IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 3 Issue 12, December 2016
ISSN (Online) 2348 – 7968 | Impact Factor (2015) - 4.332
www.ijiset.com

Two Algorithms for Coloring Permutation Graphs

H.El-Zohny(1), M.M. Khalil(2), S.AbdElrahman(3)

1,2,3Mathematics Department, Faculty of science AlAzhar University Cairo- Egypt

Abstract: -
In the first part of this paper we introduce new two algorithms for coloring permutation graphs.by this algorithm we can divide the permutation graph into a number of independent sets. We can apply this algorithm on two permutation graphs. In the second part we introduce chromatic number and some operations for two permutation graphs.
Keywords: - algorithm, permutation graph, Cartesian product, chromatic number, perfect graph.

Algorithm:
In mathematics, computer science, and related subjects, an algorithm is a finite sequence of steps expressed for solving a problem. Algorithm can be expressed as

Intersection graph:-

A graph G= (V, E) is called intersection graph for a finite family F of a none empty set if there is a one to one correspondence between F and V such that two sets in F have non empty intersection if and only if their correspondence vertices in V are adjacent. Any undirected graph G may be represented as an intersection graph:-for each vertex v_i of G form a set s_i consisting of the edges incident to v_i, then two such sets have a non-empty intersection if and only if the corresponding vertices share an edge.

Permutation graph:-

An undirected graph G = (V,E) with vertices V = {1, 2, . . . , n} is called a permutation graph if there exists a permutation π on {1, 2,. . . ,n}

such that for all i, j∈V , (i-j)(π^{-1}(i)- π^{-1}(j))<0 , If and only if i and j are joined by an edge, where π^{-1} is the position in the sequence where the number i can be found. Geometrically, the integers 1,2,…n are drawn in order on areal line called as upper line and π(1), π(2),…, π(n) on a line parallel to this line called as lower line such that for each i∈N, I is directly below π(i).next ,for each I ,a line segment is drawn from I on the lower line to I on the upper line and it is denoted by l(i).then there is an edge (i,j)in Gif and only if the line segment l(i)for I intersect the line segment l(j)for j.

Permutation graph is a class of intersection graph for the intersection of line segments.

Permutation diagram and it's permutation graph in figure (1),where the permutation is 72481935106

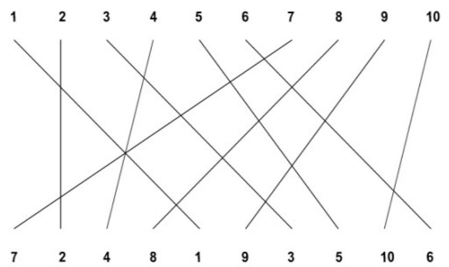

Figure (1) Permutation diagram

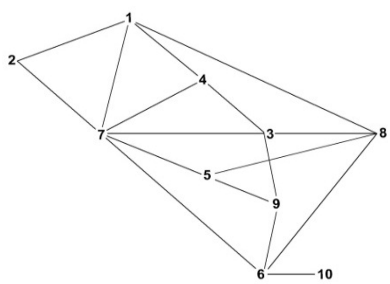

Figure (2) Permutation graph

Graph coloring:-

Defined as coloring the nodes of the graph with the minimum number of colors without any two adjacent nodes having the same color. Where the minimum number of colors for a graph is denoted by $\chi(G)$.in other words coloring of a graph is a mapping $c:v\rightarrow s$,where "s" is a finite set of colors, such that if $vw\in E$ then $c(v)\neq c(w)$.

clique:-a clique of G is a set of vertices where every pair of vertices are adjacent.

Clique number:-defined as the size of the largest clique of that graph and is denoted by $\omega(G)$

Perfect graph:-

A graph G is perfect if for induced sub graph H of G , $\omega (H)=\chi(H)$.

The degree of a vertex: the number of its adjacent vertices (the number of the neighbors of the vertex)

Main results

Algorithm (1) for coloring permutation graphs:

In this algorithm we use the largest degree ordering (LDO) algorithm and combining it with the incident degree ordering IDO.

Largest degree ordering:

It chooses a vertex with the highest number of neighbors. Intuitively, LDO can be implemented to run in O (n^2)

Incidence degree ordering:

The incidence degree ordering of a vertex is defined as the number of its adjacent colored vertices.

The algorithm works by choosing the largest degree ordering among the vertices and when we found that there are two nodes having the same degree, the IDO was used to choose between them.

The algorithm :-

Input: graph (set of nodes) $(n_1 ,n_2,\ldots, n_m)$,m colors, m colors.

Output: colored nodes and total number of colors.

d; represent the degree of a node in the graph.

ID; represent the incident degree of anode in the graph.

C_j; the color classes [which is also the minimum number of colors needed to color the graph].

```
Begin

No of colored nodes=0; Cj=0;  Cj=Cj+1

do

{

Max=-1

For I=1 to m

{

   If (!colored ((ni))

      {

       If(d>max)

         {max=d

          Index=i

      }

       Else if(d=max)

         if(ID(ni)>ID(nindex)

         Index=i
```

Else (d<max)

break

}

Color (n_{index})

Put n_{index} in C_j

If n_{index} is adjacent to the vertices in the class C_j go to the class C_{j+1}

No of colors nodes=no of colored nodes+1

}

While (no of colored nodes <m)

}

End.

Now we indicate how this algorithm runs:

Step (1)

Max=-1

For i=1, 1not colored

4=d(1)>max=-1

{max=4,index=1}

For i=2, 2 not colored

2=d(2)<max

{do nothing}

For i=3, 3 not colored

4=d(3)=max=4

So test else if condition

ID(3)!>ID(1)

{do nothing}

For i=4, 4 not colored

3=d(4)<max=4

{do nothing}

For i=5, 5 not colored

3=d(5)<max=4

{do nothing}

For i=6, 6 not colored

4=d(6)=max=4

So test condition of else if

ID(6)!>ID(1)

{do nothing}

For i=7, 7 not colored

5=d(7)>max=4

So change {max=5, index=7}

For i=8; 8 not colored

3=d(8)<max=5

So{do nothing}

For i=9; 9 not colored

3=d(9)<max=5

So{do nothing}

For i=10; 10 not colored

1=d(10)<max=5

So {do nothing}

At the final of the first step the algorithm color n_{index}=7 , put node 7in the class C_1 which mean the vertex 7 colored with the first possible one color.

Step (2)

Max=-1

For i=1, 1not colored

4=d(1)>max=-1

{Max =4,index=1}

For i=2, 2 not colored

2=d(2)<max

{do nothing}

For i=3, 3 not colored

4=d(3)=max=4

So test else if condition

ID(3)!>ID(1)

{do nothing}

For i=4, 4 not colored

3=d(4)<max=4

{do nothing}

For i=5, 5 not colored

3=d(5)<max=4

{do nothing}

For i=6, 6 not colored

4=d(6)=max=4

So test condition of else if

ID(6)=ID(1)

So {do nothing}

For i=7; 7 colored

For i=8; 8 not colored

4=d(8)=max=4

So test condition of else if

ID(8)!>ID(1)

So {do nothing}

For i=9; 9 not colored

3=d(9)<max=5

So{do nothing}

For i=10; 10 not colored

1=d(10)<max=5

So {do nothing}

At the final of the second step the algorithm colored n_{index}=1, since (1,7)∈E, the vertex 1 colored with different color, so put the vertex 1in the color class C_2.

Step (3)

Max=-1

For i=1; 1 colored

For i=2; 2 not colored

2=d(2)>max=-1

So {max=2, index=2}

For i=3; 3 not colored

4=d(3)>max=2

So {max=4, index=3}

For i=4; 4 not colored

3=d(4)<max=4

{do nothing}

For i=5, 5 not colored

3=d(5)<max=4

{do nothing}

For i=6, 6 not colored

4=d(6)=max=4

So test condition of else if

ID(6)=ID(3)

So {do nothing}

For i=7; 7 colored

For i=8; 8 not colored

4=d(8)=max=4

So test condition of else if

ID(8)!>ID(3)

So {do nothing}

For i=9; 9 not colored

3=d(9)<max=4

So{do nothing}

For i=10; 10 not colored

1=d(10)<max=4

So {do nothing}

IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 3 Issue 12, December 2016
ISSN (Online) 2348 – 7968 | Impact Factor (2015) - 4.332
www.ijiset.com

At the final of the step (3) the algorithm color the vertex 3,put the vertex 3 in the second class C_2 since the edge $(3,7)\in E$,$C_2=\{1,3\}$.

Step(4)

For i=1; 1colored

For i=2; 2 not colored

2=d(2)>max=-1

So {max=2, index=2}

For i=3; 3 colored

For i=4; 4 not colored

3=d(4)>max=2

So{max=3, index=4}

For i=5; 5 not colored

3=d(5)=max=3

So test condition of else if

ID(5)!>ID(4)

So {do nothing}

For i=6; 6 not colored

4=d(6)>max=3

So {max=4, index=6}

For i=7; 7 colored

For i=8; 8 not colored

4=d(8)=max=4

So test condition of else if

ID(8)>ID(6)

So {max=4,index=8}

For i=9; 9 not colored

3=d(9)<max=4

So{do nothing}

For i=10; 10 not colored

1=d(10)<max=4

So {do nothing}

At the final of step (4) the algorithm color the vertex 8 .put the vertex 8 in the color class C_1, $C_1=\{7,8\}$.

Step(5)

Max=-1

For i=1; 1 colored

For i=2; 2 not colored

2=d(2)>max=-1

So {max=2, index=2}

For i=3; 3 colored

For i=4; 4 not colored

3=d(4)>max=2

So{max=3, index=4}

For i=5; 5 not colored

3=d(5)=max=3

So test condition of else if

ID(5)!>ID(4)

So {do nothing}

For i=6; 6 not colored

4=d(6)>max=3

So {max=4, index=6}

For i=7; 7 colored

For i=8; 8 colored

For i=9; 9 not colored

3=d(9)<max=4

So {do nothing}

For i=10; 10 not colored

1=d(10)<max=4

So {do nothing}

At the final of step (5) the algorithm color the vertex 6 .put the vertex 6 in the color class C_2, C_2={1,3,6}.

Step(6)

For i=1; 1 colored,

For i=2; 2 not colored

2=d(2)>max=-1

So {max=2, index=2}

For i=3; 3 colored

For i=4; 4 not colored

3=d(4)>max=2

So{max=3, index=4}

For i=5; 5 not colored

3=d(5)=max=3

So test condition of else if

ID(5)!>ID(4)

So {do nothing}

For i=6; 6 colored

For i=7; 7 colored

For i=8; 8 colored

For i=9; 9 not colored

3=d(9)=max=3

So test condition of else if

ID(9)!>ID(4)

So {do nothing}

For i=10

1=d(10)<max=3

So {do nothing}

At the final of step (6) we color the vertex (4),since the neighbors of 4 are in C_1,C_2,the vertex 4 must be put in C_3.

Step (7)

For i=1; 1 colored

For i=2; 2 not colored

2=d(2)>max=-1

So {max=2, index=2}

For i=3; 3 colored

For i=4; 4 colored

For i=5; 5 not colored

3=d(5)>max=2

So {max=3,index=5}

For i=6; 6 colored

For i=7; 7 colored

For i=8; 8 colored

For i=9; 9 not colored

3=d(9)=max=3

So test condition of else if

ID(9)!>ID(5)

So {do nothing}

For i=10

1=d(10)<max=3

So {do nothing}

At the final of step (7) we color the vertex (5),

the vertex (5) can't be put in C_1,put it in C_2.

Step(8)

For i=1; 1 colored

For i=2; 2 not colored

2=d(2)>max=-1

So {max=2, index=2}

For i=3; 3 colored

For i=4; 4 colored

For i=5; 5 colored

IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 3 Issue 12, December 2016
ISSN (Online) 2348 – 7968 | Impact Factor (2015) - 4.332
www.ijiset.com

For i=6; 6 colored

For i=7; 7 colored

For i=8; 8 colored

For i=9; 9 not colored

3=d(9)>max=2

So {max=3,index=9}

For i=10; 10 not colored

1=d(10)<max=2

So {do nothing}

At the final of step (8) we color the vertex (9),put the vertex (9) in the class C_1 ,C_1={7,8,9}.

Step(9)

For i=1; 1 colored

For i=2; 2 colored

For i=3; 3 colored

For i=4; 4 colored

For i=5; 5 colored

For i=6; 6 colored

For i=7; 7 colored

For i=8; 8 colored

For i=9; 9 colored

For i=10; 10 not colored

1=d(10)>max=-1

At the final of step (9) we color the vertex (2), the vertex (2) can't be put in C_1or C_2 so put it in C_3,C_3={4,2}.

Step (10)

For i=1; 1 colored

For i=2; 2 colored

For i=3; 3 colored

For i=4; 4 colored

For i=5; 5 colored

For i=6; 6 colored

For i=7; 7 colored

For i=8; 8 colored

For i=9; 9 colored

For i=10; 10 not colored

1=d(10)>max=-1

So color the vertex (10), put it in C_1

C_1={7,8,9,10}

Finally

C_1= {7,8,9,10} C_2={1,3,6,5} C_3={2,4}

we have $\chi(G)$ =3,

which is equal to longest clique of the graph, w(G)=3=$\chi(G)$,this indicate that permutation graph is perfect graph.

Also by this algorithm we divide the graph into independent sets,C_1={7,8,9,10}, C_2={1,3,6,5} C_3={2,4}.where C_1,C_2,C_3 are color classes, where C_1, C_2 are called maximum independent sets in the permutation graph.

Algorithm (2) for coloring permutation graphs:

In this algorithm we combine the saturation degree ordering (SDO), and the largest degree ordering(LDO).

Saturation degree ordering (SDO)

The saturation degree of a vertex is defined as the number of its adjacent differently colored vertices.

This algorithm works as the SDO, but when there are two nodes having the same degree, we use the LDO to choose between them.so there are two criteria to choose the next node to be colored:

*the number of colors surrounding the vertex, SDO.
*the number of vertices surrounding the vertex, LDO.

Algorithm (2)

Input: set of nodes (graph) ($n_1,n_2,\ldots,n_m$) ,m colors.
Output: colored nodes and total number of colored nodes.

Begin
No. of colored nodes =0; C_j=0;
$C_j=C_j+1$
do
{
Max=-1
For I=1 to m
{if(!colored(n_i))
{
D=SD(n_i)
If(d>max)
{
Max=d
Index=i
}
Else if (d=max)
If(degree(n_i)>degree(n_{index}))
Index=i
Else (d<max)
break
}
Color (n_{index})
Put n_{index} in C_j, if there is an edge between n_{index} and the colored vertices in C_j, put it into $C_{j+1.}$
No of colored nodes=no of colored nodes+1
While (no of colored nodes<m)
}

We now indicate how algorithm(2) runs
At first max=-1, C_j=0

Step (1)

For i=1; 1 not colored
0=d(1)>max=-1
So {max=0,index=1}
For i=2; 2 not colored
0=d(2)=max=0
So test the condition of else if
Degree(2)!>degree(1)
So {do nothing}
For i=3; 3 not colored
0=d(3)=max=0
So test the condition of else if
Degree(3)!>degree(1)
So {do nothing}
For i=4; 4 not colored
0=d(4)=max=0
So test the condition of else if
Degree(4)!>degree(1)
So {do nothing}
For i=5; 5 not colored
0=d(5)=max=0
So test the condition of else if
Degree (5)!>degree(1)
So {do nothing}
For i=6; 6 not colored
0=d(6)=max=0
So test the condition of else if
Degree (6)!>degree(1)
So {do nothing}
For i=7; 7 not colored
0=d(7)=max=0
So test the condition of else if
Degree (7) >degree(1)
So {index=7}
For i=8; 8 not colored
0=d(8)=max=0
So test the condition of else if
Degree(8)!>degree(7)
So {do nothing}
For i=9; 9 not colored
0=d(9)=max=0
So test the condition of else if
Degree (9)!>degree(7)
So {do nothing}
For i=10; 10 not colored
0=d(10)=max=0
So test the condition of else if
Degree (10)!>degree(7)
So {do nothing}

At the finish of step (1),the algorithm color the vertex (7) and put it in color class C_1,C_1={7}.

Step(2)

IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 3 Issue 12, December 2016
ISSN (Online) 2348 – 7968 | Impact Factor (2015) - 4.332
www.ijiset.com

max=-1
For i=1; 1 not colored
1=d(1)>max=-1
So {max=1,index=1}
For i=2; 2 not colored
1=d(2)=max=1
So test the condition of else if
Degree (2)!>degree(1)
So {do nothing}
For i=3; 3 not colored
1=d(3)=max=1
So test the condition of else if
Degree (3)!>degree(1)
So {do nothing}
For i=4; 4 not colored
1=d(4)=max=1

So test the condition of else if
Degree (4)!>degree(1)
So {do nothing}
For i=5; 5 not colored
1=d(5)=max=1
So test the condition of else if
Degree (5)!>degree (1)
So {do nothing}
For i=6; 6 not colored
1=d(6)=max=1
So test the condition of else if
Degree (6)!>degree (1)
So {do nothing}
For i=7; 7 colored
For i=8; 8 not colored
0=d(8)<max=1
So {do nothing}
For i=9; 9 not colored
0=d(9)<max=1
So {do nothing}
For i=10; 10 not colored
0=d(10)<max=1
So {do nothing}
At the end of the first step we color the vertex (1),put it in C_2since $(1,7)\in E$, C_2={1}.

Step (3)
Max=-1
For i=1; 1 colored
For i=2; 2 not colored
2=d(2)>max=-1
{max=2,index=2}
For i=3; 3 not colored
1=d(3)!>max=2
So {do nothing}
For i=4; 4 not colored
2=d(4)=max=2
Degree (4)>degree(2)
{max=2, index= 4}
For i=5; 5 not colored
1=d(5)!>max=2
So {do nothing}
For i=6; 6 not colored
1=d(6)!>max=2
So {do nothing}
For i=7; 7 colored
For i=8; 8 not colored
1=d(8)!>max=2
So {do nothing}
For i=9; 9 not colored
0=d(9)!>max=2
So {do nothing}
For i=9; 9 not colored
0=d(9)!>max=2
So {do nothing}
For i=10; 10 not colored
0=d(10)!>max=2
So {do nothing}
At the end of the step(3) we color the vertex(4),the vertex (4)can't be put in C_1or C_2 so C_3={4}.

Step (4)
Max=-1
For i=1; 1 colored
For i=2; 2 not colored
2=d(2)>max=-1
{max=2,index=2}
For i=3; 3 not colored
2=d(3)=max=2
So test the condition of else if
Degree (3)> degree(2)
So {max=2, index =3}
For i=4; 4 colored
For i=5; 5 not colored
1=d(5)!>d(3)=2
So {do nothing}

For i=6; 6 not colored
1=d(6)!>d(3)=2
So {do nothing}
For i=7; 7 colored
For i=8; 8 not colored
1=d(8)!>d(3)=2
So {do nothing}
For i=9; 9 not colored
0=d(9)!>d(3)=2
So {do nothing}
For i=10; 10 not colored
0=d(10)!>d(3)=2
So {do nothing}
At the end of step(4) color the vertex 3,put it in C_2.C_2={1,3}.

Step (5)

Max=-1
For i=1; 1 colored
For i=2; 2 not colored
2=d(2)>max=-1
{max=2,index=2}
For i=3; 3 colored
For i=4; 4 colored
For i=5; 5 not colored
1=d(5)!>d(2)=2
So {do nothing}
For i=6; 6 not colored
1=d(6)!>d(2)=2
So {do nothing}
For i=7; 7 colored
For i=8; 8 not colored
1=d(8)!>d(2)=2
So {do nothing}
For i=9; 9 not colored
1=d(9)!>d(2)=2
So {do nothing}
For i=10; 10 not colored
1=d(10)!>d(2)=2
So {do nothing}
At the end of step (5) we color the vertex 2, put it in C_3,C_3={4,2}.

Step (6)

Max=-1
For i=1; 1 colored
For i=2; 2 colored
For i=3; 3 colored
For i=4; 4 colored
For i=5; 5 not colored
1=d(5)>max=-1
{max=1, index=5}
For i=6; 6 not colored
1=d(6)=max=1
So test the condition for else if
Degree(6)>degree(5)
So {max=1, index=6}
For i=7; 7 colored
For i=8; 8 not colored
1=d(8)!>d(6)=1
So {do nothing}
For i=9; 9 not colored
1=d(9)!>d(6)=1
So {do nothing}
For i=10; 10 not colored
1=d(10)!>d(6)=1
So {do nothing}
At the end of step (6) we color the vertex 6, put it in C_2, C_2={1,3,6}.

Step (7)

Max=-1
For i=1; 1 colored
For i=2; 2 colored
For i=3; 3 colored
For i=4; 4 colored
For i=5; 5 not colored
1=d(5)>max=-1
{max=1, index=5}
For i=6; 6 colored
For i=7; 7 colored
For i=8; 8 not colored
1=d(8)!>d(5)=1
So{do nothing}
For i=9; 9 not colored
1=d(9)!>d(5)=1
So {do nothing}
For i=10; 10 not colored
1=d(10)!>d(5)=1
So {do nothing}
At the end of step (6) we color the vertex 5, put it in C_2,C_2={1,3,6,5}.

Step (8)

Max=-1
For i=1; 1 colored
For i=2; 2 colored
For i=3; 3 colored
For i=4; 4 colored
For i=5; 5 colored
For i=6; 6 colored
For i=7; 7 colored
For i=8; 8 not colored
1=d(8)>max=-1
So {max=1,index=8}
For i=9; 9 not colored
1=d(9)=max=1
So test the condition for else if
Degree(9)!>degree(8)
So {do nothing}
For i=10; 10 not colored
1=d(10)=max=1
So test the condition for else if
Degree(10)!>degree(8)
So {do nothing}
At the end of step (7) we color the vertex 8, put it in C_1,C_1={7,8}.

Step (9)

Max=-1
For i=1; 1 colored
For i=2; 2 colored
For i=3; 3 colored
For i=4; 4 colored
For i=5; 5 colored
For i=6; 6 colored
For i=7; 7 colored
For i=8; 8 colored
For i=9; 9 not colored
1=d(9)>max=-1
So {max=1,index=9}
For i=10; 10 not colored
1=d(10)=max=1
Test the condition of else if
Degree(10)!>degree(9)
So {index=9}
At the end of step (8) we color the vertex 9, put it in C_1,C_1={7,8,9}.

Step (10)

Max=-1
For i=1; 1 colored
For i=2; 2 colored
For i=3; 3 colored
For i=4; 4 colored
For i=5; 5 colored
For i=6; 6 colored
For i=7; 7 colored
For i=8; 8 colored
For i=9; 9 colored
For i=10; 10 not colored
1=d(10)>max=-1
{max=1,index=10}
Color n_{index}(10)
At the end of step (10) we color the vertex 10, put it in C_1,C_1={7,8,9,10}.
Finally the color classes are
C_1={7,8,9,10},
C_2={1,3,6,5},C_3={4,2 }

Algorithm (2) run in O (n^3).

Some operations on the chromatic number in permutation graphs

Compound of two permutation graphs:-
consider two graphs $G_1=(V_1,E_1)$, $G_2=(V_2,E_2)$,$V_1 \cap V_2=\emptyset$, $E_1 \cap E_2=\emptyset$, the join of G_1,G_2 is G_1+G_2= G=(V,E) where $V=V_1 \cup V_2$, $E=E_1 \cup E_2 \cup E_3$ where $E_3=\{v_iv_j : v_i \in V_1, v_j \in V_2\}$ we represent E_3 by symmetric binary relation $\pi \subseteq V_1 X V_2$ and the compound graph G by $G_1 \pi G_2$.

1) If $\pi = \emptyset$ then we get the union of G_1 and G_2 denoted by $G_1 \cup G_2$. Consider the two permutations (231), (654) the following two permutation graphs as shown in the figure (3).

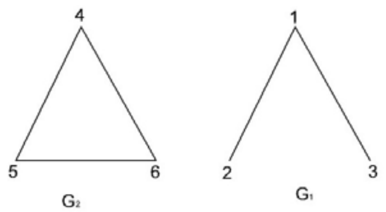

Figure (3)

IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 3 Issue 12, December 2016
ISSN (Online) 2348 – 7968 | Impact Factor (2015) - 4.332
www.ijiset.com

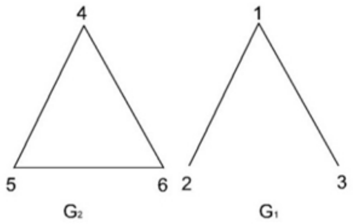

Figure (4) G=(V₁ ∪ V₂)

Apply algorithm (1) on $G_1 \cup G_2$, we find that $\chi(G_1 \cup G_2)=\max\{G_1,G_2\}$

2) If $\pi = V_1 x V_2$ [that is all edges between V_1, V_2], we get the join of G_1 and G_2, denoted by $G_1 \vee G_2$.as shown in the figure

Apply algorithm (1) on $G_1 \vee G_2$, we find that $\chi(G_1 \vee G_2) > \max\{\chi(G_1),\chi(G_2)\}$

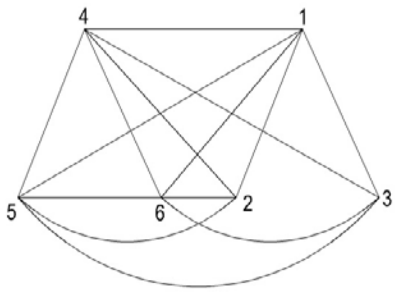

Figure (5) G=(V₁ ∨ V₂)

The Cartesian product of two permutation graphs:

The Cartesian product of two permutation graphs G and H is the graph G×H whose vertex set is the Cartesian product V(G)×V(H) and whose edges are the pairs $(g,h),(g_1,h_1)$ for which one of the following holds:

1) $g = g_1$ and $hh_1 \in E(H)$ or
2) $gg_1 \in E(G)$ and $h=h_1$.

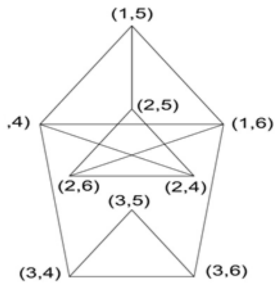

Figure (6)
Cartesian product for two permutation graphs

As shown in figure (6)

The Cartesian product of two graphs is connected if and only if G and H are connected and bipartite if and only if both G and H are bipartite.
The chromatic number of $G \times H = \max\{\chi(G),\chi(H)\}$
Apply algorithm (1) on the graph in figure (6) we found $\chi(G \times H) = 3$, which is the max of $\chi(G),\chi(H)$.

References

[1] Intersection graphs: an introduction, annals of pure and applied mathematics,2013.

[2] Finding a maximum independent set in a permutation graph, information processing letters 36(1999)19-23.

[3] Perfect graphs and the perfect graph theorems peter ballen.

[4] New graph coloring algorithms, American journal of mathematics and statistics 2(4):739-741,2006.

[5] Application of chromaticity for Cartesian products ,international journal of innovative science, engineering & techonology,vol.2 issue 1,January 2015.