RT&A, No 4(80)
Volume 19, December, 2024

DETERMINATION OF STABILITY AND RELIABILITY OF SHORTEST PATHS IN A GRAPH THROUGH LISTS OF LABELS IN DIJKSTRA'S ALGORITHM

G.Sh. Tsitsiashvili

•

[1]Insitute for Applied Mathematics, Far Eastern Branch of Russian Academy Scinces, Russia
guram@iam.dvo.ru

Abstract

In this paper, the problem of determining all shortest paths is solved in a weighted graph. For a weighted graph, the path length is defined as the sum of the lengths of its edges. This problem is solved by generalizing the well-known Dijkstra algorithm by introducing a list of labels. In the list of labels at each vertex of the graph, the first label determines the length of the shortest path. The second label is defined by a set of vertices, from which directed edges exit to the vertex in question. To reduce the required memory and determine the reliability of the shortest paths, the number of edges of the shortest paths entering the vertices of the graph is introduced and recursively calculated. The stability of shortest paths is calculated recursively, as the number of edges, paths entering the vertices of the graph and deviating from the minimum length by a given amount. These results extend to unweighed and planar graphs.

Keywords: weighted graph, shortest path, vertex neighbours, path length, reliability, stability

1. Introduction

In this paper, we consider the problem of enumerating all the edges of the shortest paths from the starting vertex to other vertices of a directed graph (digraph) or from the face of a planar graph to the outer face. This problem is solved for weighted and planar graphs at the request of specialists in the field of transport, cartography and biotechnology, where it is currently relevant [1], [2].

For a weighted graph, the path length is defined as the sum of the lengths of its edges. For a planar graph, the path length is the number of boundaries between the faces that the path from the face to the outer face intersects. The problem of determining the minimum length from the starting vertex to other vertices of the graph in a weighted digraph is solved on the basis of the Dijkstra algorithm [3] or on the basis of the Bellman algorithm [4], [5]. A similar problem, but in an unweighed graph, is solved based on the wave algorithm [6]. And the problem of determining the length of the shortest path from each face to the outer face in a planar graph is solved using the concept of a set of neighbours - surrounding a face of a planar graph with other faces.

In this paper, the algorithms under consideration are supplemented by the concept of a list of labels. The concept of a label list is taken from mathematical programming (see, for example, [7], [8]). The list of labels consists of the first label, which determines the length of the shortest path to the vertex from the starting vertex or the length of the shortest path from the face to the outer face. For weighted graphs, the second label is a list of vertices from which the edges of the shortest paths are directed to the vertex in question. Thus, using the second labels at the vertex of a weighted or unweighed digraph, it is possible to list all the edges of the shortest paths to this vertex of the graph. For a planar graph, the second label is a list of faces that are adjacent to the face in question in the shortest paths to the outer face. Thus, using the lists of labels, it is possible to determine all the shortest paths from the starting vertex to the other vertices of the weighted digraph. Similarly, for a planar graph, using the lists of labels at the edges of the graph, you can list all the shortest paths to the outer face.

2. Inclusion of new labels in Dijkstra's algorithm

The Dijkstra algorithm solves the problem of determining the length of the shortest paths from the selected vertex of a weighted (equipped with a positive edge length) graph G. However, there is no algorithm for enumerating all edges included in the shortest paths. At this point of the paper, a modification of Dijkstra's algorithm is being built, which allows us to list all the edges of the shortest paths using lists of additional labels. Dijkstra's algorithm is constructed for a directed weighted graph (without multiple edges) G with a set of vertices V, consisting of n vertices, a set of edges E and a set of edge weights $\{s(e) > 0,\ e \in E\}$. In the graph G there is a starting vertex $\mathbf{0}$ from which all shortest paths to the other vertices of this graph are searched.

The first and second vertex labels of the graph G are determined by a recurrent procedure. First, set $R(\mathbf{0}) = 0,\ R(v) = \infty,\ v \in V,\ v \neq \mathbf{0};\ S(v) = \emptyset,\ v \in V,\ G_0 = G,\ V_0 = V,\ E_0 = E$. Next, for $0 \leq j \leq n$, select the vertex k_j from the condition $R(k_j) = \min_{v \in V_j} R(v)$, redefine the labels $R(v),\ S(v),\ v \in V_j,\ v \neq k_j$ and construct the graph G_{j+1} from the following conditions.

(**A**). If $R(k_j) + s(k_j, v) < R(v)$, then $R(v) := R(k_j) + s(k_j, v),\ S(v) := k_j$.

(**B**). If $R(k_j) + s(k_j, v) = R(v)$, then $R(v) := R(v),\ S(v) := S(v) \cup k_j$.

(**C**). If $R(k_j) + s(k_j, v) > R(v)$, then $R(v) := R(v),\ S(v) := S(v)$.

After such a redefinition of the labels, the vertex k_j is assumed to be visited and the graph G_{j+1} is constructed by removing the vertex k_j from the set of vertices V_j of the graph G_j and from the set of edges E_j incident k_j in E_j. As a result, a graph G_{j+1} is constructed with a set of vertices V_{j+1} and a set of edges E_{j+1}.

In the proposed modification of Dijkstra's algorithm, a list of the first and second labels is mapped to each vertex $v \in V$. At the end of the algorithm, the first label is $R(v)$ is equal to the length of the shortest path from the vertex $\mathbf{0}$ to the vertex v. The second label $S(v)$ is the set of neighbours u of the vertex v in the graph G, through which all the last edges (u, v) of the shortest paths from $\mathbf{0}$ to the vertex v pass. In the list of vertex labels $v \in V$, the first label is separated from the second label by the icon ; .

For the first label, this recurrent procedure coincides with Dijkstra's algorithm [3], [6]. And for the second label, using matinduction, it is easy to prove that for each j all the last edges (u, v) of the shortest paths from $\mathbf{0}$ to the vertex v form the set of vertices $S(k_j)$. Thus, the presented modification of Dijkstra's algorithm allows using the second vertex labels to determine all the shortest routes in the graph G from the starting vertex $\mathbf{0}$ to the remaining vertices of the graph.

Example 1. Figure 1 shows an example of a weighted undirected graph (a special case of a digraph with the same weights of multi directional edges) with vertex numbers in circles. Table 1 shows the lists of labels for the vertices of this graph.

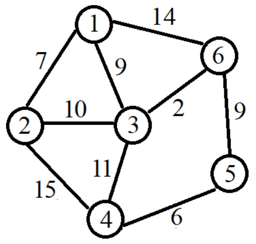

Figure 1. An example of a weighted graph.

Vertices	1	2	3	4	5	6
Label Lists	0	7;1	9; 1	20; 3	20; 6	11;3

Table 1. Lists of labels at the vertices of a weighted graph.

Remark 1. It should be noted that the presence of several vertices in the second label may be a slight change in the weights of the edges eliminated. It follows that in the case of the general situation in the (non-degenerate case), the second label at the vertices of a weighted graph can consist of only one vertex. This procedure reduces the computational complexity and the amount of memory required to implement this algorithm. However, the presence of several vertices in the second label provides the decision maker with additional degrees of freedom and makes these decisions more reliable. With small changes in the edge weights, in some cases it is even possible, on the contrary, to increase the number of vertices in the second label to increase reliability. This circumstance can be used, for example, in transport systems.

3. Edges of shortest paths in an unweighed graph

Let's now consider an unweighed graph G with a set of vertices V and a set of edges E. For such a graph, let's put $s(e) = 1$, $e \in E$. Therefore, we can proceed to the modified Dijkstra algorithm described in the previous section. However, it is more convenient to use the wave algorithm. Let $V_0 = \{a\}$ and denote V_1 the set of all neighbors of the vertex a. Let $V^1 = V \setminus V_1$, define V_2 a collection of vertices of the set V^1, which include edges from the vertices of the set V_1. Next, let's put $V^2 = V \setminus (V_1 \cup V_2)$. Continuing by induction , we get $V^k = V \setminus (V_1 \cup V_2 \cup ... \cup V_k)$,then V_{k+1} is the collection of all vertices of the set V^k, which include edges from the vertices of the set V_k. It is possible to assume the length of the shortest path from the vertex **0** to any other vertex v of the graph (the first vertex label v) to be equal to k, if $v \in V_k$. Then the second vertex label $v \in V_k$ becomes the set of vertices u of the set V_{k+1} such that the edges $(v, u) \in E$.

Figure 2 shows an example of an unweighed graph. Here, the starting vertex is marked in green, the vertices of the set V_1 are orange, and the vertices of the set V_2. are blue in the table. 2 lists of graph vertex labels are given.

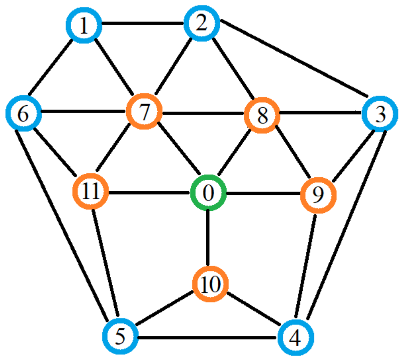

Figure 2. An example of an unweighted graph.

Vertices	0	7	8	9	10	11	1	2	3	4	5	6
Label Lists	0	1; 0	1; 0	1; 0	1; 0	1; 0	2; 7	2;7,8	2; 8,9	2; 9,10	2; 10,11	2; 7,11

Table 2. Lists of labels at the vertices of an unweighed graph.

Remark 2. For an unweighted graph, it is also sufficient to write out a set of sets V_k, $k = 1, \ldots,$ for which it is not difficult to restore the second vertex labels using the graph G. The presence of several vertices in the second label plays an important role in increasing the reliability of connections between proteins in the protein network represented by an unweighed graph.

4. Shortest paths in a planar graph

Consider the planar graph Γ[9, Chapter 1] and denote $U_0 = \{u\}$ the set of its faces with the exception of the outer face. We will look for the shortest paths from the faces of the set U_0 to the outer face. Here, the shortest path is understood to be such a polyline that intersects the minimum number of boundaries between the faces. Each face $u \in U_0$ is mapped to a subset of faces $L(u) \subseteq U_0$, touching the boundaries with the face u. Let's call $L(u)$ the set of neighbours of the face u. The face adjacent to the outer face is called the boundary, and not adjacent to the outer faces - internal. Let's denote $V_0 \subseteq U_0$ the set of boundary faces and $U_1 = U_0 \setminus V_0$ - the set of inner faces.

Define recurrent algorithm

$$U_{K+1} = \{u \in U_k : \ l(u) \subseteq U_k\}, \ V_k = U_k \setminus U_{k+1}. \tag{1}$$

We continue this recursion until the step n, when for the first time $U_{n+1} = \emptyset$ or $U_{n+1} = U_n$. It is not difficult to prove that for any $k \le n$, $u_k \in V_k$, there exists a face $v_{k-1} \in V_{k-1}$ such that $v_{k-1} \in S(v_k)$. Moreover, for any $j > 1$ intersection of $L(v_k) \cap V_{k-j} = \emptyset$. Thus, the minimum number of boundaries that a poly line must cross from the face $u_k \in V_k$ to the outer face is $k+1$. Therefore, $k+1$ can be determined by the first label of the face u_k. In turn, the second label can be taken as $L(u_k) \cap V_{k-1}$. Moreover, the directed edges coming out of the vertex u_k are represented as (u_k, u_{k-1}), $u_{k-1} \in L(u_k) \cap V_{k-1}$. It should be noted that, unlike the first two paragraphs for a planar graph, the second label is defined differently.

Example 2. Figure 3 shows an example of a planar graph in which the faces of the set V_0 are colored blue, the faces of the set V_1 are orange, and the faces of the set V_2 are green. Table 3 lists the labels of these faces.

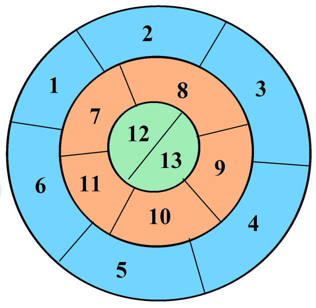

Figure 3. An example of a planar graph with faces highlighted on it.

Vertices	12	13	7	8	9	10	11	1,2,3,4,5,6
Label Lists	3; 7,8,11	3; 8,9,10,11	2;1,2,6	2; 2,3	2; 3,4	2; 4,5	2;5,6	1

Table 3. Lists of labels at the vertices of a planar graph.

Remark 3. In a planar graph, the presence of several vertices in the second label is quite common. Moreover, in cartographic applications it is not necessary to write out the second labels, it is enough to specify the sets V_k, $k = 0, \dots, n$, and represent these sets as shaded faces.

5. Reliability and stability of shortest paths

However, building label lists increases the amount of memory required to implement this algorithm. Therefore, we can replace the second label with the number of neighbours u of the vertex v in the graph G, through which all the last edges (u, v) of the shortest paths from $\mathbf{0}$ to the vertex v pass, denoting this number $N(v)$. To do this, you can only slightly change the algorithm $(\mathbf{A})$, $(\mathbf{B})$, $(\mathbf{C})$, replacing it with the following algorithm.

First we put $N(\mathbf{v}) = 0$; $v \in V$, $R(\mathbf{0}) = 0$, $R(v) = \infty$, $v \in V$, $v \ne \mathbf{0}$; $G_0 = G$, $V_0 = V$, $E_0 = E$. Next, for $0 \le j \le n$, select the vertex k_j from the condition $R(k_j) = \min\limits_{v \in V_j} R(v)$, redefine the labels $R(v)$, $N(v)$, $v \in V_j$, $v \ne k_j$ and construct the graph G_{j+1} from the following conditions.

$(\mathbf{A.1})$. If $R(k_j) + s(k_j, v) < R(v)$, then $R(v) := R(k_j) + s(k_j, v)$, $N(v) := 1$.

(**B.1**). If $R(k_j) + s(k_j, v) = R(v)$, then $R(v) := R(v),\ N(v) := N(v) + 1$.

(**C.1**). If $R(k_j) + s(k_j, v) > R(v)$, then $R(v) := R(v),\ N(v) := N(v)$.

After such a redefinition of the labels, the vertex k_j is assumed to be visited and the graph G_{j+1} is constructed by removing the vertex k_j from the set of vertices V_j of the graph G_j and from the set of edges E_j incident k_j E_j. As a result, a graph G_{j+1} is constructed with a set of vertices V_{j+1} and a set of edges E_{j+1}.

Obviously, the set of values $\{N(v),\ v \in V\}$ can be considered as a characteristic of the reliability of the shortest paths of the graph G. After all, it defines each vertex of the graph in terms of the number of edges of the shortest paths entering it. On the other hand, this set is quite simple to calculate and practically does not require additional memory.

Let us now turn to the question of the stability of shortest paths with variations in edge weights. This question is conveniently related to the computational complexity of the algorithm for determining the reliability characteristics of $N(v)$. For this purpose, the fluctuation value $\varepsilon > 0$ is set and the following modification of the previously introduced algorithm is constructed.

First we put $N(v) = 0;\ v \in V$, $R(\mathbf{0}) = 0,\ R(v) = \infty,\ v \in V,\ v \neq \mathbf{0}$; $S(v) = \varnothing,\ v \in V$; $G_0 = G$, $V_0 = V,\ E_0 = E$. Then for $0 \leq j \leq n$ we select the vertex k_j from the condition $R(k_j) = \min_{v \in V_j} R(v)$, redefine redefine labels $R(v),\ N(v),\ v \in V_j,\ v \neq k_j$ and construct a graph G_{j+1} from the conditions.

(**A.2**). If $R(k_j) + s(k_j, v) < R(v) - \varepsilon$, then $R(v) := R(k_j) + s(k_j, v),\ N(v) := 1$.

(**B.2**). If $|R(k_j) + s(k_j, v) - R(v)| \leq \varepsilon$, then $R(v) := R(v),\ N(v) := N(v) + 1$.

(**C.2**). If $R(k_j) + s(k_j, v) > R(v) + \varepsilon$, then $R(v) := R(v),\ N(v) := N(v)$.

After such a redefinition of the labels, the vertex k_j is assumed to be visited and the graph G_{j+1} is constructed by removing the vertex k_j from the set of vertices V_j of the graph G_j and from the set of edges E_j incident k_j in E_j. As a result, a graph G_{j+1} is constructed with a set of vertices V_{j+1} and a set of edges E_{j+1}. This modification of Dijkstra's algorithm allows us to calculate how much edge weight perturbations affect the final result of determining the value of $N(v),\ v \in V$.

Let's now give a table of values of $N(v),\ v \in V$ for examples 1, 2.

Vertices	0	7	8	9	10	11	1	2	3	4	5	6
Lists of $N(v)$	0	1	1	1	1	1	1	2	2	2	2	2

Table 4. Values $N(v)$ at the vertices of an unweighed graph for example 1.

Vertices	12	13	7	8	9	10	11	1,2,3,4,5,6
Values $N(v)$	3	4	3	2	2	2	2	1

Table 5. Values $N(v)$ for faces of planar graph in example 2.

6. Conslusion

In this paper, the emphasis is not on determining the lengths of the shortest paths in the graph, but on determining the shortest paths themselves. As a matter of fact, the paper does not give an algorithm for enumerating all shortest paths. The lists of labels introduced in the work only allow you to determine the shortest paths from some vertex to the starting vertex or from some face to the outer face. However, the proposed technique of labelling lists, especially the second label, makes it possible to follow well-known algorithms for determining the lengths of shortest paths: Dijkstra algorithm, wave algorithm, etc. This allows, by introducing a second label, to recursively determine its change when moving to a new vertex/face of the graph. A feature of the proposed algorithms is the introduction of a second list label into them, which at the same time is an empty or non-empty set of graph vertices. In this paper, the emphasis is not on evaluating the computational complexity of the algorithm and on estimating the amount of memory required for implementation. Therefore, this algorithm can be used for relatively small weighted graphs. It should be noted that the requirements for the uniqueness of vertices in the second label of the graph, on the one hand, reduces the computational complexity of the presented algorithms and the required amount of memory. On the other hand, the presence of several vertices in the second label makes it possible to increase the reliability of the decision about the shortest paths in the graph. Moreover, in various applications (transport systems, cartography, biotechnology) these conditions have different effects on the convenience of the above algorithms.

The research was carried out within the state assignment for IAM FEB RAS (N 075-00459-24-00).

References

[1] Tsitsiashvili G. Sh., Bocharnikov V. N., Krasnopeev S. M. (2024). Zoning of areas of the region in proximity to the external border. *Mathematical game theory and its applications*, 16 (1), p. 78-91. (In Russian).

[2] Tsitsiashvili G. Sh., Bulgakov V. P. (2022). New Applied Problems in the Theory of Acyclic Digraphs. *Mathematics*, 10(1), 45.

[3] Dijtstra E. W. (1959). A note on two problems in connection with graphs. *Numerische Mathematik*, 1 (1), p. 269-271.

[4] Bellman R. (1958). On a Routing Problem. *Quarterly of Applied Mathematics*, 16 (1), p 87-90.

[5] Ford L. R., Jr., Fulkerson D. R. Flows in Networks. Princeton University Press, 1962.

[6] Cormen Th. H., Leiserson Ch. E., Rivest R. L. Introduction to Algorithms. MIT Press and McGraw-Hill, 1990.

[7] https://ru.wikipedia.org/wiki/

[8] Goloveshkin A. V., Mikhalkovich S. S. (2022). Stable algorithmic binding to an arbitrary section of the program code. *Software systems: theory and applications*, 13 (1), p. 3-33. (In Russian).

[9] Prasolov V. V. Elements of combinatorial and differential topology. Moscow: MCNMO, 2004. (In Russian).