CS-204: Design and Analysis of Algorithms

220001037, 220001047, 220001055

April 12, 2024

1 3-Coloring Graph

To check whether a given graph is 3-colourable or not(Refer Figure 2):

- First, colour the top 3 vertices of the graph with the given3-colours(say T,F,N)
- Now connect the output of each clause with F and N.
- So, the output of the clause must be T.
- Connect the variables v_i and $\overline{v_i}$ with N
- So, possible colours for variables v_i and $\overline{v_i}$ is either T or F
- Connect literals of the clauses to the v_is
- So, for a graph to be 3-coloured, the output of the clauses should be T. So, the possible colours for the variables connected to literals would be TTT,FFF,TFF,FTT,TTF,FTF,TFT,FFT

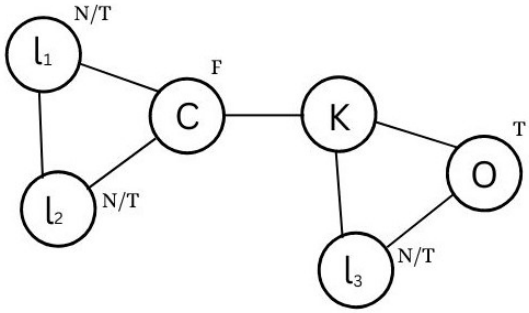

Figure 1:

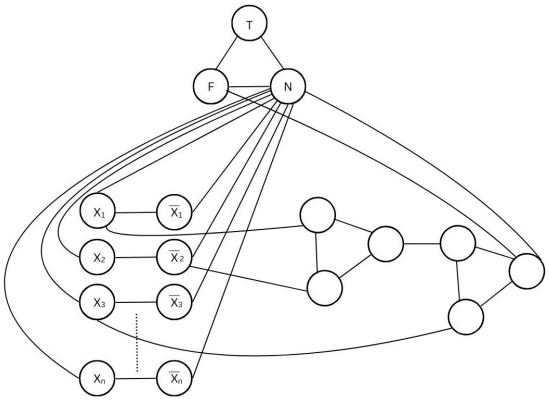

Figure 2: Check for C = (X_1 V $\overline{X}_2$ V X_3)

If we check for the case of FFF(Refer Figure 1) i.e., If literals are connected to false variables Literals should be either T/N and Let the connecting vertex has a F , Since Output has to be T L3 should be N and Now K will have no colour with regard to the definition of 3-coulouring of graph .So, Its not 3-colourable . One can check similarly for the remaining Cases to be 3-colourable.

Note that here : There are n number of variables and m number of clauses. So According to the connections between the taken Tree Graph ,There will be 6m(each clause -6)+2n (Variable and its compliment)+3(F,T,N at top) nodes and 7m(clause)+3m(connection b/w them and variables)+n(every variable and its compliment)+ 2n(both vertex and compliment connected to N)+2m (every clause output with F and N)+3(between the Top 3 F,T and N) Edges.

1.1 3-colouring Problem is NP

If any problem is in NP, then, given a certificate, which is a solution to the problem and an instance of the problem (A graph G(V, E) and an assignment of the colors (c1, c2, c3) where each vertex is assigned a color from this three colors (c1, c2, c3)), then it can be verified (Check whether the solution given is correct or not) that the certificate in polynomial time. This can be done in the following way:

For each edge u, v in graph G verify that the color c(u) != c(v)

Hence, the assignment can be checked for correctness in the polynomial-time of the graph with respect to its edges O(V+E).

1.2 3-colouring Problem is NP Hard

In order to prove that the 3-coloring problem is NP-Hard, perform a reduction from a known NP-Hard problem to this problem. Carry out a reduction from which the 3-SAT problem can be reduced to the 3-coloring problem. Let us assume that the 3-SAT problem has a 3-SAT formula of m clauses on n variables denoted by x_1, x_2,..., x_n. The graph can then be constructed from the formula in the following way:

- For every variable x_i Construct a vertex v_i In the graph and a vertex vi' denoting the negation of the variable x_i.
- For each clause c in m, add 5 vertices corresponding to values c_1, c_2, ..., c_5.
- Three vertices of different colors are additionally added to denote the values True, False, and Base (T, F, B) respectively.
- Edges are added among these three additional vertices T, F, B to form a triangle.
- Edges are added among the vertices v_i and v_i' and Base (B) to form a triangle.

The following constraints are true for graph G:

- For each of the pairs of vertices v_i and v_i', either one is assigned a TRUE value and the other, FALSE.
- For each clause c in m clauses, at least one of the literal has to hold TRUE value for the value to be true.

A small OR- gadget graph therefore can be constructed for each of the clause c = (u V v V w) in the formula by input nodes u, v, w, and connect output node of gadget to both False and Base special nodes.Let us consider the formula f = (u' V v V w') AND (u V v V w'). (Figure 2)

Now the reduction can be proved by the following two propositions:

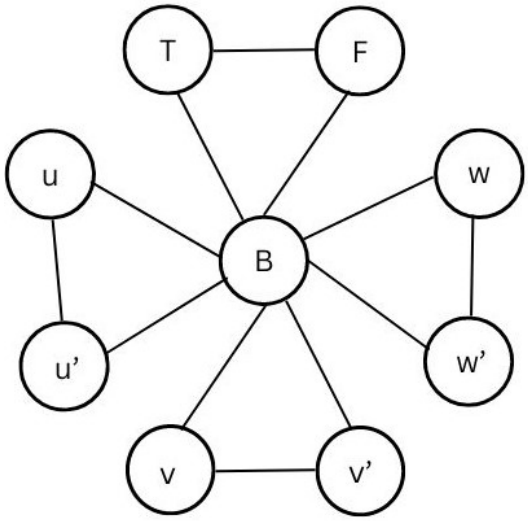

Figure 3: Arrangement of Vertices and Edges

- Let us assume that the 3-SAT formula has a satisfying assignment, then in every clause, at least one of the literals xi has to be true, therefore, the corresponding vi can be assigned to a TRUE color and vi' to FALSE. Now, extending this, for each clause the corresponding OR-gadget graph can be 3-colored. Hence, the graph can be 3-colored.
- Let us consider that the graph G is 3-colorable, so if the vertex vi is assigned to the true color, correspondingly the variable xi is assigned to true. This will form a legal truth assignment. Also, for any clause c_j = (x V y V z), it cannot be that all the three literals x, y, z are False. Because in this case, the output of the OR-gadget graph for c_j has to be colored False. This is a contradiction because the output is connected to Base and False. Hence, there exists a satisfying assignment to the 3-SAT clause.

Hence proved, 3-coloring is an NP-Complete problem.

2 Approximation Algorithm

An approximation algorithm is a method used to solve optimization problems where finding the exact optimal solution is computationally infeasible or too time-consuming. Instead of guaranteeing the best possible solution, an approximation algorithm aims to provide a solution that is "close enough" to the optimal solution, usually within a certain factor or bound.

Lets take the Vector cover Problem and find its Approximation algorithm Problem statement:

Claim: $\frac{|VC_{\text{approx}}|}{|VC_{\text{opt}}|} \leq 2$

Proof:

Lower Bound on $|VC_{\text{opt}}|$:

By definition, $|VC_{\text{opt}}|$ is the smallest size of a vertex cover that covers all edges in the graph. Therefore, $|VC_{\text{opt}}|$ must cover at least one endpoint of each edge. Since the algorithm adds both endpoints of each edge to the vertex cover, we have $|VC_{\text{approx}}| \geq |VC_{\text{opt}}|$.

Upper Bound on $|VC_{\text{approx}}|$:

Each edge in the graph contributes at most 2 to $|VC_{\text{approx}}|$ (since both endpoints are added to the vertex cover by the algorithm). Therefore, $|VC_{\text{approx}}| \leq 2 \cdot |E|$, where $|E|$ is the number of edges in the graph.

Combining Bounds:

Dividing both sides of the inequality $|VC_{\text{approx}}| \geq |VC_{\text{opt}}|$ by $|VC_{\text{opt}}|$, we get:

$$\frac{|VC_{\text{approx}}|}{|VC_{\text{opt}}|} \geq 1$$

Dividing both sides of the inequality $|VC_{\text{approx}}| \leq 2 \cdot |E|$ by $|VC_{\text{opt}}|$, we get:

$$\frac{|VC_{\text{approx}}|}{|VC_{\text{opt}}|} \leq \frac{2 \cdot |E|}{|VC_{\text{opt}}|}$$

Since $|E| \leq |VC_{\text{opt}}|$ (each vertex cover must cover at least one endpoint of each edge), we have:

$$\frac{|VC_{\text{approx}}|}{|VC_{\text{opt}}|} \leq \frac{2 \cdot |VC_{\text{opt}}|}{|VC_{\text{opt}}|} = 2$$

Conclusion: The proof shows that the size of the vertex cover produced by the algorithm is at most twice the size of the optimal vertex cover. Therefore, the approximation ratio is 2.