



Clique Cover Problem

Heuristic Metaheuristic Algorithm

Group No: 10

Student ID:

1505002

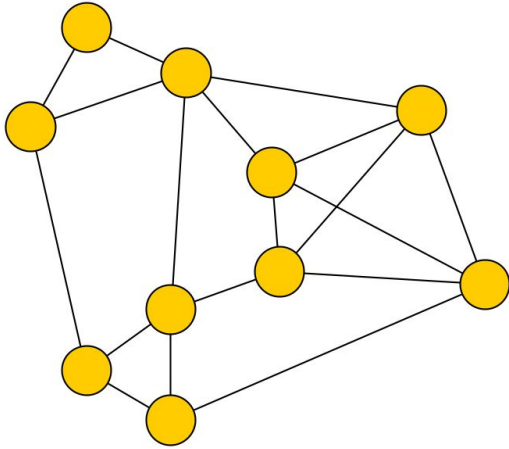
1505044

1505057

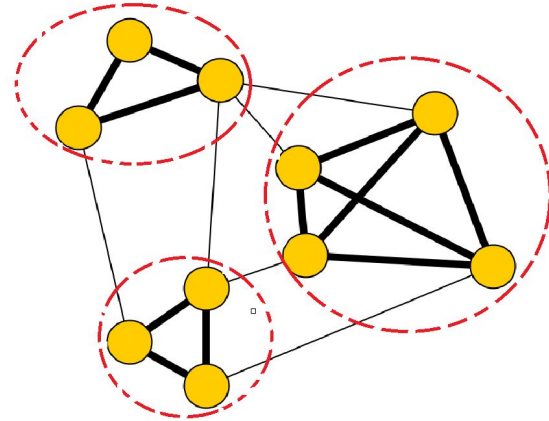
1505097

1505101

Clique Cover Problem



Given a graph $G(V,E)$ and integer $k = 3$.



True because the vertices of G can be partitioned into 3 sets S_i , where two vertices in the same sets S_i are adjacent.



Tabu Search for Vertex Clique Cover

- In this section we will present the tabu search method for vertex clique cover.
- It is a modification of the algorithm presented by Hertz and Werra's Tabucol.



Problem Description

- **INPUT:** Given an undirected graph $G(V,E)$ and a number k .
- **OUTPUT:** Output a clique cover of graph G with k cliques with minimum cost.



Representation

- We will number each vertex from 1 to $|V|$. And color of a vertex from 1 to K .
- Maintain a list P of size $|V|$ where each element is pair (vertex_number, color) i.e. P is the vertex set with color assigned.



Cost Function

- **COST(P):**

Number of pair (i,j) where v_i and v_j having same color and e_{ij} (edge connecting v_i and v_j) does not exist in $E(G)$.

- We will output the P with minimum cost.



Neighbourhood of Current Solution

- Our tabu search space consist of all possible colorings of $V(G)$ of size $|V|^k$.
- Given a current solution P , neighbourhood of P all the solution by changing the color of one vertex only. So the neighbourhood of P is of size $|V|*(K-1)$. Which will be reduced using a tabu list.



Tabu List

- Out tabu list is a fixed size queue Q. It is first in first out data structure.
- Each element of the queue is a tuple of length three (vertex, prev_color, next_color). Here prev_color is the color of the vertex in P and next_color is the color which we changed to by going the neighbour P' of P.



Algorithm

1. **Initialize** random configuration **P** and **conflicts** with inf and an empty queue **Q**
2. **WHILE** conflicts $\neq 0$ **and not** reached iteration/timelimit:
3. **Comment:** Find non-tabu move with largest improvement of cost function ie minimum P' , vertex, prev_color, next_color, cost := largest_improvement_move(P)
4. **Comment:** Make the move
 $P := P'$
5. **Comment:** Maintain Queue fixed size
 IF Q.SIZE() == MaxQsize:
 Q.DEQUEUE()



Continuation of Algorithm

6. Comment: Add to tabu list

 Q.ENQUEUE((vertex, prev_color, next_color))
7. Conflicts = COST(P)
8. **IF** COST(bestP) > COST(P):
 bestP = P
9. **Return** bestP



Complexity

- Using a queue along with hashmap with fast searching the largest_improvement_move will take $K * |E|$ time. Thus giving the overall complexity per iteration is $O(K * E)$



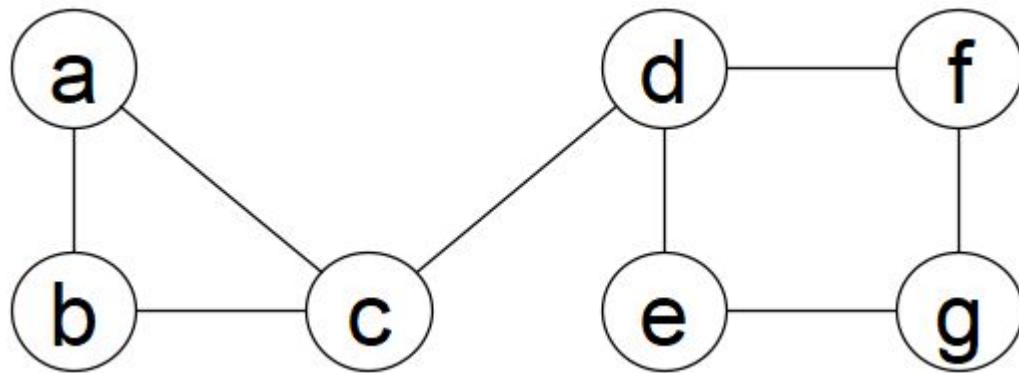
Greedy Clique Covering (GCC)

Input: graph $G = (V, E)$

permutation $P = [P_1, P_2, \dots, P_n]$ of vertices in V

Output: clique covering S of G

GCC



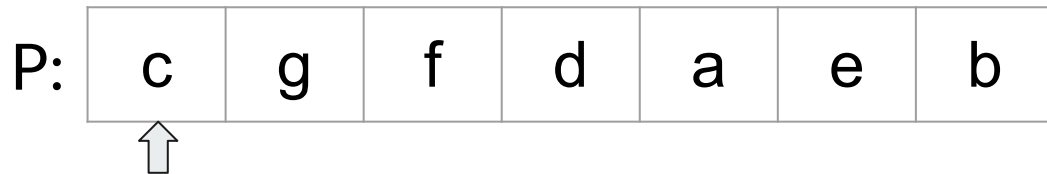
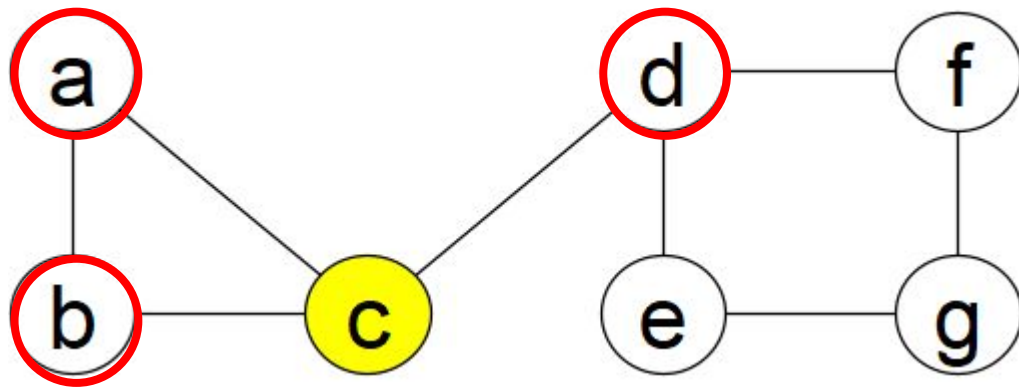
P:

c	g	f	d	a	e	b
---	---	---	---	---	---	---

sizes

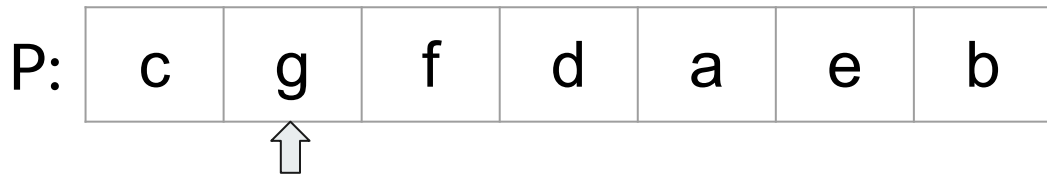
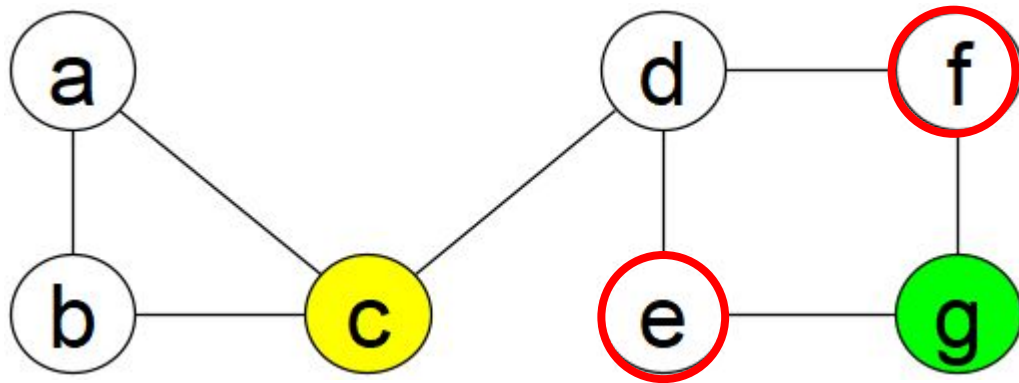
cliques


GCC



<i>sizes</i>	<i>cliques</i>
1	{c}


GCC



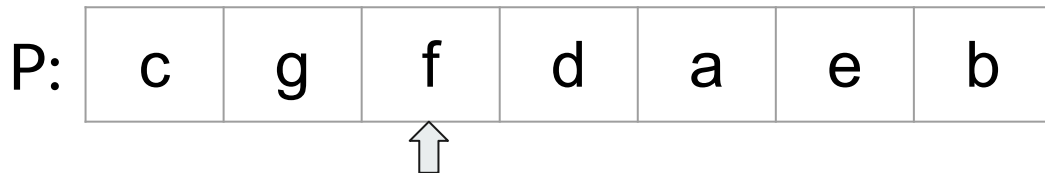
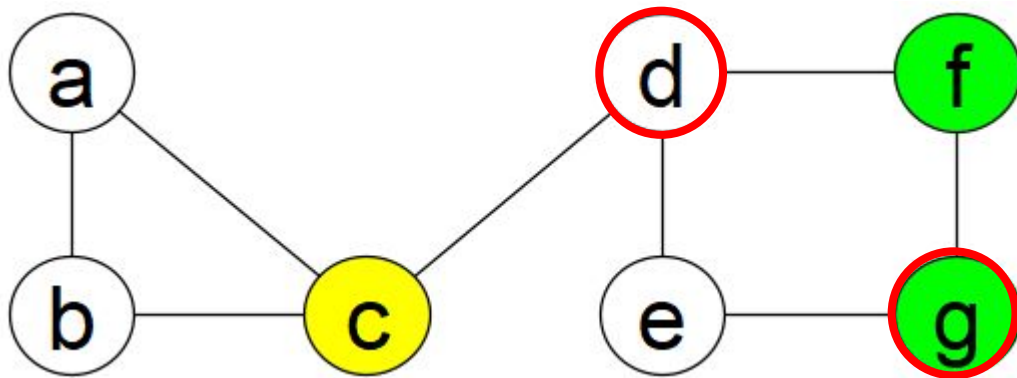
sizes

1
1

cliques

{c}
{g}


GCC



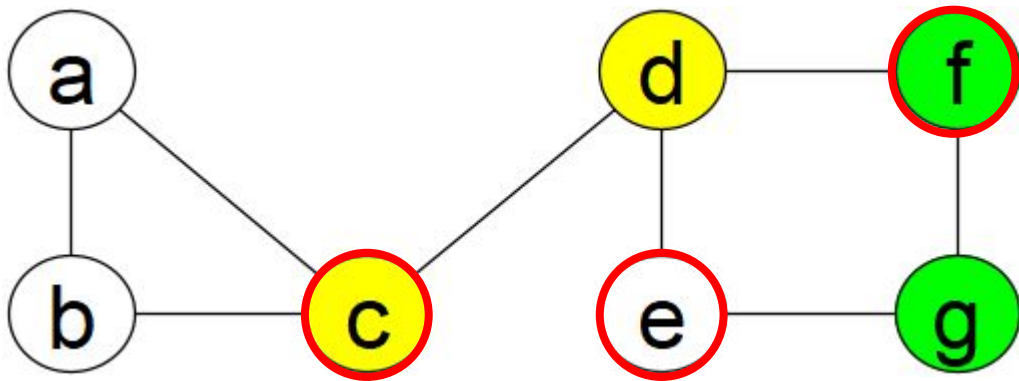
sizes

1
2

cliques

{c}
{g,f}


GCC



P:

c	g	f	d	a	e	b
---	---	---	---	---	---	---



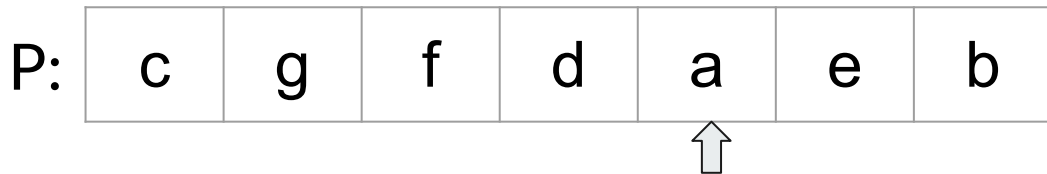
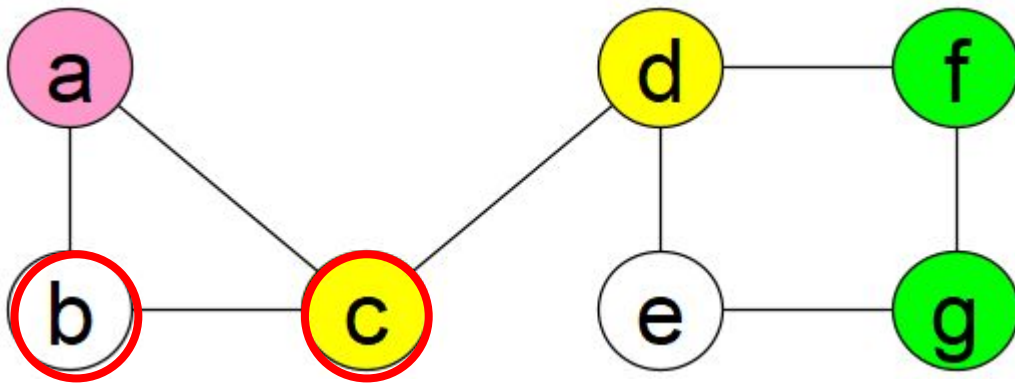
sizes

2
2

cliques

{c,d}
{g,f}


GCC



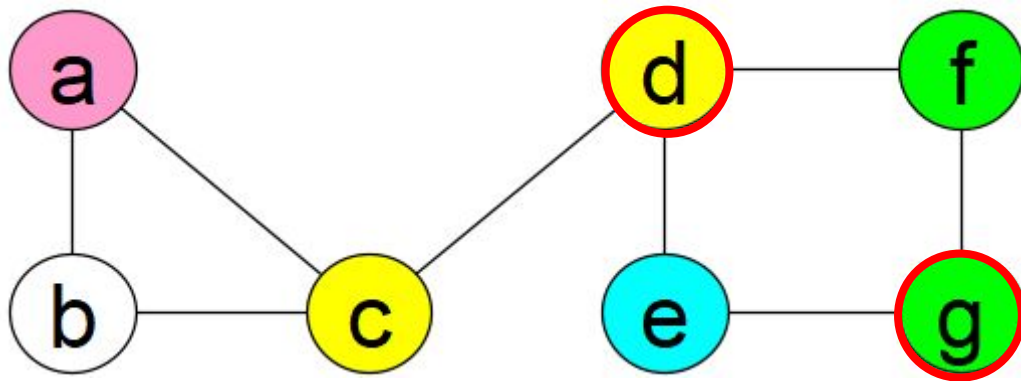
sizes

2
2
1

cliques

{c,d}
{g,f}
{a}


GCC



P:

c	g	f	d	a	e	b
---	---	---	---	---	---	---



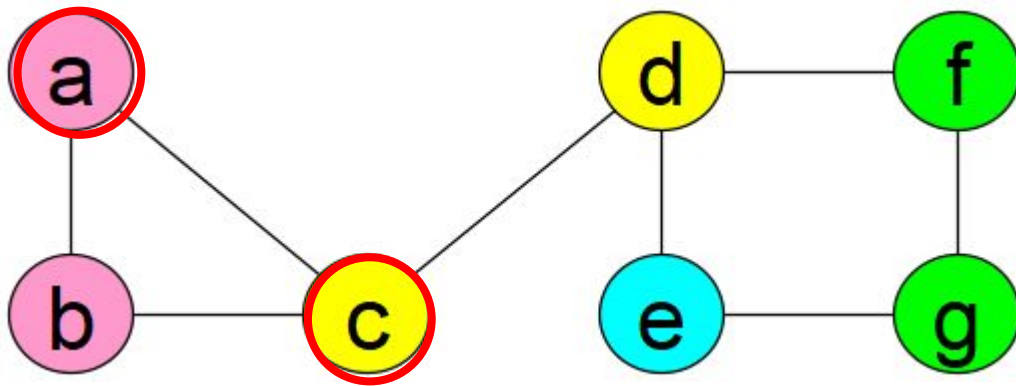
sizes

2
2
1
1

cliques

{c,d}
{g,f}
{a}
{e}


GCC



P:

c	g	f	d	a	e	b
---	---	---	---	---	---	---

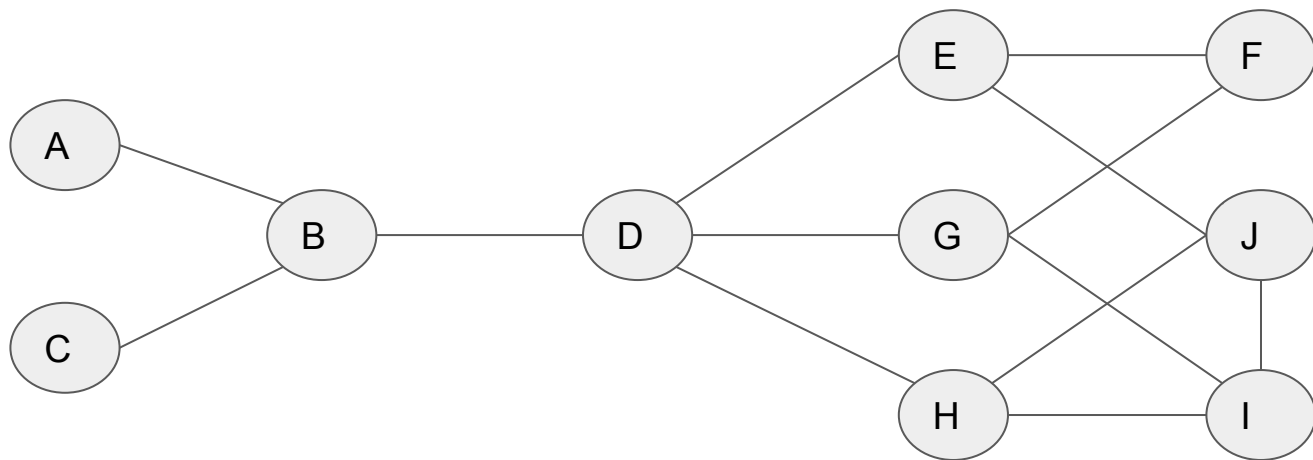


sizes

2
2
2
1

cliques

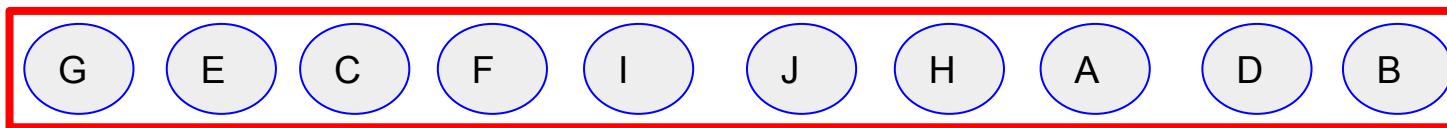
{c,d}
{g,f}
{a,b}
{e}

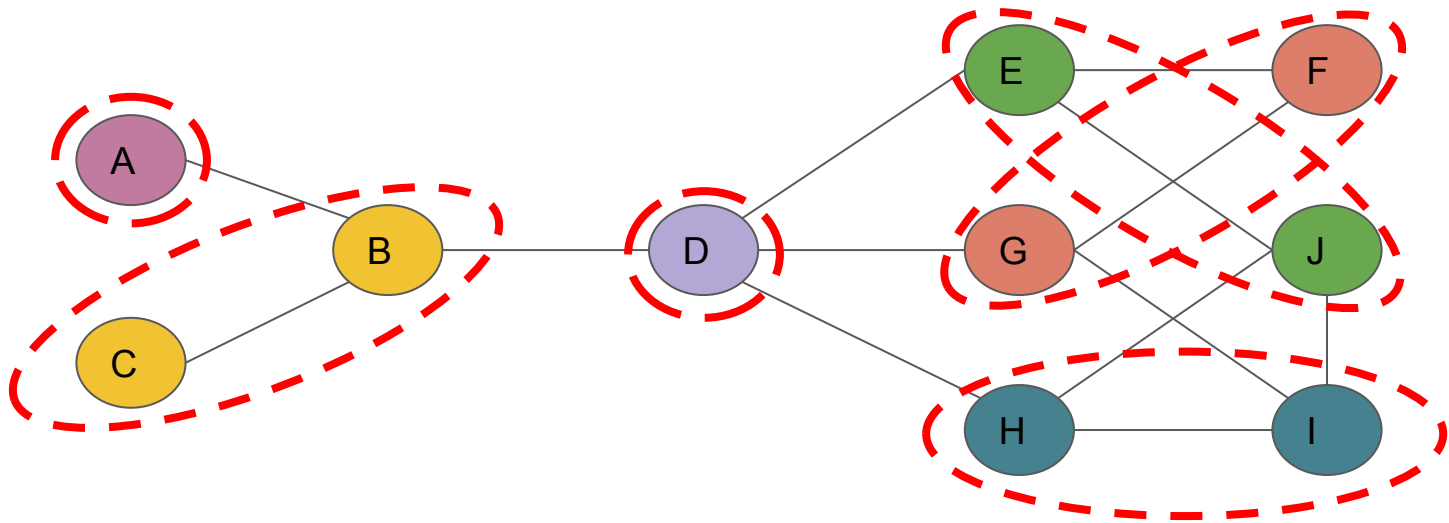


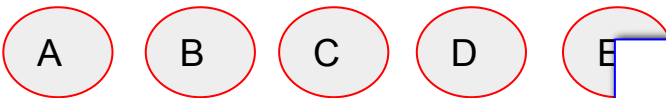
Vertices :



After
permutation :





Vertices : 

After
permutation :

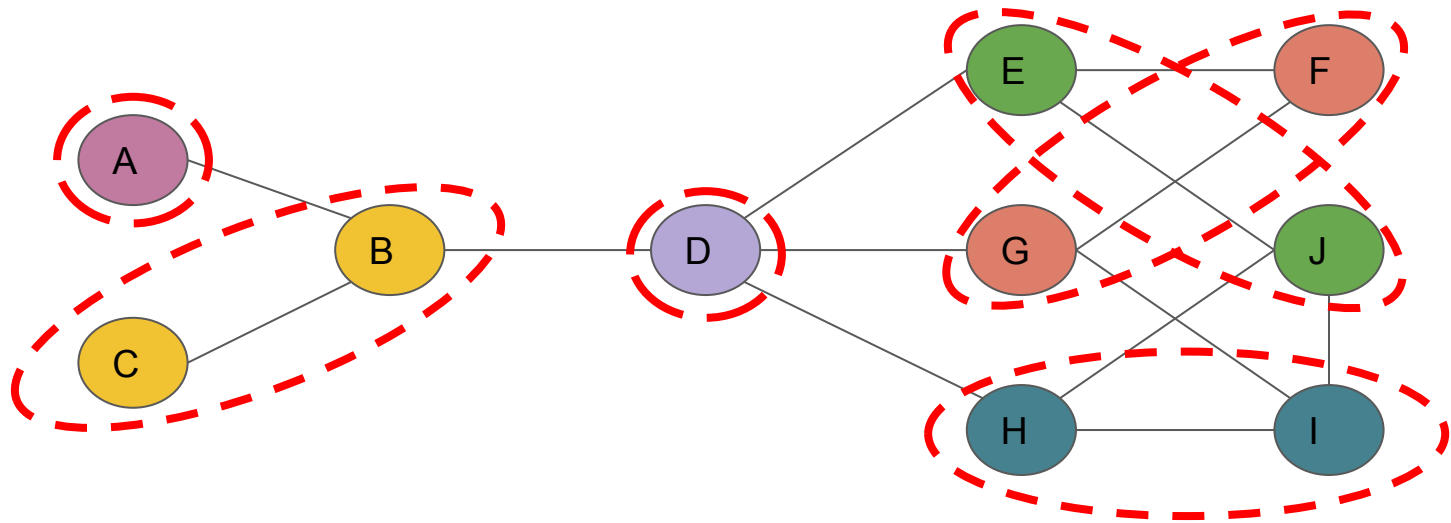


Is the result optimal?



Iterative Greedy Algorithm

- Stochastic Local Search
- Iterative Process
- Improve Greedy solution in each step
- Suitable for covering problems



$$Q_1 = \{G, F\}$$

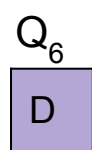
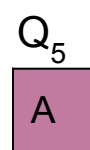
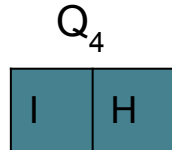
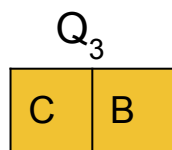
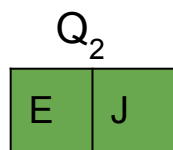
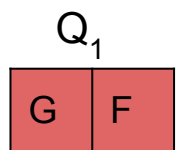
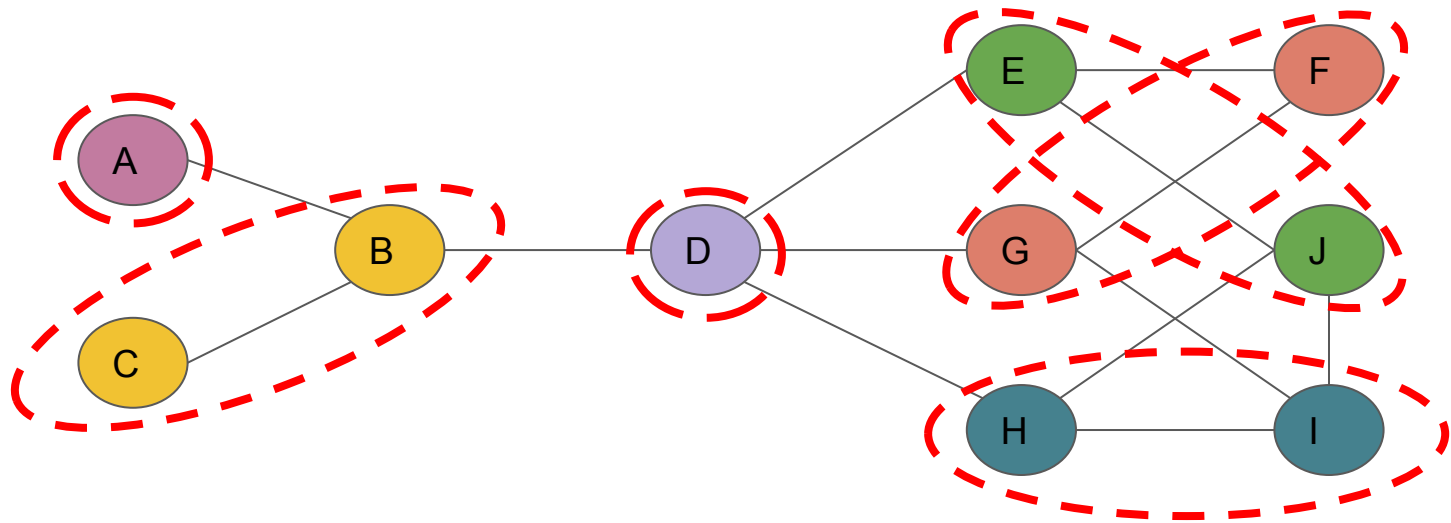
$$Q_2 = \{E, J\}$$

$$Q_3 = \{C, B\}$$

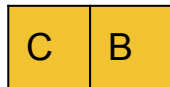
$$Q_4 = \{I, H\}$$

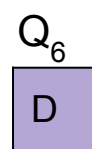
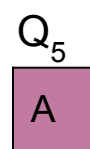
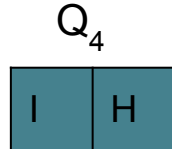
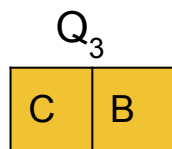
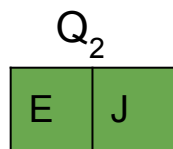
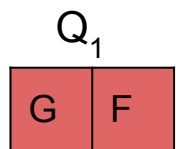
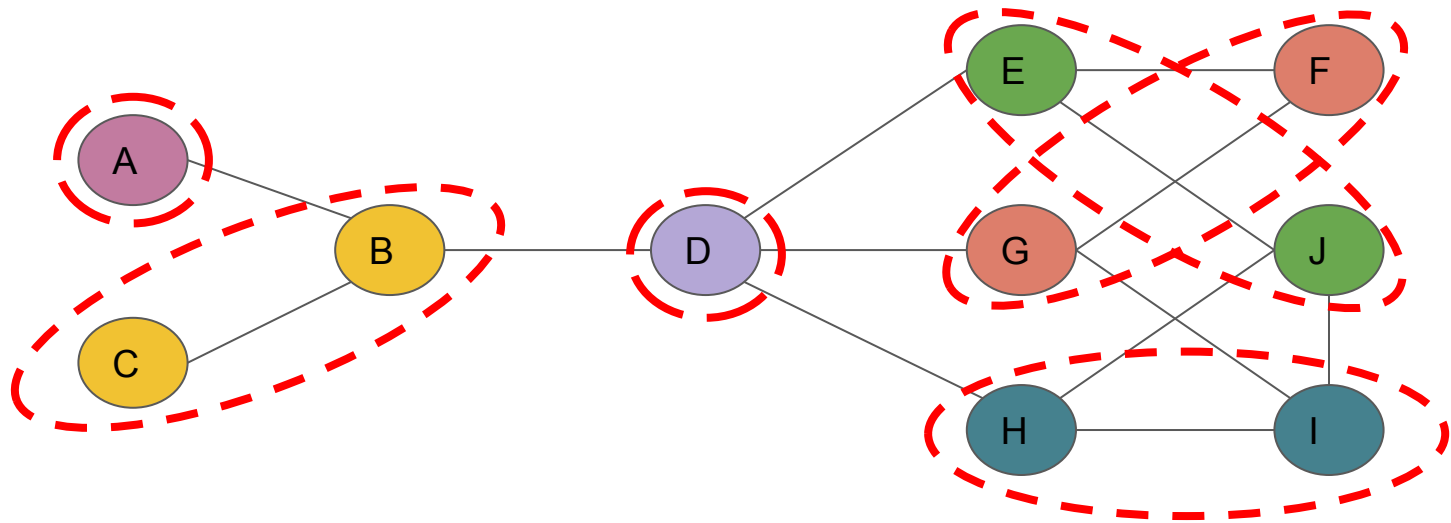
$$Q_5 = \{A\}$$

$$Q_6 = \{J\}$$



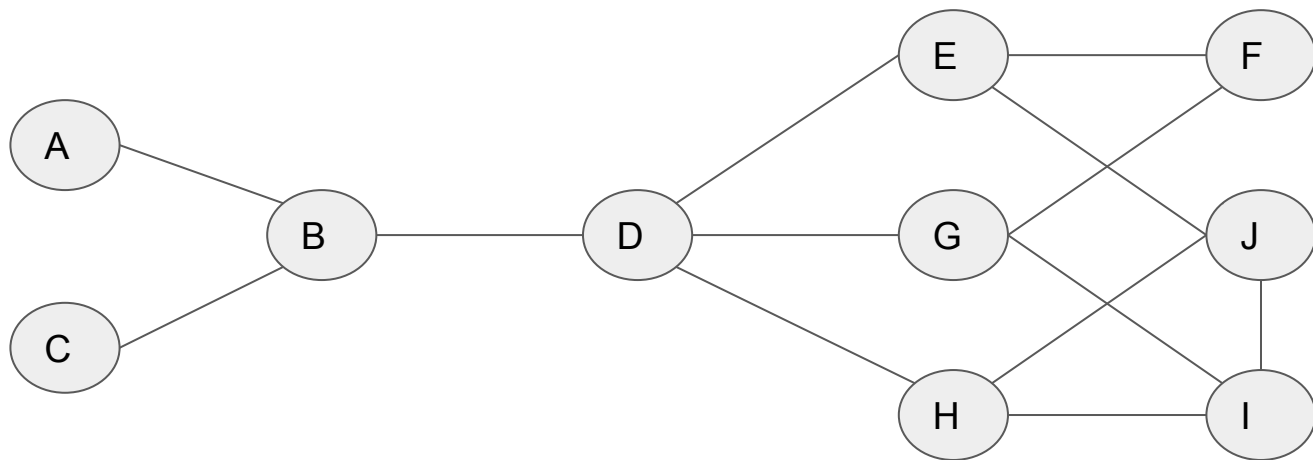
After
permutation :





After
permutation :

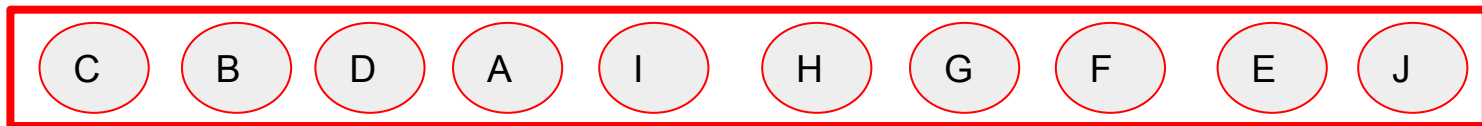


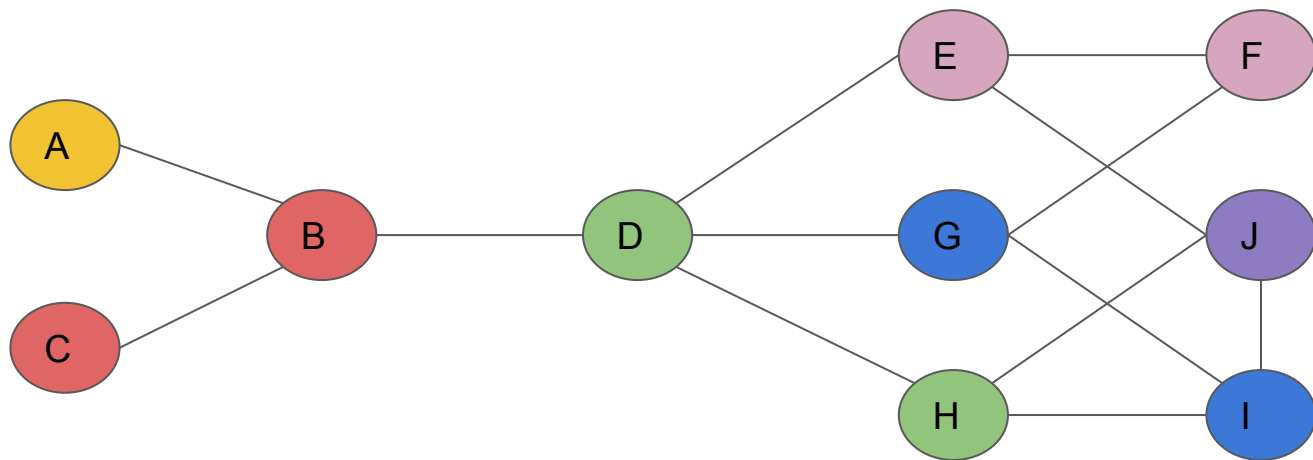


Block List :



Vertices :

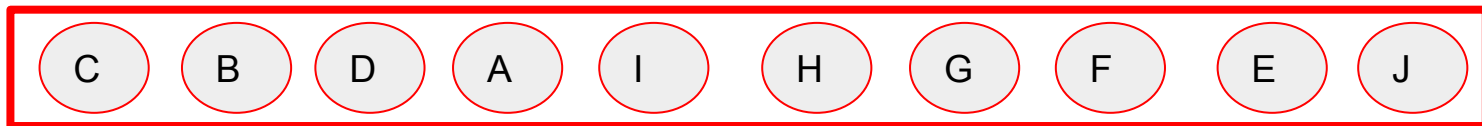


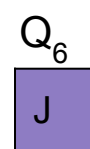
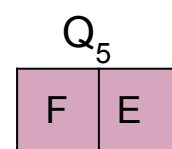
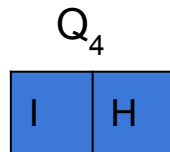
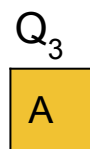
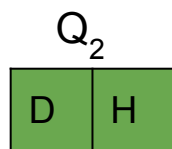
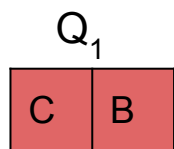
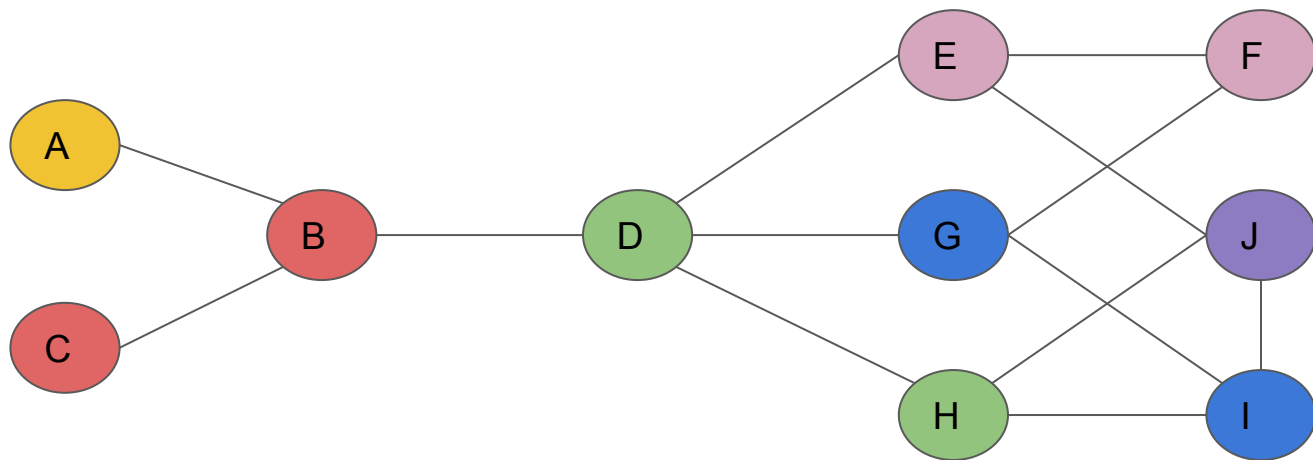


Block List :

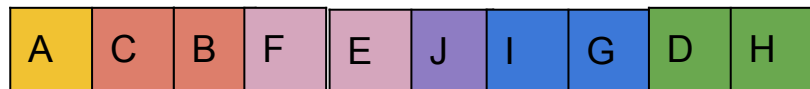


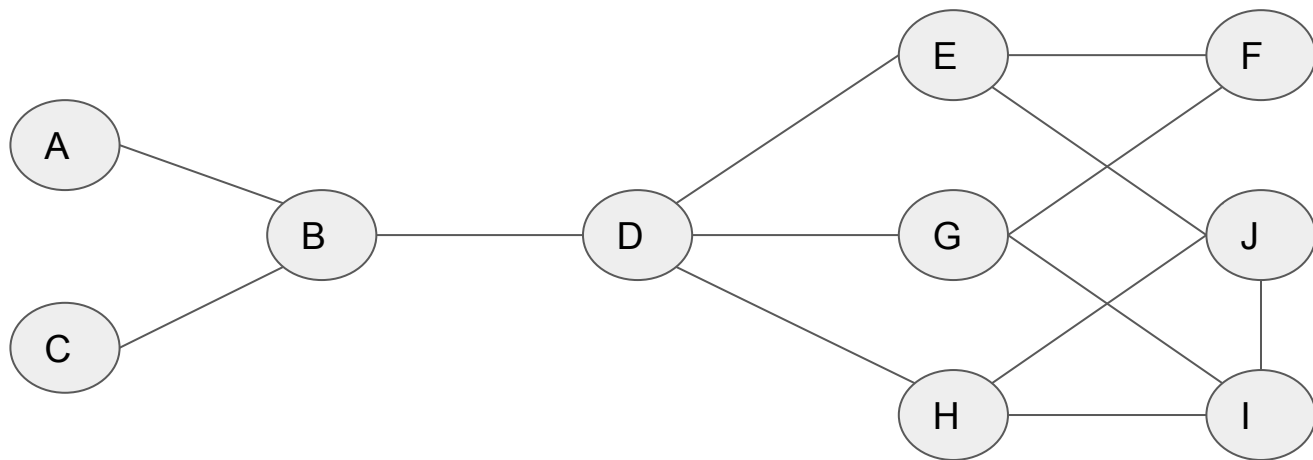
Vertices :



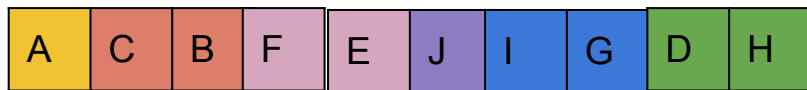


After
permutation :



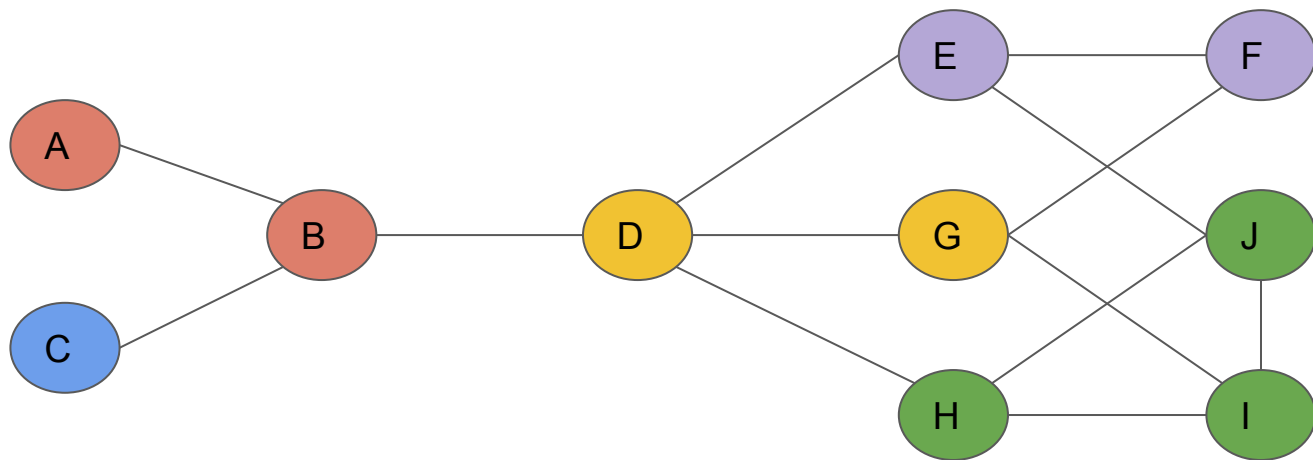


Block List :

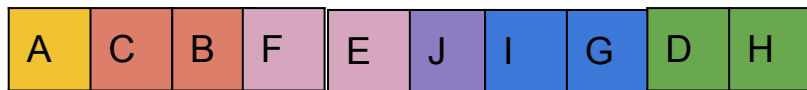


Vertices :



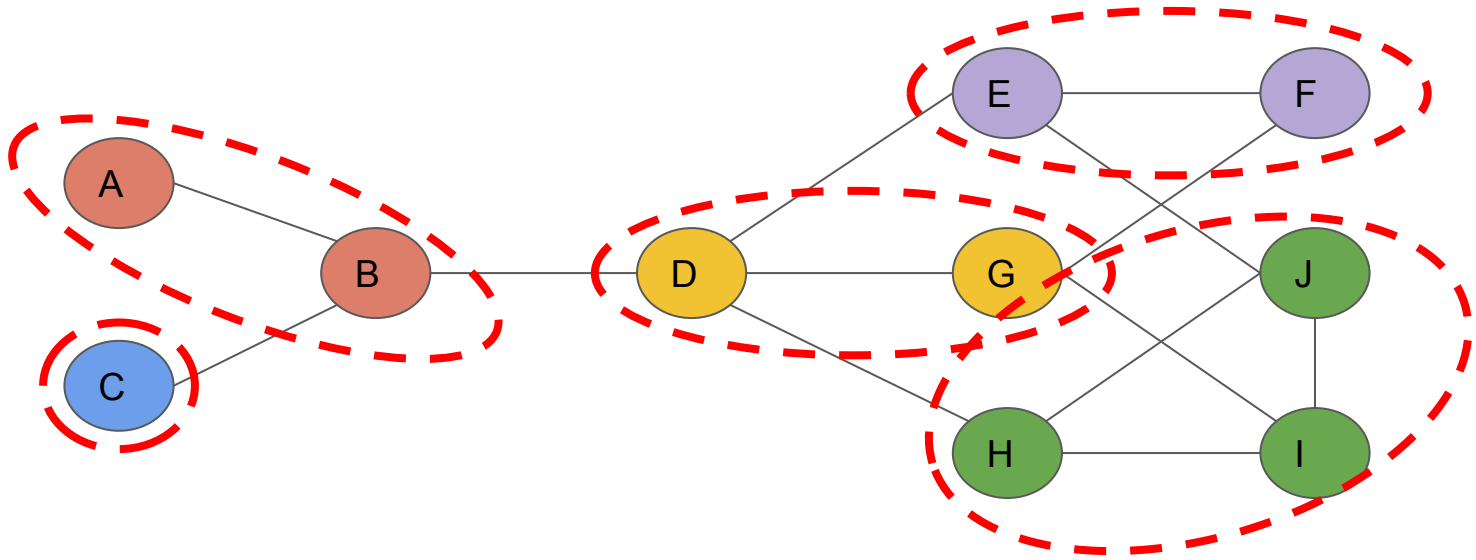


Block List :

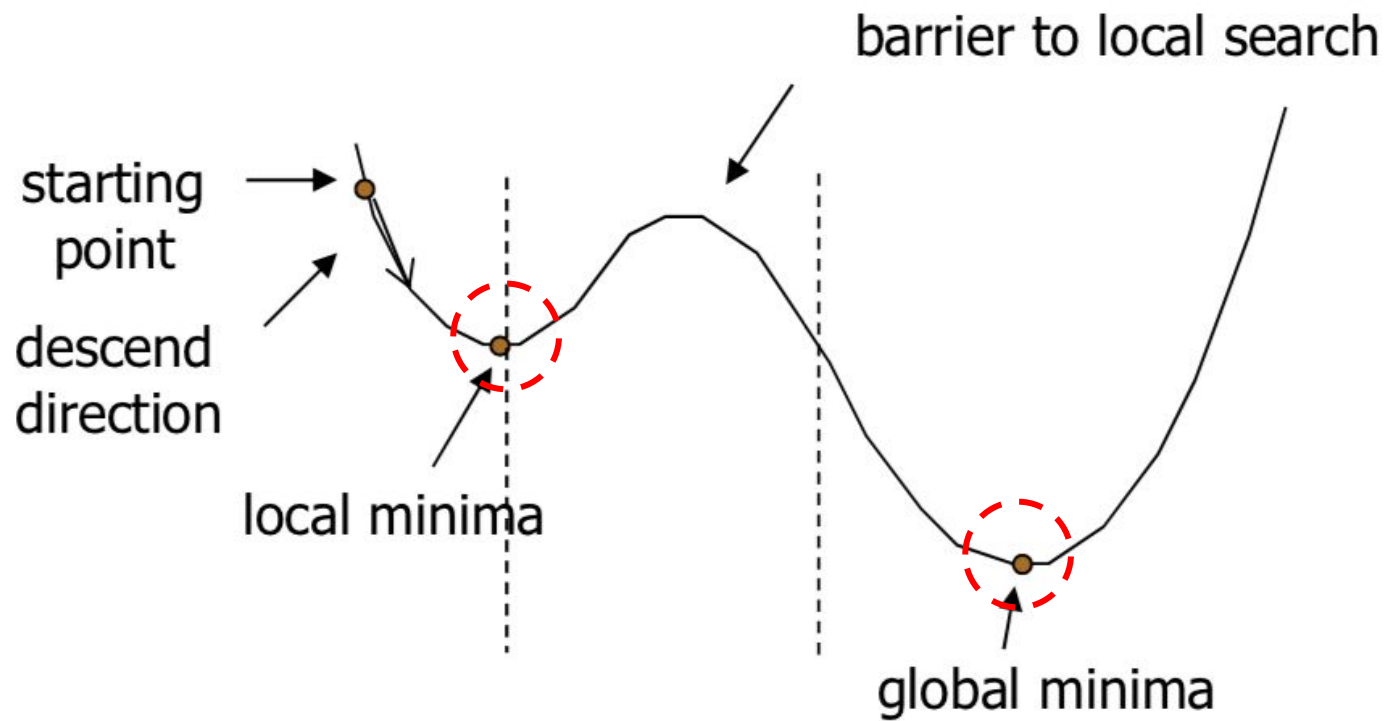


Vertices :





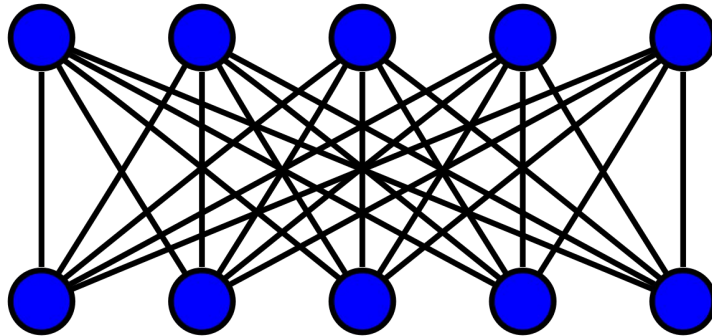
There is a problem !



RESULT FOR TRIANGLE-FREE GRAPHS

Triangle-Free Graph : A triangle-free graph is an undirected graph in which no three vertices form a triangle of edges.

The size of the maximum clique in a triangle-free graph is $\omega \leq 2$.





RESULT FOR TRIANGLE-FREE GRAPHS

We build our results on a relatively widely used method of fitness levels. We divide the search space into levels such that each level contains all solutions with the same number of cliques. Then, Lemma 1 can be used to find an upper bound for the expected running time of our algorithm.



RESULT FOR TRIANGLE-FREE GRAPHS

Lemma 1 : The expected optimization time I of a stochastic search algorithm that works at each time step with a population of size 1 and produces at each time step a new solution from the current solution is upper bounded by:

$$I \leq \sum_{i=1}^{m-1} \frac{1}{p_i}.$$

m = the number of fitness levels

p_i = the minimum probability that in time step i , the stochastic change will cause an improvement

Lemma 2 : For paths, the initial solution for IG can contain at most $\lceil 2/3n \rceil$ cliques and there are at most $\lceil n/3e \rceil$ 1-cliques in the result



RESULT FOR TRIANGLE-FREE GRAPHS

We now show that in expectation, IG finds the optimal vertex clique covering in polynomial time for *triangle-free graphs*.

Theorem 1. For triangle-free graphs on n vertices and m edges, the expected time for IG to find the optimal vertex clique covering is upper bounded by $O(n^5 m^2)$.

Proof. Based on Lemma 1, the initial clique covering contains $O(n)$ more cliques than the optimum. These will determine our fitness levels.



SOME DEFINITIONS

- Cliques of size one will be called *1-cliques*
- Two-vertex cliques will be called *2-cliques*
- An improvement occurs if random changes cause *1-cliques* move so that some pair of *1-cliques* are next to each other and form a 2-clique.



Fig a: 1-clique

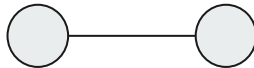


Fig b: 2-clique

SOME DEFINITIONS

First Fit rule : The choice of the first clique from the input permutation.

Block-jump : Operator *block jump* takes a chosen block representing a clique and puts it to the first position in the permutation. The other blocks are then shifted to the right.



Figure a : Illustration of the block jump operator



RESULT FOR TRIANGLE-FREE GRAPHS (Proof)

We first look at what happens if block jump occurs.

- **For 2-Clique :** Only the ordering of the *2-cliques* can be changed. No vertex can be taken by a *2-clique*, since that would create a triangle.
- **For 1-Clique :** *1-clique* will form a *2-clique* with its nearest following neighbour in the permutation.

RESULT FOR TRIANGLE-FREE GRAPHS (Proof)

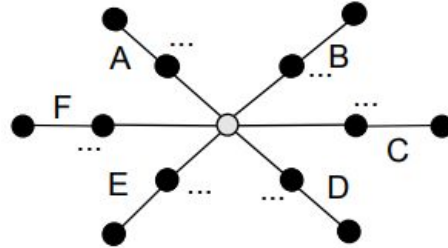


Figure 2. An illustration of the situation with a 1-clique between several 2-cliques

In Figure 2, we illustrate the situation, when a *1-clique* was left between several *2-cliques*. Each *1-clique* must necessarily have only *2-cliques* around it. If it did not have, it would be joined with another *1-clique* in a *2-clique*.



RESULT FOR TRIANGLE-FREE GRAPHS (Proof)

Let X now be the waiting time until a block jump of this 1-clique occurs. Before the final move of the *1-clique*, there are $X - 1$ block jump operations. The direction of the movement of this *1-clique* is determined by which neighbour (in Figure 2, determined by blocks **A-F**) comes first in the permutation.

Now, we will have two cases, what can happen during this waiting time.

- ❖ **Case 1.** None of the blocks around the 1-clique jumped.
- ❖ **Case 2.** Some block around the 1-clique jumped.



RESULT FOR TRIANGLE-FREE GRAPHS (Proof)

Case 1. None of the blocks around the 1-clique jumped.

- For $X = 1$ (i.e., it takes only one move to choose the 1-clique) with probability $1/(d+1) \leq 2/n$. In this case, no other moves could surely be chosen.
- For $X > 1$, we observe that for our 1-clique vertex v with $\deg(v)$ neighbours, the probability of this event will be:

$$\left(1 - \frac{\deg(v)}{X-1}\right)^{X-1} = \left(1 - \frac{\deg(v)}{X-1}\right)^{\frac{X-1}{\deg(v)} \deg(v)} \leq e^{-\deg(v)}.$$



RESULT FOR TRIANGLE-FREE GRAPHS (Proof)

Case 2. Some block around the 1-clique jumped.

We are interested in which of the $\deg(v)$ blocks was the last to jump. The probability of this case is at least $1 - e^{-\deg(v)} - 2/n$, because of the bound shown in Case 1.

We will now argue that this portion of probability is distributed fairly among all $\deg(v)$ neighbour blocks. This is because the probability of block jump is uniformly distributed among the blocks.

Hence, the probability of changing the direction of movement of 1-clique is at least $(1 - e^{-\deg(v)} - 2/n) / \deg(v)$ for each neighbour block.

RESULT FOR TRIANGLE-FREE GRAPHS (Proof)

Let us now consider the event in Case 1. None of the blocks around the 1-clique jumped, i.e., the direction will be determined by the block, which is currently the first in the permutation. Based on the previous arguments, the probability that one fixed neighbour block (in Figure 2, one of the blocks A–F) was the first one in the beginning of the waiting time, is uniformly distributed, too. Therefore, *1-cliques* actually perform fair **random walks** on the triangle-free graph.

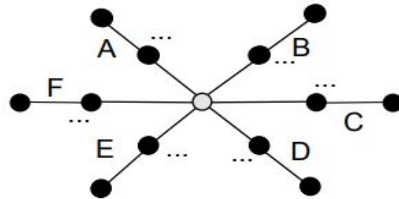


Figure 2. An illustration of the situation with a 1-clique between several 2-cliques



RESULT FOR TRIANGLE-FREE GRAPHS (Proof)

- From the cover time of random walks, it takes $O(nm)$ block jump moves of a 1-clique to visit each vertex at least once.
- For two such random walks, we have that it takes $O(n^2m^2)$ block jump moves in expectation for two 1-cliques to arrive at two adjacent vertices.
- $O(n)$ is the time needed to obtain a block jump of the 1-clique and $O(n)$ is the complexity of GCC.
- We have $O(n)$ fitness levels, on which all this happens.

Therefore, the expected time to obtain the optimum is bounded by

$$O(n^2m^2) * O(n) * O(n) * O(n) = O(n^5m^2).$$



GRAPHS WITH NON-OVERLAPPING TRIANGLES

- Let us consider the initial permutation being generated such that non-overlapping triangles are placed into it as blocks
- The rest of the permutation is generated uniformly at random
- Now we show that such a modification of IG guarantees that the optimal clique covering is found in expected polynomial time

GRAPHS WITH NON-OVERLAPPING TRIANGLES

Let G be a graph with maximum clique size $\omega = 3$. Let S be an optimum and let S' be a sub-optimum for CCP in G . There are three cases of how IG can overestimate the number of cliques $\nu(G)$

Case 1: Instead of a 2-clique in S , there are two 1-cliques in S'



GRAPHS WITH NON-OVERLAPPING TRIANGLES

Let G be a graph with maximum clique size $\omega = 3$. Let S be an optimum and let S' be a sub-optimum for CCP in G . There are three cases of how IG can overestimate the number of cliques $\nu(G)$

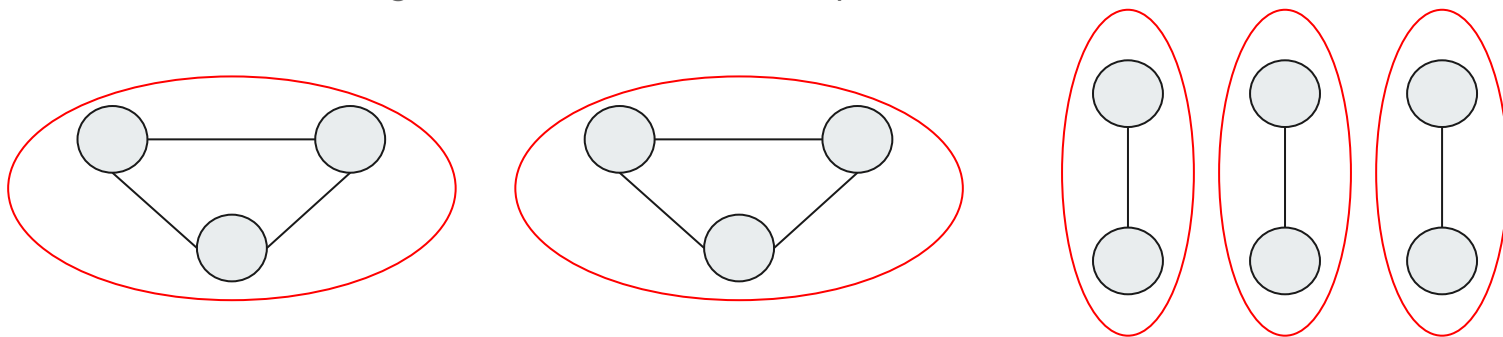
Case 2: Instead of a triangle in S , there is a 2-clique and a 1-clique in S'



GRAPHS WITH NON-OVERLAPPING TRIANGLES

Let G be a graph with maximum clique size $\omega = 3$. Let S be an optimum and let S' be a sub-optimum for CCP in G . There are three cases of how IG can overestimate the number of cliques $\nu(G)$

Case 3: Instead of two triangles in S , there are three 2-cliques in S'





GRAPHS WITH NON-OVERLAPPING TRIANGLES

Now we claim that all possible ways of overestimation represented by S' represent compositions of these three cases.

Proof: So the three cases are self explanatory. To exclude the existence of other ways, we use simple enumeration.

- Substitution of any number of 2-cliques by 1-cliques is a composition of events included in Case 1.
- Substitution of one triangle by three 1-cliques is a composition of Case 1 and Case 2.
- If we consider three triangles, the first two can be substituted based on Case 2 or Case 3 and the last triangle will remain for Case 2.
- If we consider four or more triangles, we can apply Case 2 and Case 3 iteratively. The resulting 2-cliques are further divided according to Case 1.



GRAPHS WITH NON-OVERLAPPING TRIANGLES

Lemma A: Let G be a graph with maximum clique size $\omega = 3$. If there is a suboptimal clique covering S' of G , which contains more triangles than an optimum S , then S' must also contain at least two 1-cliques.

Proof: Let c_1, c_2 and c_3 be the numbers of cliques in S with 1, 2 or 3 vertices, respectively. Let c'_1, c'_2 and $c'_3 = c_3 + d$ be the respective values for S' , and for $d \geq 1$. All vertices must be covered and S must contain less cliques than S' . Hence:

$$\begin{aligned}c_1 + 2c_2 + 3c_3 &= c'_1 + 2c'_2 + 3(c_3 + d) = n, \\c_1 + c_2 + c_3 &< c'_1 + c'_2 + c'_3 + d.\end{aligned}$$

From these formulas, $3d = (c_1 - c'_1) + 2(c_2 - c'_2)$ and $d > (c_1 - c'_1) + (c_2 - c'_2)$. This implies that $(c_2 - c'_2) > 2d$ and, thus, $(c_1 - c'_1) < -d$. The value $-d$ can be at most -1 , i.e. $c'_1 > 1$.



GRAPHS WITH NON-OVERLAPPING TRIANGLES

Lemma B: Let G be a graph with maximum clique size $\omega = 3$. Let IG begin with a suboptimal clique covering S' with at least as many triangles as in an optimal clique covering S for G . Then, IG cannot get stuck in a local optimum and will be in the global optimum if the number of extra 1-cliques in the solution is minimal.

Proof:

- Let S contain c_1 1-cliques, c_2 2-cliques and c_3 triangles. The analogous values for S' are c'_1, c'_2 and $c'_3 \geq c_3$.
- The premises imply that an improvement cannot be obtained by making the number of triangles higher.
- Therefore, it is necessary that $c'_2 < c_2$ and $c'_1 > c_1$.
- Since c_1 1-cliques are present in S , the only way to obtain an improvement is to reduce the number of extra 1-cliques by 2 and increase the number of 2-cliques by 1.
- This holds for all sub-optima, which proves the second statement.



GRAPHS WITH NON-OVERLAPPING TRIANGLES

Lemma B: Let G be a graph with maximum clique size $\omega = 3$. Let IG begin with a suboptimal clique covering S' with at least as many triangles as in an optimal clique covering S for G . Then, IG cannot get stuck in a local optimum and will be in the global optimum if the number of extra 1-cliques in the solution is minimal.

Proof(Continued):

- For the first statement, let us suppose that IG got stuck.
- Then two of the triangles must have been substituted by three 2-cliques.
- However, this is in contradiction with the fact that these three 2-cliques must lie between the triangles and block jump cannot cause a transformation, in which vertices between 2 different blocks are regrouped to 3 blocks in between.



GRAPHS WITH NON-OVERLAPPING TRIANGLES

Let G be a graph on n vertices with maximum clique size $\omega = 3$, containing only non-overlapping triangles. Let P be the initial permutation for IG, constructed by placing the triangles into P as blocks first and the rest of vertices are placed into P uniformly at random. Then, IG will find the optimal solution in $O(n^5 m^2)$ time in expectation.

Proof: Based on **Lemma A**, the initial solution must be a global optimum or it contains a 1-clique. Let us suppose that it is a sub-optimum. **Lemma B** implies that the following process is a minimisation of the number of extra 1-cliques and getting stuck in local optima is avoided.

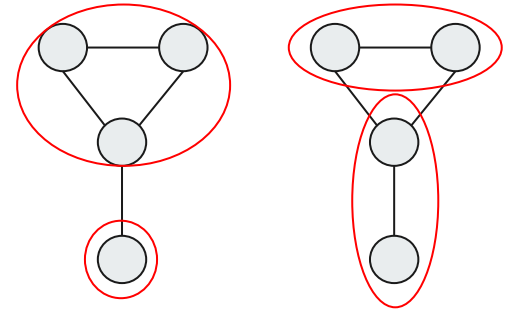
- In each time step, we have a situation, in which a 1-clique is stuck between 2-cliques and triangles.
- The probability of moving towards each direction is naturally determined by which block comes first.
- This is not influenced by the fact that we can have triangles. We have to examine two cases:

GRAPHS WITH NON-OVERLAPPING TRIANGLES

Proof(Continued):

Case 1:

- If the 1-clique is not in a triangle, it can be surrounded by 2-cliques or triangles, to which it can be connected only by a single edge (since it is in no triangle).
- In the figure, we depict the situation, when it is adjacent to a vertex in a triangle block. Such a 1-clique can be freely enhanced to a 2-clique and reduced back to 1-clique afterwards.
- Such a transformation can occur between two optima, which shows that such a clique does not contribute to the number of extra 1-cliques.

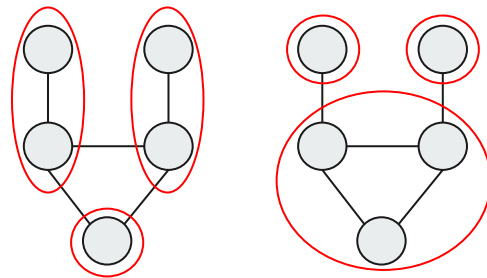


GRAPHS WITH NON-OVERLAPPING TRIANGLES

Proof(Continued):

Case 2:

- In this case, the 1-clique is in a single triangle. The other vertices of the triangle must be separated into different cliques.
- Since they cannot be in a triangle, they must each be in its own 2-clique, as shown by the figure.
- When block jump is applied to the 1-clique, this 1-clique is transformed into the triangle, and a new 1-clique can emerge on the opposite side of the triangle.
- This position depends on the ordering of cliques on the other side, for which the probability is uniformly distributed, leading to validity of the fair random walk argument.





GRAPHS WITH NON-OVERLAPPING TRIANGLES

Let G be a graph on n vertices with maximum clique size $\omega = 3$, containing only non-overlapping triangles. Let P be the initial permutation for IG, constructed by placing the triangles into P as blocks first and the rest of vertices are placed into P uniformly at random. Then, IG will find the optimal solution in $O(n^5 m^2)$ time in expectation.

Proof(Continued): In each suboptimal solution, we have that the number of extra 1-cliques is at most 2. Therefore, by applying the same cover time arguments as in Triangle Free Graphs, we have that the expected time to obtain the optimum is upper bounded by $O(n^5 m^2)$.



That's all. Thank You. Any Questions?