n \geq bN_S + N_{\bar{S}} \quad (2.3)$$

$$\xi_b \geq \left\lceil \frac{n}{b} \right\rceil \geq \left\lceil \frac{bN_S + N_{\bar{S}}}{b} \right\rceil = N_S + \left\lceil \frac{N_{\bar{S}}}{b} \right\rceil \quad (2.4)$$

$$N_S \leq \xi_b - \left\lceil \frac{N_{\bar{S}}}{b} \right\rceil \quad (2.5)$$

Combining (2.1), (2.2) and (2.5), we deduce:

$$\lambda_{A_b}(G) \leq \xi_b - \left\lceil \frac{N_{\bar{S}}}{b} \right\rceil + N_{\bar{S}} \quad (2.6)$$

$$\leq \xi_b - \left\lceil \frac{\rho \xi_b}{b} \right\rceil + \rho \xi_b \quad (2.7)$$

$$\leq \xi_b(1 + \rho) - \left\lceil \frac{\rho \xi_b}{b} \right\rceil \quad (2.8)$$

Two cases are possible:

1. If  $\rho\xi_b \leq m$ , then  $\lceil \frac{\rho\xi_b}{b} \rceil = 1$  and

$$\lambda_{A_b}(G) \leq \xi_b(1 + \rho) - 1 \quad (2.9)$$

and

$$\rho_{A_b} \leq (1 + \rho) - \frac{1}{\xi_b} \quad (2.10)$$

2. If  $\rho\xi_b > m$ , then

$$\rho_{A_b} \leq (1 + \rho) - \frac{\rho}{b} \quad (2.11)$$

□

**Corollary 2.2.** *If First-Fit guarantees a competitive ratio  $\rho$  on the problem of covering a graph of a given class  $\mathcal{G}$  with cliques, then  $FF_b$  guarantees a performance ratio of  $\rho_{FF_b}$  on the problem of covering a graph of  $\mathcal{G}$  with cliques of bounded size  $b$ , where  $\rho_{FF_b} \leq (1 + \rho) - \frac{1}{l}$  and  $l = \min \left\{ \theta_b, \frac{b}{\rho} \right\}$ .*

### 2.2.2 On the Bounded First-Fit Algorithm

Bouille and Plumettaz have studied the problem of coloring permutation graphs online with color-classes of bounded size  $b$  and have shown that  $FF_b$  guarantees a performance ratio of at most  $\frac{b}{2} + \frac{1}{2}$ . Although not mentioned in their work, their proof shows that this bound remains true for general graphs. For the sake of completeness, we provide a proof of this result here.

**Theorem 2.3** ([BP06]).  *$FF_b$  guarantees a performance ratio  $\rho_{FF_b} \leq \frac{b}{2} + \frac{1}{2}$  on the Online Bounded Graph Coloring problem with bound  $b \geq 2$ .*

*Proof.* Suppose that we have a graph  $G = (V, E)$ . Let us estimate  $\lambda$ , the maximum number of colors that  $FF_b$  may need to use on  $G$ . We consider  $\lambda_1$ , the number of color-classes of size 1, and  $\lambda_{\geq 2}$ , the number of color-classes of size at least 2.

The vertices in color-classes of size 1 must all be adjacent, otherwise  $FF_b$  would put them in the same color-class. Thus,  $G$  contains a clique of size at least  $\lambda_1$  and thus  $\chi_b \geq \lambda_1$ . The remaining vertices are put in color-classes of size at least two. Since  $|V| \leq \chi_b b$ , there number of remaining vertices is at most  $\chi_b b - \lambda_1$ . Thus  $\lambda_{\geq 2} \leq (\chi_b b - \lambda_1)/2$  and therefore,

$$\lambda \leq \lambda_1 + \frac{\chi_b b - \lambda_1}{2} = \frac{\lambda_1}{2} + \frac{\chi_b b}{2} \quad (2.12)$$

Therefore, the performance ratio  $\rho_{FF_b}$  is

$$\rho_{FF_b} \leq \frac{\lambda}{\chi_b} \quad (2.13)$$

$$\rho_{FF_b} \leq \frac{\lambda_1}{2\chi_b} + \frac{b}{2} \leq \frac{1}{2} + \frac{b}{2} \quad (2.14)$$

This ends the proof of Theorem 2.3 □

Since coloring a graph is equivalent to covering its complement with cliques, it is easy to derive the following corollary:

**Corollary 2.4.**  *$FF_{kb}$  guarantees a performance ratio  $\rho_{FF_{kb}} \leq \frac{b}{2} + \frac{1}{2}$  on Min Bounded Clique Covering with  $b \geq 2$ .*

When dealing with problems of covering graphs with color-classes of bounded size, the case  $b = 1$  is of course trivial and therefore not interesting. If  $b = 2$ , covering a graph with a minimum number of cliques is equivalent to finding a maximum matching in this graph. In this case, we can state the following lemma:

**Lemma 2.5.** *Online Min Bounded Clique Covering with  $b = 2$  can be solved with a performance ratio of  $\frac{3}{2}$  by  $FF_{kb}$  and this bound is optimal.*

*Proof.* The upper bound is given by Corollary 2.4 with  $b = 2$ . This bound can be reached with the following instance: present two vertices  $v_1$  and  $v_2$  and the edge  $v_1v_2$ . If the algorithm puts  $v_1$  and  $v_2$  in the same color-class, present  $v_3$  and the edge  $v_3v_1$  and  $v_4$  and the edge  $v_2v_4$ ; otherwise, present  $v_3$  and the edge  $v_2v_3$  and  $v_4$  and the edge  $v_3v_4$ . □

**Remark 2.6.** *The instance presented to show the lower bound of Lemma 2.5 can be repeated indefinitely, so it holds for any  $\theta_b \geq 2$ . The upper bound is also valid for any  $\theta_b$ . Thus, so is Lemma 2.5.*

Again, since coloring a graph is equivalent to covering its complement with cliques, we have:

**Corollary 2.7.** *Online Min Bounded Coloring with  $b = 2$  can be solved with a performance ratio of  $\frac{3}{2}$  by  $FF_b$  and this bound is optimal.*

Therefore, when dealing with bounded coverings of graphs, this work will concentrate on cases where  $b \geq 3$ .

## 2.3 The Biology Research Center Problem

Min Clique Covering on interval graphs has been studied before in its offline version and it is known that it can be solved polynomially [Sch03]. Recently, it has also been studied in the case where the cliques have a maximum size  $b$  [FJQS04]. In both cases, the intervals are covered with cliques

using the First-Fit algorithm after sorting the intervals in increasing order of their *right* ends. This allows us to claim that First-Fit also finds an exact solution in the online case, when the intervals are presented in increasing order of their right ends.

Online, however, this model is not very natural. On the contrary, the Biology Research Center Problem shows a very natural reason to study online Min Clique Covering on interval graphs when the intervals are presented in increasing order of their *left* ends.

For technical reasons, let us start by considering the case where the size of the cliques is not bounded.

**Lemma 2.8.** *If an interval graph  $G$  is presented online from in increasing order of the interval's left ends, then First-Fit gives an exact clique covering of  $G$ .*

*Proof.* Consider the time when interval  $I_i = [l_i, r_i]$  is presented. If First-Fit opens a new color-class  $K_p$ , then, there exists an interval  $I_j = [l_j, r_j]$  in  $K_{(p-1)}$  such that  $r_j < l_i$ . Because of the order of presentation of the intervals, no interval presented after  $I_i$  will overlap  $I_j$ , and thus no interval presented after  $I_i$  will be put in  $K_{(p-1)}$ .

Suppose that, each time a new color-class  $K_p$  is opened for some interval  $I_i$ , one interval of  $K_{(p-1)}$  which does not overlap  $I_i$  is marked in red. Suppose also that the last presented interval is marked in red. Then, at the end of the algorithm, the intervals marked in red form a stable set and the size of this stable set is equal to the number of color-class opened by First-Fit. Therefore, the number of cliques is optimal.  $\square$

It is now possible to state the following theorem:

**Theorem 2.9.** *If the intervals are presented in increasing order of their left ends, the competitive ratio  $\rho_b$  of online Min Bounded Clique Covering on an interval graph is  $2 - \frac{1}{b}$ .*

*Proof.* Using Corollary 2.2 and Lemma 2.8, it is obvious that

$$\rho_b \leq 2 - \frac{1}{b} \tag{2.15}$$

Note that in the case where  $\theta_b \leq b$ , we have  $\rho \leq 2 - 1/\theta_b \leq 2 - 1/b$ . We must now prove that  $\rho_b \geq 2 - 1/b$  for any online algorithm. The instance presented by Adversary 1 reaches that goal.

Since the cliques  $K_i$  are of size  $b$ , any algorithm  $A$  will use at least two colors for each  $i \in [1..(b-1)]$ : one for  $K_i$  and one for  $I_i$  if  $K_i$  is of type  $\beta$ , two

**Adversary 1**

---

**Input:** An online algorithm  $A$  for covering an interval graph presented in increasing order of the left ends of the intervals with cliques of bounded size  $b$ .

**Output:** An instance on which  $A$  will achieve a performance ratio  $\rho_b \geq 2 - 1/b$ .

- 1: Let  $[l_1, r_1]$  be a large interval.
  - 2: Repeat
  - 3: **for**  $i \in [1..(b-1)]$  **do**
  - 4:   Present a clique  $K_i$  of  $b$  intervals  $[l_i, r_i]$ .
  - 5:    $\varepsilon \leftarrow \frac{r_i - l_i}{100}$
  - 6:   **if**  $A$  has put all intervals of  $K_i$  in the same color-class **then** */\*We say that  $K_i$  is of type  $\beta$ \*/*
  - 7:     Present a small interval  $I_i = [l_i, (l_i + 10\varepsilon)]$
  - 8:      $l_{(i+1)} \leftarrow l_i + 11\varepsilon$
  - 9:      $r_{(i+1)} \leftarrow (r_i + l_i)/2$
  - 10:   **else** */\*We say that  $K_i$  is of type  $\alpha$ \*/*
  - 11:      $l_{(i+1)} \leftarrow (r_i + l_i)/2 + \varepsilon$
  - 12:   **end if**
  - 13: **end for**
  - 14: Present one interval  $[l_b, r_b]$ .
- 

or more for  $K_i$  if  $K_i$  is of type  $\alpha$  (in this case,  $I_i$  is not presented afterwards). In addition, since the interval presented at Step 14 overlaps only intervals belonging to  $K_i$ s of type  $\beta$ , it will be put into a new clique. Thus, the number  $\lambda_A$  of color-classes used by  $A$  on this instance will be:

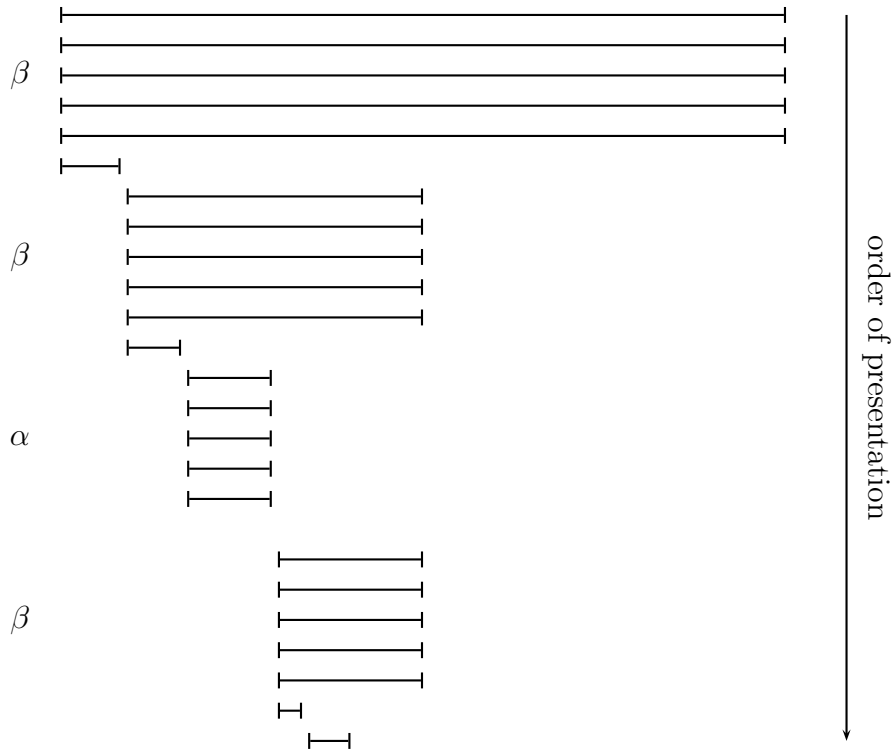
$$\lambda_A = 2(b-1) + 1 = 2b - 1 \quad (2.16)$$

Let us suggest a different solution. For each  $K_i$  of type  $\beta$ , put  $(b-1)$  intervals of  $K_i$  plus the interval  $I_i$  into one color-class. For each  $K_i$  of type  $\alpha$ , put  $K_i$  into one color-class. There remains exactly one interval for each clique  $K_i$  of type  $\beta$  (at most  $(b-1)$ ), plus the interval presented at Step 14. They can all be put in one color-class. This solution shows that the instance can be covered with exactly  $b$  color-classes of size at most  $b$ . Therefore,

$$\rho_b \geq \frac{2b-1}{b} = 2 - \frac{1}{b} \quad (2.17)$$

Together with (2.15), (2.17) ends the proof of Theorem 2.9.  $\square$

**Remark 2.10.** *If  $b$  is large, Min Bounded Clique Covering tends to be the same as Min Clique Covering. However, the performance ratio given in Theorem 2.9 then tends to 2, and not to 1 as one might expect. One must see*



**Figure 2.1:** *Illustration of an instance presented by Adversary 1 for  $b = 5$ , supposing that  $A$  puts the intervals of the cliques  $K_i$ ,  $i \in [1..4]$  as indicated on the left:  $K_i$  is of type  $\alpha$  if  $A$  has put its intervals in two or more different color-classes; it is of type  $\beta$  if  $A$  has put all its intervals in the same color-class.*

that, if  $b$  becomes large, the adversary will chose a larger graph (Figure 2.1) to achieve the lower bound of  $2 - 1/b$ . If the size of the graph is bounded, and  $b$  is large compared to that size,  $2 - 1/b$  may not be achieved as a lower bound, however, it remains an upper bound.

## 2.4 Chemical Compounds Refrigeration

This section studies the Chemical Compounds Refrigeration Problem: Intervals are presented in arbitrary order and the complete set of intervals must be covered online with cliques. Remember that this problem differs from the Biology Research Center Problem, where the intervals are presented in a specific order.

### 2.4.1 Analysis of First-Fit

It has been proved in [JPS03] that the performance ratio of online Min Clique Covering on interval graphs is 2 if the intervals are presented in arbitrary order. and that this bound is achieved by  $FF_k$ . Using this result and Corollary 2.2, we have that the performance ratio  $\rho_{FF_{kb}}$  of  $FF_{kb}$  on the online Min Bounded Clique Covering problem on an interval graph is:

$$\rho_{FF_{kb}} \leq 3 - \frac{1}{l} \leq 3 - \frac{1}{\theta_b} \quad (2.18)$$

where  $l = \min\{\theta_b, \frac{b}{2}\}$  again.

**Remark 2.11.** *This upper bound enhances Kouakou's result [Kou07] slightly. He showed an upper bound of 3.*

Let us now prove that this bound is nearly tight.

**Proposition 2.12.** *The performance ratio  $\rho_{FF_{kb}}$  of  $FF_{kb}$  on the online Min Bounded Clique Covering problem on an interval graph is:*

$$3 - \frac{3}{\theta_b} \leq \rho_{FF_{kb}} \leq 3 - \frac{1}{\theta_b}$$

*Proof.* Consider the instance presented by Adversary 2. First-Fit will use three color-classes for each  $i$ : one for the clique  $K_i$ , one for both intervals  $I_{(i,1)}$  and  $I_{(i,2)}$  and one for the interval  $I_{(i,3)}$ . Thus, the total number  $\lambda_b$  of color-classes used by  $FF_{kb}$  will be :

$$\lambda_b = 3k \quad (2.19)$$

Consider this alternative solution: for each  $i$ , put  $I_{(i,1)}$ ,  $I_{(i,2)}$  and  $(b-2)$  intervals of  $K_i$  in one color-class. There remains at most  $b$  intervals which can also be covered with one color-class. This solution uses  $k+1$  color-classes. Thus,  $\theta_b \leq k+1$ . This result, together with (2.19) implies that

$$\rho_{FF_{kb}} \geq \frac{3k}{k+1} \geq 3 - \frac{3}{\theta_b} \quad (2.20)$$

Together with (2.18), this concludes the proof of Proposition 2.12.  $\square$



---

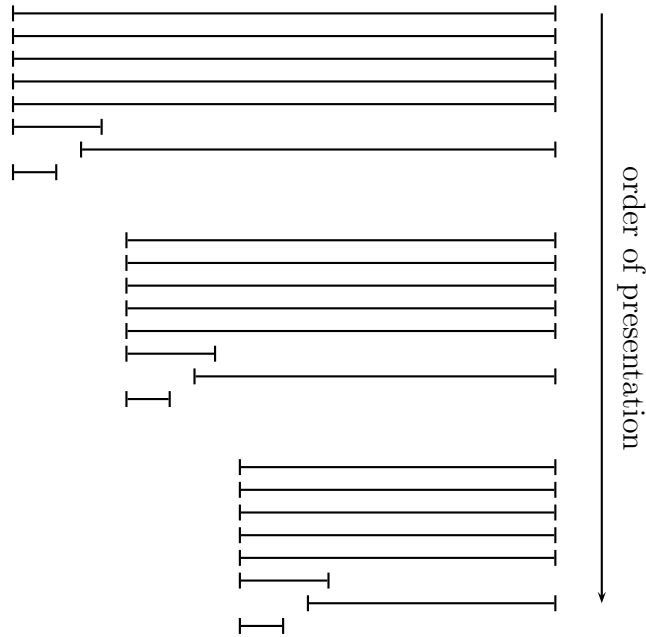
**Adversary 2**

---

**Input:** The algorithm  $FF_{kb}$ .

**Output:** An instance on which  $FF_{kb}$  will have a performance ratio of at least  $3 - \frac{3}{\theta_b}$ .

- 1: Let  $[l_i, r_i]$  be a large interval.
  - 2:  $k \leftarrow \lfloor \frac{b}{3} \rfloor$ .
  - 3: **for**  $i \in [1..k]$  **do**
  - 4:   Present a clique  $K_i$  of  $b$  intervals  $[l_i, r_i]$ .
  - 5:    $\varepsilon \leftarrow \frac{r_i - l_i}{100}$
  - 6:   Present  $I_{(i,1)} = [l_i, l_i + 3\varepsilon]$
  - 7:   Present  $I_{(i,2)} = [l_i + 2\varepsilon, r_i]$
  - 8:   Present  $I_{(i,3)} = [l_i, l_i + \varepsilon]$
  - 9:    $l_{i+1} \leftarrow l_i + 4\varepsilon$
  - 10:    $r_{i+1} \leftarrow r_i$
  - 11: **end for**
- 



**Figure 2.2:** *Illustration of the instance presented by Adversary 2.*

### 2.4.2 A Lower Bound For All Algorithms

The following proposition is given by Kouakou. We translate it here for the sake of completeness.

**Proposition 2.13** ([Kou07]). *For all  $\varepsilon \in \mathbb{R}_+$  and  $b \in \mathbb{N}, b \geq 3$ , no algorithm can guarantee a performance ratio  $\rho_b = (2 - \varepsilon)$  on online Min Bounded Clique Covering on interval graphs.*

*Proof.* Consider the instance presented by Adversary 3 with the following notation: for each  $i$ , if  $A_b$  puts  $I_{(i,1)}$  and  $I_{(i,2)}$  in the same color-class, we say that  $I_{(i,1)}$  and  $I_{(i,2)}$  form a clique of type  $\beta$ . Otherwise, we say that  $I_{(i,1)}$  and  $I_{(i,2)}$  form a clique of type  $\alpha$ .

Let  $\lambda_b$  be the number of color-classes used by  $A_b$  on the instance presented by Adversary 3. At the end of the execution, let  $N_\alpha$  (respectively  $N_\beta$ ) be the number of cliques of type  $\alpha$  (respectively  $\beta$ ). We have that  $N_\alpha + N_\beta = k$ . The cliques of type  $\beta$  are put in  $N_\beta$  color-classes. The cliques of type  $\alpha$  are put in  $2N_\alpha$  color-classes. For each  $i$ ,  $I_{(i,3)}$  is only presented if  $I_{(i,1)}$  and  $I_{(i,2)}$  form a clique of type  $\beta$ .  $I_{(i,3)}$  does not overlap  $I_{(i,2)}$ . Therefore,  $A_b$  will need to open a new color-class for each  $I_{(i,3)}$ . There are  $N_\beta$  intervals  $I_{(i,3)}$ . Similarly, the interval presented at Step 32 must also be put in a new color-class. Thus,

$$\lambda_b = 2N_\alpha + N_\beta + N_\beta + 1 = 2k + 1 \quad (2.21)$$

Consider this alternative solution: Put all cliques of type  $\alpha$  in one color-class. For each  $i$  such that  $I_{(i,1)}$  and  $I_{(i,2)}$  form a clique of type  $\beta$ : put  $I_{(i,1)}$  and  $I_{(i,3)}$  in one color-class  $K_i$  and, if  $i \neq k$ , add  $I_{(i,2)}$  to  $K_j$ , where  $j$  is the smallest  $p$  such that  $p > i$  and  $I_{(p,1)}$  and  $I_{(p,2)}$  form a clique of type  $\beta$ . This solution uses  $N_\alpha + N_\beta + 1 = k + 1$  cliques. Thus,

$$\rho_b \geq \frac{2k + 1}{k + 1} > 2 - \varepsilon \quad (2.22)$$

This ends the proof of Proposition 2.13. □

**Remark 2.14.** *Proposition 2.13 and Proposition 2.12 show that the performance ratio  $\rho_b$  of the problem of online covering an interval graph with cliques of bounded size is*

$$2 \leq \rho_b < 3 - \frac{1}{\theta_b} \quad (2.23)$$

---

**Adversary 3**

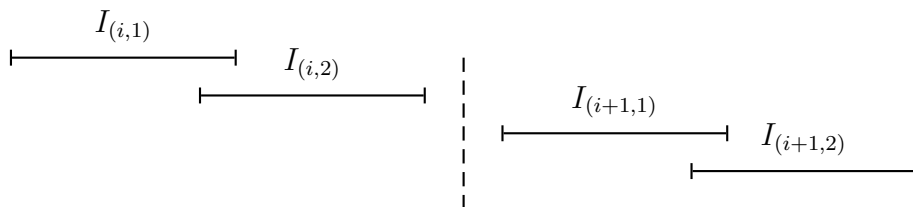
---

**Input:** An online algorithm  $A_b$  for covering an interval graph with cliques and a real positive number  $\varepsilon$ .

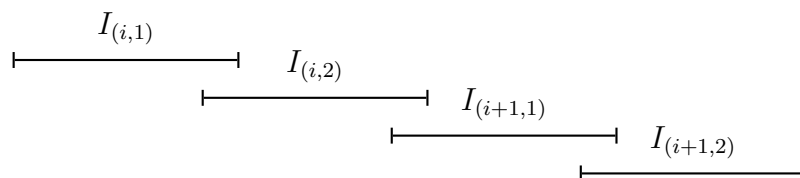
**Output:** An instance on which  $A_b$  will have a performance ratio strictly greater than  $(2 - \varepsilon)$ .

- 1: Let  $[l_{(1,1)}, r_{(1,1)}]$  be an interval of any length.
- 2: Let  $k \in \mathbb{N}$  be such that  $k > (\frac{1}{\varepsilon} - 1)$
- 3:  $i \leftarrow 1$  /\*A counter\*/
- 4: **loop**
- 5:    $\delta \leftarrow r_{(i,1)} - l_{(i,1)}$  /\*The length of an interval\*/
- 6:    $\mu \leftarrow \frac{\delta}{100}$  /\*A small number used to make the intervals overlap or not.\*/
- 7:    $l_{(i,2)} \leftarrow r_{(i,1)} - 2\mu$
- 8:    $r_{(i,2)} \leftarrow l_{(i,2)} + \delta$
- 9:   Present intervals  $I_{(i,1)} = [l_{(i,1)}, r_{(i,1)}]$  and  $I_{(i,2)} = [l_{(i,2)}, r_{(i,2)}]$
- 10:   **while**  $A_b$  covers  $I_{(i,1)}$  and  $I_{(i,2)}$  with two different clique **do**
- 11:      $i \leftarrow i + 1$ ; **if**  $i > k$  **then goto** 30
- 12:      $l_{(i,1)} \leftarrow r_{(i-1,2)} + 2\mu$
- 13:      $r_{(i,1)} \leftarrow l_{(i,1)} + \delta$
- 14:      $l_{(i,2)} \leftarrow r_{(i,1)} - 2\mu$
- 15:      $r_{(i,2)} \leftarrow l_{(i,2)} + \delta$
- 16:     Present intervals  $I_{(i,1)} = [l_{(i,1)}, r_{(i,1)}]$  and  $I_{(i,2)} = [l_{(i,2)}, r_{(i,2)}]$
- 17:   **end while** /\*See Figure 2.3 for an illustration.\*/
- 18:   **while**  $A_b$  covers  $I_{(i,1)}$  and  $I_{(i,2)}$  with the same cliques **do**
- 19:      $i \leftarrow i + 1$ ; **if**  $i > k$  **then goto** 30
- 20:      $l_{(i,1)} \leftarrow r_{(i-1,2)} - 2\mu$
- 21:      $r_{(i,1)} \leftarrow l_{(i,1)} + \delta$
- 22:      $l_{(i,2)} \leftarrow r_{(i,1)} - 2\mu$
- 23:      $r_{(i,2)} \leftarrow l_{(i,2)} + \delta$
- 24:     Present intervals  $I_{(i,1)} = [l_{(i,1)}, r_{(i,1)}]$  and  $I_{(i,2)} = [l_{(i,2)}, r_{(i,2)}]$
- 25:   **end while** /\*See Figure 2.4 for an illustration.\*/
- 26:    $i \leftarrow i + 1$ ; **if**  $i > k$  **then goto** 30
- 27:    $l_{(i,1)} \leftarrow r_{(i-2,1)} + \mu$
- 28:    $r_{(i,1)} \leftarrow l_{(i,1)} + \frac{\delta}{2k}$
- 29: **end loop**
- 30: **for each**  $i$  such that  $I_{(i,1)}$  and  $I_{(i,2)}$  form a clique of type  $\beta$  **do**
- 31:   Present an interval  $I_{(i,3)}$  of length  $\delta$  centered on  $l_{(i,1)}$
- 32:   **if**  $i = k$  **then** Present an interval of length  $\delta$  centered on  $r_{(i,1)}$
- 33: **end for**

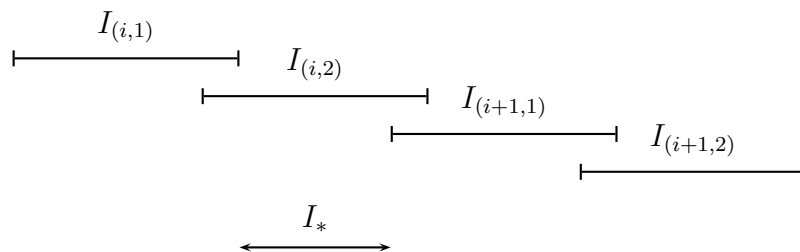
---



**Figure 2.3:** *Illustration of Step 16 of Adversary 3: Here,  $I_{(i,1)}$  and  $I_{(i,2)}$  form a clique of type  $\alpha$ . Therefore,  $I_{(i+1,1)}$  does not overlap  $I_{(i,2)}$ .*



**Figure 2.4:** *Illustration of Step 24 of Adversary 3. Here,  $I_{(i,1)}$  and  $I_{(i,2)}$  form a clique of type  $\beta$ . Therefore,  $I_{(i+1,1)}$  overlaps  $I_{(i,2)}$ .*



**Figure 2.5:** *Illustration of the steps 27 and 28 of Adversary 3: They make sure that any upcoming interval will be presented within interval  $I_*$ .*

## 2.5 Conclusion

Table 2.1 summarizes the known results on online covering interval graphs with cliques, or online coloring co-interval graphs. The upper bound on Min Bounded Coloring on a co-interval graph presented in arbitrary order is achieved by  $FF_b$  and this bound is tight.

		Presentation model		
		Increasing order of		Random
		$l_i$	$r_i$	
Coloring	Unbounded	1	1	2
Case	Bounded	$2 - \frac{1}{b}$	1	$2 \leq \rho_b \leq 3 - \frac{1}{\theta_b}$

**Table 2.1:** *Performance ratio of the problem of online coloring co-interval graphs*



# Chapter 3

## Comparability and Permutation Graphs

In this chapter, based on [DLB07], we consider online coloring of two classes of graphs, namely comparability graphs (graph representation of partially ordered sets)<sup>1</sup> and permutation graphs (graphs of inversions in permutations).

Note that permutation graphs are a subclass of comparability graphs [Gol04]. Therefore, some results that are valid for one class are also valid for the other class.

Furthermore, we also study a generalization of the coloring problem, called *cocoloring*. Both coloring and cocoloring in the considered classes of graphs have applications in industrial contexts [DEdW07, DSKLZ06, SM04, SM07].

From the point of view of theory of ordered sets, this work can be seen as online partitioning of posets into antichains (or into chains and antichains).

Section 3.1 gives some definitions and notations. Online coloring is studied in Section 3.2 and online cocoloring is studied in Section 3.3.

**Related works.** Besides the papers cited above, most of which are studies in offline cases, several online problems have been studied on posets (See for instance [BR97]). In particular, online partitioning posets into chains has been widely studied [Fel97, Kie81, KPT94]. From a graph theoretical point of view, it consists in covering comparability graphs with cliques online. To our knowledge, online coloring of comparability graphs has been essentially studied in the particular case of permutation graphs [NP00, DSKLZ06].

---

<sup>1</sup>All notions will be formally defined in section 3.1

**Main results.** For online Min Coloring, we study the First-Fit algorithm in several different models and give a tight analysis. Among others, we show that even in some restricted models, First-Fit has a performance ratio that is at least  $O(\sqrt{n})$ . We then devise an algorithm with a performance ratio of  $\frac{\chi+1}{2}$  in the general model. Moreover, we analyze the behaviour of First-Fit on online Min Bounded Coloring and show that it has a performance ratio of  $\frac{b}{2} + \frac{1}{2}$ . For the online cocoloring problem, we show that the performance ratio is  $\frac{n}{4} + \frac{1}{2}$  and we give better bounds in some more restricted cocoloring problems.

### 3.1 Definitions and Notations

**Definition 3.1** (Orientation and transitive orientation). *Given an undirected graph  $G = (V, E)$ ,  $G^d = (V, E^d)$  is called an orientation of  $G$  if*

$$v_1v_2 \in E \Leftrightarrow (v_1, v_2) \in E^d \text{ xor } (v_2, v_1) \in E^d$$

Moreover, if

$$(v_1, v_2) \in E^d \text{ and } (v_2, v_3) \in E^d \Rightarrow (v_1, v_3) \in E^d$$

then this orientation is transitive.

Cocoloring was introduced in [LS77].

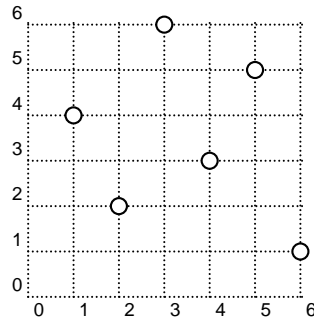
**Definition 3.2** (Cocoloring). *Cocoloring a graph  $G = (V, E)$  is partitioning  $V$  into cliques and stable sets. The cochromatic number of a graph  $G$ , denoted by  $z(G)$ , is the smallest number of such sets needed to cover all vertices. A  $k$ -cochromatic graph is a graph  $G$  such that  $z(G) = k$  and a  $k$ -cocolorable graph is a graph  $G$  such that  $z(G) \leq k$ .*

**Definition 3.3** (Comparability graph). *A comparability graph is an undirected graph which admits a transitive orientation.*

Every transitive orientation of a comparability graph can be seen as the graph representation of some partially ordered set (poset), where the vertices represent the elements of the set and there is an arc from  $a$  to  $b$  if  $a < b$  and no arc between  $a$  and  $b$  if these elements are not comparable.

Clearly, in comparability graphs, a clique of order  $k$  corresponds to a *chain* of length  $(k-1)$ . Moreover, a stable set in a comparability graph corresponds to an antichain in the related poset. Consequently, a coloring (resp. a clique covering) on a comparability graph is equivalent to a partitioning of a poset into antichains (resp. chains). Cocoloring a comparability graph is equivalent to partitioning a poset into chains and antichains.





**Figure 3.1:** A lattice representation of the permutation  $\pi = [4, 2, 6, 3, 5, 1]$ .

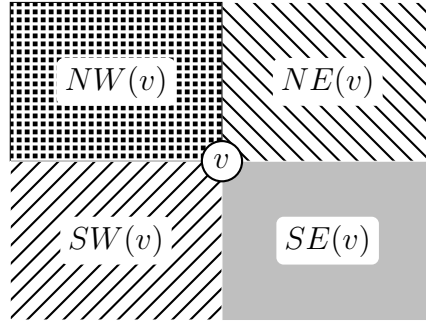
**Definition 3.4** (Permutation graph). A permutation graph is an undirected graph for which there exists a permutation such that every vertex of the graph corresponds to an element in the permutation and two vertices are adjacent if and only if the corresponding elements appear in reverse order in the permutation.

Note that permutation graphs are comparability graphs. More precisely, a permutation graph is a comparability graph, the complement of which is also a comparability graph (co-comparability graph) [Gol04]. Consequently, every hardness result stated for permutation graphs also holds for comparability and co-comparability graphs; conversely, every competitive analysis on comparability graphs also holds for permutation graphs. Coloring a permutation graph is equivalent to partitioning a permutation into increasing subsequences. Cocoloring a permutation graph is equivalent to partitioning a permutation into monotone subsequences; in some works, cocoloring a permutation is called *monotone partitioning* [DSKLZ06].

Both permutation graphs and comparability graphs are perfect graphs. Coloring such graphs offline can be done optimally in polynomial time [Gol04]. On the other hand, cocoloring permutation graphs (and consequently comparability graphs) is NP-hard [Wag84].

In our online model, the graph is presented together with a transitive orientation; consequently, this work can be seen as online partitioning of posets into antichains (or into chains and antichains).

**Definition 3.5** (Lattice representation). Throughout this work, we will use a very common representation for permutation graphs, called the lattice representation. The permutation is represented in a two-dimensional plane: the  $y$ -axis represents the values of the elements of the permutation and the  $x$ -axis represents the position of these elements. A point  $(x, y)$  in the plane means that an element of value  $y$  is at position  $x$  in the permutation.



**Figure 3.2:** *The four cardinal regions defined for the vertex  $v$ .*

This means that, whenever a new point is presented, we are given its position relatively to points previously presented along each axis but not its absolute position in the permutation. It is equivalent to present a permutation graph  $G$  together with a transitive orientation and also a transitive orientation of  $\overline{G}$  or to present a sequence of numbers one has to decompose into monotone sequences. The related permutation can be computed when all elements are known. In the last section of this chapter, we mention the *discrete latticial model*, where the latticial representation is drawn in  $\{1, \dots, n\}^2$ ,  $n$  being the order of the graph.

Given the lattice representation, for each vertex  $v$ , we denote by  $v|_x$  and  $v|_y$  the coordinates of vertex  $v$  in the latticial representation of the graph. Furthermore, we define four regions of the plane, which are inspired from the four cardinal points:

**Definition 3.6** (Cardinal regions).

$$\begin{aligned} NW(v) &= \{(x, y) : x < v|_x \text{ and } y > v|_y\} \\ NE(v) &= \{(x, y) : x > v|_x \text{ and } y > v|_y\} \\ SW(v) &= \{(x, y) : x < v|_x \text{ and } y < v|_y\} \\ SE(v) &= \{(x, y) : x > v|_x \text{ and } y < v|_y\} \end{aligned}$$

It is immediate to see that

$$(v, w) \text{ is an arc} \Leftrightarrow w \in SE(v) \Leftrightarrow v \in NW(w) \quad (3.1)$$

We will also use this terminology to specify directions on the plane.

## 3.2 Coloring

In 3.2.1, we define a way to present a permutation graph in a given direction and characterize the directions making First-Fit exact. In 3.2.2, we propose an analysis of First-Fit for bipartite permutation graphs. In 3.2.3, we devise an algorithm which significantly improves the bound for online Min Coloring in comparability and permutation graphs. Finally, in 3.2.4, we analyze First-Fit on online Min Bounded Coloring of comparability and permutation graphs.

### 3.2.1 Preliminaries

Given the lattice representation of a permutation graph  $G$ , it is well known that it can be colored exactly if the vertices are presented from west to east, since First-Fit colors  $G$  exactly using this order [Gol04]. This order of presentation corresponds to presenting the permutation from left to right. Considering this, it is natural to wonder whether it is easy to color a permutation graph online if the vertices are presented in some other direction. It can be presented, for instance, from east to west, from south-west to north-east and so on. More precisely, given a fixed direction  $\vec{u} \in \mathbb{R}^2$ , the graph is said to be presented in the direction  $\vec{u}$  if

$$(v_1|_x, v_1|_y) \cdot \vec{u} < (v_2|_x, v_2|_y) \cdot \vec{u} \Rightarrow v_1 \text{ is presented before } v_2$$

If  $(v_1|_x, v_1|_y) \cdot \vec{u} = (v_2|_x, v_2|_y) \cdot \vec{u}$ , then  $v_1$  may be presented before or after  $v_2$ .

**Proposition 3.7.** *If a permutation graph  $G$  is presented from north-west to south-east or from south-east to north-west ( $\vec{u} = (x, y)$  such that  $x \cdot y \leq 0$  and  $(x, y) \neq (0, 0)$ ) in a latticial model, then First-Fit colors  $G$  exactly.*

*Proof.* This proof is inspired from Chvátal's proof [Chv84]. Let us suppose  $x \geq 0, y \leq 0$  and  $(x, y) \neq (0, 0)$ . Then, for all vertices  $v_1$  and  $v_2$  such that  $v_2 \in NW(v_1)$  and  $v_1 \neq v_2$ ,  $v_2$  is presented before  $v_1$ .

Let us suppose that the color  $k$  is attributed to  $v_k$ . Then, there exists a point  $v_{k-1}$  with color  $(k-1)$  such that  $v_{k-1}$  is presented before  $v_k$  and the related vertices in  $G$  are linked. Thus,  $v_{k-1} \in NW(v_k)$ .

By the same argument, we show that there exist  $v_i, 1 \leq i \leq k-1$ , where  $v_i$  is of color  $i$  and  $v_i \in NW(v_{i+1})$ .

Then  $\{v_i, i \in \{1, \dots, k\}\}$  constitute a clique of order  $k$

The proof is similar if  $x \leq 0, y \geq 0$  and  $(x, y) \neq (0, 0)$ . □

**Corollary 3.8.** *If a graph  $G$  is presented from west to east, from east to west, from south to north or from north to south on a latticial plane, then First-Fit finds the exact solution of Min Coloring on  $G$ .*

**Proposition 3.9.** *If a permutation graph  $G$  is presented from south-west to north-east or from north-east to south-west ( $x \cdot y > 0$ ) in a latticial model, then no algorithm can guarantee an optimal coloring for any arbitrary comparability graph  $G$ , even if  $G$  is a  $P_4$ .*

*Proof.* Let us suppose  $x = 1$  and  $y = 1$ . The proof is similar for other cases. Let us present  $v_1$  at  $(1, 3)$  and  $v_2$  at  $(4, 1)$ .  $v_1$  is given color 1 and  $v_2$  is given color 2. Present  $v_3$  at  $(5, 4)$ . Three cases are possible.

1. If  $v_3$  is given color 3 then three colors are used already. Present  $M_4$  at  $(9, 2)$ , it will be given color 2 or 4.
2. If  $v_3$  is given color 1 then present  $v_4$  at  $(3, 8)$ .  $v_4$  must be given color 3.
3. If  $v_3$  is given color 2 then present  $v_4$  at  $(9, 2)$ .  $v_4$  must also be given color 3.

In all cases, the graph presented is a bipartite  $P_4$  and is colored with at least 3 colors.  $\square$

Let us conclude the preliminaries with some remarks about symmetries. Consider two permutation graphs  $G = (V, E)$  and  $G' = (V', E')$ , where the latticial representations of  $G$  and  $G'$  are symmetric to each other with respect to the  $x$ -axis (or  $y$ -axis). For any vertex  $v$  of  $G$ , let  $v'$  be its symmetric vertex in  $G'$ . Clearly,  $(u, v) \in E \Leftrightarrow (u', v') \notin E'$ . Thus,  $G'$  is the complement of  $G$ .

Consider now a permutation graph  $G'' = (V'', E'')$  obtained from  $G$  by symmetry respectively to the axis given by the vector  $\vec{u} = (1, 1)$ . Clearly,  $G''$  is isomorphic to  $G$ .

Thus, we can deduce proposition 3.10.

**Proposition 3.10.** *Any algorithm for coloring permutation graphs presented in direction  $\vec{u} = (x, y)$  is equivalent to an algorithm for partitioning permutation graphs presented in direction  $\vec{u}' = (-x, y)$  (or  $(x, -y)$ ) into cliques. It is also equivalent to an algorithm for coloring permutation graphs presented in direction  $\vec{u}'' = (-x, -y)$ .*

We then deduce from propositions 3.7, 3.9 and 3.10:

**Corollary 3.11.** *A permutation graph presented from south-west to north-east or from north-east to south-west ( $\vec{u} = (x, y)$  with  $xy \geq 0$  and  $(x, y) \neq (0, 0)$ ) can be partitioned into cliques exactly by  $FF_k$  while no online algorithm can partition a permutation graph presented from south-east to north-west or from north-west to south-east exactly.*

### 3.2.2 Competitive Analysis of First-Fit

In 2000, Nikolopoulos and Papadopoulos have shown that First-Fit is not  $\chi$ -bounded if the presentation order is arbitrary [NP00]: for any integers  $\chi > 0$  and  $k$ , there exists a permutation graph  $G$  such that  $\chi(G) = \chi$  and  $\chi_{\text{FF}}(G) \geq \frac{1}{2}((\chi^2 + \chi) + k(\chi^2 - \chi))$ . In this section, we study the performance of First-Fit according to different presentation orders.

We first note that it achieves a competitiveness ratio of  $\frac{n}{4} + \frac{1}{2}$  for general graphs and that this bound is tight even for bipartite graphs. We then focus on bipartite permutation graphs and show that First-Fit is  $O(\sqrt{n})$ -competitive for this class.

First-Fit is known to guarantee a ratio close to  $\frac{n}{4}$  for general graphs. Indeed, Miller [Mil04] shows that a ratio lower than  $\frac{n}{4}$  cannot be achieved even for bipartite graphs and the  $\frac{n}{4}$ -competitiveness is mentioned by Lovász et al. [LST89]. Since we did not find a proof for this claim in the literature, we give it here, and take this opportunity to slightly precise this result.

**Proposition 3.12.** (See [LST89, Mil04]) *First-Fit guarantees a competitiveness ratio of  $\frac{n}{4} + \frac{1}{2}$  for every graph of order  $n$  and this bound is tight even for bipartite graphs.*

*Proof.* The proof of the lower bound is inspired from Miller's proof and improves it a little by showing that the performance ratio of First-Fit can be as bad as  $\frac{n}{4} + \frac{1}{2}$  for bipartite graphs.

We construct a bipartite graph  $G = (V, E)$  on  $2t$  vertices and give an ordering on  $V$  that forces First-Fit to use  $t + 1$  colors. We define  $G$  by letting  $V = \{v_1, \dots, v_t, w_1, \dots, w_t\}$  and  $E = \{(v_i, w_j) : i \neq j\} \cup \{(v_t, w_t)\}$ . Now consider running First-Fit using the ordering  $\{v_1, w_1, v_2, w_2, \dots, v_t, w_t\}$ .

Clearly, First-Fit will assign color  $i$  to vertices  $v_i$  and  $w_i$ , for  $i < t$ . It will assign color  $t$  to  $v_t$  and color  $t + 1$  to  $w_t$ . Since the graph is bipartite, it can be colored with two colors. Therefore, the performance ratio  $\rho$  of First-Fit is  $\rho \geq \frac{t+1}{2} = \frac{n/2+1}{2} = \frac{n}{4} + \frac{1}{2}$ .

We prove now that the claimed competitiveness ratio is guaranteed. Let us consider an online instance consisting in a graph  $G$  of order  $n$  presented vertex by vertex in an arbitrary order.

Note first that if  $\chi(G) = 1$ , then First-Fit is exact. So we can assume that  $\chi(G) \geq 2$ .

Let  $k$  be the number of color-classes containing only one vertex. First-Fit is conceived in such a way that the related vertices constitute a clique of order  $k$  and consequently  $\chi(G) \geq k$ .

On the other hand, the number of colors used by First-Fit is at most  $k + \frac{n-k}{2} \leq \frac{n}{2} + \frac{k}{2}$ . Consequently,  $\rho \leq \frac{n}{2\chi(G)} + \frac{k}{2\chi(G)} \leq \frac{n}{4} + \frac{1}{2}$ .  $\square$

Since the negative result already holds for bipartite graphs, we focus in our next analysis on bipartite permutation graphs.

**Theorem 3.13.** *The performance ratio of First-Fit on online Min Coloring on a bipartite permutation graph is  $O(\sqrt{n})$  and this bound is tight, even if we impose a direction of presentation  $\vec{u} = (x, y)$ ,  $x \cdot y > 0$ .*

*Proof.* We start by proving that  $O(\sqrt{n})$  is an upper bound for the performance ratio of First-Fit using Claim 3.14. We then prove that the given bound is tight using Adversary 4.

Consider a bipartite permutation, that is a permutation which can be partitioned into two increasing subsequences. Let  $C_1$  and  $C_2$  be the two colors obtained by applying First-Fit in the direction  $\vec{u} = (1, 0)$  on this permutation. Recall that in this case,  $C_1$  and  $C_2$  represent an exact coloring of this graph. Without loss of generality, we can suppose that for every arc  $(v_1, v_2)$  of the associated permutation graph, we have  $v_1 \in C_1$  and  $v_2 \in C_2$ .

An increasing subsequence  $(v_i)_i$  is *alternating* with respect to  $C_1$  and  $C_2$  if  $[(v_{2i})_i \in C_1 \text{ and } (v_{2i+1})_i \in C_2]$  or  $[(v_{2i})_i \in C_2 \text{ and } (v_{2i+1})_i \in C_1]$ .

**Claim 3.14.** *Suppose that First-Fit, applied online to the permutation, gives color  $k \geq 3$  to an alternating increasing subsequence of size  $P$ . Then First-Fit must have given color  $(k - 1)$  to an alternating increasing subsequence of size  $P$  and color  $(k - 2)$  to an alternating increasing subsequence of size  $P + 1$ .*

*Proof.* Without loss of generality, we can suppose  $k = 3$ . Suppose  $P = 2p$  (The case  $P = 2p + 1$  is similar) and let  $v_1, \dots, v_{2p}$  be the increasing alternating subsequence of size  $P$  colored with color 3. Without loss of generality, we can suppose  $v_{2i+1} \in C_1$ ,  $i = 0, \dots, p - 1$  and  $v_{2i} \in C_2$ ,  $i = 1, \dots, p$  (in the other case, the proof is identical). We have that

$$\forall i \exists w_{2i+1} \in SE(v_{2i+1}), w_{2i+1} \text{ colored } 2, w_{2i+1} \in C_2 \quad (3.2)$$

$$\forall i \exists w_{2i} \in NW(v_{2i}), w_{2i} \text{ colored } 2, w_{2i} \in C_1 \quad (3.3)$$

Thus,

$$w_{2i+1} <^{(a)} v_{2i+1} <^{(b)} v_{2i+2} <^{(c)} w_{2i+2} <^{(d)} w_{2i+3} \quad (3.4)$$

(a) Because  $w_{2i+1} \in SE(v_{2i+1})$ .

(b) Because  $(v_i)$  is increasing.

(c) Because  $w_{2i+2} \in NW(v_{2i+2})$ .

- (d) Because  $w_{2i+3}$  is on the right of  $v_{2i+3}$ , thus to the right of  $w_{2i+2}$ ; and  $(w_i)$  is increasing, since all its elements have the same color.

Thus, the  $(w_j)_j$  are all different and form an increasing alternating subsequence colored with color 2,  $w_{2i} \in C_1$  and  $w_{2i+1} \in C_2$ .

Similarly, the existence of the subsequence  $(v_i)$  proves the existence of an increasing alternating subsequence  $(u_k)$  of size  $P$  and of color 1 with  $u_{2i} \in C_1$  and  $u_{2i+1} \in C_2$ .

Similarly again, the existence of the subsequence  $(w_j)$  proves the existence of an increasing alternating subsequence  $(u'_k)$  of size  $P$  and of color 1 with  $u'_{2i} \in C_2$  and  $u'_{2i+1} \in C_1$ .

In addition,  $u_1 \neq u'_1$  because  $u_1 \in C_2$  and  $u'_1 \in C_1$ .

If  $u_1 < u'_1$ , the subsequence  $u_1, u'_1, \dots, u'_{2p}$  is increasing, alternating and of size  $P + 1$ . If  $u'_1 < u_1$ , then the subsequence  $u'_1, u_1, \dots, u_{2p}$  is increasing, alternating and of size  $P + 1$ .

This ends the proof of Claim 3.14 for the case  $P = 2p + 1$ . The proof is similar if  $P = 2p$ . □

To conclude the competitive analysis, we distinguish two cases:

1. If First-Fit colors the permutation with two colors, the competitiveness ratio is one.
2. If First-Fit uses  $k \geq 3$  colors, then by iteratively applying Claim 3.14, we see that the number  $n$  of vertices is such that  $n \geq \lceil \frac{k}{2} \rceil (\lceil \frac{k}{2} \rceil + 1) \geq (\frac{k}{2})^2$ .

Thus  $k \leq 2\sqrt{n}$ . This inequality still holds for  $k = 2$ . This proves that a performance ratio of  $\sqrt{n}$  is guaranteed.

In order to conclude the proof of Theorem 3.13, let us use Adversary 4, illustrated in Figure 3.3 to prove that the bound is tight. This proof is given here for the direction  $\vec{u} = (1, 1)$ ; by a simple rotation, one can easily see that it holds for any  $\vec{u} = (x, y)$ ,  $x > 0$ ,  $y > 0$ .

For any  $k$  and any  $j$ , the vertex  $v_{(k,j)}$  is given color  $j$  because it is the smallest color that is admissible in this region. Thus, at each step  $k$ , a new color-class is opened. Let  $\lambda_k$  be the total number of color-classes opened when the step  $k$  is reached. We have  $\lambda_k = k$ .

Also, at each step  $k$  we add  $k$  vertices, the number  $n_k$  of vertices used is thus  $n_k = n_{k-1} + k$ , which induces that  $n_k = \frac{k(k+1)}{2}$ . We have  $\lambda_k = O(\sqrt{n_k})$ . This is true in particular at the last step: let the total number of colors used be  $\lambda$  and the total number of vertices be  $n$ , we have  $\lambda = O(\sqrt{n})$  and this concludes the proof of Theorem 3.13. □

**Adversary 4**

---

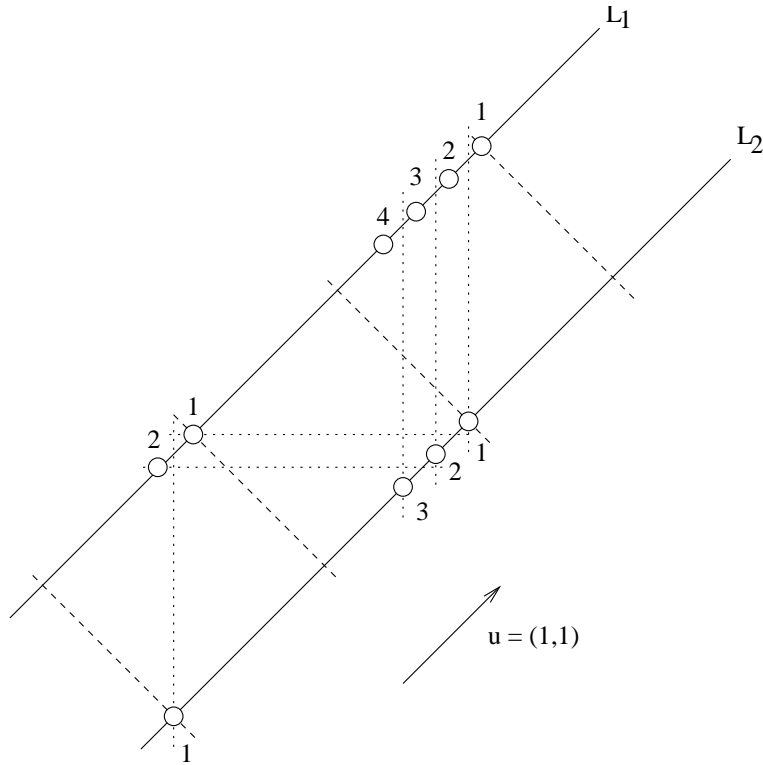
- 1: Draw two parallel lines  $L_1$  and  $L_2$  on a plane, such that  $L_1$  is above  $L_2$  and their direction vector has two positive components. These lines represent the colors of the vertices in an optimal offline coloring.
  - 2:  $k := 1$
  - 3:  $j := 1$
  - 4: Present a vertex  $v_{(k,j)}$  on  $L_2$ . It will be given color 1.
  - 5: **for**  $k := 2..K$  **do**
  - 6:      $j := k$
  - 7:     **if**  $v_{(k-1,j-1)}$  is on  $L_2$  **then**
  - 8:         Present  $v_{(k,j)}$  on  $L_1$  such that  $v_{(k,j)}|_x = v_{(k-1,j-1)}|_x - \varepsilon$
  - 9:     **else** */\* $v_{(k-1,j-1)}$  is on  $L_1$ \*/*
  - 10:         Present  $v_{(k,j)}$  on  $L_2$  such that  $v_{(k,j)}|_y = v_{(k-1,j-1)}|_y - \varepsilon$
  - 11:     **end if**
  - 12:     **for**  $j := (k-1)..2$  **do**
  - 13:         **if**  $v_{(k-1,j-1)}$  is on  $L_2$  **then**
  - 14:             Present  $v_{(k,j)}$  on  $L_1$  such that  $v_{(k-1,j)}|_x < v_{(k,j)}|_x < v_{(k-1,j-1)}|_x$
  - 15:         **else** */\* $v_{(k-1,j-1)}$  is on  $L_1$ \*/*
  - 16:             Present  $v_{(k,j)}$  on  $L_2$  such that  $v_{(k-1,j)}|_y < v_{(k,j)}|_y < v_{(k-1,j-1)}|_y$
  - 17:         **end if**
  - 18:     **end for**
  - 19:      $j := 1$
  - 20:     **if**  $v_{(k-1,j)}$  is on  $L_2$  **then**
  - 21:         Present  $v_{(k,j)}$  on  $L_1$  such that  $v_{(k,j)}|_x = v_{(k-1,j)}|_x + \varepsilon$
  - 22:     **else** */\* $v_{(k-1,j)}$  is on  $L_1$ \*/*
  - 23:         Present  $v_{(k,j)}$  on  $L_2$  such that  $v_{(k,j)}|_y = v_{(k-1,j)}|_y + \varepsilon$
  - 24:     **end if**
  - 25: **end for**
- 

A *co-bipartite* graph is a graph which can be partitioned into two cliques. By Remark 1.36 and Proposition 3.10, we get:

**Corollary 3.15.** *The performance ratio of  $FF_k$  for online partitioning of a co-bipartite permutation graph is  $O(\sqrt{n})$  and this bound is tight, even if we impose a direction of presentation  $\vec{u} = (x, y)$ ,  $x \cdot y < 0$ .*

**Remark 3.16.** *For the bipartite case, the result of Nikolopoulos and Papadopoulos [NP00] states that online coloring does not admit a constant performance ratio. Our result improves it to  $O(\sqrt{n})$ .*





**Figure 3.3:** *Illustration of the principle of Adversary 4. The vertices are presented from South-West to North-East ( $\vec{u} = (1,1)$ ). The numbers close to them represent their colors attributed by First-Fit. The dotted lines are level-lines representing the  $x$  or  $y$  coordinates of the vertex they cross. The only purpose of these lines on this figure is helping to see whether a given vertex is to the left or the to right of some other vertex. The dashed lines show that each group of vertices with  $k = \text{constant}$  is presented after the group of vertices with  $k - 1$  in the direction of  $\vec{u}$ .*

### 3.2.3 A Better Performance Ratio

Given a comparability graph, it is well known that an exact  $k$ -coloring  $C_1, \dots, C_k$  can be found such that, for every arc  $(v, w)$ ,  $v \in C_i$  and  $w \in C_j$  with  $i < j$  [Gol04]. We call such a coloring an *increasing coloring*. This notion will be useful to understand the algorithm used in the proof of the following theorem.

**Note added in proof.** *Theorem 3.17 is a result in [Tro95]. It was found independently by the author in this thesis.*

**Theorem 3.17.** *There exists an online algorithm for coloring comparability graphs (presented with a transitive orientation) guaranteeing a competitiveness ratio of  $\frac{\chi+1}{2}$  and this bound is tight, even for permutation graphs presented in a latticial model.*

*Proof.* Let us consider Algorithm 5, which computes a proper coloring for a comparability graph presented vertex by vertex together with a transitive orientation.

---

**Algorithm 5**

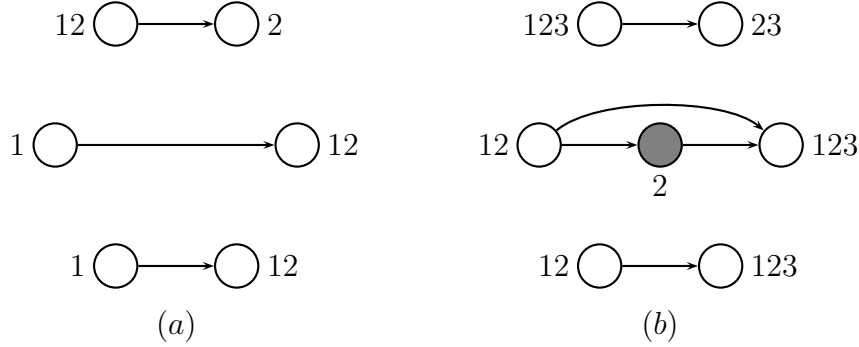
---

**Input:** A comparability graph  $G$  delivered online, vertex by vertex.

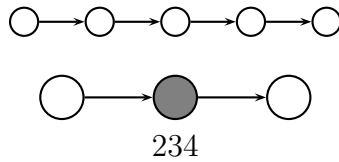
**Output:** A  $\frac{\chi(\chi+1)}{2}$ -coloring of  $G$ , where  $\chi$  is the chromatic number of  $G$ .  $\chi$  is unknown before the end of the algorithm.

- 1: **while**  $G$  is not completely presented **do**
  - 2:   Let  $G'$  be the subgraph of  $G$  induced by the currently presented vertices. Define  $k := \chi(G')$  and accept a new vertex  $v^*$ .
  - 3:   In the graph defined by  $G' \cup \{v^*\}$ , consider a longest path  $P$  containing  $v^*$ . Let  $l$  be the number of vertices in  $P$ .
  - 4:   **if**  $l > k$  **then** rename all the currently attributed colors as indicated here: for each vertex  $v$ , rename its color by concatenating the name  $C(v)$  of its color with  $C_r(v) + 1$  on the right. Increase  $k$  by 1.
  - 5:   Let  $p$  be the rank of  $v^*$  in  $P$ . Give color  $p \dots (k - l + p)$  to  $v^*$ .
  - 6: **end while**
- 

Colors used by Algorithm 5 are named by ascending sequences of consecutive integers. If  $C(v)$  is the color of a particular vertex  $v$ , we will call  $C_r(v)$  the right-most integer of  $C(v)$  and  $C_l(v)$  the left-most integer of  $C(v)$ . Sometimes, when there is no ambiguity on the vertex, we might just note  $C$ ,  $C_l$  and  $C_r$ . Note that  $C$  is a color (thus a sequence of consecutive integers), while  $C_l$  and  $C_r$  are integers. The principle of the algorithm is the following: at each online step, the color assigned to the vertex presented



**Figure 3.4:** *Illustration of Step 4 of Algorithm 5. The colors next to the nodes represent the colors attributed by Algorithm 5. Part (a) represents the graph before the presentation of vertex  $v$  and part (b) the same graph after the presentation of  $v$  (marked in gray). Since the chromatic number of the graph has increased, the colors are renamed.*



**Figure 3.5:** *Illustration of Step 5 of Algorithm 5. This graph contains two cliques. One of size 5 and one of size 3. For the sake of visibility, not all arcs of the cliques are drawn on this picture. The gray vertex is the newly presented vertex. In an optimal increasing coloring, it could be in  $C_2$ ,  $C_3$  or  $C_4$ . Therefore, Algorithm 5 assigns color 234 to it.*

is the sequence of integers corresponding to the possible colors the vertex can be assigned in every optimal increasing coloring in the already presented graph. The current coloring is also renamed in such a way that the sequence used to color every vertex always contains the possible colors in an optimal increasing coloring of the current graph.

Note that, at each step, Algorithm 5 has to compute a longest path containing  $v^*$  in a comparability graph. This can be done in polynomial time by computing a maximum clique in the neighborhood of  $v^*$ . The complexity of the steps 2 and 3 is  $O(|V'| + |E'|)$  each [Gol04]. Thus, the whole complexity of Algorithm 5 at each online step is also  $O(|V'| + |E'|)$ .

The proof of Theorem 3.17 is based on three claims.

**Claim 3.18.** *If any vertex  $v$  in the comparability graph is part of two different paths, both of maximum length, then  $v$  will hold the same rank in both paths.*

*Proof.* Assume  $v$  does not have the same rank in both paths. Let  $P_1$  be the path where  $v$  has the smallest rank and  $P_2$  be the other path. We can make a new path consisting of all elements of  $P_2$  preceding  $v$ ,  $v$  and all elements of  $P_1$  with a rank larger than  $v$ . This path will be longer than both  $P_1$  and  $P_2$ , which is in contradiction.  $\square$

**Claim 3.19.** *For every vertex  $v$  with color  $C$ , there exists a path with length  $\chi(G') + C_l - C_r$ , where  $v$  is exactly at rank  $C_l$ .*

*Proof.* This is obviously true when  $v$  is first assigned a color. Afterwards, every time  $\chi(G')$  increases by one,  $C_r$  also increases by one by Step 4 of the algorithm. So the sequence which was used in Step 3 will always verify this property.  $\square$

**Claim 3.20.** *Algorithm 5 computes a proper coloring.*

*Proof.* Let  $v^*$  be the last introduced vertex. Let  $v$  be a vertex with the same color as  $v^*$ ; we have to show that  $v$  and  $v^*$  are not in a same path.

By contradiction, let us first suppose that  $v$  is before  $v^*$  on some path  $P$ . By Claim 3.19,  $v$  and  $v^*$  both belong to a path ( $P_v, P_{v^*}$  respectively) of length  $\chi(G') + C_l(v^*) - C_r(v^*)$  at position  $C_l(v^*)$ . Moreover,  $P_{v^*}$  is a path containing  $v^*$  of maximum length in the already presented instance.

The path consisting of the elements of  $P_v$  preceding  $v$ ,  $v$ ,  $v^*$  and the elements of  $P_{v^*}$  with a rank larger than  $v^*$  would be longer than  $P_{v^*}$ , which is a contradiction.

A similar argument holds if  $v$  is after  $v^*$  on  $K$ .  $\square$

---

**Adversary 6**

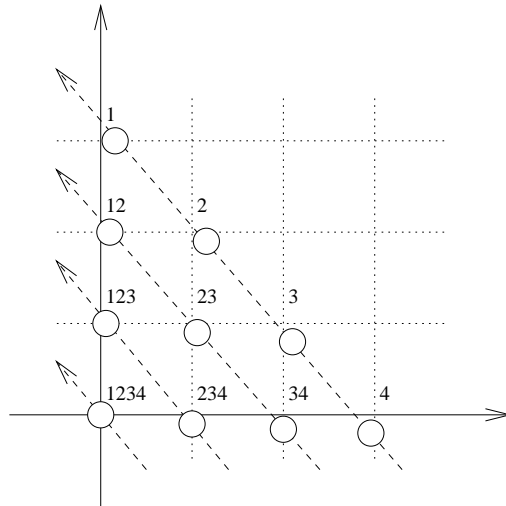
---

- 1: **for**  $k := 1..K$  **do**
- 2:   **for**  $x := k..0$  **do**
- 3:      $y := k - x$
- 4:     Put a vertex at coordinate  $(x + \varepsilon_y, y - \varepsilon_x)$ , where

$$\varepsilon_y := \begin{cases} 0 & \text{if } y = 0, \\ \varepsilon_{y-1} < \varepsilon_y < \varepsilon_{y-1} + \frac{1}{K} & \text{if } y \neq 0. \end{cases}$$

$$\varepsilon_x := \begin{cases} 0 & \text{if } x = 0, \\ \varepsilon_{x-1} < \varepsilon_x < \varepsilon_{x-1} + \frac{1}{K} & \text{if } x \neq 0. \end{cases}$$

- 5:   **end for**
  - 6: **end for**
- 



**Figure 3.6:** *Illustration of the Adversary 6. Vertices are inserted in the order south-west to north-east, and on each line in the order given by the arrow.*

The colors used by the algorithm are all subintervals of  $[1, \dots, \chi]$ . There are exactly  $\chi(\chi + 1)/2$  such subintervals. This concludes the analysis of Algorithm 5.

Let us now show that the bound is tight, even for permutation graphs. Adversary 6 presents a graph  $G$  in a continuous latticial model, such that  $\chi(G) = K$  and Algorithm 5 will use all subintervals of  $[1..K]$  as colors. This ends the proof of Theorem 3.17. □

**Corollary 3.21.** (Theorem 3.17, by Proposition 3.10) *There exists an online algorithm for partitioning co-comparability graphs (presented with a transitive orientation of a complementary graph) into cliques guaranteeing a competitiveness ratio of  $\frac{\theta+1}{2}$  and this bound is tight, even for permutation graphs. In particular, it gives an algorithm for partitioning permutations into decreasing sequences.*

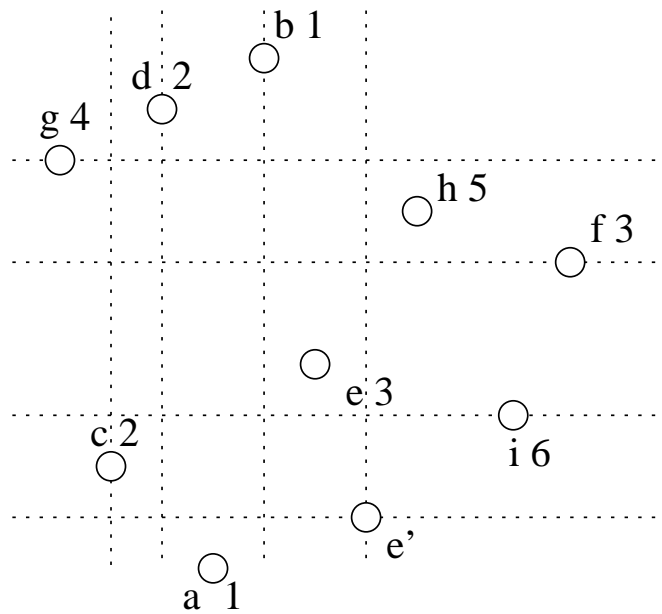
**Remark 3.22.** *It is known that the performance ratio of online Min Coloring on a co-comparability graph is an exponential function of  $\chi$  [Kie81, KPT94]. Moreover, with some restrictions on the order of presentation of the vertices, it is possible to guarantee a performance ratio that is a polynomial function of  $\chi$  [Fel97]. For the case of permutation graphs, which are comparability and co-comparability graphs, we achieve a performance ratio which is a polynomial function of  $\chi$  for a quite general online model.*

**Remark 3.23.** *Theorem 3.17 tells us that the class of comparability graphs is  $\chi$ -bounded by the binding function  $\frac{\chi(\chi+1)}{2}$ . For comparability graphs with a bounded chromatic number, it leads to a constant competitiveness ratio. In particular, Algorithm 5 guarantees 3 colors for bipartite graphs, which is optimal by proposition 3.9, and 6 colors for 3-colorable comparability graphs. It is worth noting that the class of bipartite graphs (and thus also the class of comparability graphs) is known to be not  $\chi$ -bounded [GL88] if the transitive orientation is not given. This hypothesis on the online model allows us to reduce the best known bound [Mil04] for online coloring of bipartite graphs from  $2 \log_2 n$  to 3.*

**Lemma 3.24.** *No online algorithm guaranteeing 1 color on stable sets and 3 colors on bipartite permutation graphs (given with a transitive orientation) can guarantee less than 6 colors for 3-colorable permutation graphs.*

*Proof.* If  $\chi(G) = 3$ , we devise an adversary that presents the vertices as shown on figure 3.7. The letters near the vertices represent their order of presentation and the numbers represent the colors that any admissible algorithm, in the sense of Lemma 3.24, would have to attribute to these vertices. Vertex  $e'$  is not actually presented and is only used for the proof. Let us explain for each vertex why it can get only one color:

vertex  $a$  is the first vertex; it is given color 1. When vertex  $b$  is presented,  $G$  is still 1-colorable. So, in order to respect the 1-bound,  $b$  must be given color 1. Vertex  $c$  is adjacent to vertex  $a$ ; it must be given color 2. Vertex  $d$  could not be given color 3, because if it was, an adversary could present the vertex  $e'$ , which would then have to be given color 4, thus not respecting the 3-bound on 2-colorable graphs. Vertex  $f$  must be given color 3. Vertices  $g$ ,  $h$  and  $i$  can be given only one color each.  $\square$



**Figure 3.7:** Graph delivered to force any admissible algorithm to use at least  $3(\chi - 1)$  colors on a  $\chi$ -colorable graph  $G$ . The letters near the vertices represent their order of presentation. The numbers represent their colors. The proof of Lemma 3.24 explains why vertex  $d$  cannot be colored with color 3. The dotted lines are a help to visualize the placement of vertices relatively to each-other.

### 3.2.4 Bounded Coloring

In [DEdW07], Demange, Ekim and de Werra mention that Min Coloring on permutation graphs has applications in robotics, namely for deciding how many robots are needed to collect objects of different sizes disposed along a line. It is natural to consider that a robot may not be able to transport more than a given amount  $b$  of objects. One then has to solve Min Bounded Coloring on permutation graphs. In [SM07], Spieksma and Moonen mention a similar application in the steel industry for online Min Bounded Coloring on permutation graphs. This section studies the behavior of First-Fit on online Min Bounded Coloring on both permutation and comparability graphs. We consider that the vertices may be presented in any order.

**Theorem 3.25.** *For any  $b \geq 3$  the performance ratio  $\rho_{FF_b}$  of  $FF_b$  on a permutation graph is*

$$\frac{b}{2} \left( 1 + \frac{1}{2\chi - 1} \right) \leq \rho_{FF_b} \leq \frac{b}{2} + \frac{1}{2}$$

*Proof.* The upper bound is given by Theorem 2.3. Let us show the lower bound by considering the instance presented by Adversary 7. Note that on lines 7, 8, 13 and 14 of Adversary 7,

$$\chi - i - j \geq 2 \quad \forall (i, j) \tag{3.5}$$

For an easier understanding of the equations hereafter, the reader can always assume that terms in  $\varepsilon$  are negligible compared to terms in  $\eta$ , which are themselves negligible compared to terms in natural numbers.

Let us now prove that vertices  $u_{(i,j+k)}^l$ , where  $k$  is a strictly positive natural number such that the vertex  $u_{(i,j+k)}^l$  exists and  $l \in \{1, 2\}$ , are all linked by an edge to  $u_{(i,j)}^2$ .

$$u_{(i,j+k)}^1|_x = \chi - i + i\eta + 2(j+k)\varepsilon = \chi - i + i\eta + 2j\varepsilon + \varepsilon + (2k-1)\varepsilon \tag{3.6}$$

$$u_{(i,j+k)}^1|_x = u_{(i,j)}^2|_x + (2k-1)\varepsilon \tag{3.7}$$

$$u_{(i,j+k)}^1|_x > u_{(i,j)}^2|_x \tag{3.8}$$

$$u_{(i,j)}^2|_y = \chi - i - j + i\eta + 2j\varepsilon + \varepsilon \stackrel{(3.5)}{\geq} 2 + i\eta + 2j\varepsilon + \varepsilon \tag{3.9}$$

$$u_{(i,j)}^2|_y \geq 1 + i\eta + 2j\varepsilon + \varepsilon + 1 > 1 + i\eta + 2j\varepsilon + k\varepsilon \tag{3.10}$$

$$u_{(i,j)}^2|_y > u_{(i,j+k)}^1|_y \tag{3.11}$$



---

**Adversary 7**

---

**Input:** A natural number  $b \geq 3$

**Output:** A permutation graph  $G$  presented online such that  $b$ -FF will have a performance ratio of  $\frac{b}{2} + \frac{1}{2\chi-1}$

1: Let  $p$  be any odd natural number.

2:  $\chi \leftarrow \lfloor \frac{b \cdot p + 1}{2} \rfloor$  /\*Once completely presented,  $G$  will be a graph with chromatic number  $\chi$ , as usually.\*/

3: Let  $\eta$  be a real number such that  $\eta \ll \frac{1}{2\chi}$

4:  $\varepsilon \leftarrow \frac{\eta}{2\chi}$

5: **for** int  $i = 0..(\chi - 2)$  **do**

6:   **for** int  $j = 0..(\chi - 2 - i)$  **do**

7:     Present a vertex  $u_{(i,j)}^1 = (\chi - i + i\eta + 2j\varepsilon, 1 + i\eta + 2j\varepsilon)$

8:     Present a vertex  $u_{(i,j)}^2 = (\chi - i + i\eta + 2j\varepsilon + \varepsilon, \chi - i - j + i\eta + 2j\varepsilon + \varepsilon)$

9:   **end for**

10: **end for**

11: **for** int  $i = 0..(\chi - 2)$  **do**

12:   **for** int  $j = 0..(\chi - 2 - i)$  **do**

13:     Present a vertex  $v_{(i,j)}^1 = (\chi - i + \chi\eta + (i + 1)\eta - 2j\varepsilon, 1 + \chi\eta + (i + 1)\eta - 2j\varepsilon)$

14:     Present a vertex  $v_{(i,j)}^2 = (j + 1 + \chi\eta + (i + 1)\eta - 2j\varepsilon + \varepsilon, 1 + \chi\eta + (i + 1)\eta - 2j\varepsilon + \varepsilon)$

15:   **end for**

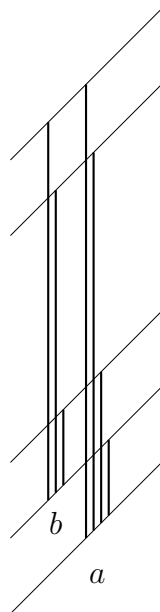
16: **end for**

17: **for** int  $i = 0..(\chi - 1)$  **do**

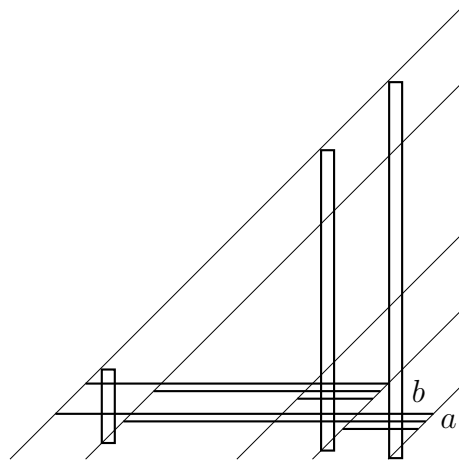
18:   Present a vertex  $w_i^1 = (\chi - i + \chi\eta + i\eta + \varepsilon, 1 + \chi\eta + i\eta + \varepsilon)$

19: **end for**

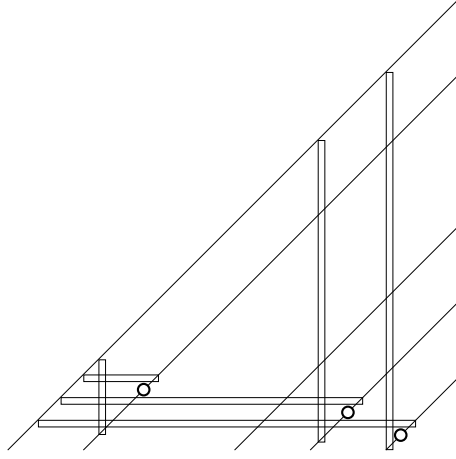
---



**Figure 3.8:** *Illustration of loop 5 to 10 of Adversary 7. The oblique lines represent the color-classes in an offline exact coloring; they have a slope of 1. The vertical lines represent the two vertices presented at steps 7 and 8:  $u_{(i,j)}^1$  is at the southern end of the line and  $u_{(i,j)}^2$  is at the northern end of it; of course,  $u_{(i,j)}^2$  is slightly more to the east than  $u_{(i,j)}^1$ . The group a represents the first passage through the loop 6 to 9 ( $i = 0$ ) and the group b represents the second passage through this loop ( $i = 1$ ). Note that, for the sake of visibility,  $\varepsilon$  has been made very big on the picture. It must in fact be very small, as described in the adversary.*



**Figure 3.9:** *Illustration of loop 11 to 16 of Adversary 7. The oblique lines represent the color-classes in an offline exact coloring; they have a slope of 1. The vertical boxes represent the sets of points presented at steps 5 to 10 (see Figure 3.8). The horizontal lines represent the two vertices presented at steps 13 and 14:  $v_{(i,j)}^1$  is at the eastern end of the line and  $v_{(i,j)}^2$  is at the western end of it; of course,  $v_{(i,j)}^2$  is slightly more to the south on the vertical axis than  $v_{(i,j)}^1$ . The group a represents the first passage through the loop 12 to 15 ( $i = 0$ ) and the group b represents the second passage through this loop ( $i = 1$ ). Note that, for the sake of visibility,  $\varepsilon$  has been made very big on the picture: the correct value of  $\varepsilon$  is given in the adversary.*



**Figure 3.10:** *Illustration of loop 17 to 19 of Adversary 7. The oblique lines represent the color-classes in an offline exact coloring; they have a slope of 1. The vertical (respectively horizontal) boxes represent the sets of points presented at steps 5 to 10 (respectively 11 to 16) (see Figure 3.8, respectively Figure 3.9). The small circles represent the vertices  $w_i^1$ .*

(3.8) and (3.11) show that  $u_{(i,j)}^2$  and  $u_{(i,j+k)}^1$  form a clique.

$$u_{(i,j+k)}^2|_x = u_{(i,j)}^2|_x + 2k\varepsilon > u_{(i,j)}^2|_x \quad (3.12)$$

$$u_{(i,j+k)}^2|_y = \chi - i - (j+k) + i\eta + 2(j+k)\varepsilon + \varepsilon \quad (3.13)$$

$$u_{(i,j+k)}^2|_y = \underbrace{\chi - i - j + i\eta + 2j\varepsilon + \varepsilon - k + 2k\varepsilon}_{u_{(i,j)}^2|_y} \quad (3.14)$$

$$u_{(i,j+k)}^2|_y < u_{(i,j)}^2|_y \quad (3.15)$$

(3.12) and (3.15) show that  $u_{(i,j)}^2$  and  $u_{(i,j+k)}^2$  form a clique. Thus, for a given  $i$ , each time the adversary enters the loop starting at Step 6, a new color-class is opened.

We now want to prove that the vertices  $u_{(i+k,j_a)}^l$  are always linked to the vertices  $u_{(i,j_b)}^1$  for all natural number  $k, j_a, j_b$  such that the vertices  $u_{(i+k,j_a)}^l$

and  $u_{(i,j_b)}^1$  exist and  $l \in \{1, 2\}$ . First, note that

$$2j_a\varepsilon - k - k\eta < 2j_b\varepsilon \quad \forall k, j_a, j_b \quad (3.16)$$

$$u_{(i+k,j_a)}^1|_x = \chi - i + i\eta + 2j_a\varepsilon - k + k\eta \quad (3.17)$$

$$u_{(i,j_b)}^1|_x = \chi - i + i\eta + 2j_b\varepsilon \quad (3.18)$$

Thus, by (3.16),  $u_{(i+k,j_a)}^1|_x < u_{(i,j_b)}^1|_x$ . Note now that, for all  $j_a$  and  $j_b$ ,  $k\chi > j_b - j_a$ . Thus,  $2k\chi\varepsilon > 2(j_b - j_a)\varepsilon$  and  $k\eta > 2j_b\varepsilon - 2j_a\varepsilon$ . Besides that,

$$u_{(i+k,j_a)}^1|_y = 1 + i\eta + 2j_a\varepsilon + k\eta \quad (3.19)$$

$$u_{(i,j_b)}^1|_y = 1 + i\eta + 2j_b\varepsilon \quad (3.20)$$

Thus,  $u_{(i+k,j_a)}^1|_y > u_{(i,j_b)}^1|_y$  and  $u_{(i+k,j_a)}^1$  and  $u_{(i,j_b)}^1$  are always linked by an edge for all  $k, j_a, j_b$ . Let us now look at  $u_{(i+k,j_a)}^2$ .

$$u_{(i+k,j_a)}^2|_y = \chi - (i+k) - j_a + (i+k)\eta + 2j_a\varepsilon + \varepsilon \quad (3.21)$$

Since  $\chi - (i+k) - j_a \geq 2$ ,

$$u_{(i+k,j_a)}^2|_y \geq 1 + i\eta + 2j_a\varepsilon + 1 + k\eta + \varepsilon > 1 + i\eta + 2j_b\varepsilon \quad (3.22)$$

$$u_{(i+k,j_a)}^2|_y > u_{(i,j_b)}^1|_y \quad (3.23)$$

And

$$u_{(i+k,j_a)}^2|_x = \chi - i + i\eta + 2j_a\varepsilon - k + k\eta + \varepsilon < \chi - i + i\eta + 2j_b\varepsilon \quad (3.24)$$

$$u_{(i+k,j_a)}^2|_x < u_{(i,j_b)}^1|_x \quad (3.25)$$

Thus,  $u_{(i+k,j_a)}^l$  and  $u_{(i,j_b)}^1$  form a clique for all  $k, j_a, j_b$  such that the vertices  $u_{(i+k,j_a)}^l$  and  $u_{(i,j_b)}^1$  exist. Each time the adversary passes through Step 5, a new set of colors must be introduced. Furthermore, for every pair  $(i, j)$ ,  $u_{(i,j)}^1$  and  $u_{(i,j)}^2$  are in a stable set. Since they are presented consecutively, they are colored with the same color.

One can prove similarly that, each time the adversary goes through Step 12, a new color must be introduced and that  $v_{(i,j)}^1$  and  $v_{(i,j)}^2$  are colored with the same color. In addition, the same reasoning can be used to prove that:

- All vertices  $v_{(i_0,j_a)}^1$  are linked to the vertices  $u_{(i,j_b)}^2$  for all  $i \leq i_0$ .
- All vertices  $v_{(i_0,j_a)}^1$  are linked to the vertices  $u_{(i,j_b)}^1$  for all  $i > i_0$ .

- All vertices  $v_{(i_a, j_0)}^2$  are linked to the vertices  $u_{(i, j_b)}^1$  for all  $i \leq \chi - j_0 - 1$ .
- All vertices  $v_{(i_a, j_0)}^2$  are linked to the vertices  $u_{(i, j_b)}^2$  for all  $i > \chi - j_0 - 1$ .

Thus, the vertices  $v$  may not have the same colors as the vertices  $u$ . Finally, all vertices  $w_{i_0}$  are linked to:

- All vertices  $u_{(i, j)}^2$  for all  $i_0 \leq i$ .
- All vertices  $u_{(i, j)}^1$  for all  $i_0 > i$ .
- All vertices  $v_{(i, j)}^2$  for all  $i_0 \leq i$ .
- All vertices  $v_{(i, j)}^1$  for all  $i_0 > i$ .

The total number of colors used by First-Fit on this instance is thus  $\frac{\chi(\chi-1)}{2}$  for the loop starting at Step 5,  $\frac{\chi(\chi-1)}{2}$  for the loop starting at Step 11, and  $\chi$  for the loop starting at Step 17, for a total of  $\chi^2$  colors.

Let us now look for an exact unbounded coloring of this instance. First, draw  $\chi$  straight lines on the plane with equations  $d_i : y = x - i, i \in [0..(\chi-1)]$ .

It is easy to verify that the vertex  $u_{(i, j)}^1$  is on the line  $d_{\chi-i-1}$  and that the vertex  $u_{(i, j)}^2$  is on the line  $d_j$ . Thus, on each line, there are  $\chi - 2$  vertices of type  $u$ .

Similarly, the vertex  $v_{(i, j)}^1$  is on the line  $d_{\chi-i-1}$  and that the vertex  $v_{(i, j)}^2$  is on the line  $d_j$ . Thus, on each line, there are  $\chi - 2$  vertices of type  $v$ .

Finally, the vertex  $w_i^1$  is on the line  $d_{\chi-i-1}$ . Thus, on each line, there is exactly one vertex of type  $w$ .

On each line  $d_i$ , there are thus  $2\chi - 1$  vertices. In an exact increasing unbounded coloring, each line could be used as one color, and all colors would have the same size. Let us suggest the following bounded coloring: consider the color-classes of this exact increasing unbounded coloring and cut them in chunks of sizes at most  $b$ . Since  $\chi = \lfloor \frac{b \cdot p + 1}{2} \rfloor$ , this gives us a solution with  $\lambda_b = p\chi$  colors. On each line, at most one chunk will have a size strictly smaller than  $b$  (note that, if  $b$  is odd,  $\chi = \frac{b \cdot p + 1}{2}$ , all chunks have exactly size  $b$  and this solution is optimal). The performance ratio  $\rho$  of  $FF_b$  is thus

$$\rho \geq \frac{\chi^2}{\chi b} \geq \frac{\chi^2}{p\chi} = \frac{\chi}{p} = \frac{b}{2} \left( 1 + \frac{1}{2\chi - 1} \right) \quad (3.26)$$

This ends the proof of Theorem 3.25. □

**Corollary 3.26.** *For any  $b \geq 3$  the performance ratio  $\rho_{FF_b}$  of  $FF_b$  on a comparability or co-comparability graph is*

$$\frac{b}{2} \left( 1 + \frac{1}{2\chi - 1} \right) \leq \rho_{FF_b} \leq \frac{b}{2} + \frac{1}{2}$$

### 3.3 Cocoloring

The problem we deal with in this section is a generalization of Min Coloring. Roughly speaking, given a graph, we want to cover its vertex set not only with stable sets but also with cliques. When dealing with cocoloring, each color-class is either a clique or a stable set.

**Definition 3.27** (Min Cocoloring). *Given a graph  $G = (V, E)$ , Min Cocoloring is the problem of partitioning the vertex set  $V$  into a minimum number of color-classes.*

**Definition 3.28.** *A  $k$ -cocolorable graph is a graph that can be partitioned into  $k$  color-classes.*

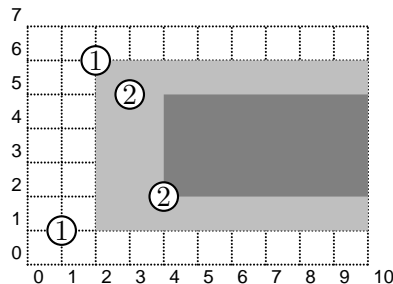
**Remark 3.29.** *First-Fit can trivially be adapted to partitioning a graph into color-classes. Let us denote it by  $FF_z$ .  $FF_z$  puts each vertex in the first possible color-class as it is presented.*

**Remark 3.30.** *As long as a color-class contains only one vertex, it is not determined whether this color-class is a stable set or a clique.*

In the offline case, cocoloring is known to be more difficult than coloring in comparability graphs (NP-hard) [Wag84]. A natural question is whether it is also more difficult in the online case. Indeed, in this section, we will point out that online cocoloring is as difficult in permutation graphs than in general graphs. For this reason, we will restrict us to permutation graphs and we will consider some relaxations of the online model allowing interesting results. We restrict ourselves to 2-cocolorable graphs. We first consider restrictions on the way the adversary may present the graph: we adopt the discrete latticial model and the permutation graph is presented from west to east. The second relaxation gives more freedom to the algorithm: we allow it a bounded delay before deciding the color of the presented vertices.

#### 3.3.1 A Dramatic Bound

Di Stefano et al. [DSKLZ06] have shown that two natural greedy algorithms can guarantee a performance ratio of  $\frac{n}{4} + \frac{1}{2}$  for the problem of online cocoloring of a permutation graph. One can observe that this result also holds for general graphs. In what follows, we point out that no algorithm can guarantee a better ratio, even for split permutation graphs. We recall that a graph is called split if its vertices can be partitioned into one clique and one stable set.



**Figure 3.11:** *Illustration of the proof of Proposition 3.31.ii. The numbers of the vertices represent the color-classes attributed by the adversary. The light gray (respectively dark gray) region represents the region where all vertices will be presented by the adversary after a second vertex has been put in color-class 1 (respectively 2).*

**Proposition 3.31.**

- i.*  $FF_z$  guarantees a performance ratio of  $\frac{n}{4} + \frac{1}{2}$  for online cocoloring general graphs.
- ii.* No algorithm for online cocoloring graphs can guarantee a performance ratio better than  $\frac{n}{4} + \frac{1}{2}$  even for split permutation graphs presented from west to east on a latticial plane.

*Proof.*

*i.* This proof is very similar to the proof given in [DSKLZ06]. It is straightforward to verify that  $FF_z$  cocolors exactly graphs with cochromatic number 1. So, we can assume that the cochromatic number is at least 2. Then, it is very simple to see that  $FF_z$  leaves at most one vertex alone in its color-class, since any two vertices form either a clique or a stable set. Thus, the ratio  $\frac{n}{4} + \frac{1}{2}$  immediately follows ( $\frac{n}{4}$  if  $n$  is even).

*ii.* We devise an adversary which presents a split permutation graph and forces  $\lceil \frac{n}{2} \rceil$  colors: The adversary delivers the latticial representation of the permutation from west to east. While the algorithm makes color-classes of size one, the elements can be presented at any  $y$ -coordinate. As soon as the algorithm makes a color-class of size two (increasing or decreasing), all the upcoming elements must be presented strictly within the interval of the two elements of this color-class, thus making it impossible to put any upcoming element in this color-class (See Figure 3.11). Thus, no color-class has a cardinality larger than 2, and the number of color-classes used is bigger or equal to  $\lceil \frac{n}{2} \rceil$ .  $\square$



**Remark 3.32.** *Note that Proposition 3.31.ii. holds even if the graph representing the permutation is a threshold graph (A threshold graph is a graph that can be partitioned into one clique and one stable set, such that for each pair of vertices  $v_1$  and  $v_2$  of the stable set,  $N(v_1) \subseteq N(v_2)$  or  $N(v_2) \subseteq N(v_1)$ . On a lattice plane, a permutation threshold graph has the highest element of the stable set lower than the lowest element of the clique).*

### 3.3.2 Split Permutation Graphs in a Discrete Latticial Model

Let us now consider the discrete latticial model, where the vertices have coordinates in  $\{1, 2, \dots, n\}^2$ ,  $n$  being, as usual, the size of the vertex set  $V$ . Moreover, the vertices are presented from west to east.

Di Stefano et al. [DSKLZ06] prove that one can force  $\frac{\log_2 n}{2}$  colors on a 3-cocolorable permutation graph. A slight modification of their proof allows to enhance their result with the following lemma, which can be seen as the “discrete counterpart” of Proposition 3.31:

**Lemma 3.33.** *It is possible to force  $\frac{\log_2 n}{2}$  colors on a permutation split graph, thus achieving a lower bound of  $\frac{\log_2 n}{4}$  on the performance ratio of Min Cocoloring on permutation split graphs.*

*Proof.* Consider Adversary 8 that presents split permutation graph of size  $n = 2^p$ , from west to east, where  $p$  is a positive integer.

---

#### Adversary 8

---

- 1:  $l \leftarrow 0$ ;  $h \leftarrow 2^p$ .
  - 2: **repeat**
  - 3:    $m \leftarrow \text{round} \left( \frac{l+h}{2} \right)$ .
  - 4:   Introduce a vertex  $v$  with  $v|_y = m$ .
  - 5:   **if** vertex  $v$  is put in a color-class of type clique **then**
  - 6:      $h \leftarrow m - 1$
  - 7:   **else** /\*vertex  $v$  is put in a color-class of type stable set or in a new color-class\*/
  - 8:      $l \leftarrow m + 1$
  - 9:   **end if**
  - 10: **until** ( $l = m$  or  $h = m$ )
  - 11: Fill the positions that have no vertex with a stable set above  $m$  and a clique below  $m$ .
- 

It is straightforward to see that at most  $\log_2 n = p$  vertices are presented before Step 11 is executed. Moreover, these  $\log_2 n$  first vertices are assigned

to at least  $\frac{\log_2 n}{2}$  color-classes. To complete the proof, we just have to note that it is always possible for the adversary to correctly execute Step 11 by filling the remaining positions with a stable set above  $m$  and a clique below  $m$ . Since the complete graph is 2-cocolorable, we have the competitive ratio of  $\frac{\log_2 n}{4}$ .  $\square$

**Theorem 3.34.** *The performance ratio of cocoloring split permutation graphs presented from west to east on a discrete latticial plane is bounded above by  $\log_2(n) + \frac{3}{2}$ .*

*Proof.* In order to prove Theorem 3.34, we use Algorithm 9, which, as will be shown here, guarantees this bound, and the two following claims.

---

**Algorithm 9**

---

**Input:** A permutation graph of size  $n$  presented online from west to east on a discrete latticial plane.

**Output:** A cocoloring of this permutation graph using at most  $3 + 2 \log_2 n$  color-classes.

- 1: Introduce a dummy vertex at  $(-1, -1)$  and put it in color-class  $s_1$ .
  - 2: Introduce a dummy vertex at  $(-1, n + 1)$  and put it in color-class  $c_1$ .
  - 3: **for each** new vertex  $v_m$  **do** /\*Vertices are numbered according to their  $x$ -coordinate:  $v_m|_x = m$ \*/
  - 4:    $S_m \leftarrow \max_{v \neq v_m} \{v|_y : \exists v' \in NE(v)\}$
  - 5:    $C_m \leftarrow \min_{v \neq v_m} \{v|_y : \exists v' \in SE(v)\}$
  - 6:   **if**  $S_m < C_m$  **then**
  - 7:     **if**  $(C_m - v_m|_y)^2 < (S_m - v_m|_y)^2$  **then** /\* $v_m$  is closer to  $C_m$ \*/
  - 8:       Put  $v_m$  in the first available color-class  $c_i$ .
  - 9:     **else** /\* $v_m$  is closer to  $S_m$ \*/
  - 10:       Put  $v_m$  in the first available color-class  $s_i$ .
  - 11:     **end if**
  - 12:   **else** /\* $S_m > C_m$ , so the point where the stable set crosses the clique has passed\*/
  - 13:     **if**  $v_m|_y > S_m$  **then**
  - 14:       Put  $v_m$  in the first available color-class  $s_i$ .
  - 15:     **else** /\* $v_m|_y < C_m$ \*/
  - 16:       Put  $v_m$  in the first available color-class  $c_i$ .
  - 17:     **end if**
  - 18:   **end if**
  - 19: **end for**
- 

Let  $s_m$  (respectively  $c_m$ ) be the vertex at height  $S_m$  (respectively  $C_m$ ). We say that a vertex *is clique* (respectively *is stable*) if it must be put in the

clique (respectively in the stable set) in all possible split decomposition of  $G$ .

**Claim 3.35.** *If  $S_m \geq C_m$ , then no vertex  $v$  will be presented between  $S_m$  and  $C_m$  ( $C_m \leq v|_y \leq S_m$ ) in the future. Moreover, all vertices that will be presented in the future above  $S_m$  (respectively below  $C_m$ ) constitute a stable set (respectively a clique) and will be colored as such by Algorithm 9.*

*Proof.* Suppose  $S_m > C_m$ .  $\exists s'_m \in NE(s_m)$  and  $\exists c'_m \in SE(c_m)$ . Since  $G$  is a split graph, one of the vertices  $s'_m$  and  $s_m$  is stable and one of the vertices  $c'_m$  and  $c_m$  is clique. A vertex  $v$  such that  $C_m \leq v|_y \leq S_m$  would be adjacent to a vertex that is stable ( $s_m$  or  $s'_m$ ) and thus be clique; furthermore, it would be non-adjacent to a vertex that is clique ( $c_m$  or  $c'_m$ ) and thus be stable, which is a contradiction.

$\forall v \in NE(s_m)$  presented when  $S_m \geq C_m$ ,  $v \in NE(c_m) \cap NE(c'_m)$ . Thus,  $v$  must be stable in all possible split decompositions of  $G$ . A similar reasoning shows that  $\forall v \in SE(c_m)$  must be clique in all possible split decompositions of  $G$ .  $\square$

Since  $S_m$  increases and  $C_m$  decreases along the execution of the algorithm, as soon as the condition at Step 6 is false, the rest of the graph is covered with at most 2 new color-classes. Let us now calculate the number of colors used before this condition is evaluated to false.

**Remark 3.36.** *As long as  $S_m < C_m$ ,  $SE(s_m) \cap NE(c_m)$  contains exactly 1 vertex.*

**Claim 3.37.** *We note  $\Delta_i^s$  the difference ( $C_{m_i} - S_{m_i}$ ) at the time when color-class  $s_i$  was opened. We define  $\Delta_j^c$  similarly for color-class  $c_j$ . Then, if  $s_i$ ,  $i \geq 2$  is opened,*

$$\Delta_i^s \leq \frac{1}{2} \Delta_{i-1}^s \tag{3.27}$$

and if  $c_j$ ,  $j \geq 2$  is opened,

$$\Delta_j^c \leq \frac{1}{2} \Delta_{j-1}^c \tag{3.28}$$

*Proof.* We will prove Claim 3.37 for  $\Delta_i^s$ . The proof for  $\Delta_j^c$  is similar.

At the time when a first vertex  $v_i$  is put in  $s_i$ , there exist  $v_{i-1}$  in  $s_{i-1}$ , with  $v_{i-1} \in NW(v_i)$ . Let  $S'_{m_{i-1}}$  and  $C'_{m_{i-1}}$  be the values of  $S_m$  and  $C_m$  when  $v_{i-1}$  was presented.

$$\Delta_{i-1}^s \geq \left| C'_{m_{i-1}} - S'_{m_{i-1}} \right| \tag{3.29}$$

Since  $v_{i-1}$  was put in color-class  $s_{i-1}$ ,

$$\left| v_{i-1}|_y - S'_{m_{i-1}} \right| \leq \left| C'_{m_{i-1}} - v_{i-1}|_y \right| \tag{3.30}$$

Now,  $v_i|_y < v_{i-1}|_y$  and  $v_i|_x > v_{i-1}|_x$ .

Let  $c_i$  and  $s_i$  be the vertices such that  $S_{m_i} = s_i|_y$  and  $C_{m_i} = c_i|_y$ . According to Remark 3.36,  $c_{m_i} \leq v_{i-1}|_y$ . Thus,

$$\Delta_i^s \leq \left| v_{i-1}|_y - S'_{m_{i-1}} \right| \leq \frac{1}{2} \left| S'_{m_{i-1}} - C'_{m_{i-1}} \right| \leq \frac{1}{2} \Delta_{i-1}^s \quad (3.31)$$

□

Suppose now that the algorithm uses  $t$  color-classes before the condition at Step 6 evaluates to false, and that these colors appeared in the order  $k_1, k_2, \dots, k_t$ .  $\forall i, k_i \in S$  (Stable colors) or  $k_i \in C$  (Clique colors). We note  $\Delta_i$  the difference  $C_{m_i} - S_{m_i}$  at the time when  $k_i$  was used for the first time.  $\Delta_1 \geq \Delta_2 \geq \dots \Delta_t$ .

Since, out of 3 colors, at least two are of the same type, Claim 3.37 tells us that

$$\Delta_{2k+1} \leq \frac{1}{2} \Delta_{2k-1} \quad (3.32)$$

Now,  $\Delta_1 \leq n$ . Thus,

$$\Delta_{2k+1} \leq \frac{1}{2^k} n \quad (3.33)$$

and

$$\Delta_{2k+2} \leq \frac{1}{2^k} n \quad (3.34)$$

So

$$\Delta_t \leq \frac{1}{2^{(t/2)-1}} n \quad (3.35)$$

Since  $\Delta_t \geq 1$ ,  $2^{(t/2)-1} \leq n$  and  $\frac{t}{2} - 1 \leq \log_2 n$ . Thus,

$$t \leq 2 \log_2 n + 1 \quad (3.36)$$

Altogether, Algorithm 9 has used at most  $3 + 2 \log_2 n$  colors. □

**Corollary 3.38.** *Algorithm 9 can be transformed into an online algorithm for cocoloring 2-cochromatic graphs with at most  $7 + 2 \log_2 n$  color-classes, leading to a competitiveness ratio of  $\frac{7}{2} + \log_2 n$ .*

*Proof.* The modified algorithm begins as First-Fit until it needs a third color, then it switches to  $\text{FF}_k$  until it needs 5 colors and finally it turns to Algorithm 9 by using the 5<sup>th</sup> color as a first color of Algorithm 9. Since the graph is revealed along the direction  $(1, 0)$ , First-Fit and  $\text{FF}_k$  are exact. If the graph is bipartite it will be colored exactly; if it is cobipartite, it will be colored with 4 color-classes and if it is a split graph, it will be colored with at most  $7 + 2 \log_2 n$  color-classes. □

### 3.3.3 Delayed Cocoloring

Next, we look at two relaxations of online Min Cocoloring where the algorithm is allowed a delay before deciding the color-class of the presented vertices. We consider that one vertex is delivered at each time unit.

We start with a case where the graph presented is a split permutation graph, presented from west to east. The algorithm is allowed to wait for at most one time unit before deciding the color-class of each vertex presented.

**Theorem 3.39.** *Online Min Cocoloring on a split permutation graph  $G$  delivered online from west to east on a latticial plane can be exact if the algorithm is allowed a delay of one time unit before deciding the color-class of each vertex presented.*

---

**Algorithm 10**


---

*Proof.* **Input:** A split permutation graph  $G$  presented online from west to east on a latticial plane.

**Output:** An exact cocoloring of  $G$ .

```

1: for each new vertex  $v$  do
2:   if  $v$  cannot be put in color-class  $c$ , respectively  $s$  then
3:     Put  $v$  in  $s$ , respectively in  $c$ 
4:     /*The next test is for the case of  $G$  being a threshold graph*/
5:   else if  $v$  is the last vertex of  $G$  then
6:     Put  $v$  in  $s$  or in  $c$ 
7:   else
8:     Wait for the next vertex  $v'$  to be presented
9:     if  $v' \in NE(v)$  then
10:      Put  $v$  in color-class  $s$ 
11:     else /* $v' \in SE(v)$ */
12:      Put  $v$  in color-class  $c$ 
13:     end if
14:   end if
15: end for

```

---

Algorithm 10 gives an exact cocoloring of any split permutation graph  $G$  delivered online from west to east on a latticial plane. We prove this by induction on  $v$ .

Suppose the already attributed color-classes correspond to an admissible split-decomposition of  $G$ .  $v$  has been presented, but not yet assigned to a color-class, and  $v'$  is being presented.

If  $v$  is above the lowest vertex in color-class  $c$  or below the highest vertex in color-class  $s$ , then we are in the case of Step 2, and the assignment of  $v$  is still admissible.

Suppose  $v$  is above the highest vertex in color-class  $s$  and below the lowest vertex in color-class  $c$ . Suppose  $v' \in NE(v)$  (the proof is similar if  $v' \in SE(v)$ ). If there exist a representation for which  $v$  can be put in the color-class  $c$ , necessarily, the color-class of  $v'$  will be  $s$  (because  $v'$  is non-adjacent to a vertex in the color-class  $c$ ). But in this case, one could change the color-class of  $v$  into  $s$  and still have an admissible cocoloring. Thus, putting  $v$  in the color-class  $s$  is never wrong.

Initialization of the induction: the same reasoning holds for the first two vertices. We can consider that there is a vertex with color  $s$  at  $(0, -\infty)$  and a vertex with color  $c$  at  $(-1, +\infty)$ .  $\square$

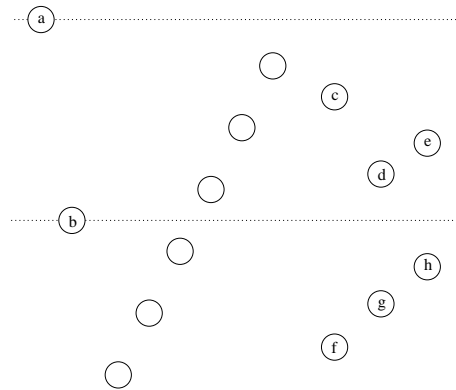
Given this surprising result, we look now at a less relaxed model, where the presented graph  $G$  may not be covered with less than 3 color-classes.

**Theorem 3.40.** *Even if an algorithm is allowed to wait for  $(n-4)$  time units before coloring a vertex, it is not possible to find an exact cocoloring for a 3-cocolorable graph presented online, even if it is presented from west-to-east on a latticial plane.*

*Proof.* The proof uses Figure 3.12. The vertices without label represent a stable set of size  $n - 5$ . The vertices are presented from west to east.  $a$  is in color-class 1. Once the vertices of the stable set are presented,  $b$  must be assigned to a color-class. If  $a$  and  $b$  are put in the same color-class, the adversary will present vertices  $c$ ,  $d$  and  $e$ . Else, the adversary will present  $f$ ,  $g$  and  $h$ . In any case, the algorithm will have to use at least four color-classes on the graph, while three would have been sufficient.  $\square$

## 3.4 Conclusion

In this chapter, we have given a tight analysis of First-Fit for the problem of online coloring comparability and permutation graphs and have shown that its performance ratio is  $O(\sqrt{n})$ . We also have presented an algorithm which dramatically improves the performance ratio of this problem to  $\frac{\chi+1}{2}$ , and have given a tight analysis of it. Furthermore, we have shown that First-Fit has a performance ratio between  $\frac{b}{2} + \varepsilon$  and  $\frac{b}{2} + \frac{1}{2}$  on Min Bounded Coloring in permutation and comparability graphs. The next chapter shows an application for these results.



**Figure 3.12:** *Illustration of the proof of Theorem 3.40. The vertices without label represent a stable set of size  $n-5$ . The vertices are presented from west to east.  $a$  is colored 1. Once the vertices of the stable set are presented,  $b$  must be colored. If  $a$  and  $b$  are colored with the same color-class, the adversary will present vertices  $c$ ,  $d$  and  $e$ . Else, the adversary will present  $f$ ,  $g$  and  $h$ .*

Finally, we have analyzed online Min Cocoloring on the same type of graphs: we have given the exact performance ratio  $(\frac{n}{4} + \frac{1}{2})$  of online Min Cocoloring in permutation and comparability graphs and have given results for some variations of the same problem.





# Chapter 4

## The Track Assignment Problem

This chapter, part of which is based on [DDSLB07], intends to initiate an analytical study of the computational complexity of some online shunting problems. Consider a train station consisting of a set of parallel tracks. Each track can be approached from one side only or from both sides. The departure times of the trains are fixed according to a given time table. The problem is to assign a track to each train as soon as it arrives to the station and such that it can leave the depot on time without being blocked by any other train; the total number of used tracks must be minimized.

We show that this problem can be modeled with online coloring of graphs. Depending on the constraints, the graphs can be overlap graphs<sup>1</sup> or permutation graphs. Thus, this chapter shows a nice application for the results of Chapter 3. More importantly, since it is natural to consider that the number of trains per track is limited, this application provides a nice framework for the analysis of the bounded version of online coloring of the involved graphs.

**Related works.** The track assignment problem is closely related to the more general shunting problem, that concerns the rolling stock allocation on a railway infrastructure under time, space and operational constraints. The shunting problem occurs in practical optimization problems like, e.g., the storage of trams or buses in a depot outside the rush hours, the rearrangement of railroad cars among different trains, the freight car distribution, and also the assignment of trains to platforms in a station [BBH<sup>+</sup>99, CDS07, DHMR00, DP<sup>v</sup>HKZ96, DSK04, FLKH05, GM01, HDM<sup>+</sup>06, HDS07, HSC00, Ros03, WZ00].

---

<sup>1</sup>Overlap graphs are also known as *circle graphs* [Gol04].

**Main results.** Starting from the Online Track Assignment Problem, this chapter studies online coloring of permutation and overlap graphs.

We show that if a permutation graph is presented from west to east on a latticial plane, then online Min Bounded Coloring has a performance ratio of  $2 - \frac{1}{\min\{b,k\}}$ , where  $k$  is the best known upper bound on  $\chi_b$ . We note that this result also holds for a comparability graph presented in a perfect order. If a permutation graph is presented on a latticial plane, starting from the origin and growing towards both east and west, we show that online Min Coloring has a performance ratio of  $2 - \frac{1}{\chi}$ . These results are applied to an online version of the problems described in [CDS07], and give an optimal solution for the Train Depot and the Big Train Station, both with the Midnight Condition. We show an upper bound of 2 for the Small Train Station Problem.

The results on overlap graphs provide bounds for the Train Depot Problem when the Midnight Condition is not fulfilled:

- i. Unbounded online coloring of an overlap graph has no constant competitive ratio: the ratio is at least  $O(\sqrt{n})$ , even for a bipartite overlap graph presented in increasing order of the left ends of its intervals.
- ii. The performance ratio can be upper-bounded by  $2\sqrt{M}$  if  $M \leq M_0$ , and by  $\log M (\lceil \log M / \log \log M \rceil + 1)$ , if  $M > M_0$ ; where  $M$  is the maximum length of the intervals, 1 is the minimum length, and  $M_0$  is such that  $2\sqrt{M_0} = 3 \log(M_0)$ . For large value of  $M$ , the ratio is  $O(\log^2 M / \log \log M)$ .

## 4.1 Introduction

Consider a train station. In the general case, the train station has two ends. Trains arrive from one end and leave from the same or from the other end. While the trains are in the station, they are stored on a track, which may be along a platform (for a station), or not (for a train depot). In order to save upkeep costs, one wants to use as few tracks as possible, under the condition that, at the time of leaving, a train is never blocked behind another train.

Since the trains may accumulate lateness during the day, the time of arrival of each train is unpredictable. The tracks must thus be assigned online, as the trains arrive, on the basis of departure times and previous assignments.

In some cases, we also take into account the finite length of a track: tracks have a finite capacity and may contain only a bounded number of trains. Throughout this work, we consider that this bound is fixed to an integer  $b$ .

This problem can be derived into several different models, which will be considered in this paper. As will be shown, these cases are equivalent to online graph coloring problems, on overlap or permutation graphs, depending on the exact model. For each model, we will show the equivalence with the appropriate online graph coloring problem and give an analysis of it.

As mentioned in Chapter 3, coloring a permutation graph is equivalent to partitioning a permutation into increasing subsequences [DSKLZ06]. E.g., [3, 6, 5, 10, 7, 1, 11, 8, 2, 12, 9, 4] can be decomposed into [3, 6, 10, 11, 12], [5, 7, 8, 9] and [1, 2, 4]. Also, a *bounded coloring of a permutation* is a decomposition of the permutation into increasing sequences, each with a limited number of elements.

As mentioned above, this chapter deals with overlap graphs. Here is a formal definition. Considering two intervals on a line, we say that they *overlap* if and only if they share at least one common point, and none is included in the other.

**Definition 4.1** (Overlap graphs). *An overlap graph is an undirected graph for which there exists a set of intervals on a line such that every vertex of the graph corresponds to an interval and two vertices are adjacent if and only if the corresponding intervals overlap.*

In [CDS07], Cornelsen and Di Stefano studied an offline version of our problem. They used the following notations: a train is of type XY if it enters the station from side X and leaves it from side Y. X and Y can take values R (for right) or L (for left).

## 4.2 Permutation Graphs

In many situations, the track assignment problem can be reduced to coloring a permutation graph; this section concentrates on these situations. In each case, we show the equivalence between the given problem and online coloring of a permutation graph.

### 4.2.1 The Train Depot with the Midnight Condition

The first model that we study is a train depot. Trains must be stored during the night on tracks. The tracks are organized as stacks, such that the last train to enter must be the first to leave on the next morning. In order to save time and energy, one wants to make sure that, when a train departs the next morning, it is always at the top of the stack. Thus, all trains are of type RR or all trains are of type LL.

One can represent on a time-axis the intervals during which each train must be stored in the depot. Two trains may be on the same track if their intervals are completely disjoint or if the interval of one train is contained in the interval of the other train (the first train arrives before and leaves after the second). One can make a graph where each vertex represents an interval and two vertices are joined by an edge if and only if the two intervals overlap but have no containment relationship. Such a graph is called an overlap graph. To model our track assignment problem, we can thus use online coloring of overlap graphs. Since one gets the knowledge of a train at the time when it arrives, the online model must state that the vertices of the overlap graph are presented in increasing order of the left ends of their corresponding intervals along the time axis.

The case of overlap graphs will however be studied in the next section. For now, we make the natural consideration that there is a time in the night at which all trains are in the depot. This is called the *midnight condition* and it means that the intervals of the trains all share at least one point. It has been proved [Gav73] that in this particular case, the overlap graph is a permutation graph. Since we know the time of arrival and of departure of our train (once they are presented), we consider that the permutation graph is given on a lattice plan, where the  $x$ -axis represents the arrival time and the  $y$ -axis represents the opposite of the departure time.

Because of the order of presentation of the permutation, the unbounded version of this problem is straightforward and has a performance ratio of 1. Indeed, there exists a polynomial algorithm which partitions a permutation into a minimum number of increasing sequences by reading the permutation from left to right [Gol04]. Therefore, we concentrate on the bounded version of the problem.

**Lemma 4.2.** *A lower bound for the performance ratio of online Min Bounded Coloring on a  $(k, b)$ -colorable permutation graph is  $2 - 1/l$ , where  $l = \min\{b, k\}$ , even if  $k$  is not known a priori and even if the graph is presented from west to east on a latticial plane.*

*Proof.* Suppose that  $k$  is known and fixed a priori. We prove Lemma 4.2 with a permutation presented on its lattice representation described in Adversary 11. The zone  $A$  represents the admissible zone where any element in the future will be presented. At the beginning,  $A$  is the complete plane. Each time the algorithm loops,  $A$  is reduced to a part of itself (see Adversary 11).

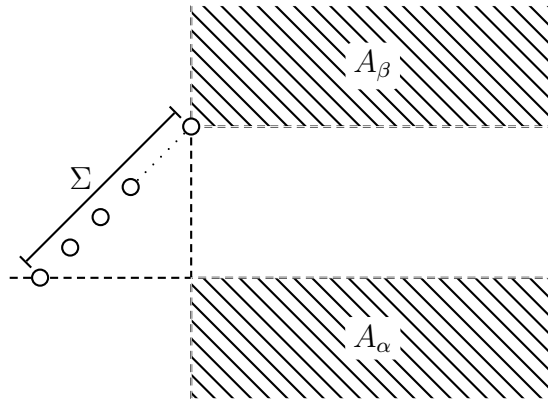
Let  $\lambda_\alpha$  be the number of colors used in all sequences of type  $\alpha$ ,  $\lambda_\beta$  be the number of colors used in all sequences of type  $\beta$  and  $\lambda_\gamma$  be the number of colors used on  $K_\gamma$ . Finally, let  $\lambda$  be the total number of colors used by the

---

**Adversary 11**

---

- 1: **repeat**
  - 2: Present a stable set  $\Sigma$  of size  $b$  in  $A$ .
  - 3: If the vertices of  $\Sigma$  are put two or more different color-classes, redefine  $A$  as the part of  $A$  which is in  $SE(\Sigma)$ , such that each future vertex in  $A$  will be adjacent to each vertex in  $\Sigma$ .  $\Sigma$  is said to be of type  $\alpha$ . If all vertices of  $\Sigma$  are put in a same color-class, redefine  $A$  as the part of  $A$  which is in  $NE(\Sigma)$ , such that each future vertex in  $A$  will be non-adjacent to each vertex in  $\Sigma$ .  $\Sigma$  is said to be of type  $\beta$ .
  - 4: **until**  $l - 1$  sequences  $\Sigma$  have been presented.
  - 5: Let  $N_\beta$  be the number of sequences  $\Sigma$  of type  $\beta$ . Present a clique  $K_\gamma$  of size  $N_\beta + 1$  in  $A$ .
- 



**Figure 4.1:** *Illustration of Adversary 11. Zone  $A$  after presenting one increasing sequence  $\Sigma$ . The zone  $A_\alpha$  (respectively  $A_\beta$ ) represents  $A$  if  $\Sigma$  is of type  $\alpha$  (respectively  $\beta$ ).*

algorithm. Clearly,

$$\lambda = \lambda_\alpha + \lambda_\beta + \lambda_\gamma \quad (4.1)$$

The definition of  $A$  after a sequence  $\Sigma$  of type  $\alpha$  allows us to say that each vertex presented after  $\Sigma$  will be adjacent to each vertex of  $\Sigma$ . Thus, any color used in  $\Sigma$  will never be used again on this instance. Besides that, since the sequences have size  $b$ , a color used on a sequence  $\Sigma$  of type  $\beta$  is saturated and may therefore not be used on any vertex not in  $\Sigma$ . Thus, no color is used in two different sequences. Therefore, if we call  $N_\alpha$  the number of sequences of type  $\alpha$  and  $N_\beta$  the number of sequences of type  $\beta$ , we have:

$$\lambda_\alpha \geq 2N_\alpha \quad ; \quad \lambda_\beta = N_\beta \quad ; \quad \lambda_\gamma = N_\beta + 1 \quad (4.2)$$

and thus

$$\lambda \geq 2N_\alpha + 2N_\beta + 1 \quad (4.3)$$

Since the algorithm presents exactly  $l - 1$  sequences of types  $\alpha$  and  $\beta$ ,

$$2N_\alpha + 2N_\beta + 1 = 2(N_\alpha + N_\beta) + 1 = 2(l - 1) + 1 = 2l - 1 \quad (4.4)$$

$$\lambda \geq 2l - 1 \quad (4.5)$$

Let us now compute the bounded chromatic number  $\chi_b$  of the instance presented above. Each sequence of type  $\alpha$  contains  $b$  vertices and can thus be colored with exactly one color. Thus, it is possible to use exactly  $N_\alpha$  colors on these sequences. Consider now the subgraph induced by the vertices of the sequences of type  $\beta$  and  $\gamma$ . By construction, we have that the sequences of type  $\beta$  form one long stable set  $\Sigma_\beta$  of size  $N_\beta b$  and each vertex of  $K_\gamma$  is stable with each vertex of  $\Sigma_\beta$ . It is possible to cover this subgraph with color-classes containing each at most  $b - 1$  vertices of  $\Sigma_\beta$  and 1 vertex of  $K_\gamma$ . Since  $|\Sigma_\beta| \leq (l - 1)b$  and since  $l \leq b$ , at most  $|K_\gamma| = N_\beta + 1$  such color-classes will be needed. Thus, one can color this subgraph with  $N_\beta + 1$  colors. Therefore

$$\chi_b \leq N_\alpha + N_\beta + 1 \leq l \quad (4.6)$$

From (4.5) and (4.6), we can deduce that the performance ratio  $\rho$  of online Min Bounded Coloring on permutation graphs presented from west to east on a latticial plane is:

$$\rho \geq \frac{\lambda}{\chi_b} \geq 2 - \frac{1}{l} \quad (4.7)$$

**Remark 4.3.** *If  $k$  is unknown in advance, we can consider the same instance as above with  $k \geq b$  and thus have  $\rho \geq 2 - 1/b$ .*

Remark 4.3 and (4.7) end the proof of Lemma 4.2. □

**Remark 4.4.** *Since permutation graphs are comparability graphs, the lower bound given in Lemma 4.2 also holds for comparability graphs. Note that if considered as a comparability graph, and if the arcs are oriented towards south-east, then the instance presented by Adversary 11 is presented in non-decreasing order of the vertices' ranks, which is a perfect order.*

**Lemma 4.5.** *The performance ratio of  $FF_b$  of online Min Bounded Coloring on a  $(k, b)$ -colorable permutation graph presented from west to east on a latticial plane is at most  $2 - 1/l$ , where  $l = \min\{b, k\}$ .*

*Proof.* The permutation is presented from west to east. Therefore, First-Fit finds an exact solution for Min Coloring on this problem [Gol04], which is equivalent to saying that it has a performance ratio of 1. By Theorem 2.1, Lemma 4.5 holds.  $\square$

**Remark 4.6.** *If the vertices are presented in nondecreasing order of their ranks, which is a perfect order for comparability graphs, the reasoning made for proving Lemma 4.5 gives the same upper bound for comparability graphs.*

**Theorem 4.7.** *The performance ratio of online Min Bounded Coloring a  $(k, b)$ -colorable permutation graph presented from west to east on a latticial plane is  $2 - 1/l$ , where  $l = \min\{b, k\}$ , and  $FF_b$  achieves this ratio.*

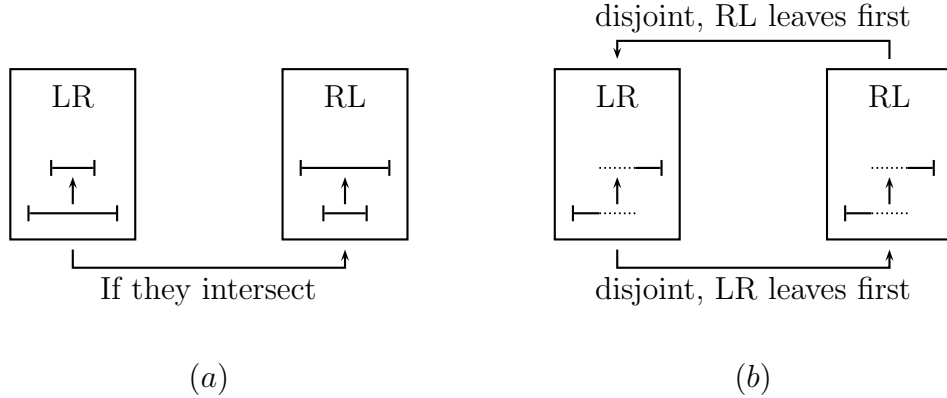
*Proof.* The proof is immediate from lemmas 4.2 and 4.5.  $\square$

By remarks 4.4 and 4.6, we get:

**Corollary 4.8.** *The performance ratio of online Min Bounded Coloring a  $(k, b)$ -colorable comparability graph presented in increasing order of the vertices' rank is  $2 - 1/l$ , where  $l = \min\{b, k\}$ , and  $FF_b$  achieves this ratio.*

## 4.2.2 The Small Train Station

In this subsection, we consider a small train station where trains pass through, but never turn back: they all are of type LR or RL. Again, the goal is to minimize the number of tracks needed while making sure that at all moments in time, a train that is scheduled to leave the station is not blocked behind another train. As this model corresponds to a typical small train station, we call it the Small Train Station Problem. In [CDS07], this model is called *Without Turning Back Trains* and it is shown that this problem is also equivalent to the coloring of a permutation graph. Figure 4.2 summarizes this proof and shows how the graph is built. Note that the midnight condition is not required here. In addition, here again, because of the dependence on the time, the intervals are presented in increasing order of their left ends.



**Figure 4.2:** Part (a) shows the conflict graph of the Small Train Station Problem as well as its transitive orientation. Part (b) shows a transitive orientation of the complement, thus proving that the graph is a permutation graph.

**Proposition 4.9.** *the performance ratio of First-Fit is unbounded by a constant on the Small Train Station Problem.*

*Proof.* Consider Adversary 12. For any integer  $k$ , it is easy to verify that First-Fit will use exactly  $i$  colors on each stable set  $\Sigma_i$ , and thus use exactly  $k$  colors. The optimum on this instance, however, is to put all intervals of type LR in one color, and all intervals of type RL in another color, thus using exactly 2 colors. Thus, for any integer  $k$ , one can force First-Fit to a performance ratio of  $\frac{k}{2}$ .  $\square$

A much better performance ratio can however be obtained.

**Proposition 4.10.** *The performance ratio  $c$  of the Small Train Station problem is bounded above by 2.*

*Proof.* The proof is straightforward by applying First-Fit separately on the permutation given by the trains entering from one side and the permutation given by the trains entering from the other side. If one never uses one track for two trains of different types, then each permutation is partitioned optimally by First-Fit. Since the optimal number of tracks for the complete permutation is at least as large as the optimal number of tracks of both

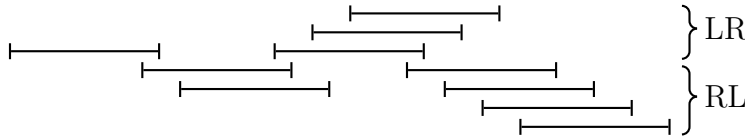


**Adversary 12**

**Input:** An integer  $k$ .

**Output:** An instance of the Small Train Station Problem on which First-Fit has a performance ratio of  $\frac{k}{2}$ .

- 1:  $i \leftarrow 1$
- 2:  $T \leftarrow \text{LR}$
- 3: **repeat**
- 4: Present a stable set  $\Sigma_i$  of  $i$  overlapping intervals, numbered from  $I_{i,1}$  to  $I_{i,i}$  of type  $T$  such that no interval of  $\Sigma_i$  overlaps any interval of type  $T$  in sets  $\{\Sigma_j | j < i\}$ . In addition, make sure that each interval  $I_{i,j}$  overlaps with intervals  $I_{(i-1),k}$  for all  $k > j$ .
- 5:  $i \leftarrow i + 1$
- 6: If  $T = \text{LR}$ , then  $T \leftarrow \text{RL}$ . Else,  $T \leftarrow \text{LR}$ .
- 7: **until**  $i = k$






**Figure 4.3:** This figure illustrates the instance presented by Adversary 12 (for  $k = 4$ ).

sub-permutations, the total amount of tracks is at most twice as large as the optimum.  $\square$

**Remark 4.11.** In the conflict graph of the Small Train Station Problem, two vertices are adjacent if and only if the corresponding intervals have a relation of inclusion. Let us call this graph an inclusion graph. As mentioned earlier (see Figure 4.2), this graph is a permutation graph. It is possible to prove that every permutation graph is also an inclusion graph:

Let  $\pi$  be a permutation containing  $n$  elements. For each element of the permutation, we note  $i$  its position and  $\pi(i)$  its value. On the axis of the real numbers, for each element of  $\pi$ , draw an interval  $[(-n + \pi(i)), i]$ . Clearly, two intervals have a relation of inclusion if and only if the corresponding elements of  $\pi$  appear in reverse order in the permutation. This allows us to build Table 4.1.

			Corresponding graph
0	0	0	Stable set
0	0	1	Co-interval $\subset$ comparability graphs
0	1	0	Overlap graphs
0	1	1	Co-inclusion = permutation graphs
1	0	0	Inclusion = permutation graphs
1	0	1	Co-overlap graphs
1	1	0	Interval graphs
1	1	1	Clique

**Table 4.1:** *Graphs of intervals on a real line. Each graph is built the following way:  $V$  is the set of intervals, and two intervals are adjacent if there is a 1 in the column corresponding to their relative position.*

### 4.2.3 The Big Train Station

Let us study the case of a big train station, where trains may enter from both sides and leave from both sides: each train may be of type LL, LR, RL or RR. We also assume that the midnight condition is fulfilled. We call this problem the Big Train Station Problem. As shown in [CDS07], this problem is again equivalent to coloring a permutation graph.

The corresponding permutation is built in the following manner: The trains departing to the left are labeled in increasing order of their departure time followed by the trains departing to the right in decreasing order of their departure time. This is exactly the order in which the trains can be positioned on one track such that they can leave on time. For the permutation, the trains arriving from the left are ordered in decreasing order of their arrival time followed by the trains arriving from the right in increasing order of their arrival time. This corresponds exactly to the order in which the trains would be positioned at midnight if only one track was given.

Here is an illustration of how the permutation for the Big Train Station Problem is built:

1. First the trains are labeled according to their departure direction and time (the intervals represent the arrival and departure times).

$$1 : [-4, 1]RL, 2 : [-2, 2]RL, 3 : [-1, 4]LL, 4 : [-3, 3]RR$$

2. Then they are permuted according to their arrival direction and time.

$$3 : [-1, 4]LL, 1 : [-4, 1]RL, 4 : [-3, 3]RR, 2 : [-2, 2]RL$$

Hence the resulting permutation is

$$[3, 1, 4, 2]$$

Online, using the above definition, the presentation of the corresponding permutation starts from a point in the middle. The left part of the permutation is then presented in increasing order of the position and the right part is presented in decreasing order of the position. There is no relation between the time of presentation of one element of one part and one element of the other part.

We slightly improve the result of [CDS07] and show that it is optimal.

**Theorem 4.12.** *The performance ratio of the Big Train Station Problem is  $2 - \frac{1}{\chi}$ .*

*Proof.* It is always possible to guarantee at most  $2\chi$  colors on the permutation by separately coloring the left part of the permutation and the right part of the permutation, with two different sets of colors and the First Fit algorithm. Since they are both presented in a perfect order, the coloring on each part will be optimal and therefore we will use at most  $2\chi$  colors. In addition, the color given to the first element can be used in both sets, since any element that can be colored with this color on the left will be in an increasing sequence with any element that can be colored with this color on the right. Doing this saves one color and uses at most  $2\chi - 1$  colors, thus guaranteeing a performance ratio of at most  $2 - \frac{1}{\chi}$ .

On the other hand, for any given  $\chi$ , it is possible to force any algorithm to use at least  $2\chi - 1$  colors. Consider Adversary 13, which presents such a permutation on the latticial plan.

---

**Adversary 13**

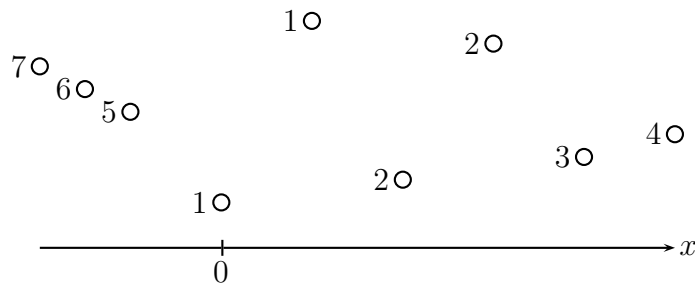
---

**Input:** A natural number  $\chi \geq 2$ .

**Output:** A permutation graph delivered online such that any algorithm will have a performance ratio of at least  $2 - \frac{1}{\chi}$ .

- 1: Let  $y_l \leftarrow 0$ ;  $y_h \leftarrow 10$ ;  $x \leftarrow 0$
  - 2: **for**  $i$  in  $[1..\chi]$  **do**
  - 3:   **repeat**
  - 4:      $y \leftarrow \frac{y_h + y_l}{2}$
  - 5:     Present a vertex  $v$  at  $(x, y)$
  - 6:      $x \leftarrow x + 1$
  - 6:     **if**  $(\text{color}(v) = i)$ , **then**  $y_l \leftarrow y$ ; **else**,  $y_h \leftarrow y$
  - 7:   **until**  $\text{color}(v) = i$
  - 8: **end for**
  - 9: Present a clique of  $\chi - 1$  elements of coordinates  $(x, y)$  with  $-1 < x < 0$  and  $y_l < y < y_h$ .
- 

Adversary 13 ensures the existence of one stable set  $\Sigma$  of size  $\chi$ , colored with  $\chi$  different colors by the algorithm, even though they could all have the same color in an exact increasing coloring. Each element of the clique that is then presented at step 9 is linked to each element of  $\Sigma$  and must therefore be colored with a new color. The algorithm must therefore use at least  $2\chi - 1$  colors, and thus has a performance ratio of at least  $2 - \frac{1}{\chi}$ . This ends the proof of Theorem 4.12.  $\square$



**Figure 4.4:** *This figure illustrates an instance which could be presented by Adversary 13. The adversary first presents the element on coordinate 0, then the elements with a positive  $x$ -coordinate, and finally the elements with a negative  $x$ -coordinate. The numbers next to the elements are the colors given by the algorithm.*

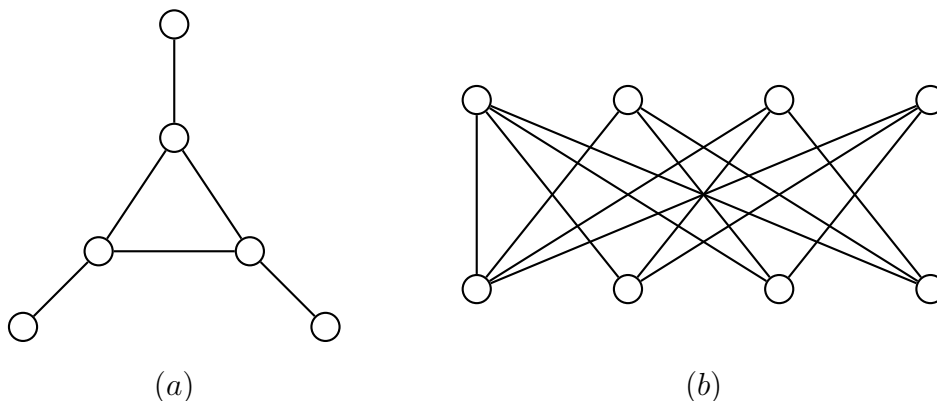
### 4.3 Overlap Graphs

The Midnight Condition, although very natural in some train stations, may not always hold. As shown in Section 4.2.1, when this condition is not satisfied, the Train Depot problem is equivalent to studying online Min Coloring on overlap graphs, where the intervals are given along a real axis in increasing order of the left ends of the intervals. This section focuses on this problem.

Clearly, overlap graphs are not perfect, since  $C_5$ , the cycle of 5 vertices, is an overlap graph. In 1980, Garey et al. have shown that finding the chromatic number of an overlap graph is NP-hard [GJMP80]. In 1992, Unger has given a polynomial algorithm for deciding whether an overlap graph is 3-colorable and has shown that deciding whether a graph is  $k$ -colorable is NP-complete for  $k \geq 4$  [Ung92].

**Remark 4.13.** *Both classes of comparability graphs and overlap graphs contain the class of permutation graphs. However, the classes of comparability graphs and overlap graphs are different. Figure 4.5 shows two graphs: in part (a), an overlap graph which is not a comparability graph and in part (b), a comparability graph which is not an overlap graph. A polynomial algorithm for recognizing comparability graphs (respectively overlap graphs) can be found in [Gol04] (respectively in [GSH89]).*

For every interval  $I_i$ , we will call  $l_i$  (respectively  $r_i$ ) the left (respectively right) end of this interval. Formally,  $I_i = [l_i, r_i]$ .



**Figure 4.5:** Part (a): an overlap graph which is not a comparability graph. Part b: a comparability graph which is not an overlap graph.

### 4.3.1 Unbounded Coloring of Overlap Graphs

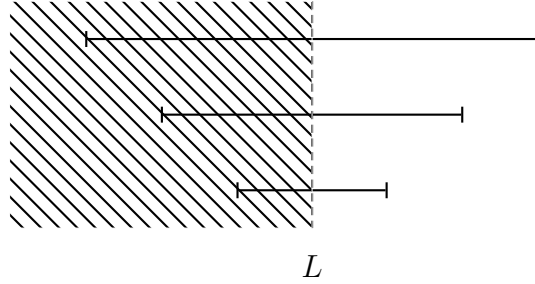
Since online coloring of overlap graphs has, to our knowledge, not been studied before, we first give some results in the case where the tracks have infinite length (or, say, a length  $n$ ), which corresponds to unbounded coloring of overlap graphs.

As a first step, we show that the bounds for FF on permutation graphs given in Theorem 3.13 still hold for overlap graphs:

**Lemma 4.14.** *The performance ratio of First-Fit on online Min Coloring on overlap graphs has a lower bound of  $O(\sqrt{n})$ , even if one imposes that the intervals are presented in increasing order of their left ends. This bound is tight for bipartite overlap graphs.*

*Proof.* Clearly, since  $O(\sqrt{n})$  is a lower bound for permutation graphs (See Theorem 3.13), which form a subclass of overlap graphs [Gav73], it is also a lower bound for overlap graphs. We must now show that this value is also an upper bound for bipartite overlap graphs.

In order to force color  $k$  on an interval  $I_k$  in an overlap bipartite graph, one needs a stable set  $\Sigma_k$  of at least  $k - 1$  intervals included in each other, colored with all colors in  $[1 \dots (k - 1)]$ . Since the intervals are presented in increasing order of their left ends, all intervals of this stable set must contain the left-most point of  $I_k$ . Similarly, to force color  $k - 1$  on one of these intervals, one also needs a stable set  $\Sigma_{k-1}$  of at least  $k - 2$  intervals colored with all colors in  $[1 \dots (k - 2)]$ . Furthermore,  $\Sigma_k$  and  $\Sigma_{k-1}$  must be completely disjoint since the graph is bipartite. Therefore, if we call  $n(k)$  the number of



**Figure 4.6:** *Illustration of Claim 4.15: On the left of the limit  $L$ , there can be anything. On the right side, there is nothing but the three represented intervals. All have a different color and the complete graph is bipartite.*

vertices needed to force color  $k$  to appear, we have  $n(k) \geq n(k-1) + k - 2$ . Thus,  $O(n(k)) = O(k^2)$  and  $k \leq O(\sqrt{n})$ . Since the performance ratio  $c$  is equal to  $k/\chi$ , we have  $c \leq k/2 \leq O(\sqrt{n})$ .  $\square$

Although this performance is not very good, we would like to show now that no algorithm can guarantee much better; more precisely, that no algorithm can guarantee a constant performance ratio, not even a performance ratio bounded by a function of  $\chi$ . We start with the following claim:

**Claim 4.15.** *For any algorithm, it is possible to force a stable set  $\Sigma$  of size at least three on a bipartite overlap graph such that the corresponding intervals are included in each other and there exists a limit  $L$ , such that for all intervals  $I_i \notin \Sigma$ ,  $r_i < L$  and for all intervals  $I_i \in \Sigma$ ,  $r_i > L$ , and such that each element of the stable set has a different color.*

*Proof.* Adversary 14 forces the result described in Claim 4.15.

Note that the given overlap graph is bipartite. At the end of step 1, we have a stable set of size two, colored with two different colors, and the limit  $L$  is either  $l_2$ , if  $I_2$  was colored 2, or  $r_2$  if  $I_2$  was colored 1. At the end of step 9, at least one interval must have been colored with color 3. Let  $I_{i_3}$  be this interval. Since we do not present any interval  $I_i$  such that  $i > i_3$ , we have a stable set of size at least three colored with three different colors. Note that, if  $I_{7''}$  is colored 3, we have a stable set of size 4 colored with three different

---

**Adversary 14**

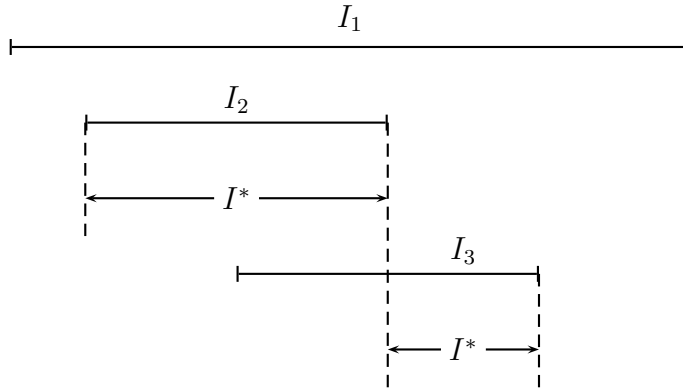
---

**Input:** Any online coloring algorithm.

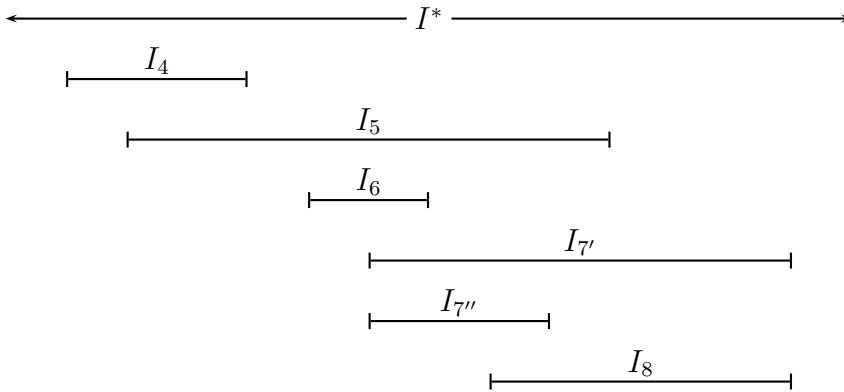
**Output:** A stable set as described in Claim 4.15.

- 1: Present an interval  $I_1$  */\* $I_1$  will have color 1.\**
  - 2: Present an interval  $I_2$  such that  $l_1 < l_2 < r_2 < r_1$
  - 3: **if**  $I_2$  has color 2 **then**
  - 4:    $I_* \leftarrow I_2$
  - 5: **else**
  - 6:   Present  $I_3$  such that  $l_2 < l_3 < r_2$  and  $r_2 < r_3 < r_1$
  - 7:    $I_* \leftarrow [r_2, r_3]$
  - 8: **end if**
  - 9: Present  $I_4$  such that  $l_* < l_4 < r_4 < r_*$
  - 10: **if**  $I_4$  has color 3, **then** STOP
  - 11: Let  $a$  be the color of  $I_4$  */\* $a \in \{1, 2\}$ \**
  - 12: Present  $I_5$  such that  $l_4 < l_5 < r_4$  and  $r_4 < r_5 < r_*$
  - 13: **if**  $I_5$  has color 3, **then** STOP
  - 14: Let  $b$  be the color of  $I_5$  */\* $b \in \{1, 2\}, b \neq a$ \**
  - 15: Present  $I_6$  such that  $r_4 < l_6 < r_6 < r_5$
  - 16: **if**  $I_6$  has color 3 **then**
  - 17:   STOP
  - 18: **else if**  $I_6$  has color  $a$  **then**
  - 19:   Present  $I_{7'}$  such that  $l_6 < l_{7'} < r_6$  and  $r_5 < r_{7'} < r_*$
  - 20: **else** */\* $I_6$  has color  $b$ \**
  - 21:   Present  $I_{7''}$  such that  $l_6 < l_{7''} < r_6$  and  $r_6 < l_{7''} < r_5$
  - 22:   **if**  $I_{7''}$  has color 3, **then** STOP
  - 23:   Present  $I_8$  such that  $r_6 < l_8 < r_{7''}$  and  $r_5 < r_8 < r_*$
  - 24: **end if**
-

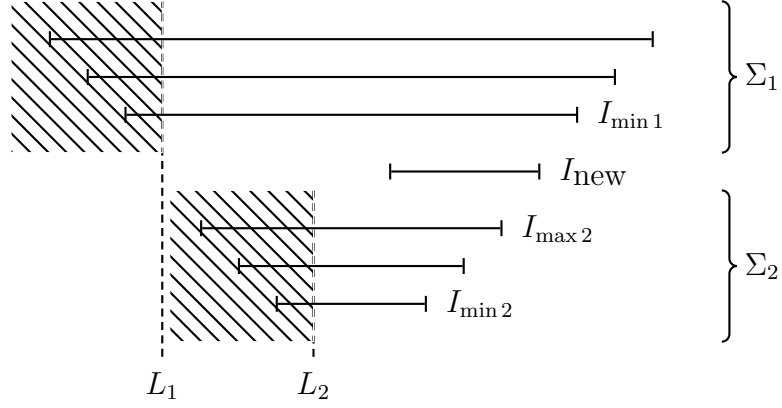




**Figure 4.7:** This figure illustrates steps 1 to 8 of Adversary 14. At the end of this part, in  $I_*$ , there are exactly two intervals, which form a stable set but are colored with two different colors.



**Figure 4.8:** This figure illustrates steps 9 to 24 of Adversary 14. At the end of this step, in  $I_*$ , there are three intervals colored with three different colors:  $I_1$ ,  $I_2$  or  $I_3$  and one element of  $\{I_i\}_{i \geq 4}$ .



**Figure 4.9:** *Illustration of the proof of Theorem 4.16.*

colors. Thus, the size of the stable set is at least 3 and at most 4. Let  $\Sigma$  be the considered stable set, we must set  $\max\{\max_{I_i \notin \Sigma} r_i, \max_{I_i \in \Sigma} l_i\} < L < \min_{I_i \in \Sigma} \{r_i\}$  to have the arrangement described in Claim 4.15.  $\square$

**Theorem 4.16.** *For any online coloring algorithm, it is possible to force any positive number of colors on a bipartite overlap graph.*

*Proof.* We prove this by induction. Suppose that it is possible to force  $k$  colors on a stable set  $\Sigma_1$  as in Claim 4.15. Construct such an instance. Let  $I_{\min 1}$  be the interval of  $\Sigma_1$  such that  $r_{\min 1} = \min_{I_j \in \Sigma_1} \{r_j\}$  and let  $L_1$  be the limit of  $\Sigma_1$  as in claim 4.15. In the interval  $[L_1, r_{\min 1}]$ , build a second stable set  $\Sigma_2$  of the same size as  $\Sigma_1$  and with the same number of colors on it. Let  $I_{\min 2}$  be the interval of  $\Sigma_2$  such that  $r_{\min 2} = \min_{I_j \in \Sigma_2} \{r_j\}$  and let  $L_2$  be the limit of  $\Sigma_2$ . Also, let  $I_{\max 2}$  be the interval of  $\Sigma_2$  such that  $r_{\max 2} = \max_{I_j \in \Sigma_2} \{r_j\}$ .

If the color-classes used on  $\Sigma_2$  are different from the ones used on  $\Sigma_1$ , define  $\Sigma = \Sigma_1 \cup \Sigma_2$  and  $L = L_2$  and  $\Sigma$  is now a stable set of the form of Claim 4.15 and covered with at least  $k + 1$  color-classes. If the color-classes used to cover  $\Sigma_2$  are the ones used to cover  $\Sigma_1$  are the same, present a new interval  $I_{\text{new}}$  such that  $L_2 < l_{\text{new}} < r_{\min 2}$  and  $r_{\max 2} < r_{\text{new}} < r_{\min 1}$  and define  $\Sigma = \Sigma_1 \cup \{I_{\text{new}}\}$  and  $L = r_{\max 2}$ .  $\Sigma$  is a stable set of the form of Claim 4.15 and covered with exactly  $k + 1$  color-classes.

Since it is possible for  $k = 3$  (see Claim 4.15), Theorem 4.16 holds.  $\square$

### 4.3.2 Overlap Graphs With Intervals of Bounded Size

**Lemma 4.17.** *In the special case where all intervals have length 1, First-Fit finds an exact coloring of an overlap graph presented in increasing order of one of the interval ends.*

*Proof.* In this case, if two intervals have the same starting point (and thus also the same end point), it is possible to put them in the same color-class. One can thus virtually consider two such intervals as just one interval, and therefore consider that no interval contains another interval. Therefore, the corresponding graph is also an interval graph. It is known that First-Fit finds an exact coloring of an interval graph presented in increasing order of the interval's left ends [Gol04].  $\square$

Lemma 4.17 gives an interesting result because, together with Theorem 4.16, it shows that there is a big difference between an overlap graph with intervals of constant size and an overlap graph with intervals of arbitrary size. Considering this, it is natural to wonder what happens if the size of the intervals is not constant, but bounded. In the remainder of this section, we will concentrate on graphs where the ratio between the shortest and the longest interval is smaller than or equal to some given integer.

Let us consider an online instance defined by a set of intervals presented in increasing order of their left ends. We assume that every interval is of size at least 1.

At each step  $t$ , interval  $I_t = [l_t, r_t]$  is presented. At this step, we define  $M(t)$  as the maximum length of intervals already presented,  $m(t)$  as their minimum length,  $\mu(t) = \log(M(t))$  and  $\mu'(t) = \mu(t)/\log(\mu(t))$ . Note that  $\mu(t)^{\mu'(t)} = M(t)$ . Note also that functions  $M$  and  $\mu$  are non decreasing and that  $\mu'$  is non decreasing for  $t, M(t) \geq 16$ . If the instance contains  $n$  intervals, then we respectively denote by  $M, m, \mu, \mu'$  the quantities  $M(n), m(n), \mu(n)$  and  $\mu'(n)$ . Let us prove the following:

**Proposition 4.18.** *There is an online algorithm which, for every collection of intervals presented from left to right, computes a coloring of the related overlap graph with at most  $(\chi M)/m$ , where  $M$  is the maximum length of intervals,  $m$  is the minimum length and  $\chi$  is the chromatic number of the related graph.*

*Proof.* We prove Proposition 4.18 by giving such an algorithm: Algorithm 15. It uses several color boxes and makes a mapping  $\mathbb{R} \rightarrow \mathbb{N} : C(x) = C_i$ . It runs as follows: for each new Interval  $I_t$ , it applies First-Fit with the color box  $C(l_t)$  while there is no inclusion. Whenever an inclusion arises for Interval

$I_t$ , then it changes  $C(x)$  for all points  $x$  such that

$$x > \left( \max_{\{I_h: l_h < l_t\}} l_h \right) + 1 \quad (4.8)$$

---

**Algorithm 15**


---

**Input:** An overlap graph  $G$  presented online from left to right.

**Output:** A  $\chi M/m$ -coloring of  $G$ , where  $M$  (respectively  $m$ ) is the length of the longest (respectively shortest) interval of  $G$ .

- 1:  $\forall i \in \mathbb{N}$ , let  $C_i$  be a box of ordered colors such that the boxes  $C_i$  are disjoint from each other.
  - 2:  $M \leftarrow +\infty$
  - 3:  $m \leftarrow 0$
  - 4:  $i \leftarrow 0$
  - 5:  $C(x) \leftarrow C_i \forall x \in \mathbb{R}$
  - 6: **for** each new interval  $I_t = [l_t, r_t]$  **do**
  - 7:   **if**  $I$  is shorter than  $m$  or longer than  $M$  **then** update  $M$  and  $m$  accordingly.
  - 8:   **if**  $I$  is included in one or more other intervals  $\{I_p = [l_p, r_p]\}$  **then**
  - 9:     Let  $\mathcal{I}$  be this set of intervals and sort  $I_p \in \mathcal{I}$  in increasing order of  $l_p$ .
  - 10:      $i \leftarrow (i + 1) \bmod \lceil M/m \rceil$
  - 11:     **for each**  $I_p \in \mathcal{I}$  **do**
  - 12:       Let  $c$  be the color of  $I_p$ .
  - 13:       **if**  $I_t$  does not overlap any interval colored with  $c$  **then**
  - 14:         Assign  $c$  to  $I_t$
  - 15:          $C(x) \leftarrow C_i$  for all  $x \in [l_p + 1, +\infty)$
  - 16:         Break; /\*goto 19\*/
  - 17:       **end if**
  - 18:     **end for**
  - 19:     **if**  $I_t$  has not been assigned a color yet **then**
  - 20:       Color  $I_t$  with the smallest possible color in  $C(l_t)$
  - 21:        $C(x) \leftarrow C_i \forall x > (\max_{\{I_h: l_h < l_t\}} l_h) + 1$
  - 22:     **end if**
  - 23:   **else**
  - 24:     Assign the smallest possible color of  $C(l_t)$  to  $I_t$ .
  - 25:   **end if**
  - 26: **end for**
- 

**Claim 4.19.** *Between two updates of  $i$ , Algorithm 15 is exact.*

*Proof.* If there is no relation of inclusion, then an overlap graph is equivalent to an interval graph and, as mentioned before, First-Fit (and therefore also Algorithm 15) gives an exact coloring.

Consider the first time that an inclusion occurs (a new interval  $I$  appears, and  $I$  is included in some other interval).  $I$  will have the same color as one of the intervals in which  $I$  is included, and the coloring is still optimal. For all intervals appearing after  $I$  but before the next update of  $i$ , one can notice that they all share at least one point with  $I$ , which is the time where  $i$  is updated. Therefore, this subgraph respects the midnight condition, and is thus a permutation graph, and is thus colored exactly with First-Fit (and therefore also Algorithm 15).  $\square$

Since  $i$  is updated at most after a length of 1, for any interval  $I_t$  we have

$$|\{C(x) : x \in I_t\}| \leq \frac{M}{m} \quad (4.9)$$

Therefore, one interval may never cross two different intervals for which  $i$  has the same value. Thus, the coloring of Algorithm 15 is a juxtaposition of  $M/m$  exact colorings of subgraphs of  $G$  and therefore, the number of colors used by Algorithm 15 is at most  $\chi M/m$ . This ends the proof of Proposition 4.18.  $\square$

The aim of the rest of this chapter is to provide an improved algorithm for the case where  $l \geq 1$ . We first propose an improved version of Proposition 4.18:

**Proposition 4.20.** *Consider an online sequence of intervals  $I_t = [l_t, r_t]$  such that the sequence  $(l_t)_t$  is non decreasing. Then, Algorithm 15 computes a coloring of the related overlap graph with at most  $\lambda$  colors, where*

$$\lambda = \chi \max_t \left\{ M(t) / \min_{h \geq t} (r_h - l_h) \right\} \quad (4.10)$$

and  $\chi$  is the chromatic number of the related graph.

*Proof.* Color boxes used by Algorithm 15 are denoted  $C_i, i \in \mathbb{N}$ . As in the proof of Proposition 4.18, it is assumed that Algorithm 15 is optimal for the sub-instance colored with a given color box. Suppose that when interval  $I_t$  is presented, an inclusion occurs and that  $C(l_t) = C_i$ . Let  $I_k = \arg \max_{\{I_h : l_h < l_t\}} l_h$ . Algorithm 15 is conceived in such a way that when the point  $l_k + 1$  is passed, one changes the color box. At this point in time, the right-most point colored with color box  $C_i$  is located at most at  $l_k + M(t_k)$ . Moreover, the intervals are presented in increasing order of their left ends.

So after less than  $M(k)/[\min_{h \geq k}(r_h - l_h)]$  changes, one can use color box  $C_i$  again. So at most

$$\max_t [M(t)/\min_{h \geq t}(r_h - l_h)] \quad (4.11)$$

color boxes are used, which concludes the proof of Proposition 4.20.  $\square$

Let us now show the following Theorem.

**Theorem 4.21.** *Consider online Min Coloring on overlap graphs associated to a collection of intervals of size at least 1 and presented in the increasing order of their left ends. There is an online algorithm guaranteeing a performance ratio of  $\log M (\lceil \log M / \log \log M \rceil + 1)$ , if  $M > M_0$  and of  $2\sqrt{M}$  if  $M \leq M_0$ , where  $M$  is the maximum length of intervals and  $M_0$  is such that  $2\sqrt{M_0} = 3 \log(M_0)$ . For large value of  $M$ , the ratio is  $O(\log^2 M / \log \log M)$ .*

*Proof.* Let us show that Algorithm 16 is eligible for Theorem 4.21. It uses interval sets  $S_i$ ,  $i \in \mathbb{N}$  and color boxes  $C_i$ ,  $i \in \mathbb{N}$ , each containing colors. We assume  $C_i \cap C_j = \emptyset$ ,  $i \neq j$ . At the beginning sets  $S_i$  are all initialized to  $S_i = \emptyset$ .

---

#### Algorithm 16

---

- 1: Let  $M_0$  be such that  $2\sqrt{M_0} = 3 \log(M_0)$
  - 2: **for each** new interval  $I_k = [l_t, r_t]$  **do**
  - 3:   Update the values of  $M(t)$ ,  $\mu(t)$  and  $\mu'(t)$ .
  - 4:   **if**  $M(t) \leq M_0$  **then**
  - 5:     Assign  $I_t$  to  $S_j$ , where  $j$  is either 2 if  $(r_t - l_t) \leq \sqrt{M(t)}$  or 3 if  $(r_t - l_t) > \sqrt{M(t)}$
  - 6:   **else**
  - 7:     Assign  $I_t$  to the set  $S_j$  such that  $\mu(t)^{j-1} < (r_t - l_t) \leq \mu(t)^j$
  - 8:   **end if**
  - 9:   Apply Algorithm 15 to the partial instance defined by  $S_j$
  - 10: **end for**
  - 11: The solution is obtained by the addition of colorings computed on each sub-instance  $S_j$ ,  $j \leq \lceil \mu' \rceil$ . The number of colors used is equal to the sum of the numbers of colors used in boxes  $C_1, \dots, C_{\lceil \mu' \rceil}$
- 

Note first that for  $M \leq M_0$ ,  $2\sqrt{M} \leq \log(M) \lceil \log(M) / \log \log(M) \rceil$  and for  $M > M_0$ ,  $2\sqrt{M} > \log(M) \lceil \log(M) / \log \log(M) \rceil$ .  $M_0$  is approximatively 20,58. Note that, if  $M(t) > M_0$ , then  $j \leq \lceil \mu'(t) \rceil$  and moreover for every  $j > \lceil \mu'(t) \rceil$ , we have  $S_j = \emptyset$ .

**Claim 4.22.** *Let us denote by  $\chi$  the chromatic number of the graph associated to the whole instance.*

- i. If  $M \leq M_0$ , then the number of colors used in box  $C_j, j = 2, 3$  is at most  $\sqrt{M}\chi$ .*
- ii. If  $M > M_0$ , then the number of colors used in box  $C_j, j \leq \lceil \mu'(t) \rceil$  is at most  $\mu\chi$  for  $j \notin \{1; 2\}$  and  $\max(\mu, \sqrt{M_0})\chi$  if  $j \in \{1, 2\}$ .*

*Proof.*

*i.* We apply Proposition 4.20 on the sub-instance defined by  $S_2$  and  $S_3$ . For the sequence of intervals in  $S_2$ , the minimum length is 1 while the maximum length is  $\sqrt{M}$ , consequently at most  $\chi\sqrt{M}$  colors are used (more precisely, Proposition 4.18 could be applied). For the sequence  $S_3$ , it is also straightforward to verify that, for every interval  $I_t$ , the following inequality holds:

$$M(t) / \min_{h \geq t} (r_h - l_h) \leq \sqrt{M} \quad (4.12)$$

*ii.* Let  $j \leq \lceil \mu' \rceil$  and consider the sub-instance defined by  $S_j$ . For intervals in  $S_j$ , the coloring is produced by Algorithm 15, using several color boxes  $C_j^1, \dots, C_j^h, \dots$ , all included in  $C_j$ . We also apply Proposition 4.20 on each  $S_j$ . Let us first consider  $j \notin \{1; 2\}$ , we then have for each  $I_t$ :

$$\max_{h \leq t, I_h \in S_j} (r_h - l_h) / \min_{h \geq t, I_h \in S_j} (r_h - l_h) \leq \mu \quad (4.13)$$

For  $i \in \{1; 2\}$ , we have

$$\max_{h \leq t, I_h \in S_j} (r_h - l_h) / \min_{h \geq t, I_h \in S_j} (r_h - l_h) \leq \max(\sqrt{M_0}; \mu) \leq 1.5\mu \quad (4.14)$$

where the last inequality is due to the definition of  $M_0$ . □

Since there are  $\lceil \mu' \rceil$  interval sets used, the number of colors used for the whole instance is at most  $2\sqrt{M}\chi$  if  $M \leq M_0$  and

$$(\mu(\lceil \mu' \rceil + 1))\chi = \chi[\log M(\lceil \log M / \log \log M \rceil + 1)] \quad (4.15)$$

if  $M > M_0$ , which concludes the proof of Theorem 4.21. □

**Remark 4.23.** *In many train depots, the trains only stay a limited amount of time and this amount of time does not differ very much from one train to the other. For such train depots, the results shown in Proposition 4.18, Proposition 4.20 and Theorem 4.21 are relevant.*

Coloring case	Presentation order	Performance ratio $\rho$
Unbounded	Perfect	$\rho = 1$
	Arbitrary	$\rho = \frac{\chi+1}{2}$
Bounded	Perfect	$\rho = 2 - \frac{1}{\min(b,k)}$
	Arbitrary	$\rho < \frac{b}{2} + \frac{1}{2}$

**Table 4.2:** *Permutation graphs and comparability graphs*

## 4.4 Conclusion

In this chapter, starting from the Online Track Assignment Problem, we have studied online coloring of permutation graphs and of overlap graphs.

Table 4.2 summarizes the results found on permutation graph and comparability graphs in this chapter and in Chapter 3. These results were applied to an online version of the problems described in [CDS07], and gave an optimal solution for the Train Depot and the Big Train Station, both with the Midnight Condition. They showed an upper bound of 2 for the Small Train Station Problem.

The results found on overlap graphs provide bounds for the Train Depot Problem when the Midnight Condition is not fulfilled.

Given the difficulty of online Min Coloring in overlap graphs, in the next chapter, we will consider another subclass of overlap graphs: trees and forests.



# Chapter 5

## Trees, Forests and Split Graphs

This chapter is a natural continuation of the previous chapters because it explores online Min Coloring and online Min Bounded Coloring on two classes of perfect graphs: forests and split graphs.

**Related works.** Online Min Coloring on particular classes of graphs has been the subject of various studies, besides the previous chapters of this thesis. Among others, the following classes have been studied:  $P_5$ -free graphs [KPT95], Interval graphs [BBE<sup>+</sup>03] and  $H$ -free bipartite graphs [BCP06]. In [GL88], Gyarfas and Lehel have studied online Min Coloring on several classes of graphs. In particular, they have given, without proof, an upper bound on the performance ratio of this problem on split graphs, and have shown that the performance ratio of online Min Coloring on trees is not bounded by a constant.

**Main results.** For trees and forests, we prove that the performance ratio of online Min Coloring is  $\frac{1}{2} \log_2(2n)$  and that the performance ratio of First-Fit on online Min Bounded Coloring is  $\rho = 1 + \frac{\lfloor \log_2(b) \rfloor}{\chi^b}$ .

We also show that it is possible to force First-Fit to use  $\log_2(2n)$  colors on a bipartite overlap graph, even if the graph is presented in the form of a family of intervals, in increasing order of the intervals left ends, and even if the corresponding graph is a tree.

For split graphs, we prove that the performance ratio of online Min Coloring is  $1 + \frac{1}{\chi}$  and that the performance ratio of First-Fit on online Min Bounded Coloring is  $\rho = 2 + \frac{1}{\chi^b} - \frac{3}{b}$ .

## 5.1 Trees and Forests

**Definition 5.1** (Tree). *A tree is a connected graph that contains no cycle.*

**Definition 5.2** (Forest). *A forest is a disjoint union of trees.*

Trees and Forests are perfect graphs. They are bipartite. Coloring a tree or a forest offline can be done in linear time. Justification and more information can be found in [BLS99].

It is quite easy to see that every forest is an overlap graph. Since we could not find a proof in the literature, we will show it here.

**Lemma 5.3.** *The class of forests is strictly included in the class of overlap graphs.*

*Proof.* It is easy to see that every overlap graph is not necessarily a forest. Consider the graph  $C_4$  shown on Figure 5.1(a). It can be represented as a set of intervals where each interval corresponds to a vertex in  $C_4$  and two vertices are adjacent if and only if the two corresponding intervals overlap (see Figure 5.1(b)). Of course,  $C_4$  is not a forest, since it contains a cycle.

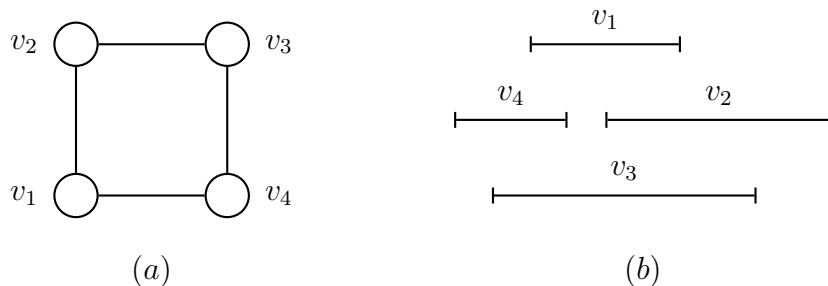
Let us now show that every tree is an overlap graph. Given a tree  $T = (V, E)$ , choose a vertex  $v_0 \in V$ . Draw an interval  $I_0 = [l_0, r_0]$ . For each neighbor  $v_i$  of  $v_0$ , draw an interval  $I_i$  centered on  $r_0$  such that if  $v_i$  and  $v_j$  are neighbors of  $v_0$ ,  $I_i \subset I_j$  or  $I_j \subset I_i$ . Repeat the same construction for each  $v_i$  (See Figure 5.2 for an illustration). The result is a set of intervals such that two intervals overlap if and only if the corresponding vertices are adjacent in  $T$ . Of course, if  $T$  is a forest, it is possible to do the same construction separately to each connected component of  $T$ .  $\square$

In this section, we consider that the trees are presented vertex by vertex. Before the tree is completely presented, the subgraph induced by the presented vertices may not be a tree, but is of course always a forest.

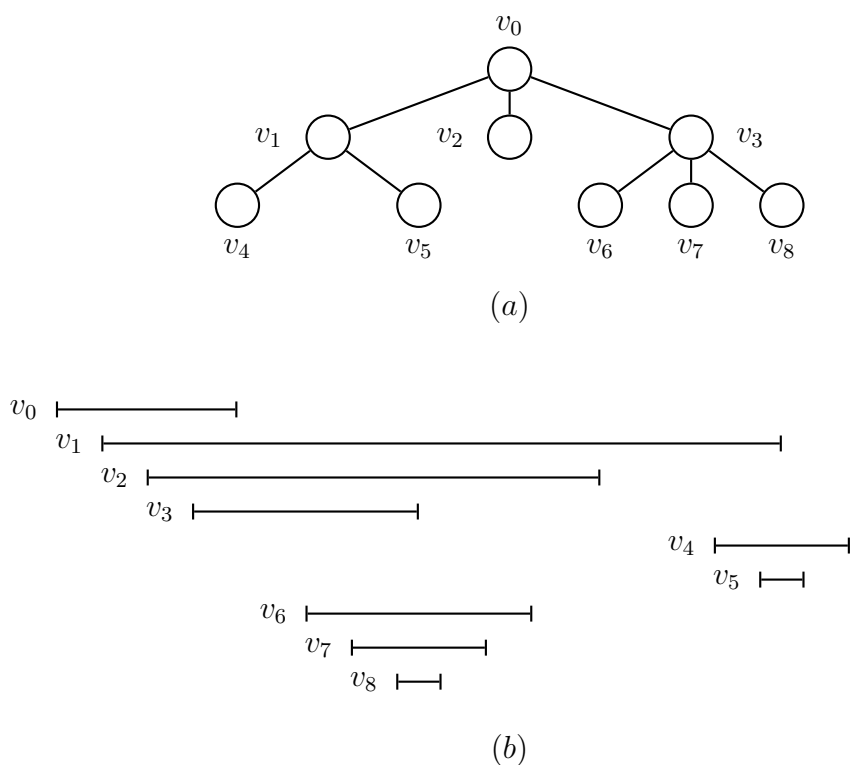
### 5.1.1 Online Coloring of Trees

**Lemma 5.4.** *For  $n \geq 2$ , First-Fit guarantees a performance ratio  $\rho \leq \frac{1}{2} \log_2(2n)$  on online Min Coloring on trees, where  $n$  is the number of vertices of the tree.*

*Proof.* First, remember that every tree of size  $n \geq 2$  is bipartite. Thus, to reach a performance ratio of  $\frac{1}{2} \log_2(2n)$ , one must force First-Fit to use  $\log_2(2n)$  colors, or to use  $\lambda$  colors with at most  $2^{\lambda-1}$  vertices. We prove this by induction on  $\lambda$ .



**Figure 5.1:** Part (a) shows the graph  $C_4$ . Part (b) shows the same graph represented as a set of intervals, showing that  $C_4$  is an overlap graph.



**Figure 5.2:** This figure illustrates the construction used to show that every tree is an overlap graph. Part (a) shows a tree  $T$ . Part (b) shows  $T$  represented under the form of overlapping intervals.

To start the induction, let us note that  $2^1 = 2$  adjacent vertices are enough to force First-Fit to use  $2^{2-1} = 2$  colors. Thus, Lemma 5.4 is true for  $\lambda = 2$ .

Suppose now that Lemma 5.4 holds for  $\lambda = k$ . Let us show that it holds for  $\lambda = k + 1$ .

Suppose that there exists a tree  $T = (V, E)$  with  $|V| < 2^k$  and an ordering on the vertices such that First-Fit uses  $k + 1$  colors on  $T$ . Let  $v_{k+1}$  be the vertex colored with color  $k + 1$ .  $v_{k+1}$  must be adjacent to a vertex colored with color  $k$ ; let  $v_k$  be this vertex. Now consider the forest  $T' = (V, E')$  where  $E' = E - \{v_k v_{k+1}\}$ . Presenting  $T'$  using the same ordering on  $V$  will result on  $v_{k+1}$  being colored with color  $k$ . Thus, we have two trees colored with  $k$  colors each and by our induction hypothesis, each of these trees has at least  $2^{k-1}$  vertices, which makes a total of  $2^k$  vertices and constitutes a contradiction.  $\square$

**Lemma 5.5.** *For any  $\lambda \geq 1$ , no algorithm can guarantee to use less than  $\lambda$  colors on a tree of size  $2^{\lambda-1}$ .*

*Proof.* Again, we prove this by induction on  $\lambda$ . Of course, it is easy to build a tree of size  $2^{1-1} = 1$  on which any algorithm will use 1 color. Thus, Lemma 5.5 holds for  $\lambda = 1$ .

Suppose that Lemma 5.5 holds for  $\lambda \leq k$ ; let us show that it holds for  $\lambda = k + 1$ .

Present  $k$  trees  $T_i, i \in [1..k]$ , each one of size  $2^{i-1}$  such that, for each tree, the algorithm will use at least  $i$  colors. By our hypothesis, it is possible to find  $k$  vertices  $v_i$  such that  $v_i \in T_i$  if  $i \neq j$ , then  $v_i$  and  $v_j$  have two different colors. Present a new vertex  $v_0$  and make it adjacent to each  $v_i, i \in [1..k]$ . The vertices  $v_i, i \in [0..k]$  have  $k + 1$  different colors. Besides that, the total number of vertices  $n$  is:

$$n = 1 + \sum_{i=1}^k 2^{i-1} = 2^k \tag{5.1}$$

$\square$

Lemma 5.5 shows that no algorithm can guarantee a performance ratio better than  $\rho = \frac{\lambda}{2} = \frac{1}{2} \log_2(2n)$ . Putting this information together with Lemma 5.4 allows us to state Theorem 5.6.

**Theorem 5.6.** *Let  $n$  be the number of vertices in a tree. The performance ratio of online Min Coloring on trees containing at least 2 vertices is  $\frac{1}{2} \log_2(2n)$  and First-Fit achieves this performance.*

A forest is the disjoint union of trees. First-Fit, applied on a forest, will behave on each tree independently from the other trees. Thus, the upper bound given by Lemma 5.4 for trees also holds for forests. Since the class of trees is a subclass of the class of forests, the lower bound given by Lemma 5.5 for trees clearly also holds for forests. Therefore,

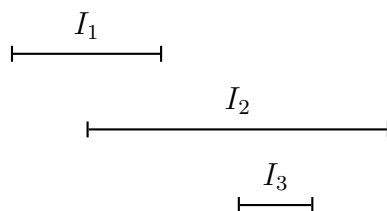
**Corollary 5.7.** *If at least one of the trees of a forest contains at least 2 vertices, then the performance ratio of online Min Coloring on this forest is  $\frac{1}{2} \log_2(2n)$ , where  $n$  is the number of vertices of the forest. First-Fit achieves this performance.*

As mentioned above, trees are overlap graphs. Section 4.3 studies overlap graphs presented in increasing order of the left ends of the intervals. Let us now see what happens if a tree is presented under the form of overlapping intervals.

**Proposition 5.8.** *It is possible to force First-Fit to use  $\log_2(2n)$  colors on a bipartite overlap graph, even if the graph is presented in the form of a family of intervals, in increasing order of the intervals left ends, and even if the corresponding graph is a tree.*

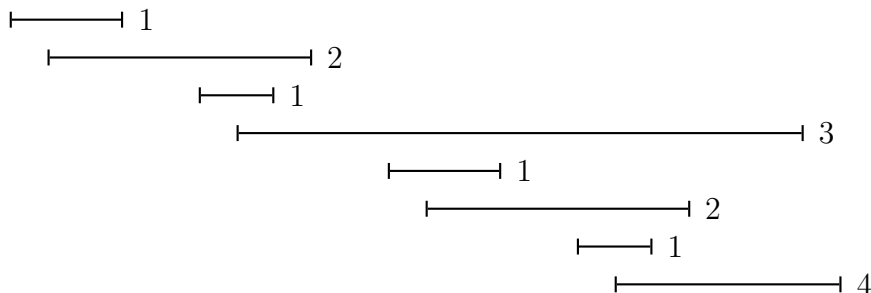
*Proof.* For each interval  $I_i$ , let  $l_i$  (respectively  $r_i$ ) be its left (respectively right) bound. Formally,  $I_i = [l_i, r_i]$ .

It is easy to build a stable set of size two, such that the intervals forming this stable set are included in each other and First-Fit will put these intervals in two different color-classes (see Figure 5.3). One can then add an interval  $I_4$  that overlaps both intervals of the stable set and will get color 3. Thus, Proposition 5.8 holds for  $n = 4$ .



**Figure 5.3:** *This figure shows a stable set of size two (intervals  $I_2$  and  $I_3$ ), such that the intervals forming this stable set are included in each other and First-Fit will put these intervals in two different color-classes.*

It is then possible to repeat the construction of Figure 5.3 in the interval  $[\max\{r_2, r_3\}, r_4]$  and thus getting a stable set  $\Sigma$  of size 3, such that First-Fit will put each interval of  $\Sigma$  in a different color-class.



**Figure 5.4:** *Illustration of the proof of Proposition 5.8 for the case  $n = 8$ . The numbers represent the colors attributed by First-Fit.*

To make First-Fit use color  $\lambda = k$ , it is enough to repeat this construction  $k - 2$  times. Each time, the number of vertices doubles. Since  $n = 4 = 2^{3-1}$  for  $\lambda = 3$ , we have  $n = 2^{k-1}$  for  $\lambda = k$ . Each step presents two trees and one vertex which is connected to both trees. Thus, the resulting graph is also a tree.  $\square$

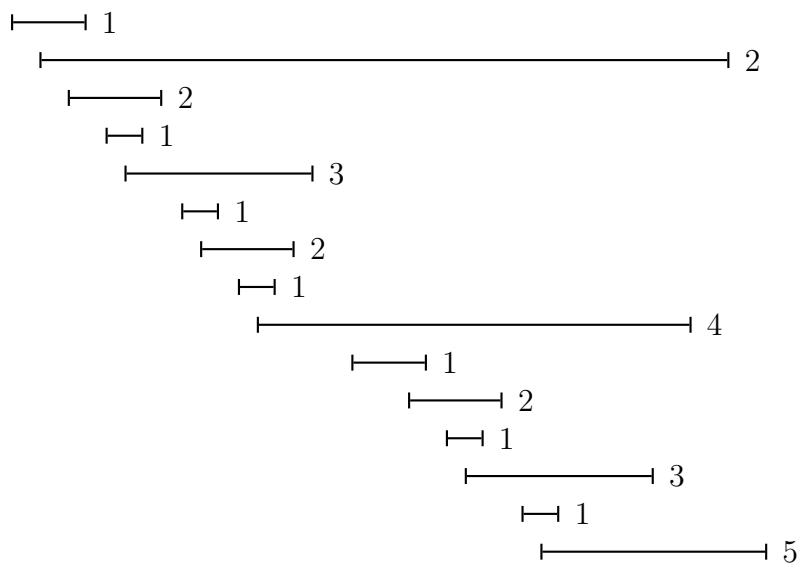
**Remark 5.9.** *Lemma 4.14 shows that it is possible to achieve  $O(\sqrt{n})$  colors on a bipartite overlap graph, which is larger than  $\log_2(2n)$  for large values of  $n$ . However, the graph used in the proof of this lemma is not a tree since it contains a cycle. Theorem 5.6 shows that it is not possible to achieve the result of Lemma 4.14 on a tree.*

**Remark 5.10.** *The number of vertices used to show Proposition 5.8 is much smaller than to number of vertices used to prove Theorem 4.16. Besides that, one can see that the construction used to show Theorem 4.16 is also a tree. However, Theorem 4.16 holds for any algorithm.*

**Remark 5.11.** *Proposition 5.8 does not generalize to overlap bipartite graphs. Indeed, Figure 5.5 shows such a graph with  $15 < 2^{5-1}$  vertices that First-Fit will color with 5 colors.*

**Remark 5.12.** *When presenting a tree with  $n = 2^\lambda$  vertices such that First-Fit will use  $\lambda + 1$  colors using the method shown in Theorem 5.6, the color that will be used the most often by First-Fit is the color 1. It will be used  $2^\lambda - 1$  times.*

Remark 5.12 will be useful in the following section.



**Figure 5.5:** *Illustration of Remark 5.11. The numbers represent the colors attributed by First-Fit.*

### 5.1.2 Online Bounded Coloring of Trees

For any bounded coloring of a tree using  $\lambda$  colors, let  $S$  be the set of color-classes containing exactly  $b$  vertices (*saturated colors*) and  $\bar{S}$  be the set of color-classes containing strictly less than  $b$  vertices (*unsaturated colors*). Define  $N_S = |S|$  and  $N_{\bar{S}} = |\bar{S}|$ . Of course,  $\lambda = N_S + N_{\bar{S}}$ . Let  $V_S$  be the set of vertices colored with saturated colors and  $V_{\bar{S}}$  be the set of vertices colored with unsaturated colors. Clearly,  $V_S \cap V_{\bar{S}} = \emptyset$  and  $V_S \cup V_{\bar{S}} = V$ .

Furthermore, let  $G_{\bar{S}} = (V_{\bar{S}}, E_{\bar{S}})$  be the subgraph of  $G$  induced by  $V_{\bar{S}}$ . Note that  $G_{\bar{S}}$  is a tree or a forest.

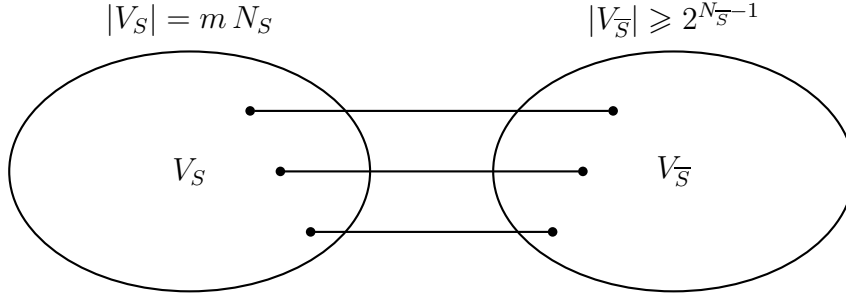
**Proposition 5.13.** *Let  $G = (V, E)$  be a tree and let  $\Omega$  be an ordering on  $V$  such that  $FF_b$  uses  $\lambda = N_S + N_{\bar{S}}$  on  $G$ . If  $N_{\bar{S}} > \log_2(b) + 1$  then it is possible to find another tree  $G'$  and an other ordering  $\Omega'$  such that  $FF_b$  will use exactly  $N_S + 1$  saturated colors and  $N_{\bar{S}} - 1$  unsaturated colors on  $G'$ .*

*Proof.* Suppose that  $N_{\bar{S}} \geq \log_2(b) + 2$ . By Theorem 5.6, we know that, to make  $FF_b$  use  $N_{\bar{S}}$  colors,  $V_{\bar{S}}$  must contain at least  $2^{N_{\bar{S}}-1}$  vertices.

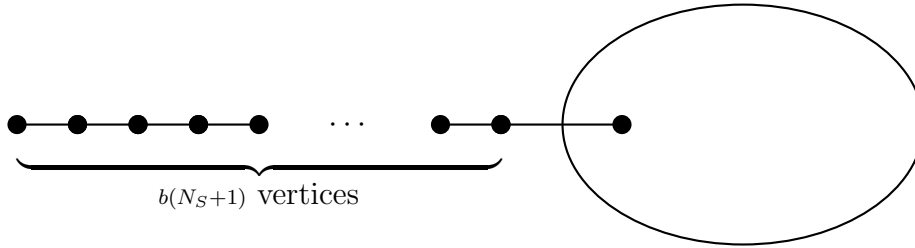
Besides that, by the same theorem, there exists a tree with  $2^{N_{\bar{S}}-2}$  vertices and an ordering on the vertices such that  $FF_b$  will use  $N_{\bar{S}} - 1$  on this tree.

However,

$$2^{N_{\bar{S}}-1} - 2^{N_{\bar{S}}-2} = 2^{N_{\bar{S}}-2} < 2^{\log_2(b)} = b \quad (5.2)$$



**Figure 5.6:** Illustrates of the proof of Proposition 5.13. As for every tree,  $bN_S + 2^{N_{\bar{S}}-1} \leq \chi_b b$ . It is then possible to build the tree shown in Figure 5.7.



**Figure 5.7:** The ellipse represents a tree with  $2^{(N_{\bar{S}}-2)}$  vertices that  $FF_b$  colors with  $N_{\bar{S}} - 1$  colors.

It is thus possible to build a new tree on which  $FF_b$  will use  $N_S + 1$  saturated colors and  $N_{\bar{S}} - 1$  unsaturated colors. Indeed, the fact that  $2^{N_{\bar{S}}-1} - 2^{N_{\bar{S}}-2} < b$  allows us to saturate one more color.  $\square$

Figures 5.6 and 5.7 illustrate the proof of Proposition 5.13. In Figure 5.7, the total number  $n$  of vertices of the tree is:

$$n = b(N_S + 1) + 2^{(N_{\bar{S}}-2)} = bN_S + b + (2^{(N_{\bar{S}}-1)} - 2^{(N_{\bar{S}}-2)}) \quad (5.3)$$

$$= bN_S + 2^{(N_{\bar{S}}-1)} + (b - 2^{(N_{\bar{S}}-2)}) \quad (5.4)$$

Since  $b - 2^{(N_{\bar{S}}-2)} \leq 0$  by hypothesis, we have that:

$$n \leq \chi_b b \quad (5.5)$$

**Lemma 5.14.**  $FF_b$  guarantees a performance ratio of at most  $1 + \frac{\lfloor \log_2(b) \rfloor}{\chi_b}$  on online Min Bounded Coloring on trees.



*Proof.* Let us calculate  $\lambda$ , the number of colors that  $FF_b$  uses on a tree. We know that if  $N_{\overline{S}} > \lfloor \log_2(b) \rfloor + 2$ , it is possible to find another coloring of the same tree such that the value of  $\lambda$  does not change and  $N_{\overline{S}} \leq \lfloor \log_2(b) \rfloor + 2$ . Since it is  $\lambda$  that we want to compute, we can suppose that  $N_{\overline{S}} \leq \lfloor \log_2(b) \rfloor + 2$ . Two cases are possible:

1.  $N_{\overline{S}} = \lfloor \log_2(b) \rfloor + 2$ . We know that  $|V_{\overline{S}}| \geq 2^{N_{\overline{S}}-1}$ . However,

$$b < 2^{N_{\overline{S}}-1} = 2^{\lfloor \log_2(b) \rfloor + 1} \leq 2b \quad (5.6)$$

$|V_S| \leq \chi_b b - |V_{\overline{S}}|$ . Thus,

$$(\chi_b - 2)b \leq |V_S| < (\chi_b - 1)b \quad (5.7)$$

But since  $N_S = \frac{|V_S|}{b}$ , we have  $N_S = \chi_b - 2$ .

2.  $N_{\overline{S}} \leq \lfloor \log_2(b) \rfloor + 1$ .  $|V_{\overline{S}}| \geq 2^{N_{\overline{S}}-1}$ . But

$$0 < 2^{N_{\overline{S}}-1} = 2^{\lfloor \log_2(b) \rfloor} \leq b \quad (5.8)$$

$|V_S| \leq \chi_b \cdot b - |V_{\overline{S}}|$ . Therefore,

$$(\chi_b - 1)b \leq |V_S| < \chi_b b \quad (5.9)$$

But since  $N_S = \frac{|V_S|}{b}$ , we have that  $N_S \leq \chi_b - 1$ .

Finally, in both cases,

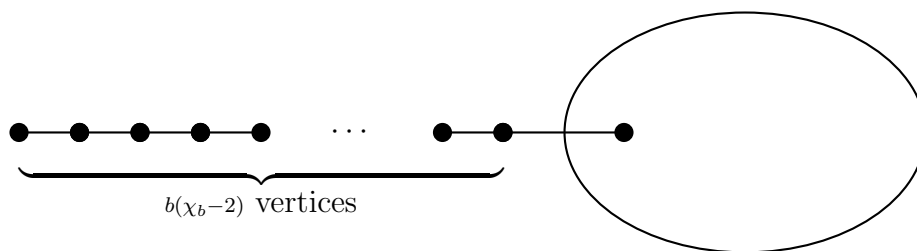
$$N_S + N_{\overline{S}} \leq \lfloor \log_2(b) \rfloor + \chi_b \quad (5.10)$$

□

**Lemma 5.15.**  *$FF_b$  has a performance ratio of at least  $1 + \frac{\lfloor \log_2(b) \rfloor}{\chi_b}$  on online Min Bounded Coloring on trees.*

*Proof.* The tree shown on Figure 5.8 forces  $FF_b$  to use  $\chi_b + \lfloor \log_2(b) \rfloor$  colors. Again, let  $G_{\overline{S}}$  the the partial graph represented by the ellipse on Figure 5.8. We know by Theorem 5.6 that such a tree exists. Furthermore, we know by Remark 5.12 that the most used color on  $G_{\overline{S}}$  will be used  $2^{\lfloor \log_2(b) \rfloor} \leq b$  times. □

**Remark 5.16.** *It is easy to see that lemma 5.15 is true for any greedy algorithm that opens a new color only when no other color can be used on a vertex.*



**Figure 5.8:** The ellipse represents a tree with  $2^{\lfloor \log_2(b) \rfloor + 1}$  vertices such that First-Fit (and thus also  $FF_b$ ) needs  $\lfloor \log_2(b) \rfloor + 2$  color-classes to cover it.

Lemmas 5.14 and 5.15 allow us to make the following proposition:

**Proposition 5.17.**  $FF_b$  has a performance ratio of exactly  $1 + \frac{\lfloor \log_2(b) \rfloor}{\chi_b}$  on online Min Bounded Coloring on trees.

**Proposition 5.18.** The performance ratio of  $FF_b$  on an overlap graph presented as a set of intervals in increasing order of their left ends is at least  $1 + \frac{\lfloor \log_2(b) \rfloor}{\chi_b}$ .

*Proof.* Use the construction shown on Figure 5.8 and use Proposition 5.8 to build a graph with less than  $2b$  vertices on which  $FF_b$  will use  $\lfloor \log_2(b) \rfloor + 2$  colors.  $\square$

## 5.2 Split Graphs

**Definition 5.19.** A split graph is a graph in which the vertices can be partitioned into a clique and a stable set.

Split graphs were first introduced by Földes and Hammer in two papers in 1977 [FH77a, FH77b], and independently by Tyshkevich and Chernyak in 1979 [TC79]. They are chordal graphs the complement of which is also a chordal graph [FH77a]. Split graphs which are also permutation graphs have been studied in Section 3.3.2. More on split graphs can be found in [BLS99].

### 5.2.1 Online Coloring of Split Graphs

**Lemma 5.20.** First-Fit guarantees to use at most  $\chi(G) + 1$  colors on online Min Coloring on a split-graph  $G = (V, E)$ . Moreover, in order to force First-Fit to use  $\chi(G) + 1$  colors, one needs to present at least  $\chi(G) + 2$  vertices.

*Proof.* Let  $K$  be the maximum clique of  $G$  and  $\Sigma$  be the stable set induced by the vertices of  $G$  which are not in  $K$ .

Let us prove the first part of the theorem by contradiction. Suppose that First-Fit has colored  $G$  with strictly more than  $\chi(G) + 1$  colors. The vertices of  $K$  are of course colored with  $|K| = \omega(G) = \chi(G)$  colors. So there are at least two colors used on  $\Sigma$  which do not appear in  $K$ . Let  $c_1$  and  $c_2$  be these colors, with  $c_1 < c_2$ . Also, let  $v_1$  (respectively  $v_2$ ) be a vertex colored with  $c_1$  (respectively  $c_2$ ). The vertex  $v_2$  should be colored with  $c_1$  by First-Fit since no vertex of  $K$  is colored with  $c_1$  and  $v_1$  and  $v_2$  are in  $\Sigma$ , thus are not adjacent. Thus, there is a contradiction.

Let us now prove the second part of the theorem, by contradiction again. Suppose that First-Fit colors a split graph of size  $\chi(G) + 1$  with  $\chi(G) + 1$  colors. Since each color appears exactly once in the coloring, there must be a clique of size  $\chi(G) + 1$ , thus  $\omega(G) > \chi(G)$ , which is a contradiction.  $\square$

**Lemma 5.21.** *For all  $\chi \geq 2$ , no online algorithm can guarantee to use less than  $\chi(G) + 1$  colors to color a split graph, where  $\chi(G) = \chi$ , even if the size  $n$  of  $G$  is smaller or equal to  $\chi + 2$ .*

*Proof.* Consider the instance presented by Adversary 17.

If  $A$  never uses color  $c_0$  at Step 5, then we have one clique of size  $k$  and one disjoint vertex, and  $A$  has used  $k + 1$  colors. Thus, the chromatic number of the graph is  $\chi(G) = k$  and the number of colors is  $\chi(G) + 1$ . Besides that, the size of the graph is  $n = \chi(G) + 1$ .

If  $A$  uses the color  $c_0$  at a certain point in time at Step 5, then Adversary 17 still presents a clique  $K$  of size  $k$ . However, since each vertex of  $K$  is adjacent to either  $v_0$  or  $v_i^*$ , no vertex of  $K$  can have color  $c_0$ . Furthermore, the condition at Step 12 ensures that there is in  $K$  at least one vertex that is not adjacent to  $v_{i^*}$  and at least one vertex that is not adjacent to  $v_0$ . Therefore,  $G$  contains no clique of size larger than  $k$  and thus  $\chi(G) = \omega(G) = k$  and the number of colors is  $k + 1 = \chi(G) + 1$ . The number of vertices is  $n = \chi(G) + 2$ .  $\square$

**Theorem 5.22.** *The performance ratio of the problem of online coloring split graphs is exactly  $1 + \frac{1}{\chi}$  and First-Fit achieves this ratio. Furthermore, a split graph with  $\chi(G) + 2$  vertices is enough to force any online algorithm to this ratio.*

*Proof.* The proof is immediate from lemmas 5.20 and 5.21.  $\square$

**Adversary 17**

---

**Input:** An online coloring algorithm  $A$  and a natural number  $\chi \geq 2$ .**Output:** A split graph  $G$  of size at most  $\chi + 2$  and with chromatic number  $\chi$  that  $A$  will color with at least  $\chi + 1$  colors.

- 1: Present a vertex  $v_0$ . Let  $c_0$  be the color of  $v_0$ .
  - 2:  $i \leftarrow 0$
  - 3: **repeat**
  - 4:    $i \leftarrow i + 1$
  - 5:   Present a vertex  $v_i$  and an edge  $v_i v_j \forall j \in [1..(i - 1)]$ .
  - 6:   **until**  $A$  puts  $v_i$  in  $c_0$  or  $i = \chi$ .
  - 7:   **if**  $i < \chi$  **then** /\*Vertex  $v_i$  is colored with  $c_0$ \*/
  - 8:     Let  $i^* = i$
  - 9:     **repeat**
  - 10:        $i \leftarrow i + 1$
  - 11:       Present a vertex  $v_i$ , and an edge  $v_i v_j \forall j \in [1..(i - 1)], j \neq i^*$ . Furthermore,
  - 12:       **if**  $(i \bmod 2) = (i^* \bmod 2)$  **then**
  - 13:         Present an edge  $v_i v_{i^*}$
  - 14:       **else**
  - 15:         Present an edge  $v_i v_0$
  - 16:       **end if**
  - 17:     **until**  $i = \chi$
  - 18: **end if**
- 

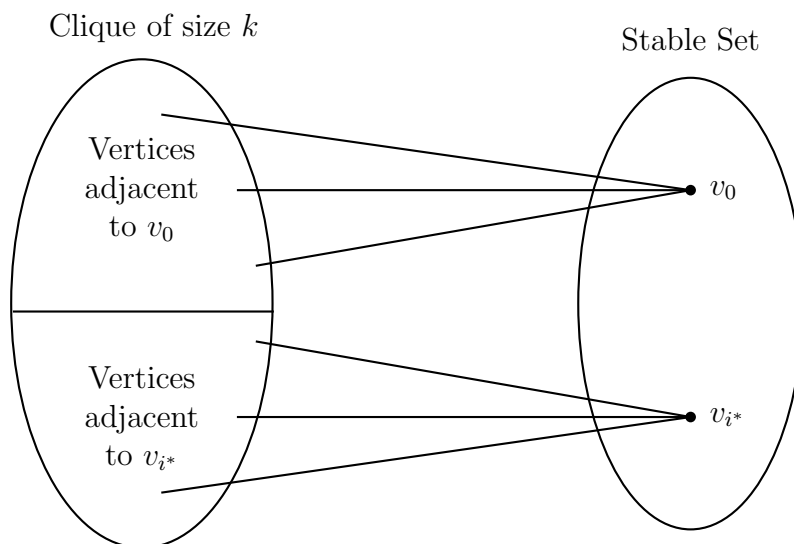
**5.2.2 Online Bounded Coloring of Split Graphs**

For any bounded coloring of a split graph using  $\lambda$  colors, let  $S$  be the set of colors containing exactly  $b$  vertices (*saturated colors*) and  $\overline{S}$  be the set of colors containing strictly less than  $b$  vertices (*unsaturated colors*). Define  $N_S = |S|$  and  $N_{\overline{S}} = |\overline{S}|$ . Of course,  $\lambda = N_S + N_{\overline{S}}$ . Let  $V_S$  be the set of vertices colored with saturated colors and  $V_{\overline{S}}$  be the set of vertices colored with unsaturated colors. Clearly,  $V_S \cap V_{\overline{S}} = \emptyset$  and  $V_S \cup V_{\overline{S}} = V$ .

Furthermore, let  $G_{\overline{S}} = (V_{\overline{S}}, E_{\overline{S}})$  be the subgraph of  $G$  induced by  $V_{\overline{S}}$ . Note that  $G_{\overline{S}}$  is a split graph.

**Theorem 5.23.** *Let  $G = (V, E)$  be a split graph.  $FF_b$  will use at most  $\left\lfloor \frac{\chi_b(b-1)-2}{b} \right\rfloor + \chi_b + 1$  color-classes to cover it and this bound is tight.*

*Proof.* We will prove Theorem 5.23 by contradiction. Suppose that  $FF_b$  uses more than  $\left\lfloor \frac{\chi_b(b-1)-2}{b} \right\rfloor + \chi_b + 1$  color-classes to cover a split graph  $G$  presented



**Figure 5.9:** *Illustration of the instance presented by Adversary 17.*

online. Let

$$\lambda = N_S + N_{\overline{S}} = \left\lfloor \frac{\chi_b(b-1) - 2}{b} \right\rfloor + \chi_b + 1 + x \quad (5.11)$$

where  $x \geq 1$ .

Since  $G_{\overline{S}} = (V_{\overline{S}}, E_{\overline{S}})$  is a split graph, we know by Lemma 5.20 that  $N_{\overline{S}} \leq \chi_b + 1$ . Thus, we can deduce from (5.11) that:

$$N_S \geq \left\lfloor \frac{\chi_b(b-1) - 2}{b} \right\rfloor + x \quad (5.12)$$

Suppose that

$$N_S = \left\lfloor \frac{\chi_b(b-1) - 2}{b} \right\rfloor + y, \quad (5.13)$$

where  $y \geq x$ . There are two possibilities:

1. If  $y > 1$ .

$$|V_S| = \left\lfloor \frac{\chi_b(b-1) - 2}{b} \right\rfloor b + yb > \chi_b(b-1) - 2 + (y-1)b \quad (5.14)$$

Since  $G$  has at most  $\chi_b b$  vertices,

$$|V_{\overline{S}}| \leq \chi_b b - |V_S| < \chi_b b - (\chi_b(b-1) - 2 + (y-1)b) \quad (5.15)$$

$$\leq \chi_b + 2 + b - yb \quad (5.16)$$

Moreover,  $N_{\overline{S}} \leq |V_{\overline{S}}|$ . Thus,

$$N_S + N_{\overline{S}} < \left\lfloor \frac{\chi_b(b-1) - 2}{b} \right\rfloor + y + \chi_b + 2 + b - yb \quad (5.17)$$

Thus

$$N_S + N_{\overline{S}} \leq \left\lfloor \frac{\chi_b(b-1) - 2}{b} \right\rfloor + \chi_b + 1 + \underbrace{(y + b - yb)}_{\leq 0 \text{ since } y, b > 1}. \quad (5.18)$$

which contradicts the hypothesis.

2. If  $y = 1$ ,

$$|V_S| > \chi_b(b-1) - 2. \quad (5.19)$$

Since  $G$  has at most  $\chi_b b$  vertices,

$$|V_{\overline{S}}| \leq \chi_b b - |V_S| < \chi_b + 2 \quad (5.20)$$

By Theorem 5.20,  $N_{\overline{S}} \leq \chi_b$ . Therefore,

$$N_S + N_{\overline{S}} \leq \left\lfloor \frac{\chi_b(b-1) - 2}{b} \right\rfloor + \chi_b + 1 \quad (5.21)$$

which contradicts the hypothesis.

This bound can be reached by presenting a stable set of size  $\lfloor \frac{\chi_b(b-1)-2}{b} \rfloor b$  and then a split graph of size  $\chi + 2$  that First-Fit will color with  $\chi + 1$  colors (see Theorem 5.21 for the construction).  $\square$

**Remark 5.24.** *Theorem 5.23 shows that the performance ratio  $\rho$  of  $FF_b$  on the problem of online coloring a split graph with colors of bounded size  $b$  is:*

$$\rho \leq 2 + \frac{1}{\chi_b} - \frac{3}{b}$$

## 5.3 Conclusion

Tables 5.1 and 5.2 summarize the main results that were proved in this chapter. In the bounded cases, the upper bounds are guaranteed by  $FF_b$  and are tight.

Moreover, we have shown that it is possible to force First-Fit to use  $\log_2(2n)$  colors on a bipartite overlap graph, even if the graph is presented in the form of a family of intervals, in increasing order of the intervals left ends, and even if the corresponding graph is a tree.

Coloring case	$\rho$
Unbounded	$\frac{1}{2} \log_2(2n)$
Bounded	$\rho \leq 1 + \frac{\lfloor \log_2(b) \rfloor}{\chi_b}$

**Table 5.1:** *Performance ratio  $\rho$  of online Min Coloring on trees and forests*

Coloring case	$\rho$
Unbounded	$1 + \frac{1}{\chi}$
Bounded	$\rho \leq 2 + \frac{1}{\chi_b} - \frac{3}{b}$

**Table 5.2:** *Performance ratio  $\rho$  of online Min Coloring on split graphs*





## Chapter 6

# Finding Hamiltonian Circuits in Quasi-Adjoint Graphs

This chapter is based on [BKLbW08]. It is motivated by a method used for DNA sequencing by hybridization presented by Blazewicz and Kasprzak in [BK06]. In this paper, the authors develop a formulation involving the search of a Hamiltonian path in order to solve a problem of DNA sequencing. They exhibit cases where the problem can be solved in polynomial time. The graphs they use are a generalization of directed line graphs. It is interesting to examine how we can generalize those graphs, while still being able to solve the Hamiltonian Path or Circuit Problem polynomially. We give here a characterization of quasi-adjoint graphs, and devise a polynomial algorithm for finding a Hamiltonian circuit.

**Related works.** DNA sequencing problems have been widely studied and, in particular, various formulations have been given in terms of combinatorial optimization of graph theoretical flavor (see references in [BK03]).

The problem of deciding whether a graph has a Hamiltonian circuit (for short, the Hamiltonian Circuit Problem) is known for a long time to be NP-complete [Kar72].

The Hamiltonian Circuit Problem remains NP-complete even for graphs having a specific structure, such as planar-cubic 3-connected graphs [GJS76], bipartite planar graphs of maximum degree 3 [ANS80], grid graphs [IPS82], maximal planar graphs [Chv85], chordal bipartite graphs and strongly chordal split-graphs [Mül96] as well as line graphs [Ber81].

However, for some other classes of graphs, such as locally connected regular graphs of degree 5 [Kik75], cographs [CLB81], proper circular arc graphs [Ber83], interval graphs [Kei85], co-comparability graphs [DS94] or directed line graphs [BHKdW99], the same problem has been shown to be

polynomially solvable.

Remark that directed line graphs are adjoints and therefore also quasi-adjoint graphs. A different generalization of directed line graphs, the partial directed line graphs, has been studied in 2007 by Apollonio and Franciosa [AF07]. However, these graphs are not related to quasi-adjoint graphs.

**Main results.** We present here a class of digraphs: the quasi-adjoint graphs. These are a super class of both the graphs used in [BK06] and the adjoints [Ber76]. A polynomial recognition algorithm in  $O(n^3)$ , as well as a polynomial algorithm in  $O(n^2 + m^2)$  for finding a Hamiltonian circuit in quasi-adjoint graphs are given. Furthermore, some results about related problems such as finding a Eulerian circuit while respecting some forbidden transitions (a path with three vertices) are discussed.

## 6.1 Definitions and Characterization

Graph theoretical terms not defined here can be found in [Ber76].

Throughout this paper, the symbol  $\subset$  always refers to a *strict* inclusion. If an inclusion is not strict, we will use the symbol  $\subseteq$ .

Furthermore, in the remaining part of this paper, we will only consider directed graphs, even when not explicitly stated.

**Definition 6.1.** For any graph  $G = (V, U)$ , we define  $m = |U|$ . As in previous chapters,  $n$  is the cardinality of  $V$ .

**Definition 6.2.** A subpath of a path  $P$  is a sequence of vertices which are consecutive in  $P$ .

**Definition 6.3.** A transition is a path consisting of 3 vertices and two arcs.

**Definition 6.4.** Let  $G = (V, U)$  be a graph and  $x \in V$ . Define  $N^+(x)$  and  $N^-(x)$  as follows:

$$\begin{aligned} N^+(x) &= \{y \in V \mid (x, y) \in U\} \\ N^-(x) &= \{y \in V \mid (y, x) \in U\} \end{aligned}$$

$N^+(x)$  is called the set of successors of  $x$  and  $N^-(x)$  is called the set of predecessors of  $x$ . For a set  $S$  of vertices,

$$N^+(S) = \cup_{x \in S} N^+(x)$$

**Definition 6.5.** Let  $G = (V, U)$  be a graph and  $x \in V$ . We define the outdegree  $d^+(x)$  (respectively the indegree  $d^-(x)$ ) of a vertex  $x$  as the number of arcs leaving (respectively entering)  $x$ . Formally:

$$\begin{aligned} d^+(x) &= |\{u \in U \mid u = (x, y) \text{ for some } y \text{ of } V\}| \\ d^-(x) &= |\{u \in U \mid u = (y, x) \text{ for some } y \text{ of } V\}| \end{aligned}$$

**Remark 6.6.** Graphs considered in this paper may be multigraphs. Therefore,  $d^+(x)$  may be different from  $|N^+(x)|$  and  $d^-(x)$  may be different from  $|N^-(x)|$ .

**Definition 6.7.** A graph is a quasi-adjoint graph if the family  $(N^+(y) \mid y \in V)$  is nested. In other words, if for any two vertices  $x$  and  $y$  the following property holds:

$$\begin{aligned} N^+(x) \cap N^+(y) \neq \emptyset \Rightarrow & N^+(x) = N^+(y) \text{ or} \\ & N^+(x) \subset N^+(y) \text{ or} \\ & N^+(y) \subset N^+(x) \end{aligned}$$

**Remark 6.8.** Berge [Ber76] gives the following definitions:

A graph is a  $p$ -graph if given any ordered pair  $x, y$  of vertices ( $x$  possibly equal to  $y$ ), there are at most  $p$  parallel arcs from  $x$  to  $y$ .

The adjoint  $G = (V, U)$  of a graph  $H = (X, V)$  is the 1-digraph with vertex set  $V$  and such that there is an arc from a vertex  $x$  to a vertex  $y$  in  $G$  if and only if the terminal endpoint of arc  $x$  in  $H$  is the initial endpoint of arc  $y$  in  $H$ .

A graph  $G$  is an adjoint if there exists some graph  $H$  such that  $G$  is the adjoint of  $H$ .

Berge [Ber76] also proves that a 1-graph  $G = (V, U)$  is the adjoint of a graph if and only if the following holds for any pair  $x, y$  of vertices in  $V$ :

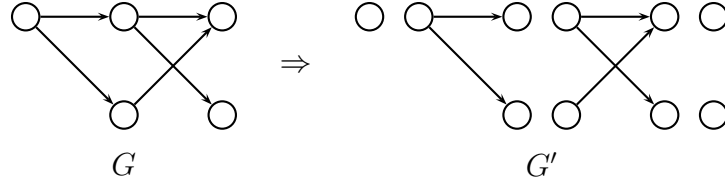
$$N^+(x) \cap N^+(y) \neq \emptyset \Rightarrow N^+(x) = N^+(y)$$

**Remark 6.9.** This statement shows that, by definition, the class of quasi-adjoint graphs strictly contains the class of adjoint graphs.

**Remark 6.10.** Quasi-adjoint graphs can be recognized in  $O(n^3)$  time by looking at every pair of vertices  $x$  and  $y$  and comparing  $N^+(x)$  and  $N^+(y)$ .

The following constructions and definitions will be used for the search of a Hamiltonian circuit in a quasi-adjoint graph  $G$ :

One can construct a new graph  $G'$  by splitting each vertex  $x$  of  $G$  into two new vertices  $x'$  and  $x''$ , and replacing each arc  $(x, y)$  by the arc  $(x'', y')$ . An example of this is given in Figure 6.1.



**Figure 6.1:** A quasi-adjoint graph  $G$  and the result of its decomposition  $G'$ . This figure also illustrates Remark 6.9: while adjoint graphs would only admit complete bipartite graphs after a decomposition, quasi-adjoint graphs also admit some incomplete bipartite graphs.

**Definition 6.11.** Each non-trivial connected component (having at least two vertices) of  $G'$  is called a cluster.

**Remark 6.12.**  $G'$  is a collection of vertex-disjoint bipartite graphs and isolated vertices. By definition, the clusters are the bipartite graphs.

For each cluster  $C$ , we divide its set of vertices into two parts: the *left part*  $L(C)$  is the set of vertices having only outgoing arcs and the *right part*  $R(C)$  is the set of vertices having only incoming arcs.

Note that the clusters resulting from the decomposition of a quasi-adjoint graph are not necessarily complete, as they would be for the adjoint of a graph. See Figure 6.1 for an example. It is possible to group vertices of  $L(C)$  into subsets such that, for any two vertices  $x$  and  $y$  from the same subset,  $N^+(x) = N^+(y)$ . As a direct consequence of the definition of quasi-adjoint graphs (Definition 6.7), each one of these subsets then belongs to one of the following categories:

- A.  $\{x \mid N^+(x) = R(C)\}$  ( $x \in L(C)$ )
- B.  $\{x \mid \exists y, z : N^+(y) \subset N^+(x) \subset N^+(z)\}$  ( $x, y, z \in L(C)$ )
- C.  $\{x \mid \exists z : N^+(x) \subset N^+(z) \text{ and } \nexists y : N^+(y) \subset N^+(x)\}$  ( $x, y, z \in L(C)$ )

**Lemma 6.13.** For every cluster  $C$ , there is at least one vertex  $x \in L(C)$  such that  $N^+(x) = R(C)$ .

*Proof.* Suppose there exists a cluster  $C$  such that there exists no vertex  $x \in L(C)$  with  $N^+(x) = R(C)$ . Consider two disjoint maximal sets  $Y_1$  and  $Y_2$  such that  $Y_i = N^+(x_i)$  for some  $x_i \in L(C)$ . Since the family  $\cup_i Y_i$  is nested, there is no chain going from  $x_1$  to  $x_2$ . Thus, the cluster is disjoint, which is a contradiction.  $\square$

## 6.2 The Hamiltonian Circuit Problem in Quasi-Adjoint Graphs

**Theorem 6.14.** *The Hamiltonian Circuit Problem in quasi-adjoint graphs can be polynomially solved in  $O(n^2 + m^2)$  time.*

*Proof.* We prove this by giving Algorithm 18, which finds a Hamiltonian circuit in a quasi-adjoint graph if there is one and gives a negative answer otherwise.

This algorithm is based on the same construction as the one used for adjoint graphs: transforming graph  $G$  into its original graph  $H$  (such that  $G$  is the adjoint of  $H$ ) and then looking for a Eulerian circuit in  $H$ . However, since clusters of quasi-adjoint graphs are not necessarily complete, as shown in Figure 6.1, we must introduce some artificial vertices to  $H$  to make sure that the one-to-one correspondence between a Eulerian circuit in  $H$  and a Hamiltonian circuit in  $G$  remains. In other words, do not make  $H$  Eulerian if  $G$  was not Hamiltonian, nor do make  $H$  not Eulerian if  $G$  was Hamiltonian.

**Remark 6.15.** *In Algorithm 18, after Step 24, each labeled arc in  $H$  corresponds to a vertex of the same name in  $G$ .*

**Claim 6.16.** *Step 6 of Algorithm 18 constructs a directed tree  $T$  with root  $Y_1$  and the sets of type  $\mathcal{C}$  as leaves.*

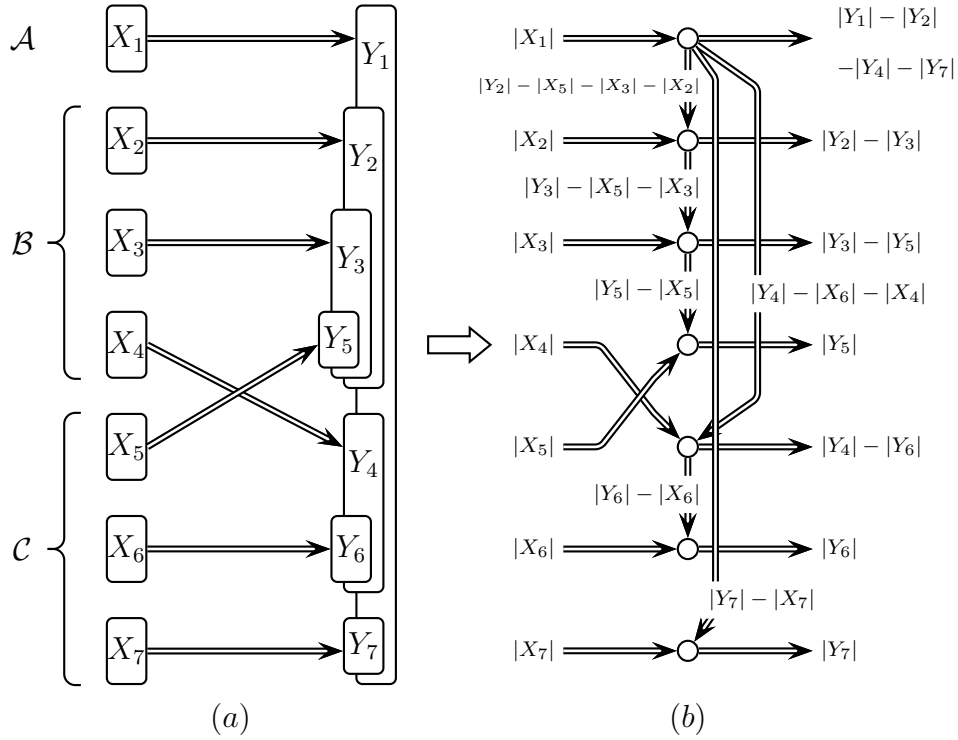
*Proof.* The arcs of  $T$  represent a relation of inclusion: an arc from  $Y_i$  to  $Y_j$  implies that  $Y_j \subset Y_i$ . Furthermore, if  $Y_j \subset Y_i$ , then there is a path from  $Y_i$  to  $Y_j$  in  $T$ . Since  $Y_1$  contains all other sets  $Y_i$ ,  $T$  is connected and  $Y_1$  is a root of  $T$ . Since the sets of type  $\mathcal{C}$  contain no other subsets, they are leaves of  $T$ . Finally, since an arc of  $T$  is a relation of strict inclusion, in the sense that an arc from  $Y_i$  to  $Y_j$  implies that there is no set  $Y_k$  such that  $Y_j \subset Y_k \subset Y_i$ . Thus, if  $Y_j \subset Y_i$ , then there is only one path from  $Y_i$  to  $Y_j$  in  $T$  and therefore  $T$  is a tree.  $\square$

**Claim 6.17.** *At Steps 4 and 16, Algorithm 18 exits only if there is no Hamiltonian circuit in  $G$ .*

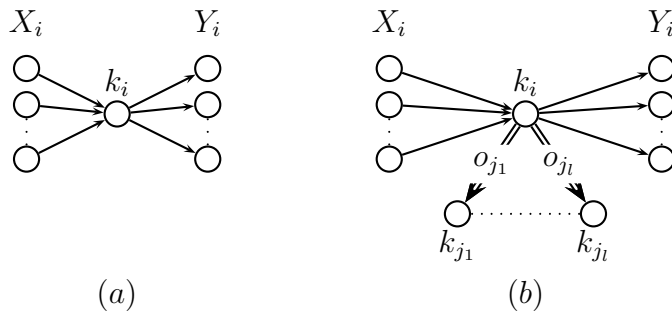
*Proof.* Suppose there is a Hamiltonian circuit in  $G$ . By construction of the cluster  $C$ , the edges that belong to both the Hamiltonian circuit of  $G$  and the cluster  $C$  define a perfect matching in  $C$ . If  $|L(C)| \neq |R(C)|$ ,  $C$  does not admit a perfect matching and therefore,  $G$  does not admit a Hamiltonian circuit. This ends the proof of Claim 6.17 for Step 4.

**Algorithm 18****Input:** A quasi-adjoint graph  $G = (V, U)$ .**Output:** A Hamiltonian circuit in  $G$  or a claim of non-existence of such a circuit.

- 1: Define an empty graph  $H = (V^H, U^H)$  with  $V^H \leftarrow \emptyset$  and  $U^H \leftarrow \emptyset$
- 2: For every  $x \in V$ , introduce two vertices  $x'$  and  $x''$  into  $V^H$ . For every arc  $(x, y) \in U$  such that  $x \neq y$ , introduce the arc  $(x'', y')$  in  $U^H$ .
- 3: **for each** cluster  $C$  of  $H$ , **do**
- 4:   **if**  $|L(C)| \neq |R(C)|$  **then exit.** There is no Hamiltonian circuit in  $G$ .
- 5:   Decompose  $L(C)$  into sets of types  $\mathcal{A}$ ,  $\mathcal{B}$  and  $\mathcal{C}$ . The unique set of type  $\mathcal{A}$  is labeled  $X_1$ . Label all other sets of  $L(C)$  with a unique identifier  $X_i$ . Let  $Y_i \leftarrow N^+(X_i)$ .
- 6:   Sort sets  $Y_i$  according to their inclusion relation: Construct a directed tree  $T = (V^T, U^T)$  with  $V^T = \cup_i \{Y_i\}$  and  $(Y_i, Y_j) \in U^T \Leftrightarrow Y_j \subset Y_i \wedge \nexists k : Y_j \subset Y_k \subset Y_i$ .
- 7:   Delete all arcs of  $C$ .
- 8:   For every leaf  $Y_i$  of  $T$ , introduce a vertex  $k_i$  in  $V^H$ . For each vertex  $x'' \in X_i$ , add the arc  $(x'', k_i)$  to  $U^H$  and for each vertex  $y' \in Y_i$ , add the arc  $(k_i, y')$  to  $U^H$ . Label the vertex  $Y_i$  of  $T$ .
- 9:   **for each** vertex  $Y_i$  of  $T$  not yet labeled such that all elements of  $N^+(Y_i)$  in  $T$  are labeled **do**
- 10:     Introduce a vertex  $k_i$  into  $V^H$ .
- 11:     For each vertex  $x'' \in X_i$ , introduce the arc  $(x'', k_i)$  into  $U^H$ .
- 12:     For each vertex  $y' \in Y_i$  such that  $N^-(y') = \emptyset$ , introduce the arc  $(k_i, y')$  into  $U^H$ .
- 13:     **for each**  $Y_j \in N^+(Y_i)$  in  $T$  **do**
- 14:        $o_j \leftarrow d^+(k_j) - |X_j|$ .
- 15:       **if**  $o_j < 0$  **then**
- 16:         **exit.** There is no Hamiltonian circuit in  $G$ .
- 17:       **else**
- 18:         add  $o_j$  arcs  $(k_i, k_j)$  to  $U^H$ .
- 19:       **end if**
- 20:     **end for**
- 21:     Label the vertex  $Y_i$  of  $T$ .
- 22:   **end for**
- 23: **end for**
- 24: In  $H$ , link each pair of vertices  $x'$  and  $x''$  by the arc  $(x', x'')$  and label this arc  $x$ .
- 25: Search for a Eulerian circuit  $\mathcal{E}$  in  $H$ . If there is none,  $G$  contains no Hamiltonian circuit. Otherwise, the (closed) sequence of labels of arcs from  $\mathcal{E}$  is the solution for the Hamiltonian Circuit Problem in  $G$ .



**Figure 6.2:** Illustration of the transformation of a cluster by Algorithm 18 (Steps 3 to 23). Part (a) represents the original cluster coming from the decomposition of graph  $G$  and part (b) represents the resulting part of  $H$ . The numbers next to the arcs represent the multiplicity of these arcs.



**Figure 6.3:** Construction by Algorithm 18. Part (a): Step 8 (type  $\mathcal{C}$ ). Part (b): steps 10 through 20 (types  $\mathcal{A}$  and  $\mathcal{B}$ ).

Algorithm 18 builds the vertices  $k_j$  such that there is a path from every vertex  $x'' \in X_i$  such that  $Y_i \supset Y_j$  to  $k_j$  and there is a path from  $k_j$  to every vertex  $y' \in Y_j$  (including the subsets of  $Y_i$ ).

At Step 14, all arcs exiting  $k_j$  have been built, and at least  $|X_j|$  must enter  $k_j$ . If  $|X_j| > d^+(k_j)$ , it means that

$$|\cup_{k|Y_k \subseteq Y_j} X_k| > |Y_j|$$

Since the vertices in  $\cup_{k|Y_k \subseteq Y_j} X_k$  do not have any other successor but the vertices in  $Y_j$ , this means that there is no possible Hamiltonian circuit in  $G$ .  $\square$

Claim 6.17 shows that Algorithm 18 exits before reaching Step 25 only if it could be shown before that there is no Hamiltonian circuit in  $G$ . We now have to show that Step 25 finds a Eulerian circuit in  $H$  if and only if there is a Hamiltonian circuit in  $G$ . In order to do so, we start with proving claims 6.18 and 6.19.

**Claim 6.18.** *For any two vertices  $x''$  and  $y'$  of the same cluster, there is a path from  $x''$  to  $y'$  in  $H$  if and only if there is an arc  $(x, y)$  in  $G$ .*

*Proof.*

$\Rightarrow$  Suppose that there is an arc  $(x, y)$  in  $G$ . Let  $X_i$  be the set containing  $x''$ . Then, by construction,  $Y_i \ni y'$ .

If  $X_i$  is of type  $\mathcal{C}$ , then Algorithm 18 builds a path  $\langle x'', k_i, y' \rangle$  at Step 8.

Consider now the case where  $X_i$  is of type  $\mathcal{A}$  or  $\mathcal{B}$ . If there exists no  $j$  such that  $y' \in Y_j$  and  $Y_j \subset Y_i$ , then, at the time when  $Y_i$  is chosen at Step 9, no  $Y_j$  containing  $y'$  has been chosen before and thus no arc entering  $y'$  has been added to  $U^H$  yet; thus  $N^-(y') = \emptyset$ ; so the algorithm builds a path  $\langle x'', k_i, y' \rangle$  at Step 12.

Finally, if  $\exists j : y' \in Y_j \wedge Y_j \subset Y_i$ , then there are the arcs  $(x'', k_i)$  and  $(k_j, y')$  in  $U^H$ . Note that, at Step 18, the algorithm puts an arc  $(k_l, k_m)$  in  $U^H$  if there is an arc  $(Y_l, Y_m)$  in  $T$ . Thus, since  $Y_j \subset Y_i$ , there is a path from  $Y_i$  to  $Y_j$  in  $T$  and also a path from  $k_i$  to  $k_j$  in  $H$ . Thus, there is a path from  $x''$  to  $y'$  in  $H$ .

$\Leftarrow$  Suppose that there is a path from  $x''$  to  $y'$  in  $H$ . By construction,  $x''$  has only one successor, that we denote  $k_i$  and  $y'$  has only one predecessor, that we denote  $k_j$ . This means that  $x'' \in X_i$  and  $y' \in Y_j$ . If  $i = j$ , then  $Y_j = Y_i = N^+(X_i)$ . Thus, there is an arc  $(x, y)$  in  $G$ . Else, it means



that there is a path from  $k_i$  to  $k_j$ . Thus, there is a path from  $Y_i$  to  $Y_j$  in  $T$  and thus,  $Y_j \subset Y_i$  and  $y \in Y_i$ , which implies that  $(x, y)$  is in  $G$ .

□

**Claim 6.19.** *At the end of Algorithm 18, every vertex  $v \in V^H$  that was part of some cluster at the end of Step 2, as well as every vertex that was introduced by the algorithm (the  $k_i$  vertices), satisfies  $d^+(v) = d^-(v)$ .*

*Proof.* There are three types of such vertices: the vertices  $x'$  and  $x''$ , and the vertices  $k_i$ , for every  $i$ .

For every vertex  $x'$ , there is only one incoming arc, added at Step 8 or at Step 12. There is also only one outgoing arc, which is  $(x', x'')$  (Step 24). Thus,  $d^+(x') = d^-(x')$ .

Similarly, for every vertex  $x''$ , there is only one incoming arc, which is  $(x', x'')$  (Step 24). There is also only one outgoing arc, added at Step 8 or at Step 11. Thus,  $d^+(x'') = d^-(x'')$ .

Consider now a vertex  $k_i$ . We want to prove that  $d^-(k_i) = d^+(k_i) \forall i$ . For every  $i$ ,

$$d^+(k_i) = |Y_i| - \underbrace{\sum_{j|Y_j \in N^+(Y_i) \text{ in } T} |Y_j|}_{\text{by Step 8 or Step 12}} + \underbrace{\sum_{j|Y_j \in N^+(Y_i) \text{ in } T} o_j}_{\text{by Step 8 or Step 18}} \quad (6.1)$$

For every  $i \neq 1$ ,

$$d^-(k_i) = \underbrace{|X_i|}_{\text{by Step 11}} + \underbrace{o_i}_{\text{by Step 18}} \quad (6.2)$$

And for  $i = 1$ ,

$$d^-(k_1) = |X_1| \quad (6.3)$$

Besides that, by Step 14:

$$o_i = d^+(k_i) - |X_i| = |Y_i| - \sum_{j|Y_j \in N^+(Y_i) \text{ in } T} |Y_j| + \sum_{j|Y_j \in N^+(Y_i) \text{ in } T} o_j - |X_i| \quad (6.4)$$

Let  $(Y_i \rightsquigarrow Y_j)$  represent a path from  $Y_i$  to  $Y_j$ , possibly a path of length 0 if  $i = j$ . We show that, for all  $i$ ,

$$o_i = |Y_i| - \sum_{j|\exists(Y_i \rightsquigarrow Y_j) \text{ in } T} |X_j|. \quad (6.5)$$

(6.5) is obvious if  $Y_i$  is a leaf of  $T$ . By induction, suppose that it holds for every  $Y_j$  such that  $Y_j \in N^+(Y_i)$  in  $T$ . Then, (6.4) can be rewritten:

$$\begin{aligned} o_i &= |Y_i| - \sum_{j|Y_j \in N^+(Y_i) \text{ in } T} |Y_j| + \sum_{j|Y_j \in N^+(Y_i) \text{ in } T} \left( |Y_j| - \sum_{k|\exists(Y_j \rightsquigarrow Y_k) \text{ in } T} |X_k| \right) - |X_i| \\ o_i &= |Y_i| - \sum_{j|Y_j \in N^+(Y_i) \text{ in } T} \left( \sum_{k|\exists(Y_j \rightsquigarrow Y_k) \text{ in } T} |X_k| \right) - |X_i| \\ o_i &= |Y_i| - \sum_{j|\exists(Y_i \rightsquigarrow Y_j) \text{ in } T} |X_j|. \end{aligned}$$

Thus, for every  $i$ ,

$$\begin{aligned} d^+(k_i) &= |Y_i| - \sum_{j|Y_j \in N^+(Y_i) \text{ in } T} |Y_j| + \sum_{j|Y_j \in N^+(Y_i) \text{ in } T} \left( |Y_j| - \sum_{k|\exists(Y_j \rightsquigarrow Y_k) \text{ in } T} |X_k| \right) \\ d^+(k_i) &= |Y_i| - \sum_{j|\exists(Y_i \rightsquigarrow Y_j) \text{ in } T} |X_j| + |X_i| \end{aligned} \quad (6.6)$$

and, for every  $i \neq 1$ , by replacing  $o_i$  by its value in (6.2):

$$d^-(k_i) = |X_i| + |Y_i| - \sum_{j|\exists(Y_i \rightsquigarrow Y_j) \text{ in } T} |X_j| \quad (6.7)$$

which, with (6.6), leads to the conclusion that, for  $i \neq 1$ ,  $d^+(k_i) = d^-(k_i)$ .

For  $i = 1$ ,  $d^-(k_1) = |X_1|$ . Besides that, Step 4 ensures that

$$|Y_1| = \sum_{j|\exists(Y_1 \rightsquigarrow Y_j) \text{ in } T} |X_j| \quad (6.8)$$

Thus, (6.6) becomes  $d^+(k_1) = |X_1|$  and, by (6.3),  $d^+(k_1) = d^-(k_1)$ . This ends the proof of Claim 6.19.  $\square$

**Remark 6.20.** *The vertices  $x'$  (respectively  $x''$ ) which were not part of any cluster at the end of Step 2 have  $d^-(x') = 0$  and  $d^+(x') = 1$  (respectively  $d^-(x'') = 1$  and  $d^+(x'') = 0$ ). Of course, if any such vertex exists,  $G$  does not contain a Hamiltonian circuit.*

**Claim 6.21.** *Algorithm 18 has a complexity of  $O(n^2 + m^2)$ .*

*Proof.* Step 1 has complexity 1. Step 2 has complexity  $n + m$ . Step 24 has complexity  $n$ . Step 25 can be done in  $O(n^2)$  [Law76].

For steps 3 to 23, since they are executed on disjoint parts of the graph, we will calculate their execution time over all passes through this loop instead of for each pass through this loop separately.

Step 4 has complexity  $2n$ . Steps 5 and 6 can be done in  $m^2$  for the creation of sets  $X_i$  and  $m^2$  again for the comparison of sets  $Y_i$  and the construction of  $T$  (both can be done at the same time); thus the complexity of these two steps is  $O(m^2)$ . The complexity of Step 7 is  $m$ . The complexity of Step 8 is at most  $3n$ .

The loop at Step 9 will be executed at most  $n$  times. The complexity of Step 10 is 1. The complexity of steps 11 and 12 is at most  $n$  each. The loop at Step 13 is executed at most  $n$  times. The steps within this loop have complexity of  $O(1)$  except Step 18, which may add at most  $n$  arcs over all passes since  $\sum_j (d^+(k_j)) = |\{y : \exists j \text{ such that } y \in Y_j\}| \leq n$ . Therefore, the complexity of the loop at Step 9 is  $O(n^2)$ .

This gives us an overall complexity of  $O(n^2 + m^2)$ . □

**Remark 6.22.** *In the special case where  $G$  is a 1-graph, we have that  $m \leq n^2$ , thus the complexity of Algorithm 18 is  $O(n^4)$ .*

Claims 6.19 and 6.18 prove that there is a Eulerian circuit in  $H$  if and only if there is a Hamiltonian circuit in  $G$ . By Claim 6.21, Algorithm 18 is polynomial. Thus, Theorem 6.14 holds. □

## 6.3 Generalizations of Quasi-Adjoint Graphs

Quasi-adjoint graphs are interesting because of the polynomiality of finding a Hamiltonian circuit. This section discusses two related problems.

Algorithm 19, if used before Algorithm 18, enlarges the class of graphs for which the Hamiltonian Circuit Problem is polynomially solvable, since it can transform some graphs into quasi-adjoint graphs.

Theorem 6.25 gives an interpretation of Algorithm 18 in terms of forbidden transitions and shows a limitation to the generalization of this idea.

### 6.3.1 Removal of Arcs

When searching for a Hamiltonian circuit in a graph, some arcs can safely be removed. We devise here the algorithm doing this and show that it does not affect the hamiltonicity of a graph.

**Algorithm 19**

---

**Input:** A graph  $G = (V, U)$ .**Output:**  $G$  with some arcs removed without changing its hamiltonicity.

- 1: Remove all loops (arcs of type  $(x, x)$ ).
  - 2: Split  $G$  into clusters. Denote the new graph  $G' = (V', U')$ .
  - 3: **for** each cluster  $C$  of  $G'$  **do**
  - 4:     Solve the problem of perfect matching in  $C$ .
  - 5:     **if** a perfect matching is found **then**
  - 6:         label all arcs composing the solution as N (Necessary).
  - 7:         **for** every not-labeled arc  $(x, y)$  of  $C$  **do**
  - 8:             Consider the subgraph of  $C$  induced by the removal of  $x$  and  $y$ .
  - 9:             Solve the problem of perfect matching in it.
  - 10:            **if** there is no solution to this problem **then**
  - 11:                Remove the arc  $(x, y)$  from  $U'$  /\*Thus, also from  $C$ .\*/ and the corresponding arc from  $U$ .
  - 12:            **else**
  - 13:                label all the arcs of this solution and the arc  $(x, y)$  as N.
  - 14:            **end if**
  - 15:         **end for**
  - 16:     **end if**
  - 17: **end for**
- 

**Claim 6.23.** *Algorithm 19 removes an arc  $(x, y)$  from a graph  $G$  only if this arc cannot be part of any Hamiltonian circuit in  $G$ .*

*Proof.* For each cluster  $C$  of  $G'$  and  $x'' \in L(C)$  and  $y' \in R(C)$ ,  $R(C)$  includes  $N^+(x'')$  and  $L(C)$  includes  $N^-(y')$ . Consider now that  $(x'', y')$  does not belong to any perfect matching in  $C$ . Then, if  $(x, y)$  is in some Hamiltonian circuit  $\mathcal{H}$  in  $G$ , there exists a vertex  $v \in G$  with  $v'' \in L(C)$  which does not have a successor in  $\mathcal{H}$ . This is a contradiction with the definition of a Hamiltonian circuit. Thus,  $(x, y)$  may not be part of any Hamiltonian circuit in  $G$ . □

### 6.3.2 About Forbidden Transitions

**Definition 6.24.** *Searching for a path with forbidden transitions is searching for a path which does not contain a forbidden transition as a subpath.*

The method of Blazewicz and Kasprzak [BK06] searches for a Eulerian path in polynomial time in graphs where some transitions are forbidden. Unfortunately, Theorem 6.25 states that this cannot be generalized.

**Theorem 6.25.** *Given any Eulerian graph  $H$  with a collection  $\mathcal{F}$  of forbidden transitions, it is NP-complete to find a Eulerian path (or circuit) which does not contain any transition from  $\mathcal{F}$ .*

*Proof.* Consider the Hamiltonian Path (Circuit) Problem in a directed graph  $G = (V, U)$ . It is always possible to introduce arcs into  $G$  so that it becomes the adjoint of some Eulerian graph  $H$ . This can be done in polynomial time by comparing the successors of every pair of vertices  $x$  and  $y$  and adding missing arcs to have  $N^+(x) \cap N^+(y) \neq 0 \Rightarrow N^+(x) = N^+(y)$ . The number of arcs introduced is smaller than  $n^2$ . When transforming  $G$  into its original graph  $H$ , each arc that was added to  $G$  results in a transition (a path of three vertices, see Definition 6.3) in  $H$ . These transitions are labeled as forbidden. Then, there exists a Eulerian path (circuit) in  $H$  that does not contain any forbidden transitions if and only if  $G$  has a Hamiltonian path (circuit).  $\square$

**Remark 6.26.** *Every cluster of graph  $G$  constructed at the beginning of Algorithm 18 can be seen as a complete bipartite graph with some missing arcs (see Figure 6.1). In order to find a Hamiltonian circuit in  $G$ , one could add all those arcs to  $G$ .  $G$  would then be an adjoint, which could be transformed into its original graph in order to find a Eulerian circuit in it. However, there would be some forbidden transitions, corresponding to the newly added arcs. As stated in Theorem 6.25, this is in general difficult. The construction of Algorithm 18, though only applicable to quasi-adjoint graphs, avoids this problem.*

## 6.4 Conclusion

In this chapter, we have defined the polynomial-time recognizable class of quasi-adjoint graphs, which extends the set of known graph classes for which the Hamiltonian Circuit Problem is polynomially solvable. We have provided a polynomial-time algorithm of complexity  $O(n^2 + m^2)$  solving the problem in these graphs, as well as another algorithm which provides some extension of this class with respect to the polynomial solvability of the Hamiltonian Circuit Problem. The class of quasi-adjoint graphs is a generalization of two known classes: the adjoints [Ber76] and the graphs modeling the problem of isothermic DNA sequencing by hybridization [BK06].



# Conclusion

Although graph theory is a well known and extensively studied field of research, many questions remain open. In this thesis, we have solved some of these. Our investigation on online coloring has turned out to be very fruitful. We could strengthen several results for this problem in special classes of graphs, most of which with practical applications. The Hamiltonian circuit problem is another domain of graph theory with many open questions, and we could enlarge the set of cases in which this problem is polynomially solvable. In both domains, this thesis has explored some paths and there are many research avenues still to be explored.

In Chapter 2, we have started with some general remarks which constitute a preliminary to the rest of the thesis: we have given a method to transform an online algorithm for covering graphs with given sets into an online algorithm for covering graphs with the same given sets, but with the additional constraint that the sets have a maximum cardinality  $b$ .

Then, we have studied online coloring of co-interval graphs and used the above quoted method for the bounded case. For the case where the colors are of bounded size and the intervals are presented in arbitrary order, we have shown that the performance ratio of the problem is somewhere between 2 and 3. Its exact value is unknown yet.

In Chapter 3, we have extensively studied online coloring of permutation and comparability graphs. In the second part of this chapter, we have studied online cocoloring of permutation graphs. Many models can be imagined that may result in different performance ratios. In particular, it would be interesting to find the exact performance ratio if the permutation graph is presented on a discrete latticial plan.

In Chapter 4, we have applied the results of Chapter 3 to a practical problem. We have continued the study of online coloring of permutation graphs and online coloring of overlap graphs. The performance ratio of the small train station problem and the big train station problem remain open in the bounded case.

In Chapter 5, we have studied online coloring of trees, forests and split

graphs. Many other classes of graphs, such as threshold graphs, claw-free graphs or cacti, have not been studied yet. See [BLS99] or [Gol04] for a long list of graphs with interesting applications. Furthermore, for each class of graphs, many models can be imagined: the vertices can be presented in a given order, they can be presented in blocks or one by one. And in each case, of course, the coloring can be bounded or unbounded.

In Chapter 6, we have introduced the quasi-adjoint graphs. We have given a polynomial recognition algorithm and a polynomial algorithm for solving the Hamiltonian circuit problem in these graphs. This chapter leads to many interesting questions such as finding easy to recognize super-classes of quasi-adjoint graphs such that the Hamiltonian circuit problem is still solvable in polynomial time, or finding a Eulerian circuit or path that avoids some given forbidden transitions.

Truly, graph theory is a very wide domain and many questions remain open.



# Bibliography

- [AF07] Nicola Apollonio and Paolo G. Franciosa. A characterization of partial directed line graphs. *Discrete Mathematics*, 307:2598–2614, 2007.
- [Alb03] Susanne Albers. Online algorithms: A survey. *Mathematical Programming*, 97:3–26, 2003. Invited paper at ISMP 2003.
- [ANS80] Takanori Akiyama, Takao Nishizeki, and Nobuji Saito. NP-completeness of the Hamiltonian cycle problem for bipartite graphs. *J. Inform. Process.*, 3(2):73–76, 1980.
- [BBE<sup>+</sup>03] Eric Bach, Joan Boyar, Leah Epstein, Lene M. Favrholdt, Tao Jiang, Kim S. Larsen, Guo-Hui Lin, and Rob van Stee. Tight bounds on the competitive ratio on accommodating sequences for the seat reservation problem. *J. Scheduling*, 6(2):131–147, 2003.
- [BBH<sup>+</sup>99] Ulrich Blasum, Michael R. Bussieck, Winfried Hochstättler, Christoph Moll, Hans-Helmut Scheel, and Thomas Winter. Scheduling trams in the morning. *Mathematical Methods of Operations Research*, 49(1):137–148, 1999.
- [BCP06] Hajo Broersma, Agostino Capponi, and Daniel Paulusma. Online coloring of h-free bipartite graphs. *Lecture Notes in Computer Science*, 3998:284–295, 2006.
- [Ber61] Claude Berge. Färbung von Graphen, deren sämtliche bzw. deren ungerade Kreise starr sind. Technical Report Wiss. Zeitung 114, Martin Luther University, Halle-Wittenberg, 1961.
- [Ber76] Claude Berge. *Graphs and Hypergraphs*. North Holland, Amsterdam, 1976.

- [Ber81] Alan A. Bertossi. The edge Hamiltonian path problem is NP-complete. *Inform. Process. Lett.*, 13(4-5):157–159, 1981.
- [Ber83] Alan A. Bertossi. Finding Hamiltonian circuits in proper interval graphs. *Inform. Process. Lett.*, 17(2):97–101, 1983.
- [BEY05] Allan Borodin and Ran El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 2nd edition, 2005.
- [BHKdW99] Jacek Blazewicz, Alain Hertz, Daniel Kobler, and Dominique de Werra. On some properties of DNA graphs. *Discrete Applied Mathematics*, 98(1-2):1–19, 1999.
- [BK03] Jacek Blazewicz and Marta Kasprzak. Complexity of DNA sequencing by hybridization. *Theor. Comput. Sci.*, 290(3):1459–1473, 2003.
- [BK06] Jacek Blazewicz and Marta Kasprzak. Computational complexity of isothermic DNA sequencing by hybridization. *Discrete Applied Mathematics*, 154(5):718–729, 2006.
- [BKLbW08] Jacek Blazewicz, Marta Kasprzak, Benjamin Leroy-Beaulieu, and Dominique de Werra. Finding hamiltonian circuits in quasi-adjoint graphs. *Discrete Applied Mathematics*, 2008. To be published.
- [BLS99] Andreas Brandstädt, Van Bang Le, and Jeremy P. Spinrad. *Graph classes: a survey*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1999.
- [BP06] Danaé Bouille and Matthieu Plumettaz. Coloration online bornée dans les graphes de permutations. Technical report, Ecole Polytechnique Fédérale de Lausanne, 2006.
- [BR97] Vincent Bouchitté and Jean-Xavier Rampon. Online algorithms for orders. *Theor. Comput. Sci.*, 175(2):225–238, 1997.
- [CDS07] Sabine Cornelsen and Gabriele Di Stefano. Track assignment. *J. of Discrete Algorithms*, 5(2):250–261, 2007.
- [Chv84] Vašek Chvátal. Perfectly ordered graphs. In *Topics on perfect graphs*, volume 88 of *North-Holland Math. Stud.*, pages 63–65. North-Holland, Amsterdam, 1984.

- [Chv85] V. Chvátal. Hamiltonian cycles. In *The traveling salesman problem*, Wiley-Intersci. Ser. Discrete Math., pages 403–429. Wiley, Chichester, 1985.
- [CLB81] D. G. Corneil, H. Lerchs, and L. Stewart Burlingham. Complement reducible graphs. *Discrete Applied Mathematics*, 3:163–174, 1981.
- [CRST06] Maria Chudnovsky, Neil Robertson, Paul Seymour, and Robin Thomas. The strong perfect graph theorem. *Annals of Mathematics*, 164(1):51–229, 2006.
- [CZZ07] Timothy Chan and Hamid Zarrabi-Zadeh. A randomized algorithm for online unit clustering. *Lecture Notes in Computer Science*, 4368:121–131, 2007.
- [DDSLB07] Marc Demange, Gabriele Di Stefano, and Benjamin Leroy-Beaulieu. Online Bounded Coloring of Permutations and Overlap Graphs. In *Proceedings of 4th Latin-American Algorithms, Graphs and Optimization Symposium (LAGOS)*, pages 213–218, 2007. ENDM 30.
- [DEdW07] Marc Demange, Tımaz Ekim, and Dominique de Werra. A tutorial on the use of graph coloring for some problems in robotics. *European Journal of Operational Research*, 2007. To appear.
- [DHMR00] Elias Dahlhaus, Peter Horak, Mirka Miller, and Joseph F. Ryan. The train marshalling problem. *Discrete Appl. Math.*, 103(1-3):41–54, 2000.
- [DLB07] Marc Demange and Benjamin Leroy-Beaulieu. Online Coloring of Comparability Graphs: Some Results. Technical report, École Polytechnique Fédérale de Lausanne, 2007.
- [DPvHKZ96] Stéphane Dauzère-Pérès, Stan van Hoesel, Leo G. Kroon, and Peter J. Zwaneveld. Routing trains through railway stations: model formulation and algorithms. *Transportation Science*, 30:181–194, 1996.
- [DS94] Jitender S. Deogun and George Steiner. Polynomial algorithms for Hamiltonian cycle in cocomparability graphs. *SIAM J. Comput.*, 23(3):520–552, 1994.

- [DSK04] Gabriele Di Stefano and Magnus Love Koci. A graph theoretical approach to the shunting problem. *Electr. Notes Theor. Comput. Sci.*, 92:16–33, 2004.
- [DSKLZ06] Gabriele Di Stefano, Stefan Krause, Marco E. Lübbecke, and Uwe T. Zimmermann. On minimum  $\ell$ -modal partitions of permutations. In *LATIN*, pages 374–385, 2006.
- [Eul36] Leonhard Euler. *Solutio problematis ad geometriam situs pertinentis*. *Opera Omnia*, 7:128–140, 1736.
- [Fel97] Stefan Felsner. Online chain partitions of orders. *Theor. Comput. Sci.*, 175(2):283–292, 1997.
- [FH77a] Stéphane Földes and Peter L. Hammer. Split graphs. Proc. 8th southeast. Conf. on Combinatorics, graph theory, and computing; Baton Rouge 1977, 311-315 (1977)., 1977.
- [FH77b] Stéphane Földes and Peter L. Hammer. Split graphs having dilworth number two. *Can. J. Math.*, 29:666–672, 1977.
- [FJQS04] Gerd Finke, Vincent Jost, Maurice Queyranne, and András Sebő. Batch processing with interval graph compatibilities between tasks. *Discrete Applied Mathematics*, 2004. To appear. Preliminary version in Cahiers Leibniz.
- [FLKH05] Richard Freling, Ramon M. Lentink, Leo G. Kroon, and Dennis Huisman. Shunting of passenger train units in a railway station. *Transportation Science*, 39:261–272, 2005.
- [Gav73] Fanica Gavril. Algorithms for a maximum clique and a maximum independent set of a circle graph. *Networks*, 3:261–273, 1973.
- [GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.
- [GJMP80] Michael R. Garey, David S. Johnson, Gary L. Miller, and Christos H. Papadimitriou. The complexity of coloring circular arcs and chords. *SIAM Journal on Algebraic and Discrete Methods*, 1(2):216–227, 1980.

- [GJQ07] Dion Gijswijt, Vincent Jost, and Maurice Queyranne. Clique partitioning of interval graphs with submodular costs on the cliques. *RAIRO Oper. Res.*, 41:275–287, 2007.
- [GJS76] Michael R. Garey, David S. Johnson, and Larry J. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1:237–267, 1976.
- [GKL99] András Gyárfás, Zoltán Király, and Jenő Lehel. Online 3-chromatic graphs I. triangle-free graphs. *SIAM J. Discrete Math.*, 12(3):385–411, 1999.
- [GL88] András Gyárfás and Jenő Lehel. Online and first-fit colorings of graphs. *Journal of Graph Theory*, 12(2):217–227, 1988.
- [GLS84] Martin Grötschel, László Lovász, and Alexander Schrijver. Polynomial algorithms for perfect graphs. *Annals of Discrete Mathematics*, 21:325–356, 1984.
- [GM01] Giorgio Gallo and Federico Di Miele. Dispatching buses in parking depots. *Transportation Science*, 35(3):322–330, 2001.
- [Gol04] Martin Charles Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Annals of discrete mathematics. Elsevier, 2nd edition, 2004.
- [GSH89] Csaba P. Gabor, Kenneth J. Supowit, and Wen-Lian Hsu. Recognizing circle graphs in polynomial time. *Journal of the Association for Computing Machinery*, 36(3):435–473, 1989.
- [HDM<sup>+</sup>06] Mohamed Hamdouni, Guy Desaulniers, Odile Marcotte, François Soumis, and Marianne van Putten. Dispatching buses in a depot using block patterns. *Transportation Science*, 40(3):364–377, 2006.
- [HDS07] Mohamed Hamdouni, Guy Desaulniers, and François Soumis. Parking buses in a depot using block patterns: A benders decomposition approach for minimizing type mismatches. *Comput. Oper. Res.*, 34(11):3362–3379, 2007.
- [HS94] Magnús M. Halldórsson and Mario Szegedy. Lower bounds for online graph coloring. *Theor. Comput. Sci.*, 130(1):163–174, 1994.

- [HSC00] Shiwei He, Rui Song, and Sohail S. Chaudhry. Fuzzy dispatching model and genetic algorithms for railyards operations. *European Journal of Operations Research*, 124(2):307–331, 2000.
- [IPS82] Alon Itai, Christos H. Papadimitriou, and Jayme Luiz Szwarcfiter. Hamilton paths in grid graphs. *SIAM J. Comput.*, 11(4):676–686, 1982.
- [Jos06] Vincent Jost. *Ordonnancement chromatique: polyèdres, complexité et classification*. PhD thesis, Université Joseph Fourier, Grenoble, France, October 2006.
- [JPS03] Jerzy W. Jaromczyk, Andrzej Pezarski, and Maciej Slusarek. An optimal competitive algorithm for the minimal clique covering in circular arc graphs. In *Proc. 19th European Workshop on Computational Geometry*, pages 48–51, Bonn, Germany, 2003.
- [Kar72] Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of computer computations (Proc. Sympos., IBM Thomas J. Watson Res. Center, Yorktown Heights, N.Y., 1972)*, pages 85–103. Plenum, New York, 1972.
- [Kei85] J. Mark Keil. Finding Hamiltonian circuits in interval graphs. *Inform. Process. Lett.*, 20(4):201–206, 1985.
- [Kie81] Henry A. Kierstead. An effective version of dilworth’s theorem. *Trans. Amer. Math. Soc.*, 268(1):63–77, 1981.
- [Kik75] P. B. Kikust. The existence of a Hamiltonian cycle in a regular graph of degree 5. In *Latvian mathematical yearbook, 16 (in Russian)*, pages 33–38, 271. Izdat. “Zinatne”, Riga, 1975.
- [Kou07] Bernard Kouakou. *Algorithmique Online et Applications*. PhD thesis, Université Paris I - Panthéon sorbonne, 2007.
- [KPT94] Henry A. Kierstead, Stephen G. Penrice, and William T. Trotter. Online coloring and recursive graph theory. *SIAM J. Discret. Math.*, 7(1):72–89, 1994.
- [KPT95] Henry A. Kierstead, Stephen G. Penrice, and William T. Trotter. Online and first-fit coloring of graphs that do not induce  $p_5$ . *SIAM J. Discret. Math.*, 8(4):485–498, 1995.

- [Law76] Eugene L. Lawler. *Combinatorial optimization: networks and matroids*. Holt, Rinehart and Winston, New York, 1976.
- [Lov72] László Lovász. Normal hypergraphs and the perfect graph conjecture. *Discrete Mathematics*, 2(3):253–267, 1972.
- [LS77] Linda Lesniak and H. Joseph Straight. The chromatic number of a graph. *Ars Combinatoria*, 3:39–46, 1977.
- [LST89] László Lovász, Michael E. Saks, and William T. Trotter. An online graph coloring algorithm with sublinear performance ratio. *Discrete Math*, 75:319–325, 1989.
- [Mil04] Avery Miller. Online graph colouring. Canadian Undergraduate Mathematics Conference, 2004.
- [Mül96] Haiko Müller. Hamiltonian circuits in chordal bipartite graphs. *Discrete Math.*, 156(1-3):291–298, 1996.
- [NP00] Stavros D. Nikolopoulos and Charis Papadopoulos. On the performance of the first-fit coloring algorithm on permutation graphs. *Inf. Process. Lett.*, 75(6):265–273, 2000.
- [PS82] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, 1982.
- [Ros03] Andrea Rossi. Il problema dell’ordinamento dei treni in un deposito: modellazione e soluzione algoritmica. Master’s thesis, Università dell’Aquila, 2003.
- [Sch03] Alexander Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*. Springer-Verlag, 2003.
- [SM04] Frits Spieksma and Linda Moonen. Partitioning a permutation graph: Algorithms and an application. In *Operations Research 2004, International Conference, Tilburg University (The Netherlands)*, pages 150–150, September 2004.
- [SM07] Frits Spieksma and Linda Moonen. Partitioning a weighted partial order. *Journal of Combinatorial Optimization*, 2007. Accepted.

- [TC79] Regina I. Tyshkevich and Arkady A. Chernyak. Canonical decomposition of a graph defined by the degrees of its vertices. *Isv. Akad. Nauk BSSR, Ser. Fiz.-Mat. Nauk*, 5:14–26, 1979.
- [Tro95] William T. Trotter. Partially ordered sets. In *Handbook of combinatorics, Vol. 1*, pages 433–480. Elsevier, Amsterdam, 1995.
- [Ung92] Walter Unger. The complexity of colouring circle graphs (extended abstract). In *STACS*, pages 389–400, 1992.
- [Wag84] Klaus Wagner. Monotonic coverings of finite sets. *Elektronische Informationsverarbeitung und Kybernetik*, 20(12):633–639, 1984.
- [WZ00] Thomas Winter and Uwe T. Zimmermann. Real-time dispatch of trams in storage yards. *Ann. Oper. Res.*, 96:287–315, 2000. Mathematics of industrial systems, IV (Valparaiso, 1996).
- [ZZ07] Hamid Zarrabi-Zadeh. Online coloring co-interval graphs. In *Proceedings of the 12th International CSI Computer Conference (CSICC 2007)*, pages 1328–1332, Tehran, Iran, February 2007.



# Curriculum Vitæ

<b>Name</b>	Benjamin Leroy-Beaulieu
<b>Nationalities</b>	French and swiss
<b>Date of birth</b>	February 22, 1979
<b>Marital Status</b>	Single
<b>Address</b>	Ch. de Rionza 19 CH - 1020 Renens

## Education

---

2004-2008	PhD student of Prof. Dominique de Werra in Operations research at EPFL, Switzerland
1997-2002	Masters degree in computer science at EPFL, Switzerland
1999-2000	Exchange student at the University of Waterloo, Ontario, Canada
1994-1997	Lycée de Gaulle-Adenauer, Bonn (Germany). Baccalauréat magna cum laude of type S (scientific)

## Professional Experience

---

2004-2008	Teaching assistant in the ROSE research group at EPFL: Discrete Mathematics, Operations Research, Graphs and Networks, Optimization
2002-2004	Software engineer at ELCA Informatik AG, Zurich
2001-2002	Diploma work as a software engineer and mathematician at Genedata AG, Basel. Production of a tool for automatically drawing biochemical pathways on a 2D plane
2000	3 months internship as a software engineer at Serial SA, Archamps, France
1998	1 month internship as a software engineer at Sony Europe, Cologne, Germany

## Languages

---

French	Native language
English	Fluently spoken and written
German	Fluently spoken; average writing skills
Italian	Basic notions

## Conferences and Talks

---

Aug. 2007	<i>Sixth International Symposium on Graphs and Optimization</i> , Cademario ‘Online Coloring of Co-interval Graphs’
May 2007	Invited talk at <i>Università dell’Aquila</i> , L’Aquila ‘The Track Assignment Problem’
Mar. 2007	Invited talk at <i>ESSEC</i> , Paris ‘The Track Assignment Problem’
Mar. 2007	<i>3e Cycle Romand de Recherche Opérationnelle</i> , Zinal ‘The Track Assignment Problem’
Jan. 2007	Invited talk at <i>EPFL</i> , Lausanne ‘Finding Hamiltonian Circuits in Quasi-Adjoint Graphs’
Dec. 2006	<i>Symposium on Algorithms with Performance Guarantee</i> , Paris, ‘Online Coloring of Overlap Graphs’
Aug. 2006	<i>Fifth International Symposium on Graphs and Optimization</i> , Leukerbad, ‘Online Coloring of Comparability Graphs’
Aug. 2006	Invited talk at <i>International Symposium on Mathematical Programming</i> , Rio de Janeiro, ‘Online Coloring of Comparability Graphs: Some Results’
Mar. 2006	<i>3e Cycle Romand de Recherche Opérationnelle</i> , Zinal ‘Online Coloring of Permutation Graphs’
Jan. 2006	<i>International Workshop on Combinatorial Optimization</i> , Aussois, ‘Online Coloring of Permutation Graphs’

## Miscellaneous

---

2008	Co-organizer of the <i>Operations Research Symposium in the honour of Thomas Liebling and Dominique de Werra</i> , Lausanne
Since 2004	Member of <i>Écologie libérale</i> , a political movement for the promotion of sustainable development with an economical approach.
Since 2008	Member of the executive committee of <i>Écologie libérale</i>