12

Chromatic scheduling

DOMINIQUE DE WERRA and ALAIN HERTZ

1. Introduction
2. Colouring with weights on the vertices
3. List-colouring
4. Mixed graph colouring
5. Co-colouring
6. Colouring with preferences
7. Bandwidth colouring
8. Edge-colouring
9. Sports scheduling
10. Balancing requirements
11. Compactness
12. Conclusion
References

Variations and extensions of the basic vertex-colouring and edge-colouring models have been developed to deal with increasingly complex scheduling problems. We present and illustrate them in specific situations where additional requirements are imposed. We include list-colouring, mixed graph colouring, co-colouring, colouring with preferences and bandwidth colouring, and we present applications of edge-colourings to open shop, school timetabling and sports scheduling problems. We also discuss balancing and compactness constraints which often appear in practical situations.

1. Introduction

We show here how graph colouring models may provide a natural tool for dealing with a variety of scheduling problems. Starting from the basic vertex-colouring model, we will introduce some variations and extensions that are motivated by their applications to some scheduling issues. In each case we give references for

https://doi.org/10.1017/CBO9781139519793.015 Published online by Cambridge University Press

further results and for extensions of the various models presented. For algorithms, see Chapter 13.

In chromatic scheduling problems we have a collection V of items, such as operations of jobs to be performed. In V there are some pairs v, w that are subject to an incompatibility condition and we call E the set of such incompatibility pairs. These data are represented by the graph $G = (V, E)$ in which the items are associated with the vertices and the incompatible pairs v, w with the edges vw between the corresponding vertices.

We also have a set $C = \{1, 2, \ldots, k\}$ of time periods (of unit duration). Assuming that each item (considered as an operation) has unit completion time, we may ask whether we can find a schedule taking the incompatibilities into account and using at most k periods of time. This is precisely the vertex k-colouring problem: there exists a feasible schedule if and only if the set V of vertices can be partitioned into subsets $S_1, S_2, \ldots, S_k$, where each S_i contains no two incompatible items.

In some instances, we may try to find the smallest set C of periods (that is, the smallest k) for which a schedule in time $k = |C|$ exists. This is the usual chromatic number $\chi(G)$ of the associated graph G. A classical application of this model is the basic school timetabling problem: the items are one-hour classes, and two classes are incompatible if they cannot be simultaneously given because they involve the same students or the same teacher.

We recall that for arbitrary graphs, the vertex k-colouring problem is NP-complete (see Garey and Johnson [21]).

2. Colouring with weights on the vertices

Previously we have assumed that all items v have the same unit completion time, which we denote by $\omega(v)$. More generally, we may have arbitrary completion times for the items; we identify these completion times with weights $\omega(v)$ which we assume for simplicity to be non-negative integers.

As before we have a set C of unit periods with $k = |C|$, and in order to complete item v we have to assign to it a subset $c(v)$ of $\omega(v)$ periods from C. In order to respect the incompatibility requirements, the assignments of $\omega(v)$ periods to the items v must be such that $c(v) \cap c(w) = \varnothing$ for each incompatible pair vw in E. This is precisely the *multicolouring problem* which has been studied, for instance, by Halldórsson and Kortsarz [28].

In the multicolouring model we observe that if $\omega(v) = r > 1$ for every v in a graph G, then the minimum number of colours needed for a multicolouring of G may be strictly smaller than $r\,\chi(G)$, as can be seen for the pentagon with $r = 2$ in Fig. 1, where the weights are indicated inside each vertex: here only $5 (< 2\,\chi(G) = 6)$ colours are needed.

We observe immediately that the multicolouring problem in a graph G is equivalent to the basic vertex-colouring problem in the graph G' obtained from

https://doi.org/10.1017/CBO9781139519793.015 Published online by Cambridge University Press

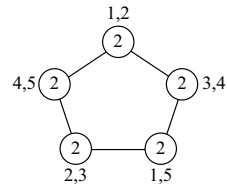


Fig. 1.

G by replacing each vertex v by vertices $v_1, v_2, \ldots, v_{\omega(v)}$, each one linked to all neighbours of v in G. We also introduce all possible edges between $v_1, v_2, \ldots, v_{\omega(v)}$.

The above multicolouring model may produce schedules in which the $\omega(v)$ periods during which a particular item v is being processed are not consecutive. We say that the schedule has some *preemptions* – that is, interruptions during the execution of some of the items. This may be acceptable in some types of application such as school timetabling: if v represents the $\omega(v)$ lectures of a same topic given by a teacher to a specific set of students, a schedule will be improved if these $\omega(v)$ lectures are spread throughout the week. We see later how one can impose such a requirement for these $\omega(v)$ lectures.

There are also contexts in which interruptions during the completion of an item v are not allowed: this means that v has to be assigned a set $c(v)$ of $\omega(v)$ consecutive time periods in C. If this is required for every item v, we have an *interval colouring problem*. This model has been discussed by various authors (see Čangalović and Schreuder [41] and Bouchard *et al.* [3]). It occurs in particular when there are prohibitive set-up times needed to start or restart the processing of an item.

As an illustration, the multicolouring with $k = 4$ colours in Fig. 2(a) (where weights appear inside each vertex) is not an interval colouring. An interval colouring with $k = 5$ colours is shown in Fig. 2(b). Notice that no such interval colouring can be found with fewer than five colours.

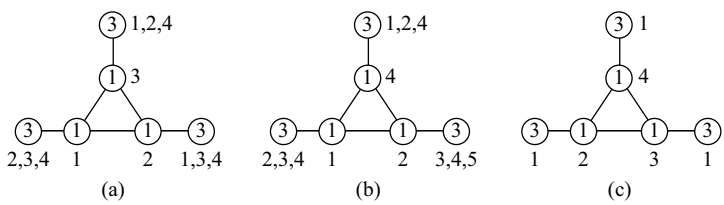


Fig. 2.

In some circumstances, the schedules constructed are designed to be repeated, day after day (say). In such cases it may be convenient to consider that the set $C = \{1, 2, \ldots, k\}$ is cyclically ordered, this means in particular that period 1 follows period k.

https://doi.org/10.1017/CBO9781139519793.015 Published online by Cambridge University Press

In the interval colouring context, we also allow intervals that contain periods k and 1. The possible intervals are now called *cyclic intervals* and the corresponding problem is the *cyclic interval colouring problem*: assign to each vertex v of the graph G a cyclic interval $c(v)$ in C of length $\omega(v)$, in such a way that for any two incompatible items v, w we have $c(v) \cap c(w) = \varnothing$. Cyclic interval colourings have been considered by de Werra and Solot [14]. The multicolouring in Fig. 2(a) is a cyclic interval colouring with four colours.

A related situation occurs in the batch scheduling problem: as before we have a set V of items with a family E of pairwise incompatibilities; each item v again has an integral weight $\omega(v)$ that corresponds to its processing time. We will not allow the items to start their processing individually, but we rather partition the set V into a number k (to be determined) of batches $S_1, S_2, \cdots, S_k$, where each S_i is a subset of items that have no incompatibilities, and therefore whose items can be processed simultaneously. The completion time of a batch S_i is then defined as $p(S_i) = \max_{v \in S_i} \omega(v)$. It is then required to find a value k and a partition into batches $S_1, S_2, \ldots, S_k$ for which $p(S_1) + \ldots + p(S_k)$ is minimum.

Such a problem arises when we have items v to sterilize by placing them into an oven for a duration of at least $\omega(v)$ consecutive time units. Incompatibility requirements do not allow us to place certain specified pairs of items simultaneously in the oven. Finding such a schedule is called the *batch colouring problem*: it has been studied (under the name of 'weighted colouring') in Demange *et al.* [15]. An optimum batch colouring (minimizing $\sum_i p(S_i)$) is given in Fig. 2(c); here $\sum_i p(S_i) = 3 + 1 + 1 + 1 = 6$. It uses $4 (> \chi(G) = 3)$ colours and no colouring with three colours can be as good.

As a final observation we mention that if in all the above types of colouring each weight $\omega(v)$ is equal to 1, then we get back to the basic vertex-colouring problem. So all of these models are generalizations of this problem and they are motivated by various types of application.

We now consider more situations in which extensions of the basic vertex-colouring problem arise.

3. List-colouring

We return to the basic vertex-colouring model in which each weight $\omega(v) = 1$. Here we have a set $C = \{1, 2, \ldots, k\}$ of periods and each item v has a list $L(v) \subseteq C$ of periods to which it could be assigned. In the course scheduling context this means that each class v can be offered only at one of the periods in $L(v)$. Finding a basic vertex-colouring in G such that each vertex v gets a colour $c(v) \in L(v)$ is the well-known *list-colouring problem* which has been extensively studied, for reasons related to its many applications (see Chapter 6 and Voigt [42] for some basic results).

Worthy of interest is the special case in which $|L(v)| = 1$ for a particular subset V^* of items and $L(v) = C$ for all items $v \in V - V^*$. It means that the vertices $v \in V^*$

https://doi.org/10.1017/CBO9781139519793.015 Published online by Cambridge University Press

have already been coloured, and it is required to determine whether the colouring can be extended to the whole graph. This is the *precolouring extension problem*; it is often met in applications related to class scheduling. In such situations some classes are scheduled initially (in order to satisfy some requirements that may be intractable in a simple graph colouring model) and we wish to construct the rest of the schedule while keeping the precolouring fixed.

This problem is also a generalization of the basic vertex-colouring model (it is the case $V^* = \varnothing$) and is NP-hard for general graphs. Surprisingly, for a bipartite graph G with $V^* = \{v_1, v_2, v_3\}$ and $L(v_1) = \{1\}$, $L(v_2) = \{2\}$, $L(v_3) = \{3\}$, $C = \{1, 2, 3\}$, the precolouring extension problem is difficult (see Even *et al.* [20]).

We may observe that if the set $C = \{1, 2, \ldots, k\}$ of colours to be used is specified, then a list-colouring problem may be transformed into a basic vertex-colouring problem. Starting from $G = (V, E)$ and a collection of lists $L(v) \subseteq C = \{1, 2, \ldots, k\}$ for each $v \in V$, we add a clique K on new vertices $w_1, w_2, \ldots, w_k$ and join each vertex $v \in V$ to all vertices $w_i \in K$ for which $i \notin L(v)$. Then any colouring with k colours in the resulting graph gives a list-colouring of G if we call i the colour $c(w_i)$ for $i = 1, 2, \ldots, k$.

It is also appropriate to recall that in most school timetabling contexts, we usually also have a list $\bar{L}(v)$ of unavailable periods for each teacher v. But since the set $C = \{1, 2, \ldots, k\}$ of teaching periods in the week is generally specified, we easily get $L(v) = C - \bar{L}(v)$ as a list of possible periods (colours) for the classes of teacher v.

In connection with list-colouring, there is another concept that is of interest in the context of timetabling and scheduling: we say that a graph $G = (V, E)$ is *p-choosable* if there is a list-colouring of G for all assignments of lists $L(v)$ with $|L(v)| \geq p$ for each $v \in V$. Choosability has been extensively studied from a theoretical point of view (see Chapter 6, Erdős *et al.* [19] and Gutner [26]). The *choice number* $ch(G)$ is the minimum number p for which G is p-choosable. Graphs G with $ch(G) = 2$ have been characterized (see Erdős *et al.* [19]), but for $ch(G) > 2$ a complete characterization is still elusive.

In terms of timetabling, choosability is a natural concept: consider a school timetabling problem, where the dean in charge of the timetable has assigned courses to the teachers, and on this basis the graph G of conflicts between courses can be constructed. A colouring of G with k colours gives an assignment of classes to periods in the set $C = \{1, 2, \ldots, k\}$ of teaching periods. But teachers are not always available and may be invited to give a list of periods in which their own classes can be scheduled. In order to find a feasible timetable, one has to ask teachers to give lists of sufficient size. This is a problem for the dean who has to find a value p for which an acceptable timetable can be constructed, whatever set of at least p periods is given by each teacher. The dean naturally has to determine a value p that is as small as possible to avoid a rebellion of teachers. This is why it is so useful to be able to determine $ch(G)$.

https://doi.org/10.1017/CBO9781139519793.015 Published online by Cambridge University Press

4. Mixed graph colouring

Suppose now that in a basic class scheduling problem we have not only incompatibility requirements preventing some pairs of classes from being assigned to the same period, but also *precedence constraints*: these require that, for some pairs v, w of classes, v must be scheduled for some period $c(v)$ occurring earlier than the period $c(w)$ assigned to w. Such a requirement is represented in the graph G by an arc oriented from v to w. A graph $G = (V, E)$ with a set A of oriented arcs will be denoted by $G = (V, E, A)$, and is called a *mixed graph*. It follows that, given $G = (V, E, A)$ and $C = \{1, 2, \ldots, k\}$, a mixed graph colouring is an assignment of a period $c(v) \in C$ to each item v in such a way that $c(v) \neq c(w)$ for each pair $vw \in E$ and $c(v) < c(w)$ for each (ordered) pair $vw \in A$.

Mixed graph colourings have been extensively studied. We can see easily that, for a bipartite mixed graph $G = (V, E, A)$ with $C = \{1, 2, 3\}$, the problem of finding whether there is a mixed graph colouring is difficult: the precolouring extension problem at the end of Section 3 is equivalent to a mixed graph colouring problem obtained from G by introducing a set $A = \{v_1u_1, u_1u_2, u_3v_2, v_2u_4, u_5u_6, u_6v_3\}$ of arcs, where the u_i are new vertices (see Fig. 3). Properties of mixed graph colourings can be found in Ries [35], Ries and de Werra [36], Hansen *et al.* [29] and Sotskov *et al.* [40].

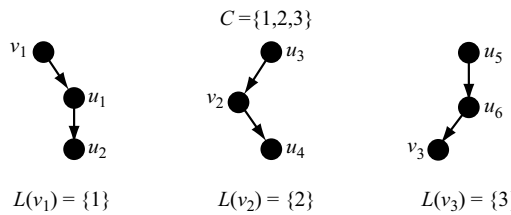


Fig. 3.

Notice that what makes the mixed graph colouring problem difficult is the presence of edges, not of arcs: if we have a graph $G = (V, E, A)$ with $E = \varnothing$, then there is a vertex-colouring satisfying $c(v) < c(w)$ for each arc vw if and only if G contains no oriented cycle. The colour of a vertex v is simply the length of a longest path ending in v.

There are strong connections between colourings and orientations of graphs: it is not difficult to see that finding a colouring with a minimum number of colours is equivalent to finding an acyclic orientation of all its edges for which the longest oriented path is as short as possible.

In this spirit we recall the well-known *Gallai–Roy theorem* which states that, for any oriented graph $\vec{G}$, the chromatic number of its unoriented copy G does not exceed the maximum number of vertices in an elementary oriented path of $\vec{G}$ (see Roy [38]).

The graph colouring problem can also be considered as formulated in the oriented graph obtained by replacing each edge vw in the initial graph G by a pair of opposite

https://doi.org/10.1017/CBO9781139519793.015 Published online by Cambridge University Press

arcs vw and wv forming a *disjunctive pair*. Let $\vec{G}$ be the resulting graph: then finding a vertex-colouring of G with k colours amounts to choosing exactly one arc from each disjunctive pair in such a way that the resulting graph G^* has no oriented cycles and no oriented path has more than k vertices.

Now for a mixed vertex-colouring, we can construct from G a graph $\vec{G}$ in the same way, while keeping the original arcs of G as they are. The mixed colouring problem is then similar to a problem with some preassignments: the choice has already been made for the arcs appearing in the original graph, and the question is whether one can extend the partial solution to the whole of $\vec{G}$.

5. Co-colouring

Another type of vertex-colouring has been introduced under the name of *co-colouring* as a natural generalization of the basic model (see Gimbel *et al.* [24]). Up to now the colour classes of colourings (the sets of vertices of the same colour) have been independent (or stable) sets. We now allow each colour class to be either an independent set or a clique, so a partition of the vertex-set V of a graph $G = (V, E)$ into p cliques and q independent sets is a co-colouring with $p+q$ colours. The smallest value of $p + q$ for which such a partition exists is the *co-chromatic number* of G. This type of colouring is motivated in particular by an application in robotics (see Demange *et al.* [17]).

Suppose that we wish to schedule the moves of a robot which is required to pick up a collection of items of different sizes along a storage line. The robot may pick up several items during a trip and it has to pile them up: this implies that, to ensure stability, larger items must be placed below smaller ones; in other words, during a trip along the storage line the robot must pick up items in decreasing order of size.

Different assumptions can be made on the possible moves of the robot along the storage line. We can number the stored items from 1 to n in order of decreasing size, so the alignment of the items along the line corresponds to a permutation Π of $\{1, 2, \ldots, n\}$. We construct the permutation graph $G(\Pi)$ corresponding to Π by assigning a vertex to each item and by linking vertices i and j if $i > j$ and i comes before j when we move from left to right along the line. So, in a move from left to right, the robot can pick up items in increasing order of their numbers. The vertices of $G(\Pi)$ corresponding to the items picked up during one trip form a independent set, so minimizing the number of left-to-right trips to pick up all the items is equivalent to partitioning the vertex-set V of $G(\Pi)$ into the minimum number of independent sets. It is the basic vertex-colouring model.

If we now assume that the robot can also move from right to left, then the items picked during any trip from right to left correspond to a clique in $G(\Pi)$. So minimizing the number of trips (with both directions allowed) amounts to finding the co-chromatic number of $G(\Pi)$.

https://doi.org/10.1017/CBO9781139519793.015 Published online by Cambridge University Press

As an illustration, suppose that there are nine items stored in line in the order 926415738. A robot moving from left to right needs four trips to pick up all the items:

138 2457 9 6.

Since the problem corresponds to a permutation graph, there is a polynomial algorithm to find a minimum collection of trips. If we also allow trips from right to left, we have a solution with three trips:

24578 369 1.

This corresponds to a minimum co-colouring. The problem is NP-hard, even in the class of comparability graphs; there is, however, a polynomial 2-approximation algorithm which guarantees that we use at most twice as many trips as necessary (see Demange *et al.* [16]).

In Demange *et al.* [17] additional variations on the possible moves of the robot are studied extensively and complexity results are given for these problems. They consider in particular the situation in which each robot (starting from the left or from the right) is allowed to make one move in one direction followed by a move in the other direction; the items picked during such a double move form a clique and an independent set in $G(\Pi)$; each colour class thus defines a *split graph* – that is, a graph whose vertex-set can be partitioned into a clique and an independent set. Partitioning the vertex-set of a graph G into split graphs is the *split colouring problem* which has been defined and studied in Ekim and de Werra [18].

6. Colouring with preferences

In the models examined so far we had essentially rigid constraints that could not be violated. But in almost all real situations there are some 'soft' constraints that may be satisfied or not. When they are violated, a certain penalty is incurred and the problem consists in finding a solution that minimizes the total penalty, or at least keeps it within a given bound. This is precisely what can be done in the vertex-colouring problem with preferences, which is another extension of the basic model.

We start with a graph $G = (V, E)$ and we introduce a subset P of pairs vw (not in E) on which some requirements are added. So we now have a graph $G = (V, E; P)$ in which P is a set of pairs with *preferences*, which we represent by dotted lines. Each edge in E is called a *strong* edge. For each pair $vw \in P$ we have a positive penalty $\omega(vw)$ which is counted whenever the preference is not satisfied.

In this first model (called M_1), the preference on vw could be that v and w should have the same colour, otherwise the penalty $\omega(vw)$ is incurred. We want to find a vertex-colouring with the smallest possible total penalty or with a total penalty that is below a given value W, and a minimum number of colours. As we can see, the number k of colours need not be specified in M_1. Figure 4 shows a colouring of a graph with $k = 3$ colours and a total penalty of 2. The colours are indicated besides the vertices and penalties are shown on the preference edges. No solution can be

https://doi.org/10.1017/CBO9781139519793.015 Published online by Cambridge University Press

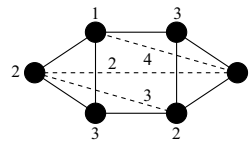


Fig. 4.

found with a smaller penalty, even if we use more colours, since there are cycles with exactly one strong edge.

When does a solution without penalty exist? This question has been studied for a long time, and we have the classical result of Cartwright and Harary [7] that in a graph $G = (V, E; P)$ there exists a vertex-colouring satisfying all preferences if and only if G contains no cycle with exactly one strong edge.

In M_1, minimizing the number of colours remains a difficult problem as can be expected. We can however recognize the situations in which there is a colouring with two colours that satisfies all preferences. As shown in Cartwright and Harary [7], a graph $G = (V, E; P)$ has a 2-colouring satisfying all preferences if and only if every cycle has an even number of strong edges.

The model M_1 is used in compilation, where one has to minimize the number of registers used. The model with $G = (V, E; P)$ has been exploited in Robillard [37]: each variable of the computer programme to be compiled is associated with a vertex of an associated graph. Whenever two variables have to be alive simultaneously (because they appear in the same instruction of the programme), they are joined by an edge. E is the set of these edges, and two variables whose vertices are joined by an edge cannot be assigned to the same register. So finding a vertex-colouring of G minimizing the number of colours gives us an assignment of variables to registers with a minimum number of registers.

Specialists in computation use an operation called *spilling*. It amounts to introducing, for some variable x, another variable x' which may be assigned to another register; practically, one copies the content of variable x at some stage of the execution of the programme into another variable x', and uses x' instead of x in the next instructions. This operation is represented in G by splitting x into two vertices x and x', and linking x and x' by a preference edge with penalty $\omega(xx')$ equal to the cost of copying the content of x into x'. In the resulting graph $G = (V, E; P)$ we may find, as before, whether there is a vertex-colouring without penalty. If so, it means that we can find a register allocation without any spilling. Such an allocation may however require a larger number of registers (colours) than are available, so we may impose a bound on k (the number of available registers) and find a vertex-colouring with at most k colours and a minimum penalty. Such a solution will require some spillings. For example, if three variables x, y, z have to be simultaneously alive, and if no spilling is allowed, then three registers are needed. By introducing an additional variable x' (by splitting x into x and x') we may use only two registers, but with a penalty for the spilling since x and x' get different colours. This is illustrated in Fig. 5.

https://doi.org/10.1017/CBO9781139519793.015 Published online by Cambridge University Press



Fig. 5.

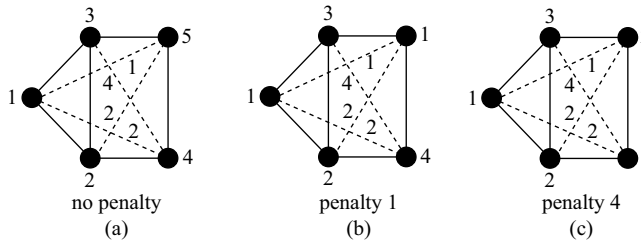


Fig. 6.

A second model (M_2) can also be considered. Here, in $G = (V, E; P)$, the preference edges vw in P are such that we would prefer to have $c(v) \neq c(w)$ in a vertex-colouring, but if we happen to have $c(v) = c(w)$ then a penalty $\omega(vw)$ is incurred. In this model M_2 the number k of colours is fixed, since by using a sufficient number of colours one may always satisfy all preferences.

Such a situation occurs rather naturally in scheduling: two operations v and w which are normally not simultaneous could be assigned to the same operator, but one may get into trouble in case of any delay in v or w; so we prefer to assign v and w to different operators, but if this is not possible then we pay a penalty $\omega(vw)$. The smallest number k of colours for which a vertex-colouring without penalty exists is the chromatic number of the graph obtained by introducing all pairs of P into E.

Finding a colouring with minimum penalty, for a fixed number k, is the *robust colouring problem* which is NP-hard in general (see Yánez and Ramírez [43]). A 5-colouring without penalty is shown in Fig. 6(a). In Figs. 6(b) and 6(c) an optimal robust 4-colouring and an optimal robust 3-colouring are shown, with penalties 1 and 4. This problem is studied by Archetti *et al.* [2].

7. Bandwidth colouring

The central model of vertex-colouring is an ideal instrument for capturing incompatibilities between pairs of items to be scheduled, but it may not be sufficient because of the scheduling applications considered. The notion of incompatibility may be refined by saying more than just 'a pair v, w of items must not be scheduled at the same time period – that is, the corresponding vertices must not be assigned the same colour'.

https://doi.org/10.1017/CBO9781139519793.015 Published online by Cambridge University Press

We may also consider more quantified requirements by saying, for instance, that two items v, w must be scheduled at periods that are separated from each other by at least t_{vw} periods. This is precisely the idea of *bandwidth colouring* (see Prestwich [33]). One may imagine many applications of scheduling in which operations requiring some set-up times must be separated by sufficiently many time units.

A graph $G = (V, E)$ is given, together with a collection of non-negative integer values t_{vw} assigned to the edges vw of E. We wish to find a number k (generally as small as possible) of colours to be used, and a partition of V into independent sets $S_1, S_2, \ldots, S_k$ such that $|c(v) - c(w)| > t_{vw}$ for each edge $vw \in E$, where $c(u)$ is the colour of u (that is, $c(u) = j$ if $u \in S_j$). If $t_{vw} = 0$ for each edge vw, the requirement simply means that $S_1, S_2, \ldots, S_k$ is a vertex-colouring of G.

Such a model may be extended in several ways. First, we could assign to each vertex v a number $\omega(v)$ of consecutive colours; this is the *interval colouring model*, which is able to handle items or operations that have different processing times. In such a case we would have requirements of the form $|f(v) - f(w)| \geq t_{vw}$, where $f(u)$ is the first colour assigned to the vertex u.

In a more general formulation we could consider that each edge vw is an oriented arc from v to w, and we introduce for each such arc a set T_{vw} of integer values that are forbidden for the difference $f(w) - f(v)$ – that is, for each arc vw we require that $f(w) - f(v) \notin T_{vw}$. For instance, if $T_{vw} = \{0\}$ for all arcs vw, and $\omega(v) = 1$ for all $v \in V$, we get the basic vertex-colouring model. If we have an interval vertex-colouring, then to avoid overlap of items v and w in the schedule, we must impose the condition:

$$f(v) + \omega(v) - 1 < f(w) \quad \text{or} \quad f(w) + \omega(w) - 1 < f(v).$$

This means that $T_{vw} = \{-\omega(w) + 1, \ldots, -1, 0, 1, \ldots, \omega(v) - 1\}$ ensures that items v and w are not simultaneously in process. To separate the two processing intervals for v and w, we may just choose a larger set than before.

This general model has been considered initially for frequency assignment problems with possible interferences. The requirements are that on neighbouring emitters the frequencies that have to be assigned to each one have forbidden values for their differences.

Observe that the orientation given to each initial edge vw is arbitrary. If we choose to have an arc wv instead of the arc vw, then we set:

$$T_{wv} = -T_{vw} = \{-a \ : \ a \in T_{vw}\}.$$

For such types of colourings, one can easily extend many classical upper bounds for the chromatic number (see de Werra and Gay [12]).

In these frequency assignment problems it is interesting to distinguish the *span* of the colouring constructed (the difference between the largest and the smallest frequencies) and the *order* (the number of different frequencies). Consider, for

https://doi.org/10.1017/CBO9781139519793.015 Published online by Cambridge University Press

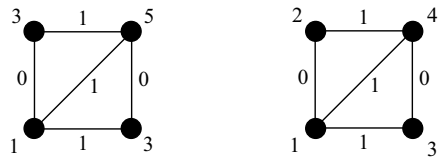


Fig. 7.

example, the graphs in Fig. 7, where the numbers on the edges vw correspond to the values t_{vw} (where $|c(v) - c(w)| > t_{vw}$): Fig. 7(a) shows that a bandwidth colouring c with minimum order 3 has span$(c) = 5 - 1 = 4$, while Fig. 7(b) shows that a bandwidth colouring c with minimum span 3 has order 4.

8. Edge-colouring

A well-known special case of the basic vertex-colouring problem is the edge-colouring model. We are given a multigraph $G = (V, E)$ and we want to partition the edge-set into matchings $M_1, M_2, \ldots, M_p$, so that no two edges in the same subset M_i have a vertex in common. The smallest number p for which such a partition exists is the chromatic index $\chi'(G)$ of G.

Let us consider the line graph $L(G)$ of G, the graph obtained by assigning a vertex v_e to each edge e of G and joining two vertices v_e and v_f in $L(G)$ by an edge if the corresponding edges e, f in G are adjacent. Then there is a one-to-one correspondence between the edge-colourings of G and the vertex-colourings of $L(G)$.

Many applications of edge-colourings have been studied. Among these, the class−teacher timetabling problem and the preemptive open shop scheduling models are the most famous.

Although it would be conceivable to study edge-colourings as special vertex-colourings, it seems to be extremely convenient to consider the edge-colourings of G directly. One reason is that the situation is more naturally visualized in some applications where G has a specific structure (such as being bipartite). Another reason is that there are techniques (such as alternating chain methods) that provide useful construction tools for edge-colourings, and these procedures are easily visualized without going to $L(G)$.

General properties of edge-colourings are presented in Chapter 5. Here we restrict our attention to specific applications related to scheduling and timetabling. The class−teacher timetabling problem is undoubtedly the most basic model, to which a variety of additional constraints may be added, according to the needs of concrete applications. We have a set T of teachers $t_1, t_2, \ldots, t_n$, a set CL of classes $c_1, c_2, \ldots, c_m$ (here a class is a group of students who follow exactly the same programme), and a collection of one-period lectures; each lecture is given by a specific teacher to a specific class.

https://doi.org/10.1017/CBO9781139519793.015 Published online by Cambridge University Press

The data of this problem may be represented by a bipartite multigraph. We associate each teacher t_j and each class c_i to a vertex of the graph, and each one-period lecture given by teacher t_j to class c_i is represented by an edge between c_i and t_j. We thus obtain a bipartite multigraph G with CL as the left-hand set of vertices, and T as the right-hand set. Let $C = \{1, 2, \ldots, k\}$ be the set of periods available. A timetable is an assignment of one period in C to each lecture in such a way that no teacher and no class is involved in more than one lecture at any period. This is an edge-colouring of G.

If no other requirements are imposed, it is known that a timetable exists if and only if the number of available periods is no less than the maximum load of all teachers and all classes; here the load of a teacher or of a class is the number of lectures in which the teacher or class is involved, and is the degree of the corresponding vertex of G. This statement follows from König's theorem which asserts that, for a bipartite graph $G = (V, E)$, $\chi'(G) = \Delta(G) = \max_{v \in V} d(v)$. Many variations and extensions of this basic timetabling model have been studied (see Burke *et al.* [6]).

Another classical application of edge-colouring in bipartite multigraphs is the preemptive open shop scheduling problem: we are given a collection $\mathcal{P} = \{P_1, P_2, \ldots, P_m\}$ of processors together with a set $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ of jobs. Each job J_j consists of operations $O_{1j}, O_{2j}, \ldots, O_{mj}$ to be processed (with interruptions allowed) on the processors in any order. We are given the processing time p_{ij} of O_{ij} for each operation on each processor. We usually assume that p_{ij} is a non-negative integer; if $p_{ij} = 0$, the operation O_{ij} does not exist. A *schedule* is an assignment of p_{ij} time units on P_i (not necessarily consecutive) for each operation O_{ij}, in such a way that at any moment no processor is involved in more than one operation and no two operations of the same job are in process.

We usually want to find a schedule S with a minimum total completion time $C_{\max}(S)$. Clearly, $C_{\max}(S) \geq \max\left\{\max_i \sum_j p_{ij}, \max_j \sum_i p_{ij}\right\}$. We may now associate a vertex to each processor P_i and a vertex to each job J_j. Each operation O_{ij} is represented by p_{ij} parallel edges between the vertices associated to P_i and to J_j. We recognize that if we identify the processors P_i of the open shop scheduling model with the classes c_i of the class−teacher timetabling model, and if we similarly identify the jobs J_j with the teachers t_j, then by considering O_{ij} as the set of lectures of t_j to c_i, we can use the edge-colouring model to solve the open shop scheduling problem. From König's theorem we see that $C_{\max}(S) = \max\left\{\max_i \sum_j p_{ij}, \max_j \sum_i p_{ij}\right\}$, and that without any loss of generality we may restrict the occurrence of preemptions to integer times.

Notice that there is also a non-preemptive open shop scheduling problem, where no interruption is allowed during the execution of an operation O_{ij}. The model to use would then be an interval edge-colouring analogous to the interval vertex-colouring. Minimizing $C_{\max}(S)$ is then an NP-hard problem as soon as the number of processors is at least 3 (see Gonzales and Sahni [25]). For two processors the problem remains easy (see de Werra [11]).

https://doi.org/10.1017/CBO9781139519793.015 Published online by Cambridge University Press

9. Sports scheduling

Another classical application of edge-colouring is the tournament scheduling problem. In its most elementary version we are given a league of $2n$ teams numbered from 1 to $2n$, and each team has to play exactly one game against every other team. This situation is represented by a complete graph K_{2n} on $2n$ vertices associated with the teams, while each edge corresponds to a game involving the two teams associated to its endpoints.

No team can play more than one game on each day, so the games that can be played on the same day form a matching in K_{2n}. Thus, finding a schedule for the $n(2n-1)$ games of the league corresponds to constructing an edge-colouring of K_{2n}. At least $2n-1$ days are needed, and it is known that $\chi'(K_{2n}) = 2n-1$, so a schedule in $2n-1$ days can be found; it corresponds to an edge-colouring of K_{2n} with matchings $M_1, M_2, \ldots, M_{2n-1}$ and $|M_i| = n$ for $i = 1, 2, \ldots, 2n-1$.

A simple way of constructing such a schedule is to define each set M_i (the games played on day i) as follows: team $2n$ plays againt team i, and team $i+k$ plays against team $i-k$ for $k = 1, 2, \ldots, n-1$, where the numbers $i+k$ and $i-k$ are taken (modulo $2n-1$) as one of the numbers $1, 2, \ldots, 2n-1$. Figure 8 shows a five-day schedule for a league of six teams.

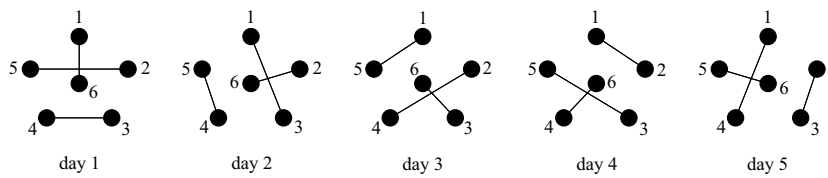


Fig. 8.

We now introduce the idea of an oriented edge-colouring to formulate and solve some related problems that arise in many situations. We assume that each of the $2n$ teams has its own stadium. When a team plays in its home stadium, it is a *Home game* (H) for this team, and when a team plays elsewhere it is an *Away game* (A). When a schedule is constructed, we have an edge-colouring $M_1, M_2, \ldots, M_{2n-1}$, but we still have to decide in which stadium each game should be played. We represent a game played between teams i and j in the home stadium of team j by an arc ij oriented from i to j. So constructing a schedule for such a sports league consists of determining an edge-colouring of K_{2n} and an orientation for each edge.

In general, one desires to have for each team a sequence of games that alternate as regularly as possible between Home games and Away games. Perfect alternation for all teams is not possible; as can be seen easily, we cannot have more than two teams with perfect alternation. When two games for a team i scheduled on consecutive days d and $d+1$ are both Home or both Away, we say that team i has a break on day $d+1$. In other words, the minimum number of breaks in a schedule for $2n$ teams is at least $2n-2$, and in fact, one can always construct a schedule where $2n-2$ teams have

https://doi.org/10.1017/CBO9781139519793.015 Published online by Cambridge University Press

M_1	$\overleftarrow{61}$	$\overrightarrow{25}$	$\overleftarrow{34}$
M_2	$\overrightarrow{62}$	$\overrightarrow{31}$	$\overleftarrow{45}$
M_3	$\overleftarrow{63}$	$\overrightarrow{42}$	$\overleftarrow{51}$
M_4	$\overrightarrow{64}$	$\overrightarrow{53}$	$\overleftarrow{12}$
M_5	$\overleftarrow{65}$	$\overrightarrow{14}$	$\overleftarrow{23}$

oriented colouring

		teams					
		1	2	3	4	5	6
days	1	A	A	H	A	H	H
	2	H	H	A	H	A	A
	3	A	H	A	A	H	H
	4	H	A	H	H	A	A
	5	A	H	A	H	A	H

Home–Away pattern

Fig. 9.

exactly one break each and the two remaining teams have no break; this is clearly optimal (see de Werra [10]). The construction is tabulated in Fig. 9 for a league of $2n = 6$ teams; the arrow above each game indicates where the game is played (where $\vec{ij}$ stands for the arc ij). We have underlined the Hs and As corresponding to breaks: teams 1 and 6 have no break, and all others have one break each.

In practice there are many additional requirements to be taken into account. One of these is the fact that some stadia are not always available: for each day with n games, n stadia are used and n stadia are unused. If the stadium of team i is not available on day d, it means that the game played by team i must be an (A) for i, and the edge adjacent to vertex i that receives colour d should be an arc ij for some j. In addition, it may be necessary to schedule some games between teams i and j on one of certain specific days; this gives another kind of list-colouring. There may be additional constraints related to the geography of the league: the sequence of opponents of a given team i may have to be arranged in such a way that team i does not have too many long distances to travel. And there are many more. So while the problem is basically still an edge-colouring problem with orientations to determine, its difficulty may be greatly increased, and other techniques such as integer programming or constraint programming have to be used (see Rassmussen and Trick [34] and Kendall *et al.* [31]).

10. Balancing requirements

We have not considered the use of resources in the scheduling applications discussed so far. However, one usually has some resources available in limited amounts. For this reason, in our chromatic scheduling models consisting of a sequence of k one-period schedules, it may be wise to balance the consumption of resources among the k periods so as to reduce as much as possible the maximum consumption in the different periods.

In the basic class−teacher timetabling problem, for instance, the number of classrooms is limited. We consider them as the resources consumed by the lectures. Assuming that the classrooms are all available during the k teaching periods of the

https://doi.org/10.1017/CBO9781139519793.015 Published online by Cambridge University Press

week, we should construct a timetable (here it is an edge-colouring $M_1, M_2, \ldots, M_k$) in such a way that $\max_i |M_i|$ is as small as possible, where the size $|M_i|$ of the matching M_i is precisely the number of classrooms needed during period i.

It is well known that whenever an edge-colouring $M_1, M_2, \ldots, M_k$ exists, there is a *balanced edge-colouring* $M'_1, M'_2, \ldots, M'_k$ with $-1 \leq |M'_i| - |M'_j| \leq 1$, for all $i, j \leq k$. The construction of such a colouring can be easily achieved by trying to balance the cardinalities of two matchings M_i and M_j with $|M_j| - |M_i| \geq 2$. If $G = (V, E)$ is the bipartite graph associated to the class−teacher timetabling problem, and if $k \geq \Delta(G)$ is the number of teaching periods, we need $\lceil |E|/k \rceil$ classrooms. If the number c of classrooms is given, then we can find a timetable in $\max\{\Delta(G), \lceil |E|/c \rceil\}$ periods.

We notice at this stage that the above reasoning assumes that lectures can be moved arbitrarily inside the set C of teaching periods. As soon as we have lists $L(e)$ of available colours for some edges e, this may not work any more.

The above discussion shows that perfect balancing is always possible for the basic edge-colouring problem. The same holds for vertex-colouring in claw-free graphs (graphs that contain no $K_{1,3}$ as an induced subgraph). The reason is that the union of any two colour classes S_i, S_j in a vertex-colouring of a claw-free graph induces a subgraph that consists of even cycles and elementary paths, so that balancing between S_i and S_j is possible. It follows that perfect balancing is possible for any vertex-colouring $S_1, S_2, \ldots, S_k$.

However, for arbitrary graphs perfect balancing is not possible in general: in $K_{1,3}$ any vertex-colouring with $k = 2$ colours has $\big||S_1| - |S_2|\big| = 2$. But using more colours may help the balancing procedure; it has been shown by Hajnal and Szemerédi [27] that, for any graph G, there exists a balanced vertex-colouring with k colours when $k \geq \Delta(G) + 1$; a shorter proof is given by Kierstead and Kostochka [32]. Notice that the set of integers k for which a graph G admits a balanced vertex k-colouring is not necessarily an interval; for example, the graph $K_{3,3}$ in Fig. 10 has a balanced 2-colouring with $S_1 = \{a, b, c\}$, $S_2 = \{d, e, f\}$, and a balanced 4-colouring with $S_1 = \{a, b\}$, $S_2 = \{c\}$, $S_3 = \{d, e\}$, $S_4 = \{f\}$, but no balanced 3-colouring.

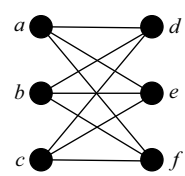


Fig. 10.

For the practice of timetabling, we should also be able to solve the following problem: for a fixed number k of periods (colours), find a schedule (a vertex-colouring) $S_1, S_2, \ldots, S_k$ such that $\max_{1 \leq i \leq k} |S_i| - \min_{1 \leq j \leq k} |S_j|$ is minimum among all possible vertex-colourings with k colours. Finding this optimal value is an NP-hard problem in general, but polynomial algorithms exist for some types of graphs

https://doi.org/10.1017/CBO9781139519793.015 Published online by Cambridge University Press

such as trees (see Hertz and Ries [30]). Partial results on balancing in vertex-colouring can be found in de Werra [8]. It has been shown in particular that if G is a graph in which no vertex belongs to more than q maximal cliques, then any vertex-colouring $S_1, S_2, \ldots, S_k$ can be transformed to satisfy:

$$|S_i| - |S_j| \leq (q-2) \min \left\{ |S_i|, |S_j| \right\} + 1, \text{ for all } i, j \leq k.$$

Another way of formulating this is to say that, for any $k \geq \chi(G)$, a graph G has a k-colouring $S_1, S_2, \cdots, S_k$ with

$$|S_1| \leq |S_2| \leq \ldots \leq |S_k| \leq (q-1)|S_1| + 1.$$

More generally, we may wonder what can be done in the case where the amount of resource available is not the same for each of the k periods of the schedule; for instance, if h_i is the amount of resource available at period i, does there always exist a schedule represented by a vertex-colouring $S_1, S_2, \ldots, S_k$ with $|S_i| \leq h_i$, for $i = 1, 2, \ldots, k$? This is generally a difficult problem. It has been shown that, even if G is the line graph of a bipartite graph G with $\Delta(G) = 3$, the problem of deciding whether a colouring S_1, S_2, S_3 exists with $|S_i| \leq h_i$ for $i = 1, 2, 3$ is NP-complete (see Even *et al.* [20]). However, if G is the line graph of a tree of bounded degree, then a polynomial algorithm exists for finding such a colouring if it exists (see de Werra *et al.* [13]).

Let us now return to the basic class−teacher timetabling problem for which the model is an edge-colouring $M_1, M_2, \ldots, M_k$ in a bipartite multigraph $G = (V, E)$. We recall that the left-hand set of vertices corresponds to the m classes c_i, and that the right-hand set corresponds to the n teachers t_j. For each pair c_i, t_j, let $m_G(c_i, t_j)$ be the number of parallel edges between c_i and t_j in G: these represent the one-period lectures of teacher t_j to class c_i. We assume now that we have to construct a timetable for a week of k days and that s is the number of teaching periods in any day of the week.

If G is the bipartite multigraph representing the data of our basic class−teacher problem, then for a solution to exist we must have:

$$ks \geq \Delta(G). \tag{1}$$

A timetable corresponds to an edge-colouring $M_1, M_2, \ldots, M_{ks}$. For $r = 1, 2, \ldots, k$, let $N_r = \bigcup_{p=(r-1)s+1}^{rs} M_p \subseteq E$. Then N_r is the set of lectures (edges) assigned to the s periods of day r.

For a subset N_r of edges, we denote by $d_{N_r}(x)$ the degree of the vertex x, and by $m_{N_r}(x, y)$ the number of edges between the vertices x and y in the subgraph of $G = (V, E)$ generated by $N_r \subseteq E$.

In a basic class−teacher timetabling problem, represented by a bipartite multigraph $G = (V, E)$ with a number k of days and s of teaching periods in a day satisfying (2), we say that an edge-colouring $M_1, M_2, \ldots, M_{ks}$ is *perfectly balanced* (on k days) if it satisfies:

$$\lfloor |E|/ks \rfloor \leq |M_t| \leq \lceil |E|/ks \rceil \quad (t = 1, 2 \ldots, ks) \tag{2}$$

https://doi.org/10.1017/CBO9781139519793.015 Published online by Cambridge University Press

(the numbers of lectures scheduled at any period are all within 1 of each other) and, for each $r = 1, 2, \ldots, k$,

$$\lfloor |E|/k \rfloor \leq |N_r| \leq \lceil |E|/k \rceil \tag{3}$$

(the total numbers of lectures scheduled on any day are all within 1 of each other).

Also,

$$\lfloor d_G(c_i)/k \rfloor \leq d_{N_r}(c_i) \leq \lceil d_G(c_i)/k \rceil \quad (i = 1, 2, \ldots, m) \tag{4}$$

(the numbers of lectures involving class c_i on any day are all within 1 of each other),

$$\lfloor d_G(t_j)/k \rfloor \leq d_{N_r}(t_j) \leq \lceil d_G(t_j)/k \rceil \quad (j = 1, 2, \ldots, n) \tag{5}$$

(the numbers of lectures involving teacher t_j on any day are all within 1 of each other), and

$$\lfloor m_G(c_i, t_j)/k \rfloor \leq m_{N_r}(c_i, t_j) \leq \lceil m_G(c_i, t_j)/k \rceil \quad (i = 1, 2, \ldots, m; j = 1, 2, \ldots, n) \tag{6}$$

(the lectures of teacher t_j to clas c_i are spread uniformly among the k days, for all teachers t_j and all classes c_i).

By analogy we say that the timetable associated with such an edge-colouring is also perfectly balanced.

One can show (see de Werra [9]) that for the basic class−teacher model there exists a perfectly balanced timetable on k days with s teaching periods per day, for any choice of k, s satisfying $ks \geq \Delta(G)$. As an example, consider the problem represented in Fig. 11(a) by the array giving $m_G(c_i, t_j)$ in entry j, i; we have $k = 3$ and $s = 2$.

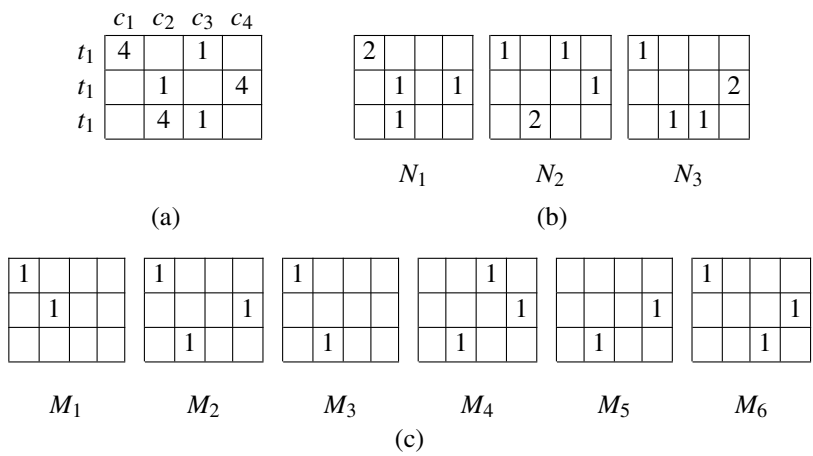


Fig. 11.

https://doi.org/10.1017/CBO9781139519793.015 Published online by Cambridge University Press

Fig. 11(b) shows the sets N_r of lectures assigned to each day $r \leq k$ $(= 3)$, and Fig. 11(c) shows the edge-colouring with $ks = 6$ colours representing the lectures assigned for each period of the week; it is obtained by constructing a balanced edge-colouring of each of the subgraphs generated by N_r. We see that the resulting edge-colouring $M_1, M_2, \ldots, M_{ks}$ of G satisfies (2).

Notice also that the above example admits a balanced timetable $M'_1, M'_2, \ldots, M'_5$ with $\Delta(G) = 5$ periods ($|M'_t| = 3, t = 1, 2, \ldots, 5$), but then the constraints (3) do not hold with $k = 3$.

11. Compactness

There are many quality criteria that should be considered when we estimate the value of a schedule. One of these is the *compactness* (see Brélaz *et al.* [5]). Going back to the basic (preemptive or non-preemptive) open shop problem, we may require that the working periods of each processor be consecutive – that is, we may have idle periods at the beginning and/or end of the set of k periods used. Similarly, we may require that the periods during which some operation of a fixed job J_j is in process be consecutive (this does not imply that the periods of processing a single operation are consecutive). If such a schedule can be found, we say that the schedule is *compact*, because each job and each processor has a compact schedule; such schedules are sometimes called 'no wait' schedules. But such a schedule (represented by an edge-colouring of a graph G) may not always exist. For example, consider the case when we have two processors and three jobs consisting each of one operation, with unit processing time on each processor: in any edge-colouring with three colours, there must be one job with a non-compact schedule (see Fig. 12(a) with an idle period for J_3). Observe, however, that there is a compact schedule that uses four colours, as illustrated in Fig. 12(b).

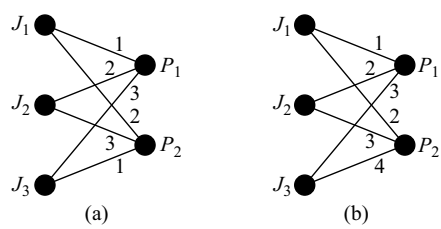


Fig. 12.

In class−teacher timetabling, it is also desirable to have schedules that are compact for each class (so idle times for the pupils between lectures are to be avoided), and hopefully also for the teachers. So, if compact schedules cannot always be found, we have to introduce a measure of distance to compactness that can be minimized, or at least bounded.

https://doi.org/10.1017/CBO9781139519793.015 Published online by Cambridge University Press

For an edge-colouring c we denote by $C(v,c)$ the set of colours assigned to the edges adjacent to vertex v. The *deficiency* $D(v,c)$ of a colouring c at a vertex v is the minimum number of integers to be added to $C(v,c)$ to form an interval. The *deficiency* $D(c)$ of an edge-colouring c of a graph $G = (V,E)$ is the sum $\sum_{v\in V} D(v,c)$. It has been shown by Giaro [22] that finding an edge-colouring with minimum deficiency is NP-hard, even if G is bipartite.

An edge-colouring of G with $D(c) = 0$ (so that $D(v,c) = 0$ for each $v \in V$) is sometimes called a *consecutive colouring*. We define the *deficiency* $\mathrm{def}(G)$ of G as the minimum value of $D(c)$ taken over all edge-colourings of G. An edge-colouring c with $D(c) = \mathrm{def}(G)$ is called *optimal*. Figure 12 illustrates the fact that $\mathrm{def}(K_{2,3}) = 0$, even if the deficiency of each edge-colouring of $K_{2,3}$ with three colours is strictly positive. Beside partial results obtained for bipartite graphs, k-regular graphs, odd cycles, wheels and complete graphs (see Giaro *et al.* [23], Schwartz [39] and Bouchard *et al.* [4]), very little is known about the deficiency of an arbitrary graph.

Analogous to what was done for interval colourings, we may introduce the idea of *cyclic compactness*: assuming that the schedule is to be repeated, we may require that the colours assigned to edges adjacent to any vertex form a cyclic interval in $C = \{1,2,\ldots,k\}$; this problem has been formulated and studied in de Werra and Solot [14] and Altinakar *et al.* [1]. Some classes of graphs have been characterized for which cyclic compact interval edge-colourings do exist, whatever the numbers of parallel edges between pairs of vertices. This provides some families of non-preemptive open shop problems with cyclic compactness requirements which can be solved by means of edge-colouring techniques.

12. Conclusion

We have shown through the motivation of applications how numerous variations and extensions of the basic vertex-colouring and edge-colouring models have been introduced and studied. Our purpose was to give an idea of the many directions that have been explored. The development of technologies will undoubtedly create a need for unsuspected colouring models in the future, thus opening original avenues of research.

In this chapter we have not concentrated on solution techniques; this was not our aim. But we should not forget that for all the models presented here there is an urgent need to develop efficient algorithmic procedures that yield either 'optimal' solutions (provided that one can reasonably define a concept of optimality), or at least good approximations to such solutions. The necessity of providing such procedures is enhanced by the fact that in all these applications we are dealing with graphs of large size.

We hope that the above colouring models, which may be viewed as the core of chromatic scheduling, will stimulate the interest of future researchers and users.

https://doi.org/10.1017/CBO9781139519793.015 Published online by Cambridge University Press

References

1. S. Altinakar, G. Caporossi and A. Hertz, On compact k-edge colorings: a polynomial time reduction from linear to cyclic, *Discrete Optimization* **8** (2011), 502–512.
2. C. Archetti, N. Bianchessi and A. Hertz, A branch-and-price algorithm for the robust graph coloring problem, *Les Cahiers du Gerad,* G-2011-75, Montréal, 2011.
3. M. Bouchard, M. Čangalović and A. Hertz, About equivalent interval colorings of weighted graphs, *Discrete Appl. Math.* **157** (2009), 3615–3624.
4. M. Bouchard, A. Hertz and G. Desaulniers, Lower bounds and a tabu search algorithm for the minimum deficiency problem, *J. Combin. Optimization* **17** (2009), 168–191.
5. D. Brélaz, Y. Nicolier and D. de Werra, Compactness and balancing in scheduling, *Zeitschrift für Operations Research* **21** (1977), 65–73.
6. E. K. Burke, D. de Werra and J. Kingston, Applications to timetabling, *Handbook of Graph Theory* (eds. J. L. Gross and J. Yellen), CRC Press (2004), 445–474.
7. D. Cartwright and F. Harary, Structural balance: a generalization of Heider's theory, *Psychological Review* **63** (1956), 277–293.
8. D. de Werra, A note on graph coloring, *RAIRO, Revue Française d'Automatique, d'Informatique et de Recherche Opérationnelle* **R-1** (1974), 49–53.
9. D. de Werra, A few remarks on chromatic scheduling, *Combinatorial programming, Methods and Applications* (ed. B. Roy), D. Reidel (1975), 337–342.
10. D. de Werra, Some models of graphs for scheduling sports competitions, *Discrete Appl. Math.* **21** (1988), 47–65.
11. D. de Werra, Graph theoretical models for preemptive scheduling, *Advances in Project Scheduling* (eds. R. Slowinski and J. Weglarz), Elsevier (1989), 171–185.
12. D. de Werra and Y. Gay, Chromatic scheduling and frequency assignment, *Discrete Appl. Math.* **49** (1994), 165–174.
13. D. de Werra, A. Hertz, D. Kobler and N. V. R. Mahadev, Feasible edge colorings of trees with cardinality constraints, *Discrete Math.* **222** (2000), 61–72.
14. D. de Werra and Ph. Solot, Compact cylindrical chromatic scheduling, *SIAM J. Discrete Math.* **4** (1991), 528–534.
15. M. Demange, D. de Werra, J. Monnot and V. Paschos, Time slot scheduling of compatible jobs, *J. Scheduling* **10** (2007), 111–127.
16. M. Demange, T. Ekim and D. de Werra, On the approximation of min split coloring and min cocoloring, *J. Graph Algorithms and Appl.* **10** (2006), 297–315.
17. M. Demange, T. Ekim and D. de Werra, A tutorial on the use of graph coloring for some problems in robotics, *Europ. J. Oper. Res.* **192** (2009), 41–55.
18. T. Ekim and D. de Werra, On split-coloring problems, *J. Combin. Optimization* **10** (2005), 211–225.
19. P. Erdős, A. L. Rubin and H. Taylor, Choosability in graphs, *Congr. Numer.* **26** (1979), 125–157.
20. S. Even, A. Itai and A. Shamir, On the complexity of timetable and multicommodity flow problems, *SIAM J. Comput.* **5** (1976), 691–703.
21. M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*, Freeman, 1979.
22. K. Giaro, The complexity of consecutive Δ-coloring of bipartite graphs: 4 is easy, 5 is hard, *Ars Combin.* **47** (1997), 287–298.
23. K. Giaro, M. Kubale and M. Malafiejski, Consecutive colorings of the edges of general graphs, *Discrete Math.* **236** (2001), 131–143.
24. J. Gimbel, D. Kratsch and L. Stewart, On cocolourings and cochromatic numbers of graphs, *Discrete Appl. Math.* **48** (1994), 111–127.

https://doi.org/10.1017/CBO9781139519793.015 Published online by Cambridge University Press

25. T. Gonzales and S. Sahni, Open shop scheduling to minimize finish time, *J. ACM* **23** (1976), 665–679.
26. S. Gutner, The complexity of planar graph choosability, *Discrete Math.* **159** (1996), 119–130.
27. A. Hajnal and E. Szemerédi, Proof of a conjecture of P. Erdős, *Combinatorial Theory and its Applications*, Balatonfüred (eds. P. Erdős, A. Rényi and V. T. Sós), North-Holland (1970), 601–623.
28. M. M. Halldórsson and G. Kortsarz, Multicoloring: problems and techniques, *Math. Foundations of Computer Science*, LNCS **3153** (2004), 25–41.
29. P. Hansen, J. Kuplinski and D. de Werra, Mixed graph coloring, *Math. Methods of Operations Research* **45** (1997), 145–160.
30. A. Hertz and B. Ries, On *r*-equitable colorings of trees and forests, *Les Cahiers du Gerad,* G-2011-40, Montréal, 2011.
31. G. Kendall, S. Knust, C. C. Ribeiro and S. Urrutia, Scheduling in sports: an annotated bibliography, *Computers Oper. Res.* **37** (2010), 1–19.
32. H. A. Kierstead and A. V. Kostochka, A short proof of the Hajnal–Szemerédi theorem on equitable coloring, *Combin. Probab. Comput.* **17** (2008), 265–270.
33. S. Prestwich, Constrained bandwidth multicoloration neighborhoods, *Proc. Computational Symp. on Graph Coloring and its Generalizations*, Ithaca (eds. D. S. Johnson, A. Mehrotra and M. Trick), (2002), 126–133.
34. R. V. Rassmussen and M. A. Trick, The timetable constrained distance minimization problem, *Annals of Operations Research* **171** (2006), 167–181.
35. B. Ries, Coloring some classes of mixed graph, *Discrete Appl. Math.* **155** (2007), 1–6.
36. B. Ries and D. de Werra, On two coloring problems in mixed graphs, *Europ. J. Combin.* **29** (2008), 712–725.
37. B. Robillard, *Vérification Formelle et Optimisation de l'Allocation de Registres*, Ph.D. Thesis, CNAM, Paris, 2010.
38. B. Roy, Nombre chromatique et plus longs chemins d'un graphe, *RAIRO, Revue Française d'Automatique, d'Informatique et de Recherche Opérationnelle* **1** (1967), 129–132.
39. A. Schwartz, The deficiency of a regular graph, *Discrete Math.* **306** (2006), 1947–1954.
40. Y. N. Sotskov, A. Dolgui and F. Werner, Mixed graph coloring for unit-time job-shop scheduling, *Internat. J. Math. Algorithms* **4** (2001), 289–323.
41. M. Čangalović and J. A. M. Schreuder, Exact colouring algorithm for weighted graphs applied to timetabling problems with lectures of different lengths, *Europ. J. Operational Research* **51** (1991), 248–258.
42. M. Voigt, List colourings of planar graphs, *Discrete Math.* **120** (1993), 215–219.
43. J. Yánez and J. Ramírez, The robust coloring problem, *Europ. J. Operational Research* **148** (2003), 546–558.

https://doi.org/10.1017/CBO9781139519793.015 Published online by Cambridge University Press