The Walking Bus Challenge

Team member:

- Boglio Simone 772263
- Croce Lorenzo 807833

Programming language:

- Java SE

Solution description

The walking bus problem starts from a given complete graph and our goal is to minimize the number of chaperons involved. This means that we must minimize the number of leaves (or maximize the number of internal nodes). We also must take into account the distance node to node (how it's explained in the document) and the total danger of the paths from nodes to the root.

We thought to use the Prim algorithm (used to create a spanning tree) and adapt it to our problem.
A brief explanation of our code is the following (this explanation uses different structures compared to the source code)

We use:

- the set of arcs ordered by crescent distance between their node M = {1-2, 3-2, 0-1, etc}
- the set of visited nodes V = {0, 1, 2, etc}
- the set of unvisited nodes NV = {3, 4, etc}\V
- the set of leaves L= {0, 1 ,etc}
- a table for the partial distance between leaves and root

Initially V and L contains only the root (the school). For each arc (a-b) we check if a or b (not both) is contained in V. In this case we check if the new node can be connected with the leaf respecting the maximum distance constraint (as indicated in the document). If the distance is respected, the node is added to V, removed from NV, the leaf is substitute by the new node, the partial distance table is updated and the cycle go over. If no one new node is found (so we are to the end of the branch), the root is added to L and the cycle restart.

The algorithm tries to connect the new node at the last insert node (the leaf). When there are no more nodes that can be appended the "branch" is finished and the leaf is no more considered for any other insertion, because the partial distance doesn't allow any other connection. When there are not arcs that respect the condition described above, the root is added to L in order to continue the exploration and start a new branch. When there are not more nodes in the NV set the tree is complete, this is our solution.

Also we built a "second" algorithm that do not use the minimum distance to choose the next node that is appended to the branch when we expand it.
To connect the next nodes in the branch this "second" algorithm use a function that takes into account:

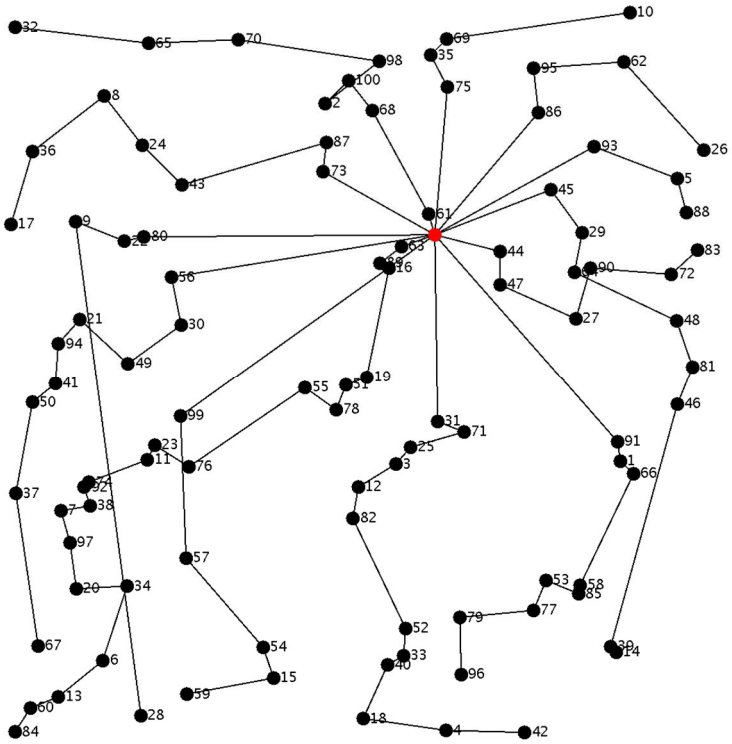
1. The distances from last point inserted and the new possible node to insert.
2. The relative angle from last point inserted and the new possible node to insert.

In this way it tries to find the best node (with a trade-off between angle and distance) in order to obtain a branch “as clean as possible” and avoid (if it is possible) arcs that exit from the “ideal way”. For “ideal way” we intend a way that goes as straight as possible (in order to minimize the alpha-distance and connect more nodes in the branch). You can imagine a straight line start from the root and the actual last point in the branch. The nodes that we find near this line, ideally, can be connected to form a valid path (if the condition about the distance is respected).
With this approach we obtain sometimes solution, respect the “first” algorithm, with one-two less nodes, the same number or one more node (this probably depends from topological distribution of the nodes).

The idea used in both approach is to create paths without cross, each node can be connected with at most one node.
Imagine this path: <root – 1 – 2 – 3>. We want to add node 4. If 4 can’t be appended to 3 because the distance condition isn’t respected, it must be connected to another node. If we add 4 to 2 (distance allowing) we create a cross, and it means create a new leaf. If we connect 4 directly to the root, we have a new leaf but the partial distance <root – 4> is minor then the partial distance <root – 1 – 2 – 4> and this means that we can add more easily a new node to 4 because the remaining distance is more.

For the practice realization of the algorithm we use Map structures indeed to Set (or List) in order to increase the performances (we need just more time for the “setup” phase).

The picture below shows graphically the result that we obtain in a problem instance (pedibus_100.dat).



The graph drawer is disabled (commented) into the source code, so the program operates with the command line only.