Technium Vol. 2, Issue 4 pp.6-12 (2020)
ISSN: 2668-778X
www.techniumscience.com

Graphs in maritime transports

Cristina Dragomir

Constanta Maritime University, Romania
cristina.dragomir@cmu-edu.eu

Manole Ionuț-Constantin

"Kemal Ataturk" National College, Medgidia, Romania
manole.ionut@gmail.com

Abstract. The rapid development of graph algorithms was primarily due to exponential progress known from the development of computers. Requested to participate in the process of combinatorial optimization, graphs¸ built a fund of theorems based on which a lot of algorithms have been developed that today form the toolbasic of this field. Applications of graph algorithms in various domains, from substantiating political decisions to macroeconomic issues, from production problems in the study of electrical networks, gives it an increased importance.

Keywords. Graphs, maritime routes, paths, algorithm, nodes, edges.

1. Introduction

Our general objective is the modeling of the many situations in everyday life using graph theory. The specific objective of this article is finding the shortest path in a graph in order to design a solution to a practical problem: determining the shortest path between two points using Dijkstra's algorithm for use in a maritime network transport route model. To achieve this specific objective, a network [1] based transport model must be analyzed to minimize transport costs.

Dijkstra's algorithm is usefull in determining a minimum cost path from a start node to each of the other vertices of the graph. The algorithm [3] can be explained on a weighted graph. At the initial time the only peak for which the minimum cost path is known is the initial peak. The following structures can be used:

- a set that will retain the peaks for which the minimum cost road is already calculated;
- a vector whose size is given by the number of nodes in the graph and which stores the cost of the minimum cost path from the starting node to any node, a path that passes only through vertices from the set of selected vertices;
- a vector that holds for each vertex in the graph the vertex that precedes it on the minimum cost path.

In the minimum road between 2 points, the road between the two points is reconstructed.

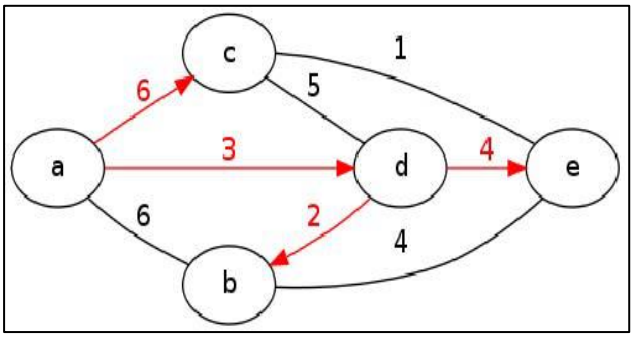

Figure 1. Dijkstra's algorithm on tree graph

2. Dijkstra's algorithm

In this section we aim at understanding the concepts of cost, relaxation of an edge, minimum path; presentation and assimilation of algorithms for calculating minimum paths.

The importance is for practical applications [4]: algorithms for determining minimum paths have multiple practical applications and represent the most commonly used class of graph algorithms:

routing within a network (of sea ships, computers etc.);

find the minimum route between two locations (Google Maps, GPS etc.);

establishing a navigation schedule in order to ensure optimal connections;

2.1 The cost of an edge and a path

Given a oriented graph G = (V, E), it's considered the function w: E → W, called the cost function, which associates to each edge a numerical value. The function range can be extended to include pairs of nodes between which there is no direct edge, in which case the value is + ∞.

The cost of a path formed by the edges p12 p23… p(n-1)n, having the costs w12, w23,…, w(n-1)n, is the sum w = w12 + w23 +… + w(n-1)n.

In the example, the cost of the path from node 1 to 5 is:

path 1: w14 + w45 = 30 + 20 = 50

path 2: w12 + w23 + w35 = 10 + 20 + 10 = 40

path 3: w13 + w35 = 50 + 10 = 60

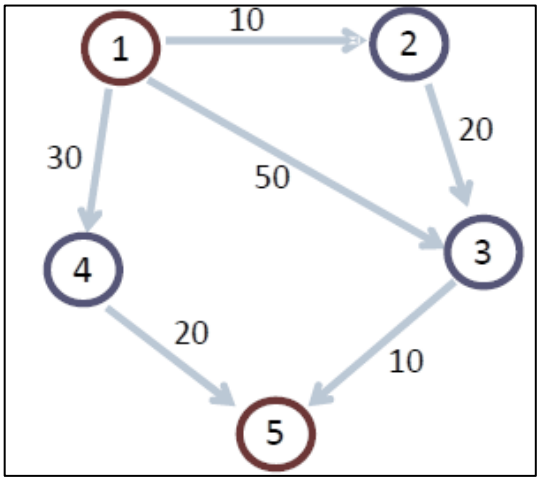

Figure 2. The cost of an edge and a path

Technium Vol. 2, Issue 4 pp.6-12 (2020)
ISSN: 2668-778X
www.techniumscience.com

2.2 The minimum cost path

The minimum cost of the path between two nodes is the minimum of the costs of the existing paths between the two nodes.

In the example above, the minimum cost path from node 1 to 5 is through nodes 2 and 3. Although, in most cases, cost is a function with non-negative values, there are situations in which a graph with negative [2] cost edges has practical relevance. Some algorithms can determine the correct path of minimum cost, including such graphs. However, it does not make sense to look for the minimum path in cases where the graph contains negative cost cycles - a minimum path [5] would have infinite length, it's cost would be reduced at each recurrence of the cycle:

In the next example, the cycle $1 \rightarrow 2 \rightarrow 3 \rightarrow 1$ has the cost -20.
path 1: w12 + w23 + w35 = 10 + 20 + 10 = 40
path 2: (w12 + w23 + w31) + w12 + w23 + w35 = -20 + 10 + 20 + 10 = 20
path 3: (w12 + w23 + w31) + (w12 + w23 + w31) + w12 + w23 + w35 = -20 + (-20) + 10 + 20 + 10 = 0

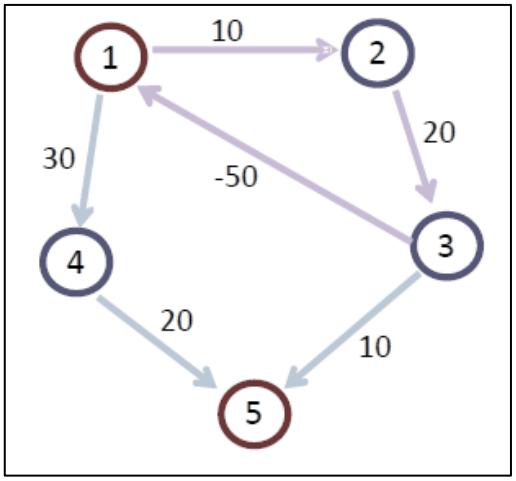

Figure 3. The minimum cost path

2.3 Relaxing an edge

The relaxation of an edge v1 - v2 consists in testing if its cost can be reduced, passing through an intermediate node u. Let w12 be the initial cost of the edge from v1 to v2, w1u the cost of the edge from v1 to u, and wu2 the cost of the edge from u to v2. If w > w1u + wu2, the direct edge is replaced by the sequence of edges v1 - u, u - v2.

In the next example, the edge 1 to 3, of cost w13 = 50, can be relaxed at cost 30, by the intermediate node u = 2, being replaced by the sequence w12, w23. Many algorithms are based on relaxation to determine the minimum path.

Technium Vol. 2, Issue 4 pp.6-12 (2020)
ISSN: 2668-778X
www.techniumscience.com

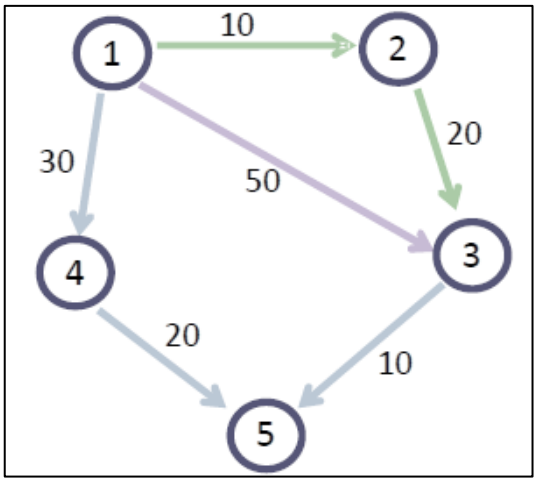

Figure 4. Relaxing an edge

2.4 Minimum single source paths

The algorithm in this section determine the minimum cost path from one source node to the rest of the nodes in the graph, based on repeated relaxations.

Dijkstra's algorithm can only be used [6] in graphs that have all non-negative edges. The algorithm is of Greedy type: the local optimum sought is represented by the cost of the path between the source node s and a node v. For each node an estimated cost d [v] is retained, initialized [7] at the beginning with the cost of the edge s → v, or with + ∞, if there is no edge.

In the following example, the source s is node 1.

The initialization will be:

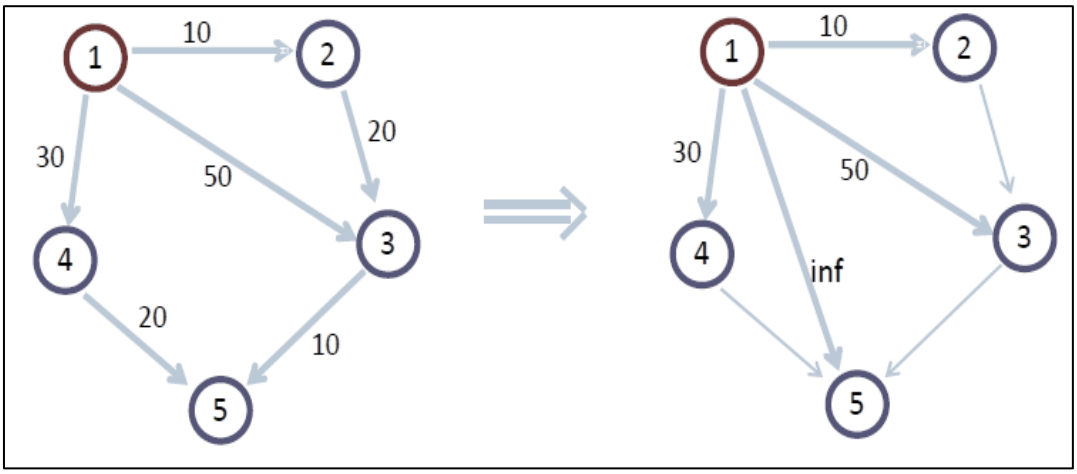

Figure 5. Initialization of source node

These paths are improved at every step, based on the other estimated costs. The algorithm repeatedly selects the node u which has, at that moment, the minimum estimated cost (compared to the source node). Next, we try to relax the rest of the costs d [v]. If d [v] <d [u] + wuv, d [v] takes the value d [u] + wuv.

In order to keep track of the edges that need to be relaxed, two structures are used: S (the set of peaks already visited) and Q (a queue with priorities, [8] in which the nodes are ordered by distance from the source) from which the node is at a minimum distance.

In S there is initially only the source, and in Q only the nodes to which there is a direct edge from the source, so which have d [node] < + ∞. In the example above, we will initialize S = {1} and Q = {2,

Technium Vol. 2, Issue 4 pp.6-12 (2020)
ISSN: 2668-778X
www.techniumscience.com

4, 3}. At the first step, node 2 is selected, which has d [2] = 10. The only node for which d [node] can be relaxed is 3. d [3] = 50> d [2] + w23 = 10 + 20 = 30.

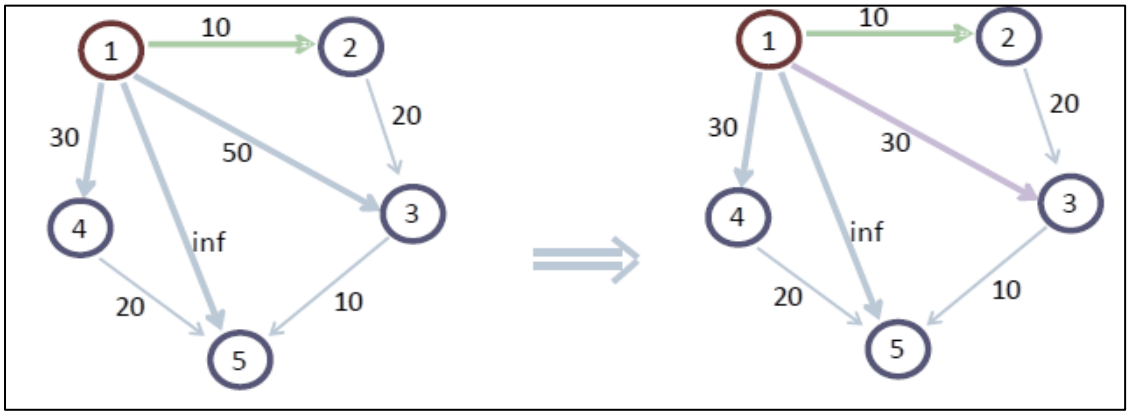

Figure 6. Relaxing nodes

After the first step, S = {1, 2} and Q = {4, 3}.
At the next step, node 4 is selected, which has d [4] = 30.
Based on it, d [5] can be modified:
d [5] = + ∞ > d [4] + w45 = 30 + 20 = 50

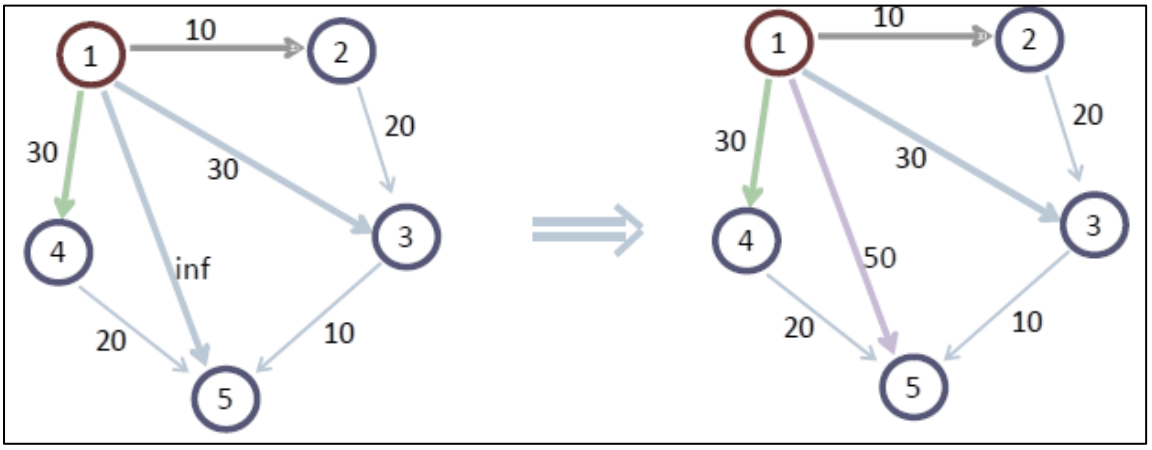

Figure 7. Node selection

After the second step, S = {1, 2, 4} and Q = {3, 5}.

At the next step, node 3 is selected, which has d [3] = 30, and d [5] changes again:
d [5] = 50> d [3] + w35 = 30 + 10 = 40.

The algorithm ends when queue Q becomes empty, or when S contains all nodes. In order to be able to determine the edges of which the minimum searched path is composed, not only its cost or end, it is necessary to retain a vector of parents P. For nodes that have a direct edge from the source, P [node] is initialized with the source, for the rest with null.

Representing the graph as an adjacency matrix leads to an inefficient implementation for any graph that is not complete, due to traversing the neighbors of node u, from the line (*), which will be executed in | V | steps for each extraction from Q, and on the whole algorithm will result | V | 2 steps.

It is preferred the representation of the graph with adjacent lists, for which the total number of operations caused by the line (*) will be equal to | E |.

The complexity of the algorithm is O (| V | 2+ | E |) if the priority queue is implemented as a linear search. In this case the function is executed in time O (| V |), and updates (Q) in time O (1). A more efficient option is to implement the queue as a binary heap. The function will be executed in time O (lg | V |); the update function (Q) will also be executed in time O (lg | V |), but the position of the node key in the heap must be known, the heap must be indexed. The obtained complexity is O (| E | Ig | V |) for a connected graph.

The most efficient implementation is obtained using a Fibonacci heap for the priority queue:

This is a complex data structure, developed especially for the optimization of the Dijkstra algorithm, characterized by a time damped by O (lg | V |) for the operation and only O (1) for the update (Q). The obtained complexity is O (| V | lg | V | + | E |), very good for rare graphs.

3. Conclusions

The Dijkstra algorithm will find the shortest path between two nodes. Also Dijkstra algorithm:
- calculates the minimum paths from one source to the other nodes;
- it cannot be used if there are negative cost edges;
- minimum complexity O (| V | lg | V | + | E |) using Fibonacci heap; in general O (| V | 2+ | E |);

Kruskal's algorithm will determine a partial minimum cost tree by connecting all nodes. Basically, Dijkastra will find a connection between the two nodes, while Kruskal will find a connection between the number of nodes.

The results obtained from those examples show that Dijkstra's algorithm is an efficient means to find the path with minimum costs from node A to node B. Also, the same results are obtained in the partial minimum cost tree using the Kruskal algorithm, but in this case procedure is much simpler with a minimum cost tree to get from point B to point A with the lowest total cost. It has been calculated that the cost of the most disadvantageous road, concluding that, in this situation, it would be 63% more expensive than the first route.

References

[1] I.C. MANOLE, E. PETAC: Social informatics and the dynamic of contemporary society. In V. Pomazan (Ed.), Proceedings of the International Conference on Interdisciplinary Studies (ICIS 2016). Interdisciplinarity and Creativity in the Knowledge Society, *London*, InTech, 2016, 81-92.

[2] J. CAHA, J. DVORSKY: Optimal path problem with possibilistic weights, In: Geoinformatics for Intelligent Transportation, Lecture Notes in Geoinformation and Cartography, *Berlin*, Springer International Publishing, 39– 50. (2015)

[3] M. GHATEE., S.M. HASHEMI: Application of fuzzy minimum cost flow problems to network design under uncertainty. Fuzzy Sets and Systems 160, 3263–3289. (2009)

[4] P. BOOMINATHAN, A. KANCHAN: Routing Planning As An Application Of Graph Theory. In: International Journal of Scientific & Technology Research, **3**, 61–66 (2014).

[5] R. MYNA, Application of Fuzzy Graph in Traffic. International Journal of Scientific & Engineering Research 2015, 6, 1692–1696.
[6] R.E. MOORE, R.B. KEARFOTT, M.J. CLOUD: Introduction to interval analysis. Society for Industrial and Applied Mathematics, *Philadelphia*, PA. (2009)
[7] T. NEUMANN: Method of Path Selection in the Graph - Case Study. TransNav, In: The International Journal on Marine Navigation and Safety of Sea Transportation 8, (2014), 557–662.
[8] T. NEUMANN: The Shortest Path Problem with Uncertain Information in Transport Networks, In: Challenge of Transport Telematics, Communications in Computer and Information Science.(2016) *Berlin*, Springer International Publishing, pp. 475–486.