DIJKSTRA ALGORITHM

The following example explains the algorithm graphically. Figure below presents the network in terms of a particular origin node (A = Start) and a particular destination node (G = Finish).

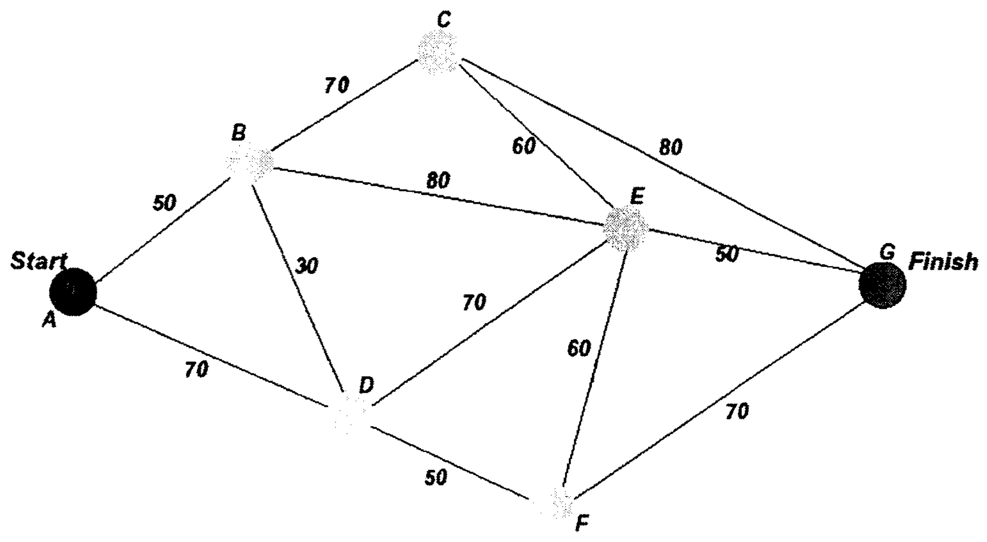

Figure 1.1: Network

In the first step the algorithm finds the node that is closest to A that has not already been put on the shortest path. In this case, it is to itself (i.e., A to A is the shortest path at this point). It thus removes A from the list of possible nodes and puts it in a shortest path node list. Next, the routine finds the node that is closest to A that has not already been put on the shortest path list. This will be node B, which is 50 units distance from A.

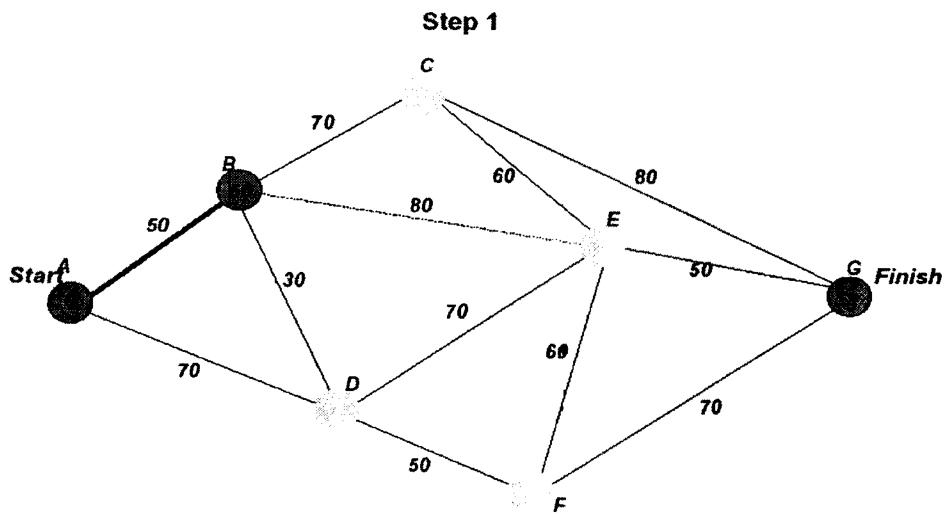

Figure 1.2: Network in Step 1

Thus, the shortest path now goes from A to B. Subsequently, node B is removed from the list of possible new nodes and is put on the shortest path list. In step 2, the routine finds the node that is closest to one of the existing nodes on the shortest path list but which has not already been put on that list. This w ill be node D, which is 70 units from A. That is, if A and B have already been put on the shortest path list, then only two nodes are connected to these - C and D. The distance from A to C is 120 (50 + 70) while the distance from A to D is 70. Thus, the routine selects node D next. Subsequently, node D is removed from the list of possible new nodes and is put on the shortest path list.

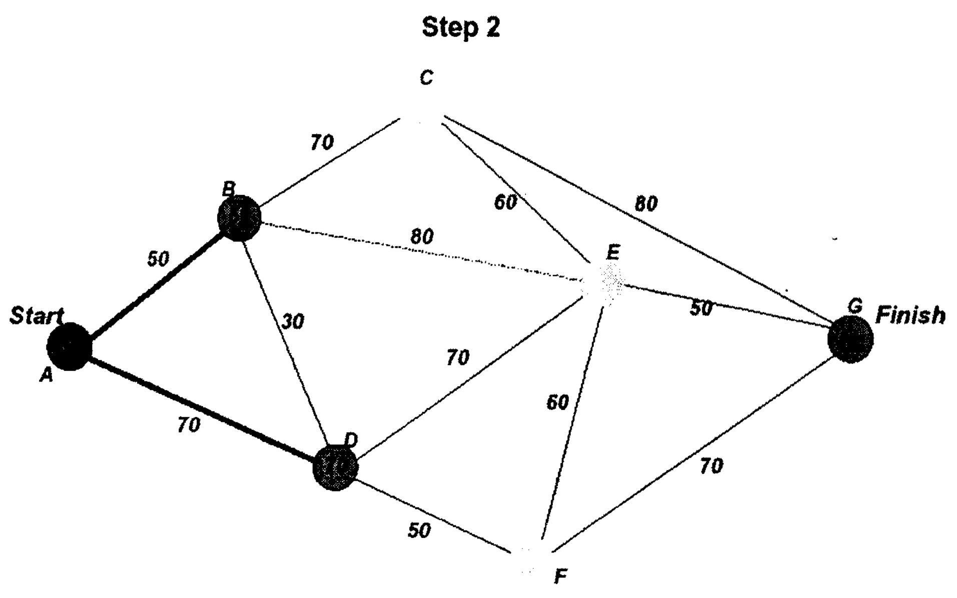

Figure 1.3: Network in Step 2

In step 3, the routine determines the node that is closest to A and which has not yet been put on the shortest path. There are two possibilities - C and F; both are 120 units distance from A. In the case of a tie, the routine chooses one, in this case node F. Subsequently, node F is removed from the list of possible new nodes and is put on the shortest path list.

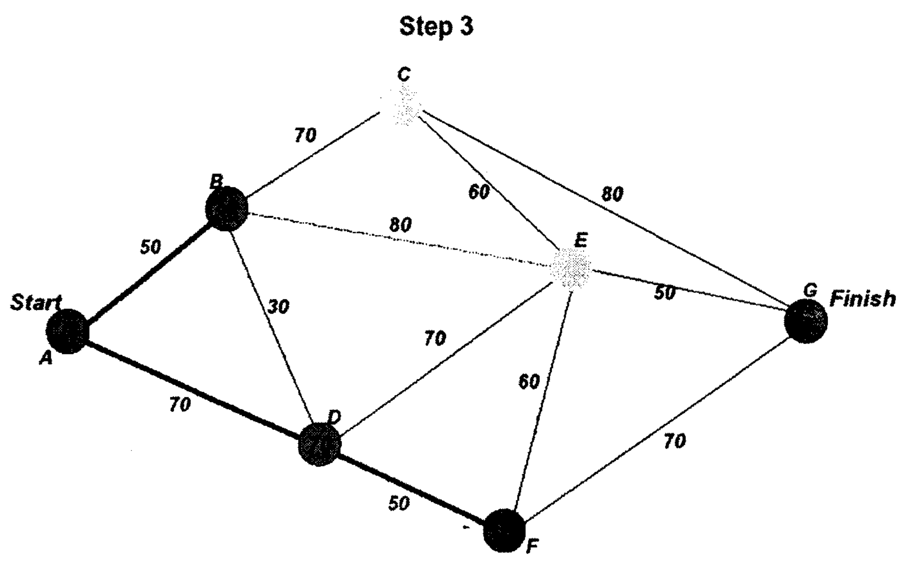

Figure 1.4: Network in Step 3

In step 4, the routine adds node C to the shortest path. Note that in step 3 if the routine had chosen node C instead of F, in this stage node F would have been selected; thus, the routine produces the same solution, just in a different order. Both nodes C and F are 120 units distance from node A. Node C is now removed the list of possible new nodes and is put on the shortest path list.

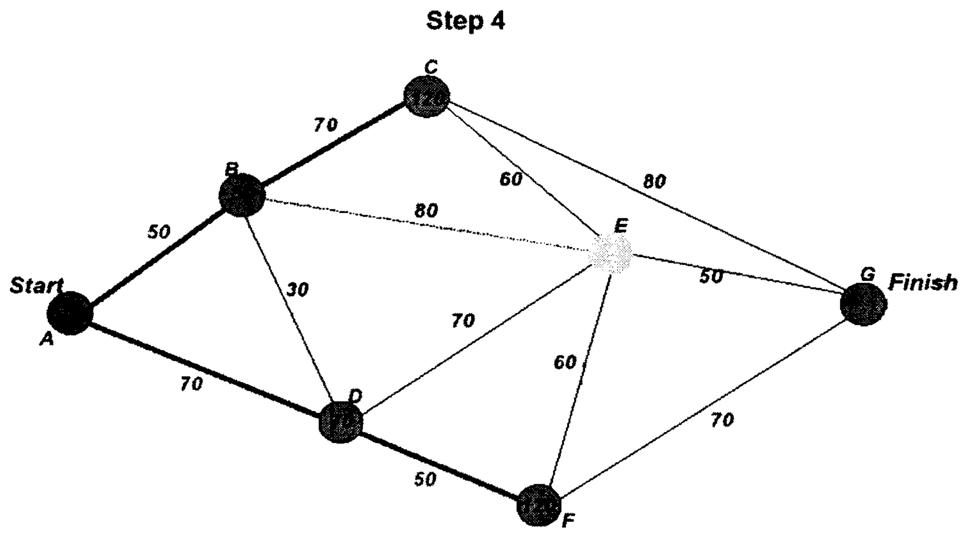

Figure 1.5: Network in Step 4

In step 5, the routine adds node E to the shortest path list because the distance to E through B is shorter than any other route that has not yet been determined (130 units from A). Notice that the path to E through C or D would have been longer than through B (180 and 140 units respectively).

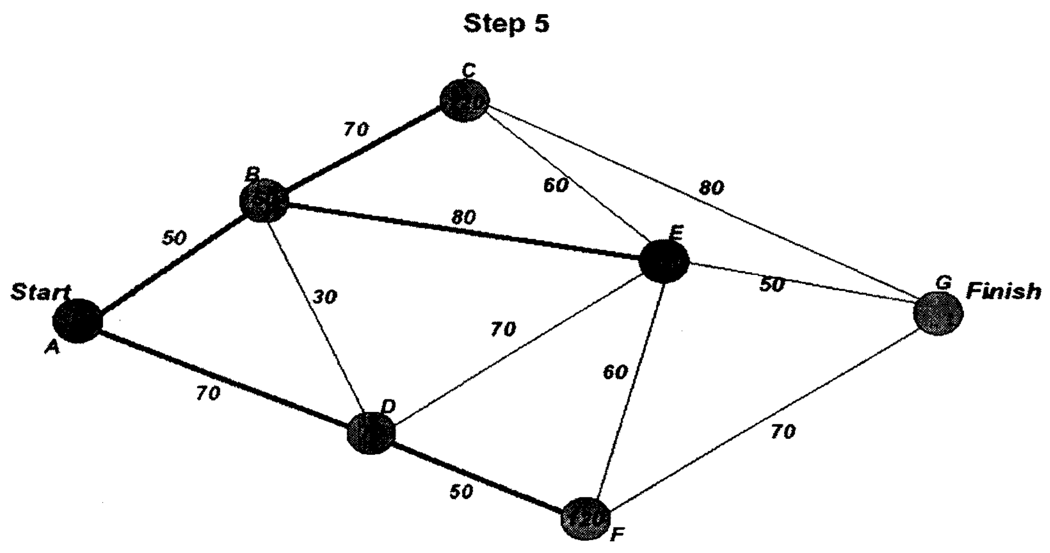

Figure 1.6: Network in Step 5

Finally, in step 6, the routine goes to the finish, node G. The path through B and E is shorter than by any other path to G (180 total units). Thus, the Dijkstra algorithm has searched every node in the network and determined a shortest path from node A to each of them. Even though we are only interested in the path from A to G, the algorithm solves all shortest paths from A to all nodes.

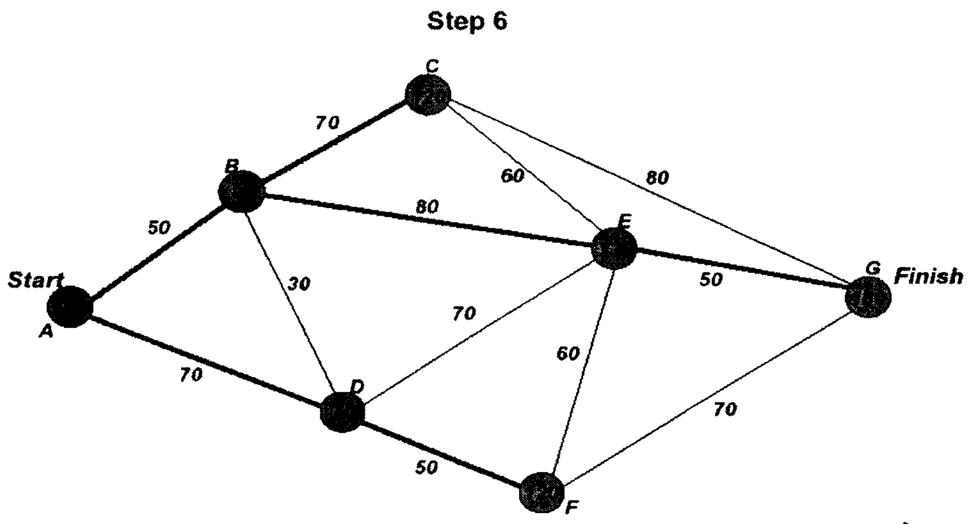

Figure 1.7: Network in Step 6

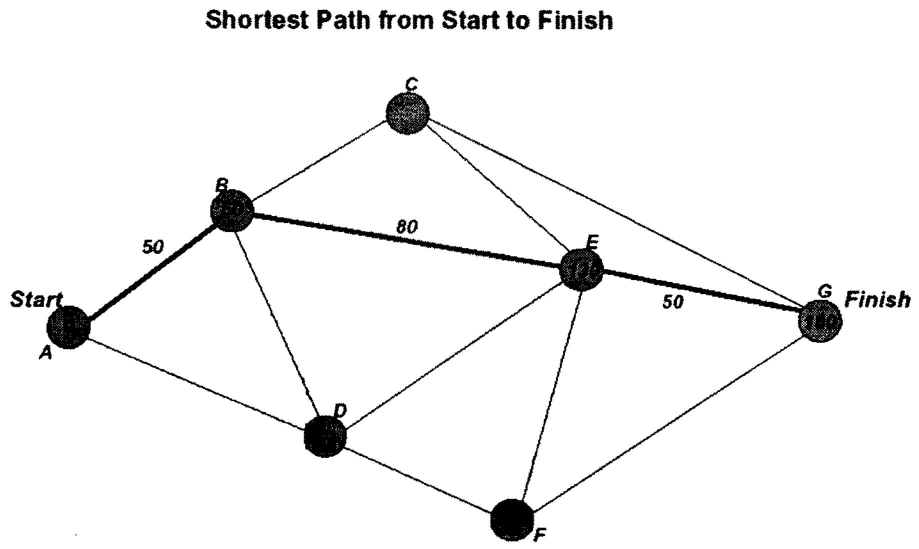

Figure 1.8: Network with the Shortest Path

As said earlier that a great number of algorithms have been developed for finding the "Shortest" or the "Best", where the best can be defined in terms of time, cost, distance, or some combination of the three. Many of these algorithms have been and are being used in Advanced Traveler Information Systems (ATIS) and Advanced Traffic Management Systems (ATMS).

ALGORITHM TO FIND ALTERNATIVES TO THE BEST ROUTE*:

An essential feature of the algorithm used to generate alternatives to the best path / route is that the magnitude of the cost inflation is adjusted automatically so that: (1) as a rule, each alternate route requested differs significantly from both the original route and also differs significantly from every other alternate route generated so far, but route duplication is never prohibited outright; (2) significant route-to-route duplication is tolerated, but route-to-route duplication seldom exceeds 70% (or some other threshold value pre-selected by the

* Alternate Route Generation – Ronald L. Blewitt Pub. No. US 2002/0143464 AI 3rd October 2002.

designer or user) until all viable route alternatives have been generated. To accomplish this behavior, the magnitude of the cost inflation is initialized to a relatively small value (20%, or some other threshold value pre-selected by the designer or user), then is automatically increased in response to the amount of route duplication encountered as alternate routes are generated. If many near-optimal alternate routes exist, very little route duplication is encountered, and the cost inflation remains near its initial value. If instead few near-optimal alternate routes exist, much route duplication occurs, and the cost inflation is quickly increased to where significantly suboptimal alternate routes are generated. In the extreme case where no near-optimal alternate routes exist, the first attempt to generate an alternate route can produce total route duplication. At that point, the cost inflation is doubled (to 40%), and the alternate route is then regenerated using that higher cost inflation.