Journal scientific and applied research, vol. 11, 2017
Association Scientific and Applied Research
International Journal

Original Contribution ISSN 1314-6289

THE SHORTEST PATH PROBLEM IN LOGISTICS

Andrey Bogdanov

KONSTANTIN PRESLAVSKI UNIVERSITY OF SHUMEN, SHUMEN 9712, 115 UNIVERSITETSKA STR.
anbog@abv.bg

Abstract: *The shortest path problem is one of the fundamental network flow problems. A large number of problems from diverse areas such as routing in telecommunication networks are an intrinsic part of this problem. The report examines different algorithms to solve this major problem for logistics.*

Key wards: *logistics management, Shortest Path Problems, algorithm.*

I. Introduction

There is wide variety of problems that go under the name „Shortest Path Problems". In graph theory, the shortest path problem is the problem of finding a path between two vertices (or nodes) in a graph such that the sum of the weights of its constituent edges is minimized.

The problem of finding the shortest path between two intersections on a road map (the graph's vertices correspond to intersections and the edges correspond to road segments, each weighted by the length of its road segment) may be modeled by a special case of the shortest path problem in graphs. Furthermore, the shortest path formulation can also serve as a submodel in larger, more complex models, such as, for example, in determining an optimal or near-optimal integer solution to the set covering formulation of the capacitated vehicle routing problem [2].

II. Exposition

A linear programming problem formulation for shortest path problem is given in the following, which represents the shortest path problem as a minimum cost network flow problem in which one unit of flow is sent from the source s to the t. Thus, all vertices except s and t are transshipment vertices with one unit of flow entering and leaving. The variable x_{ij} denotes the flow on *arc (i, j),* and the cost of sending one unit flow on *arc (i, j)* is given by c_{ij}.

$$\min \sum_{(ij)\in A} c_{il} x_{ij} \qquad (1)$$

$$\sum_{(sj)\in A} x_{sj} = 1 \qquad (2)$$

$$-\sum_{(ij)\in A} x_{ij} + \sum_{(ij)\in A} x_{ij} = 0$$
$$\forall j \in V/(ij) \qquad (3)$$

$$-\sum_{(ij)\in A} x_{ij} = -1 \qquad (4)$$

An optimal solution to this linear program may be obtained by standard linear programming solvers. However, many more effective algorithms are available for different kinds of shortest path problem.

Different algorithms are presented in this subsection, which determine the shortest paths from one source vertex to all other vertices on a directed graph. The first two are label setting algorithms and the last one is a label correcting algorithm [1]. They are based on an important property of shortest paths on a directed graph with no negative length cycles.

The vertices of an acyclic graph can be ordered such that if *(i, j)* $\in A$, then *i* < *j*. Such an ordering is referred to as topological ordering. The shortest paths from the source vertex *1* to all other vertices can be found by examining all nodes one by one in topological ordering because the shortest path to vertex *i+1* can only go through the vertices *1, ..., i*. Thus, in iteration i, we scan all *arcs (i, j)* emanating from i, and update the path length *d(j)* from vertex 1 to vertex j by *min {d(j), d(i) + cij}*.

Dijkstra's algorithm finds the shortest paths from one vertex to all other vertices of the graph with nonnegative arc lengths [2], and it allows for directed cycles in the graph. The algorithm assigns one of the two types of labels to each node: the permanent distance label and the temporary distance label.

For a given source node in the graph, the algorithm finds the shortest path between that node and every other [5]. It can also be used for finding the shortest paths from a single node to a single destination node by stopping the algorithm once the shortest path to the destination node has been determined. For example, if the nodes of the graph represent cities and edge path costs represent driving distances between pairs of cities connected by a direct road, Dijkstra's algorithm can be used to find the shortest route between one city and all other cities [4].

In both algorithms the shortest path from an origin vertex to a destination vertex can be obtained by terminating the algorithm once the destination vertex has been reached. This is a basic feature of the label setting algorithms.

The Ford-Bellman-Moore algorithm allows for arbitrary arc lengths and is a label correcting algorithm with a refined version [2, 4]. The graph can have negative arc lengths provided that no negative length cycle exists. A dynamic list of vertices called the queue is maintained, and initially, the only element of the queue is the source vertex *s*. Furthermore, $d(s) = 0$, and $d(j) = \infty$ for all other vertices *j*.

Like Dijkstra's Algorithm, Ford-Bellman is based on the principle of relaxation, in which an approximation to the correct distance is gradually replaced by more accurate values until eventually reaching the optimum solution. In both algorithms, the approximate distance to each vertex is always an overestimate of the true distance, and is replaced by the minimum of its old value with the length of a newly found path. However, Dijkstra's algorithm uses a priority queue to greedily select the closest vertex that has not yet been processed, and performs this relaxation process on all of its outgoing edges; by contrast, the Ford-Bellman algorithm simply relaxes all the edges. The algorithm may be improved in practice (although not in the worst case) by the observation that, if an iteration of the main loop of the algorithm terminates without making any changes, the algorithm can be immediately terminated, as subsequent iterations will not make any more changes. With this early termination condition, the main loop may in some cases use many fewer than $d(s) - 1$ iterations, even though the worst case of the algorithm remains unchanged.

Floyd–Warshall algorithm is an algorithm for finding shortest paths in a weighted graph with positive or negative edge weights (but with no negative cycles) [6]. A single execution of the algorithm will find the lengths (summed weights) of the shortest paths between all pairs of vertices. Although it does not return details of the paths themselves, it is possible to reconstruct the paths with simple modifications to the algorithm. The algorithm is also known as Floyd's algorithm or the WFI algorithm.

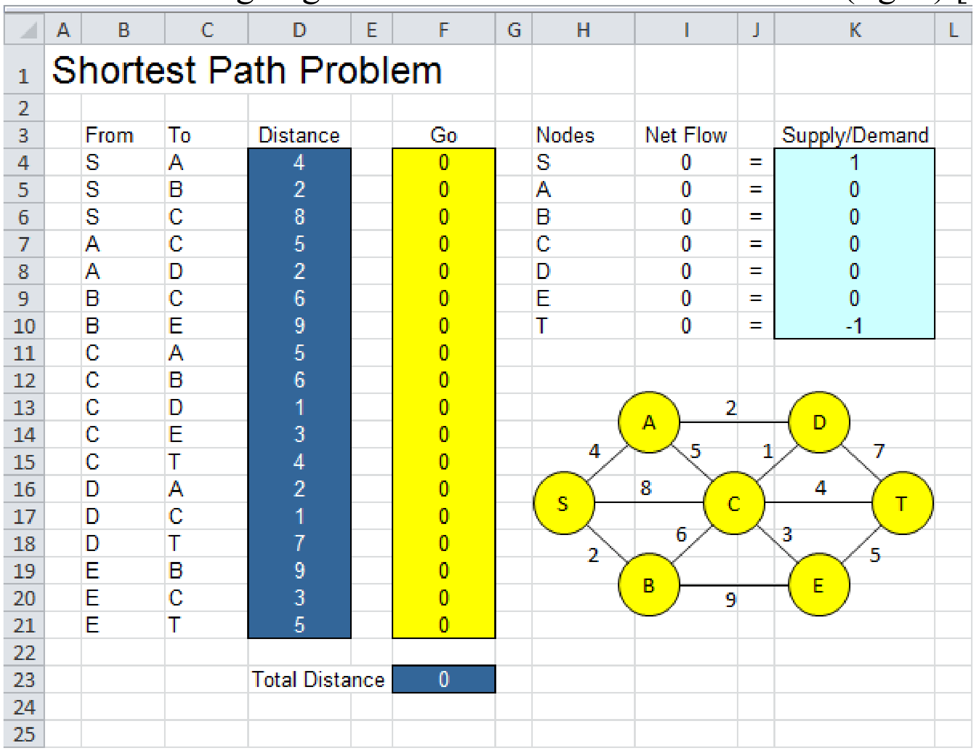
The Viterbi algorithm is a dynamic programming algorithm for finding the most likely sequence of hidden states – called the Viterbi path – that results in a sequence of observed events, especially in the context of Markov information sources and hidden Markov models. The algorithm has found universal application in decoding the convolutional codes used in both CDMA and GSM digital cellular, dial-up modems.

A generalization of the Viterbi algorithm, termed the max-sum algorithm (or max-product algorithm) can be used to find the most likely assignment of all or some subset of latent variables in a large number of graphical models, e.g. Bayesian networks, Markov random fields and conditional random fields. The latent variables need in general to be connected in a way somewhat similar to an HMM, with a limited number of connections between variables and some type of linear structure among the variables. The general algorithm involves message passing and is substantially similar to the belief propagation algorithm (which is the generalization of the forward-backward algorithm).

The shortest path problem can be solved using the solver in Excel.

For this purpose to find the shortest path from node S to node T in an undirected network. Points in a network are called nodes (S, A, B, C, D, E and T). Lines in a network are called arcs (SA, SB, SC, AC, etc).

The model we are going to solve looks as follows in Excel (fig. 1) [8].



Shortest Path Problem

From	To	Distance	Go	Nodes	Net Flow		Supply/Demand
S	A	4	0	S	0	=	1
S	B	2	0	A	0	=	0
S	C	8	0	B	0	=	0
A	C	5	0	C	0	=	0
A	D	2	0	D	0	=	0
B	C	6	0	E	0	=	0
B	E	9	0	T	0	=	-1
C	A	5	0				
C	B	6	0				
C	D	1	0				
C	E	3	0				
C	T	4	0				
D	A	2	0				
D	C	1	0				
D	T	7	0				
E	B	9	0				
E	C	3	0				
E	T	5	0				

Total Distance 0

Fig. 1 Output data to solve the task.

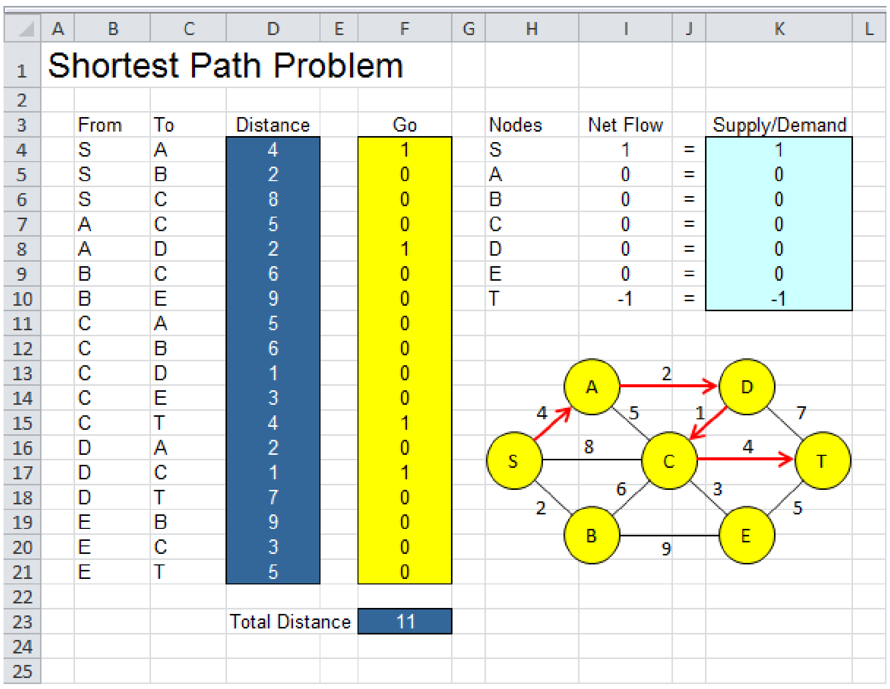
To formulate this shortest path problem, the following three questions must be answered [8].

- What are the decisions to be made? For this problem, we need Excel to find out if an arc is on the shortest path or not (Yes=1, No=0). For example, if SB is part of the shortest path, cell F5 equals 1. If not, cell F5 equals 0.

- What are the constraints of these decisions? The Net Flow (Flow Out - Flow In) of each node should be equal to Supply/Demand. Node S should only have one outgoing arc (Net Flow = 1). Node T should only have one ingoing arc (Net Flow = -1). All other nodes should have one outgoing arc and one ingoing arc if the node is on the shortest path (Net Flow = 0) or no flow (Net Flow = 0).

- What is the overall measure of performance for these decisions? The overall measure of performance is the total distance of the shortest path, so the objective is to minimize this quantity.

Explanation: The SUMIF functions calculate the Net Flow of each node. For node S, the SUMIF function sums the values in the Go column with an "S" in the From column. As a result, only cell F4, F5 or F6 can be 1 (one outgoing arc). For node T, the SUMIF function sums the values in the Go column with a "T" in the To column. As a result, only cell F15, F18 or F21 can be 1



Shortest Path Problem

From	To	Distance	Go	Nodes	Net Flow		Supply/Demand
S	A	4	1	S	1	=	1
S	B	2	0	A	0	=	0
S	C	8	0	B	0	=	0
A	C	5	0	C	0	=	0
A	D	2	1	D	0	=	0
B	C	6	0	E	0	=	0
B	E	9	0	T	-1	=	-1
C	A	5	0				
C	B	6	0				
C	D	1	0				
C	E	3	0				
C	T	4	1				
D	A	2	0				
D	C	1	1				
D	T	7	0				
E	B	9	0				
E	C	3	0				
E	T	5	0				
		Total Distance	11				

Fig. 2. Solution of the task with Excel

Conclusion: SADCT is the shortest path with a total distance of 11.

A minor modification of Dijkstra's algorithm is applied to generate the shortest paths from every vertex of a graph to a single vertex, called the sink. This time the distance label *d(i)* is associated with the path length from vertex to sink. Assume that the distance label for vertex is declared as permanent.

III. References:

[1]. Ahuja R. K., Mehlhorn, Orlin J. B. Faster algorithms for the shortest path problem. Journal of ACM, 37:213–223, 1990.

[2]. Cherkassky B. V.; Goldberg, Andrew V.; Shortest paths algorithms: theory and experimental evaluation. Mathematical Programming. 73: 2000.

[3]. Dijkstra E. A note on two problems in connection with graphs. Numerische Mathematik, 1:269–271, 1959

[4]. Hoffman K. L., Padberg M. Solving airline crew scheduling problems by branch-and-cut. Management Science, 39:657–682, 2003.

[5]. Pape U. Implementation and effciency of algorithms for the shortest route problem. Mathematical Programming, 7:212–222, 2004.

[6]. Shimbel, A. Structure in communication nets. Proceedings of the Symposium on Information Networks. New York, NY: 2003

[7]. Zwick, Uri All pairs shortest paths using bridging sets and rectangular matrix multiplication, Journal of the ACM, 49: 2002.

[8]. http://www.excel-easy.com/examples/shortest-path-problem.html