**Mathematical Formulation**

In the MLST problem, we are given a graph, Where **V** is the set of nodes, **E** is the set of edges, and **L** is the set of labels. Let |**V|=***n*, |**E**|=*m*, and |**L|=***l*. An individual (or a chromosome) in a population is a feasible solution, which is defined as a subset **C** of **L** such that all the edges with labels in **C** construct a connected subgraph of **G** and span all the nodes

In **G** . Each label in **C** can be viewed as a gene. A random individual

can be generated by adding random labels to an empty set until a feasible solution emerges.

**Encoding:**

I used binary based encoding in which I used a bit whose size is equal to number of labels in our graph.

For example if I have 4 labels , then I’ll take a 4-bit binary number in which if i-th bit is high which means the corresponding label is considered.

e.g. ‘1001’

which means label 1 and 4 is considered.

So I selected a random number from 1 to 2^(number\_of\_label)-1

Which randomly selects any combination of labels from all possible combinations.

**Crossover:**

Crossover builds one offspring from two parents. It begins by forming the union of the two parents, then sorts the labels in the union in descending order of their frequencies. The operator adds labels in their sorted order to the initially empty offspring until the offspring represents a feasible solution.

**Algorithm:**

**Crossover(s[1],s[2])**

1) Let S = union (s1, s2) and be an empty set.

2) Sort S in decreasing order of the frequency of labels in G.

3) Add labels of S, from the first to the last, to T until T represents a feasible solution.

4) Output T.



**Mutation:**

Given a solution S, a new solution mutation can be built using our mutation operation. First, a new label is added to S. Next, labels are removed (i.e., the associated edges), from the least frequently occurring label to the most, as long as remains feasible.

**Algorithm:**

**Mutation*(S)***

1) Randomly select c not in S and let T = union(S, c)

2) Sort T in decreasing order of the frequency of labels in G.

3) From the last label of the above list to the first, try to remove one label from T and keep T as a feasible solution.

4) Repeat 3) until no labels can be removed.

5) Output T.



**Fitness function:**

For fitness function we have considered the minimum number of label required such that the edges formed by these labels completely contain the entire node (spanning tree).