COL333 A3

Ojasvi Bansal(2021CS50600)
Priyadarshini Radhakrishnan(2021CS50614)

1 Approach

1.1 Part-1

- **Encoding:** We assigned ($2+k_1+k_2$) variables for each vertex in the graph(x_1,...,x_n, y_1,...,y_n, s_{11},...,s_{1k1}, s_{n1},...,s_{nk1}, p_{11},...,p_{1k1}, p_{n1},...,p_{nk1}). x_i denotes that vertex i is in G1 whereas -x_i denotes that vertex i is not in G1. y_i denotes that vertex i is in G2 whereas -y_i denotes that vertex i is not in G2. From the output of the SAT solver, we need to check the first 2n variables to check whether a vertex is in G1 or G2 or in neither.

- **Constraints:** We also tried to ensure that most of the clauses are horn clauses(clauses with 0 or 1 positive literal) to reduce the SAT solver computation time.

  - Constraint 1: Ensures a vertex i in the graph cannot belong to both the subgraphs G1 and G2. This is done by including clause "-i -i+no_vertices" which ensures that both x_i and y_i cannot be True at the same time.
  - Constraint 2: Ensures if x_i and x_j (and similarly with y) are True (that is, both vertex i and j belong to subgraph G1) then there should be an edge between them in the given graph. To avoid creating new variables for edges, we instead did the converse, i.e, if there is no edge between vertex i and j in the given graph, then x_i and x_j and similarly y_i and y_j can never be true together. This is done by adding the clauses "-i -j" and "-i+no_vertices -j+no_vertices" for vertices that don't have an edge between them.
  - Constraint 3: Ensures there are atleast k_1 vertices in G1 and atleast k_2 vertices in G2. For atleast condition, we used sequential circuit encoding. Then at the time of mapping the output of the SAT solver to the mapping file, we select any k_1 vertices indicated as True by the SAT solver if they exceed k_1 and we do the same for G2. For clauses for this constraint, please check the code in our git repo (link attached at the bottom).

1.2 Part-2

- For the second part we perform binary search on the size of the subgraph (k). The constraints used are the same as part 1 but now we only have 1+k variables for each vertex (as y, s and p variables are removed for G2). If the SAT solver returns satisfiable for a value of k (which denotes the size of the subgraph), then we increase the value of k and proceed, else we decrease the value of k and we generate the mapping file only if the value of k for which the SAT output was satisfiable, is greater than the best value found so far.

2 Code

- You can check our code here