Chapter VI: Colouring

Graph colouring is one of the fundamental topics in graph theory. It consists of assigning colours to certain elements of a graph (its vertices or its edges) subject to specific constraints. The main objective is to perform this colouring using the smallest possible number of colours while ensuring that adjacent elements do not share the same colour.

This chapter presents the essential concepts and results related to graph colouring.

In the first section, we introduce the basic definitions of vertex and edge colouring and the associated notions such as chromatic number and chromatic index.

The second section focuses on vertex colouring, which consists of partitioning the set of vertices into stable subsets.

The third section deals with arc (edge) colouring, where colours are assigned to the edges of the graph so that adjacent edges receive different colours.

The fourth section recalls some important propositions and theorems, including bounds on the chromatic number.

The fifth section presents the famous Four Colour Theorem, which applies to planar graphs and states that any planar map can be coloured with at most four colours.

Finally, the sixth section introduces the notion of perfect graphs, an important class of graphs in which the chromatic number equals the size of the largest clique for every induced subgraph.

1. **Definitions**:

Two types of colouring are defined: *vertex colouring* and *edge colouring*.

The vertex colouring (respectively edge colouring) of a multigraph $\mathbf{G}$ corresponds to assigning a colour to each vertex (respectively each edge) in such a way that two adjacent vertices (respectively edges) do not share the same colour.

We call the *chromatic number* $\boldsymbol{\gamma}(\mathbf{G})$ (respectively the *chromatic index* $\mathbf{q}(\mathbf{G})$) the minimum number of distinct colours required to perform a vertex colouring (respectively an edge colouring) of $\mathbf{G}$.

2. Vertex Colouring:

Let $\mathbf{G}=(\mathbf{X},\mathbf{U})$ a multigraph. A subset of vertices of $\mathbf{X}$ is called a *stable set* if it contains only vertices that are pairwise non-adjacent.

The stability number $\boldsymbol{\alpha}(\mathbf{G})$ is the maximum cardinality of a stable subset. A vertex colouring is therefore a partition of the vertex set into stable subsets.

$\mathbf{G}$ is said to be ***p**-chromatic* (or $\mathbf{q}$-colourable) if its vertices admit a colouring with **p** colours.

Property:

We can prove that: $\boldsymbol{\gamma}(\mathbf{G}) \geq \dfrac{|\mathbf{X}|}{\boldsymbol{\alpha}(\mathbf{G})}$.

Example:

In the graph shown below, the subset $\{\mathbf{a},\mathbf{b},\mathbf{e}\}$ forms a stable set of maximum cardinality. The minimum number of distinct colours required to achieve a proper vertex colouring of the graph is 3.

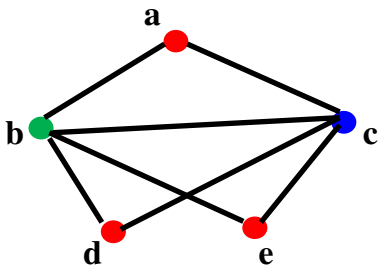


Figure 1: A minimum set stable and vertex colouring.

We have, $\gamma(\mathbf{G}) = 3 \geq \dfrac{|\mathbf{X}|}{\alpha(\mathbf{G})} = \dfrac{5}{3}$.

Brélaz (DSATUR) Algorithm (1979)

The **DSATUR** (Degree of Saturation) algorithm is a heuristic method for vertex colouring that aims to use the smallest possible number of colours.

Main idea:

At each step, the algorithm selects the *most constrained* uncoloured vertex, that is, the vertex whose neighbours already have the largest number of distinct colours (called its saturation degree). If several vertices have the same saturation degree, the one with the highest degree in the graph is chosen.

Principle:

1. Start by colouring the vertex with the highest degree using the first colour.
2. Compute the saturation degree of all uncoloured vertices.
3. Select the uncoloured vertex with the highest saturation degree (break ties by choosing the vertex with the highest degree).
4. Assign to this vertex the *smallest colour* not used by its neighbours. (The smallest colour refers to the first colour, in the ordered list of already used colours, that is not assigned to any

neighbour of the vertex under consideration. This ensures that colours are reused whenever possible, avoiding unnecessary introduction of new colours and leading to a more efficient colouring of the graph).

5. Update the saturation degrees of adjacent uncoloured vertices.
6. Repeat steps 3–5 until all vertices are coloured.

Example:

Consider the following graph $\mathbf{G} = (\mathbf{X}, \mathbf{U})$

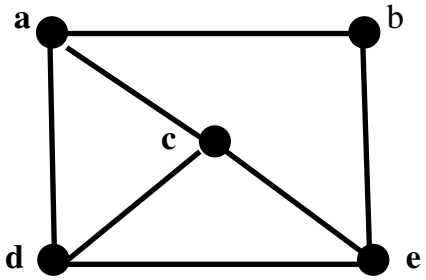


Figure 2: Vertex Colouring.

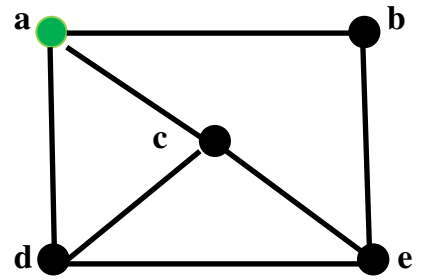
Step 1: Computation of vertex degrees

The degrees of the vertices are:

$\mathbf{d}_G(\mathbf{a}) = 3,\ \mathbf{d}_G(\mathbf{b}) = 2,\ \mathbf{d}_G(\mathbf{c}) = 3,\ \mathbf{d}_G(\mathbf{d}) = 3,\ \mathbf{d}_G(\mathbf{e}) = 3.$

Step 2: Initial colouring

Among the vertices of maximum degree, **a**, **c**, **d**, **e**, we select vertex **a** to start. Assign **Colour 1** (**green**) to vertex **a**.



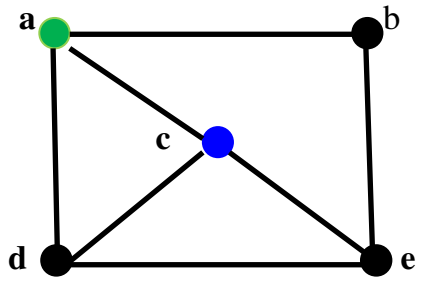
Step 3: Computation of saturation degrees

The saturation degree of a vertex is the number of distinct colours used in its neighbourhood. After colouring vertex **a**, its neighbours **b**, c, d each have one coloured neighbour, hence their saturation degree is 1.

Vertex **e** is not adjacent to **a**, so its saturation degree is 0.

Step 4: Selection of the next vertex

The vertices **b**, **c**, **d** have the highest saturation degree (equal to 1). To break the tie, we select the vertex with the highest degree. Choose vertex **c** (degree 3). Its neighbour **a** already uses Colour 1; therefore, assign Colour 2 (**blue**) to **c**.

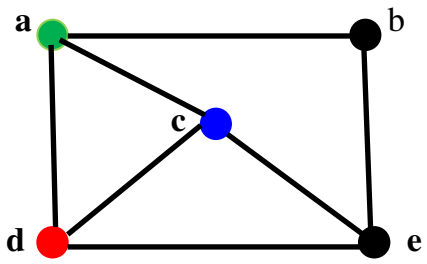


Step 5: Update of saturation degrees

- Vertex **d**, adjacent to both **a** (Colour 1) and **c** (Colour 2), now has two distinct colours in its neighbourhood, so its saturation degree becomes 2.
- Vertex **e**, adjacent to **c**, has one colour in its neighbourhood, so its saturation degree is 1.
- Vertex **b** remains with saturation degree 1.

Step 6: Next vertex selection

Vertex **d** has the highest saturation degree (2). Its neighbours are **a** (Colour 1), **c** (Colour 2), and **e** (uncoloured). The smallest available colour not used by its neighbours is **Colour 3**, thus assign Colour 3 (**red**) to vertex **d**.

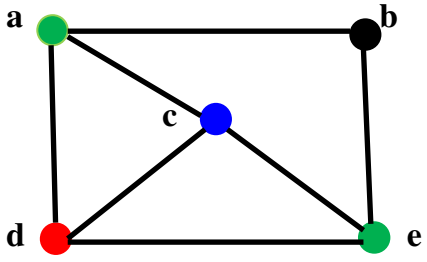


Step 7: Update of remaining vertices

- Vertex **b**: neighbours **a** (Colour 1) and **e** (uncoloured), then the saturation degree = 1.
- Vertex **e**: neighbours **b** (uncoloured), **c** (Colour 2), and **d** (Colour 3) then the saturation degree = 2.

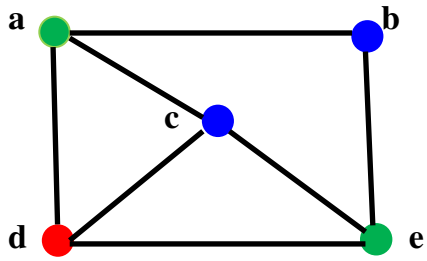
Step 8: Next vertex

Vertex **e** has the highest saturation degree (2). Its neighbours **c** and **d** use Colours 2 and 3 respectively, hence the smallest available colour is **Colour 1** (**green**). Assign Colour 1 to vertex **e**.



Step 9: Final vertex

Finally, only vertex **b** remains uncoloured. Its neighbours **a** and **e** both have Colour 1. Therefore, assign the next available colour: Colour 2 (**blue**) to **b**.



Step 10: Final colouring

The final vertex colouring is: $\mathbf{a}(1),\ \mathrm{b}(2),\ \mathrm{c}(2),\ \mathrm{d}(3),\ \mathrm{d}(1)$.

No two adjacent vertices share the same colour, and the colouring uses three colours in total. Thus, the the *chromatic number* $\boldsymbol{\gamma}(\mathbf{G}) \leq 3$

3. **Edge Colouring:**

Let us first recall the notion of a *matching*. Let $\mathbf{G}$ be a multigraph without loops. A matching is defined to be a set $\mathbf{E}$, of edges such that no two edges of $\mathbf{E}$ are adjacent.

The *edge colouring* of $\mathbf{G}$ is the smallest integer $\mathbf{q}(\mathbf{G})$ having the following property: it is possible, with $\mathbf{q}(\mathbf{G})$ colours, to colour the edges of $\mathbf{G}$ so that two adjacent edges do not share the same colour.

In other words, an edge colouring is a partition of the set of edges into classes that are *matchings*.

The following edge colouring algorithm is a simple and efficient heuristic that performs well for small or practical graphs, although it does not always guarantee the use of the minimum possible number of colours.

Algorithm:

Input: A multigraph $\mathbf{G}=(\mathbf{X},\mathbf{U})$.

Output: A colouring of the edges of $\mathbf{G}$ such that no two adjacent edges share the same colour.

Step 1: List all edges of the graph in any order.

Step 2: Start with the first edge and assign it the first colour (Colour 1).

Step 3: For each next edge $\mathbf{e}$ in the list:

- Look at all edges adjacent to $\mathbf{e}$ that are already coloured.
- Choose the smallest colour (the first colour in order) that is not used by any adjacent edge.
- Assign that colour to $\mathbf{e}$.

Step 4:

Repeat Step 3 until all edges are coloured.

Example:

Consider the graph

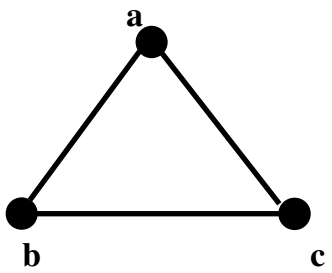


Figure 3: Edges Colouring.

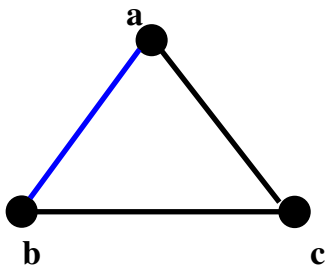
This is a *triangle graph* (a clique $\mathbf{K}_3$).

Step 1: List the edges

We list the edges in order : $\mathbf{e}_1=\{\mathbf{a},\mathbf{b}\}$, $\mathbf{e}_2=\{\mathbf{b},\mathbf{c}\}$, $\mathbf{e}_3=\{\mathbf{c},\mathbf{a}\}$.

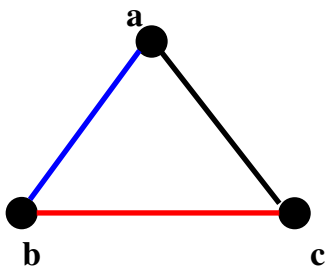
Step 2: Colour the first edge

Assign Colour 1 (blue) to edge $\mathbf{e}_1=\{\mathbf{a},\mathbf{b}\}$.



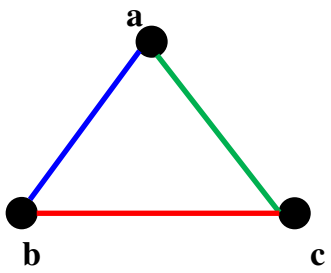
Step 3: Colour the next edge

Edge $\mathbf{e}_2 = \{\mathbf{b}, \mathbf{c}\}$ is adjacent to $\mathbf{e}_1$ (they share vertex $\mathbf{e}$. So, we cannot reuse Colour 1. Assign Colour 2 (**red**) to $\mathbf{e}_2$.



Step 4: Colour the last edge

Edge $\mathbf{e}_3 = \{\mathbf{c}, \mathbf{a}\}$ is adjacent to both $\mathbf{e}_1$ and $\mathbf{e}_2$ at $\mathbf{a}$ and $\mathbf{c}$. So, it cannot use Colour 1 or Colour 2. Asign Colour 3 (**green**) to $\mathbf{e}_3$.



The final edge colouring is: $\mathbf{e}_1(1),\ \mathrm{e}_2(2),\ \mathrm{e}_3(3)$.

No two adjacent edges share the same colour, and the colouring uses three colours in total. Thus, the the *chromatic index* $\mathbf{q}(\mathbf{G}) \leq 3$.

4. **Proposals:**

For a multigraph $\mathbf{G} = (\mathbf{X}, \mathbf{U})$, we have the following two properties:

Properties:

4. $\mathbf{q}(\mathbf{G}) \geq \max_{\mathbf{x} \in \mathbf{X}} (\mathbf{d}_{\mathbf{G}}(\mathbf{x}))$

5. $\boldsymbol{\gamma}(\mathbf{G}) \leq \max_{\mathbf{x} \in \mathbf{X}} (\mathbf{d}_{\mathbf{G}}(\mathbf{x})) + 1$

Brooks' Theorem (1941):

Let $\mathbf{G}$ be a connected simple graph with maximum degree $\mathbf{p}$. Then $\mathbf{G}$ is $\mathbf{p}$-colourable, unless
(1) $\mathbf{p} \neq 2$, and $\mathbf{G}$ is a $(\mathbf{p}+1)$-clique, or
(2) $\mathbf{p} = 2$, and $\mathbf{G}$ is an odd cycle.

Berge's Theorem (1970):

The chromatic index of a bipartite multigraph $\mathbf{G}$ with maximum degree $\mathbf{p}$ is $\mathbf{q}(\mathbf{G}) = \mathbf{p}$.

5. The Four-Colour Theorem

Every planar graph is 4-chromatic.

The Four-Colour Theorem states that it is possible, using only four distinct colours, to colour any map divided into connected regions in such a way that two adjacent regions—that is, regions sharing a common boundary (and not merely a single point)—always receive different colours.

The theorem can be expressed in several equivalent forms, such as the colouring of the faces of a polyhedron or the colouring of the vertices of a planar graph.

6. Perfect Graph

Let $\mathbf{G}=(\mathbf{X},\mathbf{U})$ be a multigraph, and let $\mathbf{w}(\mathbf{G})$ be the number of vertices in the clique subgraph of $\mathbf{G}$ that has the largest possible size. If for every subgraph $\mathbf{S}$ of $\mathbf{G}$, we have $\boldsymbol{\gamma}(\mathbf{S})=\mathbf{w}(\mathbf{S})$, then $\mathbf{G}$ is called a *perfect graph.*

Examples:

- A clique $\mathbf{K_n}$ is a perfect graph. The largest clique is the graph itself, so $\mathbf{w}(\mathbf{K_n})=\mathbf{n}$. Since all vertices are adjacent, each vertex must have a different colour, so the chromatic number is also $\boldsymbol{\gamma}(\mathbf{K_n})=\mathbf{n}$. Hence, for every subgraph of $\mathbf{K_n}$, the chromatic number equals the size of its largest clique, which satisfies the definition of a perfect graph.
- A bipartite graph is also perfect, because it can always be coloured with two colours, and its largest clique has size 2.