Chapter 12

Graph Theory and Algorithms*

12.1 Graph Coloring

Let $G = (V, E)$ is a graph. Consider the problem of assigning colors to its *vertices* with minimum number of distinct colors possible, such that no two adjacent vertices have the same color. This problem is referred as *vertex graph-coloring.*

Similarly, we can define the problem of assigning colors to the *edges* of G with minimum number of distinct colors possible, such that no two adjacent edges have the same color. This problem is called *edge graph-coloring. In this chapter the discussion of graph-coloring is restricted to vertex-coloring.*

If G has loops, then it is impossible to assign a color to this vertex such that is its color is different from itself. Thus, *graphs with loops are uncolorable.* If G has multiple edges the assigned colors will not be affected. *Throughout this chapter we will assume that all graphs are loopless.* [6]

Graph-coloring has several applications, some of which are listed below:

- Map coloring : Given a map of a country, find the minimum number of colors required to color the states such that no two adjacent states have same color. This is a famous problem is graph-coloring and is called **four color map theorem**.
- Time-Table formation : Assign time slots for final examinations such that two courses with a common student have different slots. [6]

Definition 12.1. A **vertex coloring** of a graph $G = (V, E)$ is a map $c : V \to S$ such that $c(v) = c(w)\ \forall v, w \in V$ whenever v and w are adjacent. [4]

The elements of set S are called as the available *colours*. Our interest is to find the minimum number of colours required for vertex colouring.

*Lecturer: Dr.Anand Mishra. Scribe: Gyan Prabhat *(P21AI001)*.

Definition 12.2. A *k-coloring* of a graph G is a labeling $f : V(G) \rightarrow S$, where $S = (c_1, c_2, \cdots, c_k)$ is a set of k unique colors ($|S| = k$). [6]

- The labels are **colors**.
- The vertices of one color form a **color class**.
- A k-coloring is **proper** if adjacent vertices have different labels.
- A graph is **k-colorable** if it has a proper k-coloring.

Definition 12.3. The minimum k such that G is proper k-colorable is called the **chromatic number** and is denoted by $\chi(G)$. [6]

Some examples of chromatic numbers of different graphs is shown in figure 12.1 [1]

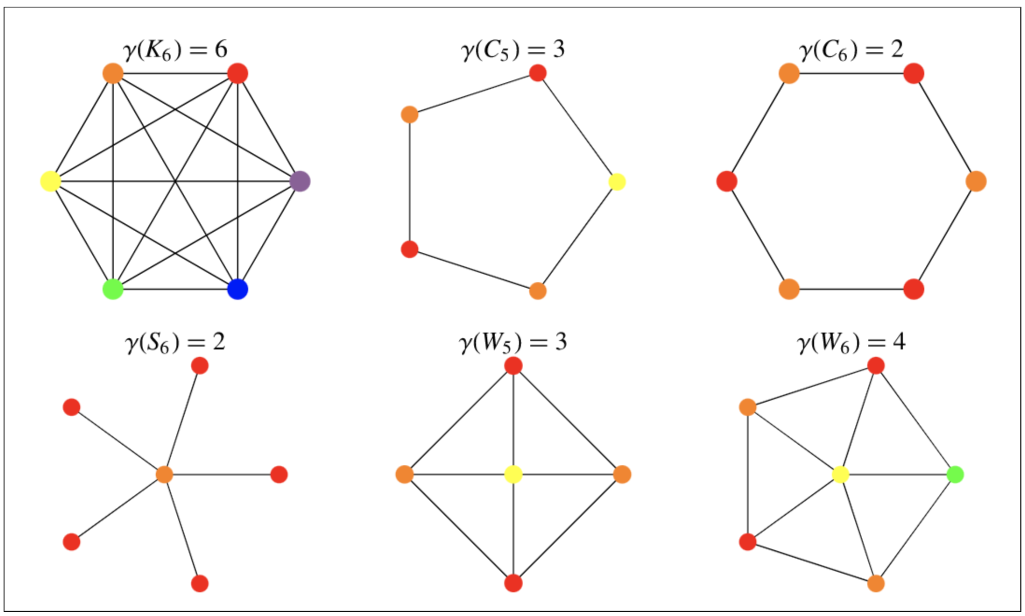


Figure 12.1: Examples of Chromatic Number

Note that, a graph G is k-colorable if $\chi(G) \leq k$ and G is k-chromatic if $\chi(G) = k$
In proper coloring, each color class is an independent set. Thus, *G is k-colorable if and only if $V(G)$ is the union of k independent sets.* Using this result, we can conclude that for a graph k-colorable and k-partite have the same meaning. [6]
Using the result mentioned above for bipartite graphs, we can conclude that :
A graph is 2-colorable if and only if it is bipartite.

Theorem 12.4. *Let $G_1, G_2, \cdots, G_k$ be the k components of a graph G. Prove that:*

$$\chi(G) = \max(\chi(G_1), \chi(G_2), \cdots, \chi(G_k))$$

Proof. Let us take any two components G_i and G_j of G such that $i \neq j$. Notice that, $\forall i, j (i \neq j) \nexists$ any edge between G_i and G_j. Hence, providing a proper coloring to $G_i \ \forall i = 1, 2, \cdots, k$ will produce a proper coloring for G.
Additionally, every proper coloring of G must restrict to a proper coloring for $G_i \ \forall i = 1, 2, \cdots, k$. Thus, we can conclude that:

$$\chi(G) = \max(\chi(G_1), \chi(G_2), \cdots, \chi(G_k))$$

□

12.1.1 Greedy Algorithm for Graph Coloring

Let us consider a graph $G = (V, E)$ where $V = \{v_1, v_2, \cdots, v_n\}$ is the vertex ordering and let $(c_1, c_2, \cdots, c_k)$ be the enumerated list of the available colors.

Step-1 : Choose any vertex v_i randomly and assign color c_1 to it.

Step-2 : Repeat the following for the $(n-1)$ vertices which are not yet colored.

a. Randomly choose any non-colored vertex c_j

b. Find all the vertices adjacent to c_j

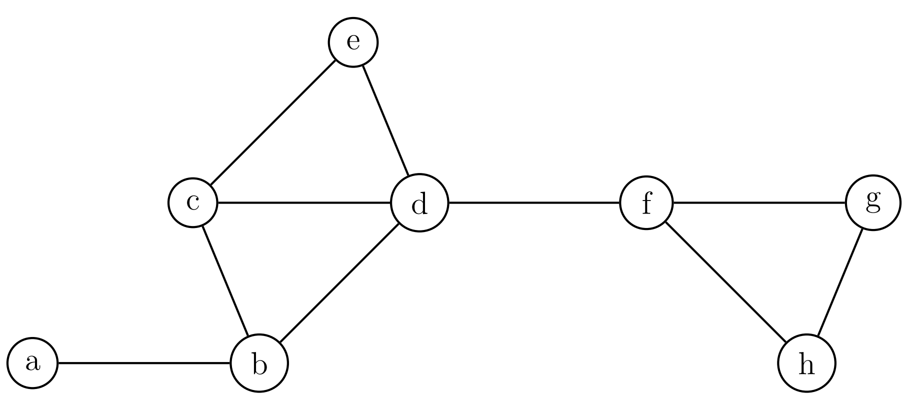
c. Select all the colors *not* used for these adjacent vertices

d. Among these colors, select the color with minimum enumeration

e. Assign this color to c_j

f. If the vertices adjacent to c_j have exhausted all previously used color, then assign a new color to c_j

Example: Let us consider the following graph G. For the vertex ordering a, b, c, d, e, f, g, h, if we apply greedy algorithm for graph coloring, we get the assignment of colors as $1, 2, 1, 3, 2, 1, 2, 3$ respectively.
Total number of unique colors required to color this graph is 3. Hence $\chi(G) = 3$ [6]



12.1.2 Bounds on Chromatic Number ($\chi(G)$)

Finding chromatic number for a graph is solvable in polynomial time only for $k = 2$, but it is a **NP-complete** problem for all $k \geq 3$. [5]

Let us consider a graph $G = (V, E)$. From the definition of chromatic number (12.3), we get that the χ(Null Graph) $= 1$ and $\chi(K_n) = n$. Thus, we can conclude:

- *Trivial lower bound* : $\chi(G) \geq 1$
- *Trivial upper bound* : $\chi(G) \leq |V(G)|$

Definition 12.5. The **clique number** of a graph G is denoted as $\omega(G)$. It is the maximum size of a set of pairwise adjacent vertices (clique) in G. [6]

Proposition 12.6. *For every graph G, $\chi(G) \geq \omega(G)$. [6]*

Proof. From the definition of *clique* (12.5), we can conclude that, vertices of a clique must be assigned distinct colors because they are pairwise adjacent. Thus, we get: $\chi(G) \geq \omega(G)$. □

Proposition 12.7. *For every graph G, $\chi(G) \leq (\Delta(G) + 1)$, where $\Delta(G)$ is the maximum degree of a vertex in G. [6]*

Proof. In a vertex ordering, each vertex has at most $\Delta(G)$ earlier neighbors. If we apply the greedy coloring algorithm, then it cannot be forced to use more than $\Delta(G) + 1$ colors. This proves constructively that $\chi(G) \leq (\Delta(G) + 1)$. □

Chromatic number of some popular graphs are shown in table 12.1

Graph (G)	$\Delta(G)$	$\chi(G)$
Star Graph $(S_n), n > 1$	$(n-1)$	2
Complete Graph (K_n)	n	n
Wheel Graph $(W_n, n > 2)$	$(n-1)$	4 (if n is even) 3 (if n is odd)
Cycle Graph $(C_n, n > 1)$	2	2 (if n is even) 3 (if n is odd)

Table 12.1: Chromatic number of S_n, K_n, W_n and C_n

Proposition 12.8. *(Welsh-Powell [1967]) If a graph G has degree sequence $d_1 \geq d_2 \geq \cdots d_n$, then: [6]*

$$\chi(G) \leq 1 + \max_i(\min\{d_i, i-1\})$$

Proof. Let $v_1, v_2, \cdots, v_n$ be the n vertices of graph $G = (V, E)$ and $d(v_i)$ be the degree of vertex i. We arrange the vertices of G in their non-increasing order of degree. With this vertex ordering, we apply the greedy coloring algorithm.
Consider v_i, it has at most $\min\{d_i, i-1\}$ earlier neighbours, so at most this many colors appear on its earlier neighbors. Hence,the color we assign to v_i is at most $1 + \min\{d_i, i-1\}$ This holds for each vertex, so we maximize over i to obtain the upper bound on the maximum color used. $\square$

12.1.3 Interval Graphs

Definition 12.9. An interval representation of a graph are such graphs for which there exists a set of n finite open intervals on the real line, such that the following two conditions hold: [3] [6]

i. There is a 1 : 1 correspondence between the intervals and the vertices of the graph.

ii. vertices are adjacent if and only if the corresponding intervals intersect.

A graph having such a representation is called an **interval graph**.

An example of an interval graph is shown in figure 12.2

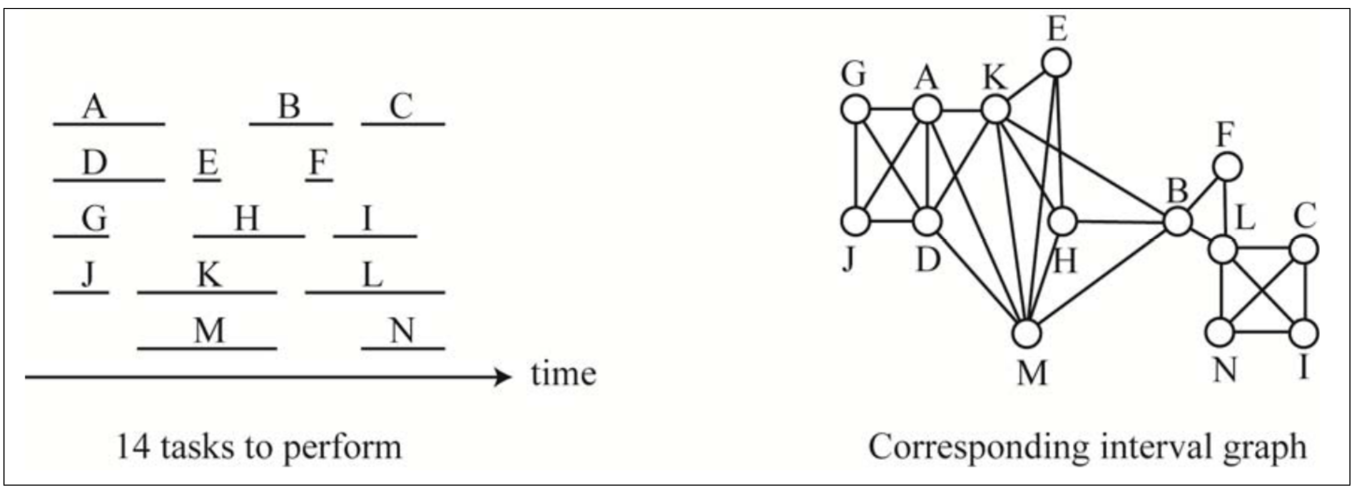


Figure 12.2: Examples of Interval Graph

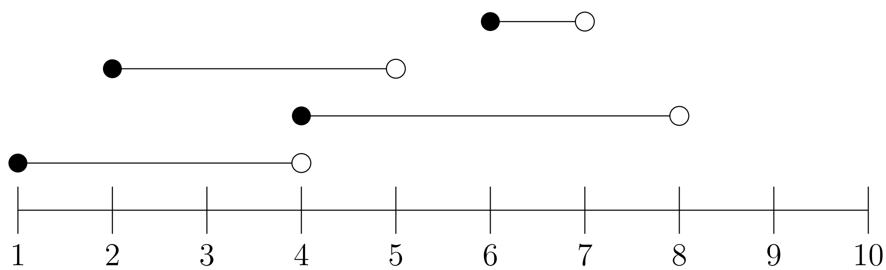
Example : Let us consider the following scheduling problem:
There are 4 events which are supposed to happen in IIT Jodhpur. The start and end time of these events are listed in table 12.2.

Event	Start Time (PM)	End Time (PM)
E1	1	4
E2	4	8
E3	2	5
E4	6	7

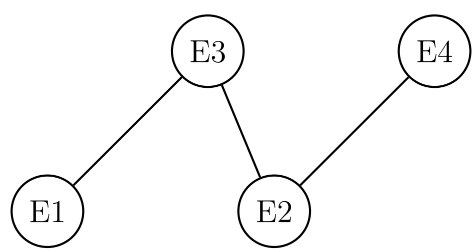
Table 12.2: Start and end time of events

i. If there is only one room available to host all these 4 events, find the maximum number of events that can be hosted such that no two events occur at the same time in the same room?

ii. Find the minimum number of rooms to host all these 4 events such that no two events occur at the same time in the same room?



Using the definition of interval graphs 12.9, let us convert the intervals shown above into a corresponding interval graph:



From this graph, we can observe that:

i. A maximum of two events can be hosted if only 1 room is available. $\{(E1, E2), (E1, E4), (E3, E4)\}$

ii. Finding the minimum number of rooms to host all these 4 events is same as finding the chromatic number of the graph. Here, the chromatic number is 2.

Not every graph is an interval graph : Any arbitrary graph can not be viewed as an interval graph.[2]

- Cycle graphs C_n for $n \geq 4$ are *not* an interval graph.
- Star Graph (S_n) and Complete graph (K_n) are interval graphs.

Definition 12.10. Let $G = (V, E)$ be any graph and $S \subset V$. Then the **induced subgraph** $G[S]$ is the graph whose vertex set is S and whose edge set consists of all of the edges in E that have both endpoints in S.

Characterization of Interval Graphs : A graph that has no *induced subgraph* as C_4 and has *transitive orientation* is an interval graph.

Heuristic to solve the Scheduling Problems

We solve the scheduling problems using greedy algorithms which are based on different heuristics.

Heuristic (1) : *Shortest event first*

Let us consider a scheduling problem where we have 4 events with their start and end time as shown in 12.3.

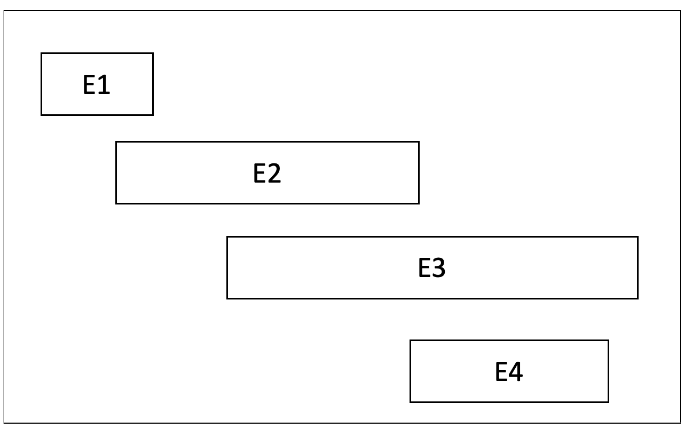


Figure 12.3: Scheduling example

Applying the Heuristic of *Shortest event first*, we get:

- We select the shortest event first, thus event *E1* is selected.
- The second shortest event is *E4*. Also, $E1 \cap E4 = \phi$, thus *E2*. Thus, *E4* is selected.

- As $E1 \cap E2 \neq \varnothing$ and $E4 \cap E3 \neq \varnothing$. Thus, no more selection is possible.
- *Conclusion* : Two events *E1* and *E4* are selected.

Let us consider a scheduling problem where we have 3 events with their start and end time as shown in 12.4.

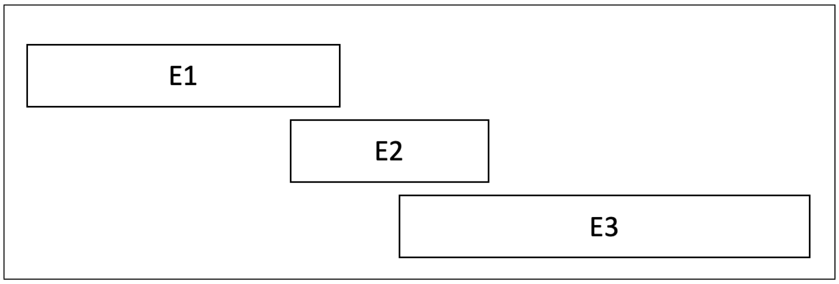


Figure 12.4: Scheduling example

Applying the Heuristic of *Shortest event first*, we get:

- We select the shortest event first, thus event *E2* is selected.
- As $E2 \cap E1 \neq \varnothing$ and $E2 \cap E3 \neq \varnothing$. Thus, no more selection is possible.
- *Conclusion* : Only one event *E2* is selected.

Clearly, we can select two events *E1* and *E3* as $E1 \cap E3 = \phi$. Thus, example 12.4 serves as an **counter-example** for the *Shortest event first* heuristic.

Heuristic (2) : *Earliest Start Time (EST)*
Start with the first event which start the earliest and then select the next possible events that begin as early as possible.

Let us consider a scheduling problem shown in 12.3. Applying the *EST Heuristic*, we get:

- Event *E1* starts the earliest. Thus, we first select event *E1*.
- The next event which starts as early as possible is *E2*. However, $E1 \cap E2 \neq \varnothing$. Thus, *E2* can not be selected.
- The next event which starts as early as possible is *E3*. Also, $E1 \cap E3 = \phi$. Thus, *E3* is selected.
- The next event which starts as early as possible is *E4*. However, $E3 \cap E4 \neq \varnothing$. Thus, *E4* can not be selected.
- *Conclusion* : Two events *E1* and *E3* are selected.

Let us consider a scheduling problem where we have 3 events with their start and end time as shown in 12.5.

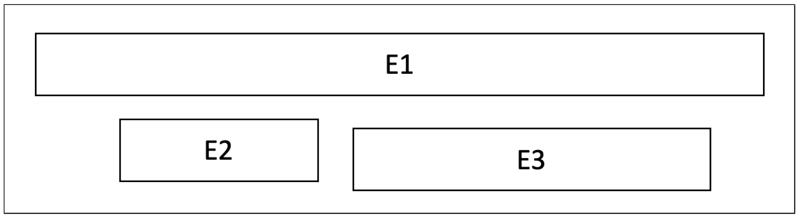


Figure 12.5: Scheduling example

Applying the *EST Heuristic*, we get:

- Event *E1* starts the earliest. Thus, we first select event *E1*.
- The next event which starts as early as possible is *E2*. However, $E1 \cap E2 \neq \varnothing$. Thus, *E2* can not be selected.
- The next event which starts as early as possible is *E3*. However, $E1 \cap E3 \neq \varnothing$. Thus, *E3* can not be selected.
- *Conclusion* : Only one event *E1* is selected.

Clearly, we can select two events *E2* and *E3* as $E2 \cap E3 = \phi$. Thus, example 12.5 serves as an **counter-example** for the *EST Heuristic*.

Heuristic (3) : *Earliest Finish Time (EFT)*
Start with the event which finishes the earliest and then select the next possible events that finish as early as possible.

Let us consider a scheduling problem shown in 12.3. Applying the *EFT Heuristic*, we get:

- Event *E1* finishes the earliest. Thus, we first select event *E1*.
- The next event which finishes as early as possible is *E2*. However, $E1 \cap E2 \neq \varnothing$. Thus, *E2* can not be selected.
- The next event which finishes as early as possible is *E4*. Also, $E1 \cap E4 = \phi$. Thus, *E4* is selected.
- The next event which starts as early as possible is *E3*. However, $E4 \cap E3 \neq \varnothing$. Thus, *E3* can not be selected.
- *Conclusion* : Two events *E1* and *E4* are selected.

Let us consider the example 12.5. Applying the *EFT Heuristic*, we get:

- Event *E2* finishes the earliest. Thus, we first select event *E2*.
- The next event which finishes as early as possible is *E3*. Also, $E2 \cap E3 = \phi$. Thus, *E3* is selected.
- The next event which starts as early as possible is *E1*. However, $E2 \cap E1 \neq \varnothing$. Thus, *E1* can not be selected.
- *Conclusion* : Two events *E2* and *E3* is selected.

Let us consider the example 12.4. Applying the *EFT Heuristic*, we get:

- Event *E1* finishes the earliest. Thus, we first select event *E1*.
- The next event which finishes as early as possible is *E2*. However, $E1 \cap E2 \neq \varnothing$. Thus, *E2* can not be selected.
- The next event which starts as early as possible is *E3*. Also, $E1 \cap E3 = \phi$. Thus, *E3* is selected.
- *Conclusion* : Two events *E1* and *E3* is selected.

The *EFT Heuristic* works well with the counter-examples identified for *EST Heuristic* and *Shortest event first* heuristic.

Bibliography

[1] Mathworld wolfram - chromatic number. 12.1

[2] Mathworld wolfram - interval graph. 12.1.3

[3] Martin C. Carlisle and Errol L. Lloyd. On the k-coloring of intervals. In *ICCI*, 1991. 12.9

[4] Reinhard Diestel. *Graph Theory (Graduate Texts in Mathematics)*. Springer, August 2005. 12.1

[5] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences)*. W. H. Freeman, first edition edition, 1979. 12.1.2

[6] Douglas B. West. *Introduction to Graph Theory*. Prentice Hall, 2 edition, September 2000. 12.1, 12.2, 12.3, 12.1, 12.1.1, 12.5, 12.6, 12.7, 12.8, 12.9