Appendix A.
Modified Bellman-Ford algorithm

Algorithms to find a minimal path from a source to destination vertex over a directed graph are collectively referred to as single source shortest path algorithms, the most common one being Dijkstra's algorithm. However, Dijkstra's algorithm does not provide a constraint on the number of edges used. By contrast, the Bellman-Ford algorithm solves for the shortest path through a process of path extension. Each iteration of the algorithm finds the shortest path containing one edge more than the previous iteration. In our case, each edge corresponds to one TLS group. The standard version of Bellman-Ford does not place any intermediate constraints limiting the number of edges used in the optimal path to any given vertex, but its methodology makes it trivial to add such a constraint. We have developed a modified version of the Bellman-Ford algorithm which imposes a global constraint on number of edges used when finding the optimal path to any destination vertex in the graph. Pseudocode for the modified algorithm is given below.

```
// Graph vertices are identified by their index in the vertices array.
Integer num_vertex
Vertex vertices[]
Record Edge {
    Integer src
    Integer dest
    Real weight
}
Edge edges[]
// Distance and predecessor are 1-dimensional arrays in Bellman-Ford.
// We use 2-dimensions to impose the edge constraint.
Integer edge_constraint
Real D(edge_constraint, num_vertex)
Integer P(edge_constraint, num_vertex)

ConstrainedBellmanFord(source_vertex)
    // Initialization
    for i in range 1 to edge_constraint
        for j in range 1 to num_vertex
          if j is source_vertex
            D(i,j) = 0.0
          else
            D(i,j) = infinity
          P(i,j) = none

    // Shortest Path Loop: the number of edges used to solve
    // for the shortest path to any vertex is at most i-1.
    for i in range 2 to edge_constraint
        for each edge in edges
          // Edge relaxation.
          if D(i,edge.dest) > (D(i-1,edge.src) + edge.weight)
              D(i,edge.dest) = D(i-1,edge.src) + edge.weight
              P(i,edge.dest) = edge.src
```