1. **Spectral clustering binary classification**

The graph will be divided into 2 clusters by using spectral clustering. Most of the partitions can include all need axioms in one cluster.

Based on the graph weighted by the dissimilarity, a new graph with the same amount of vertices and edges are generated by weighted by similarity by using the Gaussian Radial basis function kernel showed below:

$$S=e^{-D}$$

In the formula, S is the similarity matrix, D is the dissimilarity matrix. After generating the similarity matrix, spectral clustering is implemented by using python sklearn package based on the similarity matrix.

The algorithm of spectral clustering of sklearn is illustrated below:

input: similarity

output: an array construct of integers which means each element’s cluster

Step 1: Generate degree matrix D from the similarity matrix

Step 2: Generate Laplace matrix L

Step 3: Calculate normalized Laplace matrix L1 by using Ncut algorithm

Step 4: Calculate L1’s eigenvalue and each eigenvalue’s eigenvector f

Step 5: Use k-means based on the normalized eigenvector

The steps illustrated above are packaged in the sklearn library. After spectral clustering, the majority automated theorem proof (ATP) task can separate into 2 parts which have 1 part includes all the needed axioms.

2. **Greedy tree**

The second experiment is finding all greedy path if one node has more than 1 minimum dissimilarity neighbors. The same BFS process is used. If the dequeue vertex u is a terminate node, then it’s neighbor will not visited and mark u’s color to black and add to the path set. The BFS continues until the queue is empty. The process is shown below:

```
for each vertex $u \in G.V - \{s\}$
    $u.color$ = WHITE
    $u.d = \infty$
    $u.\pi$ = NIL
$s.color$ = GRAY
$s.d = 0$
$s.\pi$ = NIL
$Q = \emptyset$
ENQUEUE($Q, s$)
while $Q \neq \emptyset$
    $u$ = DEQUEUE($Q$)
    if u != terminate node:
 for each $v \in G.Adj[u]$
     if $v.color$ == WHITE
         $v.color$ = GRAY
         $v.d = u.d + 1$
         $v.\pi = u$
         ENQUEUE($Q, v$)
  if u = terminate node:
        $u.color$ = BLACK
    $u.color$ = BLACK
    path.append(u)
```

The example of the search path is illustrated in Figure 1.

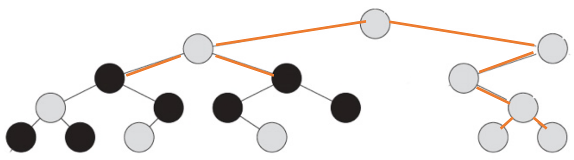

Figure 1: local beam search path

The root represents the conjecture and the black nodes represent the terminate node.

3. K-nearest neighbor algorithm to improve robustness

The K-nearest neighbor (KNN) algorithm is one of the machine learning algorithms of classification. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors, details illustrated in Figure *2*.

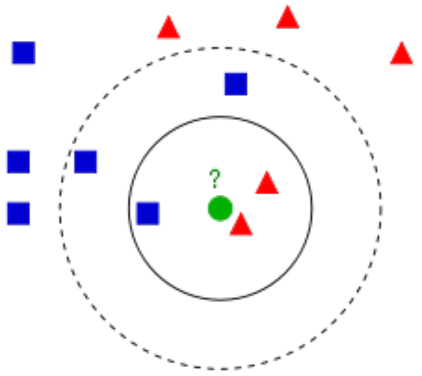

Figure 2: example of KNN

In Figure 2, the green point needs to be classified into blue or red. In this case, k=3. The green dot finds its closest 3 neighbors. In the 3 neighbors, the majority number of the category is the green dot's category.

In the axiom selection task, the KNN is used to add more axioms to the path of Local beam search all greedy path. For each axiom in the all path set, finding all neighbors and its unique dissimilarity set (UDS). The UDS is defined below:

$$\begin{gathered} \exists axiom, axiom \in Allpath, \\ \forall n, n = neighbor(axiom) \\ UDS = \bigcup \{dissimilarity(axiom, n)\} \end{gathered}$$

The UDS is ordered by non-decrease order. There are 3 kinds of UDS classified by the size of less than 3, and greater than 3 or equal to 3. If size(UDS)>=3, then this axiom can add more axioms because the first element of UDS is 0 which means the dissimilarity of this axiom itself; the second element represents the closest neighbor that is added in the local beam search; the third element represents the second closets neighbor, and so on.

In the KNN algorithm, k=1. A subset of UDS (SUDS) is generated by removing the first and the second element.