Applied Mathematics Letters 25 (2012) 1–5

Contents lists available at SciVerse ScienceDirect

Applied Mathematics Letters

journal homepage: www.elsevier.com/locate/aml

Speeding up the Floyd–Warshall algorithm for the cycled shortest path problem

Asghar Aini [a,1], Amir Salehipour [b,*]

[a] School of Computer Engineering, Shahid Sattari Air University, 13846-37945, Iran
[b] School of Industrial Engineering, Islamic Azad University-South Tehran Branch, Tehran, Iran

ARTICLE INFO

Article history:
Received 8 February 2011
Received in revised form 8 June 2011
Accepted 8 June 2011

Keywords:
The shortest path problem with a cycle
The Floyd–Warshall algorithm
The Rectangular algorithm

ABSTRACT

On a network with a cycle, where at least one cycle exists, the Floyd–Warshall algorithm is one of the algorithms most used for determining the least cost path between every pair of nodes. In this work a new algorithm for this problem is developed that requires less computational effort than the Floyd–Warshall algorithm. Furthermore, we show that the basis of our algorithm is much easier to understand, which might be an advantage for educational purposes. A small example validates our algorithm and shows its implementation.

1. Introduction

The shortest path problem is a fundamental and well-known problem in operations research related to finding a path between two nodes (vertices) of a graph such that the sum of the weights (cost, distance, time etc.) of its connecting edges is minimized [1–3]. The shortest path problem has many real-world applications, one common one being that of finding the quickest path through a road network. In this example, the nodes represent locations and the edges represent parts of a road that are weighted by the time (distance) required to travel each part. Fig. 1(a) depicts such a graph with four nodes and five arcs. The node '1' stands for the source (the depot), and the node '4' stands for the sink (the destination). According to Fig. 1(a) the shortest path from the source to the sink is through node '3' and has a cost of 5.

Generally the shortest path problem is categorized into cases without cycle(s) (Fig. 1(a)) and cases with cycle(s) (Fig. 1(b)). There are algorithms for both cases where an optimal solution is guaranteed [4,5]. In the cases with cycles there is no source, nor is there a sink (final destination). Thus, every node can be a source or sink.

In this work we study the shortest path problem with cycles on a network; however the results can be simply applied to cases on a graph. We review the Floyd–Warshall algorithm that finds both the shortest costs and the shortest routes between every pair of nodes on this network, and develop a new efficient algorithm for this problem that reduces the required computational effort of the Floyd–Warshall algorithm substantially. Besides, the understanding of our proposed algorithm is much easier than that of the currently available algorithms, especially the Floyd–Warshall algorithm, which could be beneficial for educational purposes. The remainder of this work is organized as follows. Section 2 provides a summary on the Floyd–Warshall algorithm. Section 3 discusses our proposed algorithm. Section 4 validates our algorithm by illustration with a small example. The work ends with the conclusion.

* Corresponding author. Tel.: +98 9102125091.
E-mail addresses: ainiasghar@yahoo.com (A. Aini), amir.salehipour@gmail.com (A. Salehipour).
[1] Tel.: +98 9123240430.

0893-9659/$ – see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.aml.2011.06.008

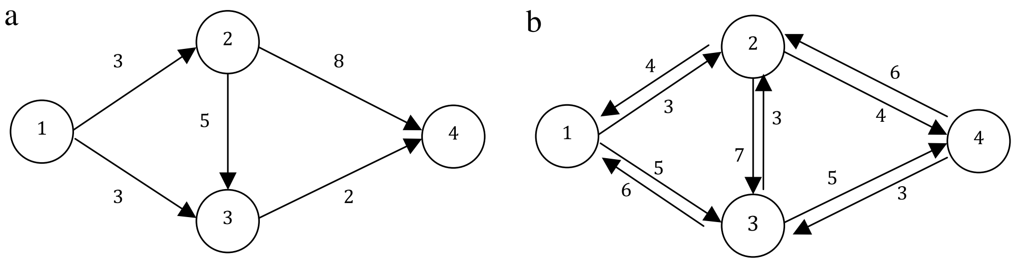

Fig. 1. Two simple networks, (a) without a cycle and (b) with cycles.

2. The Floyd–Warshall algorithm

Given a network $N(V, A)$ with node set $V = \{1, 2, \ldots, n\}$ and arc set $A = \{(i, k) : i, k \in V,\ i \neq k\}$ where $|V| = n$, and at least one cycle exists on the network, the Floyd–Warshall algorithm [6,7] is probably the most famous and one of the best algorithms for finding the shortest path between every two nodes i and k in the network N. This algorithm is based on a four-step procedure in which two square matrices D_j and R_j for $j = 0, \ldots, n$ are calculated, holding the shortest path costs and the shortest routes (sources and sinks) between every two arbitrary nodes i and k, respectively. Although the algorithm seems to be simple, it requires a lot of calculations. Given a network with n nodes, the Floyd–Warshall algorithm requires the D_j and the R_j matrices to be calculated $n + 1$ times starting from D_0 and R_0, where each has $n^2 - n$ entities. Algorithm 1 below explains the Floyd–Warshall algorithm. Details of this algorithm can be found in [6,7].

Algorithm 1. The Floyd–Warshall algorithm

Step 1. Set D_j and R_j as two square $n \times n$ matrices, where j is the stage number and n is the total number of nodes of the network.

Step 2. For $j = 0$ calculate D_0 and R_0:

$D_0 = [d_{ik}]$, where

$$d_{ik} = \begin{cases} d_{ik} & \text{if there is a direct route connecting node } i \text{ to the node } k \\ \infty & \text{if there is no direct route connecting the node } i \text{ to the node } k \\ 0 & \text{if } i = k. \end{cases}$$

$R_0 = [r_{ik}]$, where

$$r_{ik} = \begin{cases} k & \text{if there is a direct route connecting node } i \text{to the node } k \\ - & \text{if there is no direct route connecting the node } i \text{to the node } k \\ - & \text{if } i = k. \end{cases}$$

Step 3. For the remaining $j = 1, \ldots, n$ calculate the D_j and the R_j matrices as follows. Note that from now on we derive the entities of the D_j and the R_j matrices on the basis of the entities of the most recent previous matrices, i.e. the D_{j-1} and the R_{j-1} matrices:

$D_j = [d_{ik}]$ where

$$d_{ik} = \begin{cases} d_{ik} & \text{if } i = k,\ i = j, k = j \\ \min(d_{ik},\ d_{ij} + d_{jk}) & \text{otherwise.} \end{cases}$$

$R_j = [r_{ik}]$ where

$$r_{ik} = \begin{cases} k & \text{if } i = k, i = j, k = j \\ k & \text{if } d_{ik} \leq d_{ij} + d_{jk} \\ j & \text{if } d_{ik} > d_{ij} + d_{jk}. \end{cases}$$

Step 4. Repeat step 3 until the D_n and the R_n are yielded.

Now we introduce our new algorithm for the shortest path problem with cycles.

3. The Rectangular algorithm

In this section we present the major contribution of this work. This contribution is a new algorithm which reduces the amount of calculation from that required by the Floyd–Warshall algorithm substantially. This algorithm benefits from a rectangular graphical approach after which we named it. Besides requiring less computational effort and being easy to implement, the Rectangular algorithm is simple to understand, which could be an advantage for educational purposes. Algorithm 2 below explains the Rectangular algorithm. Note that the core idea of this algorithm is a set of rectangles, as illustrated in Fig. 2.

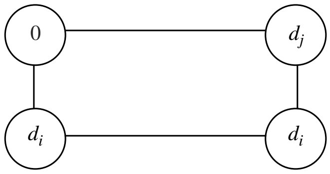

Fig. 2. Constructing a rectangle in the Rectangular algorithm. In figure, the '0' corresponds to the diagonal's zero of stage *j*.

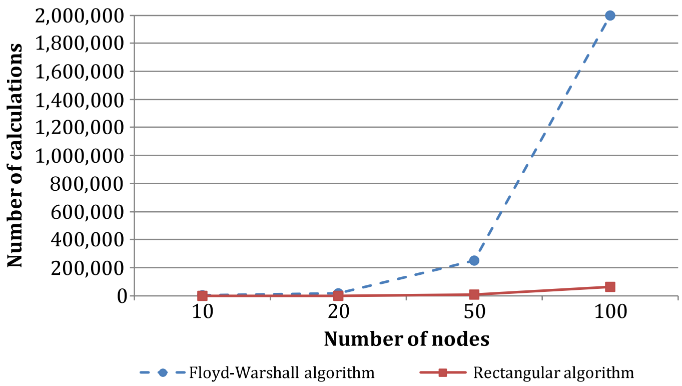

Fig. 3. The computational performance of Floyd–Warshall algorithm and the Rectangular algorithm, accomplished by performing a simulation study.

Algorithm 2. The Rectangular algorithm

Step 1. Set D_j and R_j as two square $n \times n$ matrices, where *j* is the stage number and *n* is the total number of nodes of the network.

Step 2. Derive the D_0 and the R_0 matrices by following step 2 of the Floyd–Warshall algorithm (see Fig. 2).

Step 3. For the remaining $j = 1, \ldots, n$ calculate the D_j by applying one of the following rules.

(a) If an 8 exists in any row and/or in any column of the D_j matrix, the remaining entities of that row or that column, respectively, next to it will not change. Thus they can be substituted with their values from the D_{j-1} matrix for $j \geq 1$ (Speed-Up Procedure 1).

(b) If applying rule (a) does not result in a complete D_j matrix, derive those remaining entities by drawing a set of rectangles as Fig. 2 illustrates (in fact, for each d_{ik}, $\forall i, k$ in the D_j matrix a rectangle will be formed).

(c) Derive the R_j matrix as follows. The R_j matrix is derived on the basis of the D_j matrix; thus if an entity in the D_j matrix does not change, the R_j matrix will definitely not change (as we see by comparing $\min(d_{ik}, d_{ij} + d_{jk})$ from the D_j matrix to $d_{ij} + d_{jk} < d_{ik}$ from the R_j matrix). On the other hand, if an entity changes in the D_j matrix, its associated entity in the R_j matrix can be substituted with *j* (Speed-Up Procedure 2).

Step 4. Repeat step 3 until the D_n and the R_n are yielded.

Given stage *j*, $j = 1, \ldots, n$, for each d_{ik} (for the *i*th row and *k*th column) except the diagonal zeros, a rectangle is drawn starting from the diagonal's zero of stage *j*. This diagonal zero forms the upper left corner of the rectangle (see Fig. 2). Here $d_{ik} = \min(d_{ik}, d_{ij} + d_{jk})$ as in the Floyd–Warshall algorithm (see Algorithm 1).

Obviously, in stage *j*, we cannot construct a rectangle using the *j*th row and the *j*th column. This implies that this row and this column will appear over stages (as in the Floyd–Warshall algorithm). Obviously, Speed-Up Procedure 1 and Speed-Up Procedure 2 in the Algorithm 2 have substantial effects on the speed of the Rectangular algorithm. This can be understood by considering the fact that the Floyd–Warshall algorithm requires $n^2 - n$ calculations for each matrix where a total of $2(n + 1)$ matrices are derived, while the Rectangular algorithm requires at most $n^2 - n$ calculations for each matrix, where this number can be reduced substantially by the speed-up procedures. This implies that the worst case performance of the Rectangular algorithm is similar to the normal performance of the Floyd–Warshall algorithm. A graphical explanation for this is simulated in Fig. 3.

As this figure illustrates, in a set of 100 randomly generated instances with up to 100 nodes, the time taken by the Floyd–Warshall algorithm increases rapidly. Note that, to prepare the figure for the simulation study, we have generated the computational time for the Rectangular algorithm randomly according to the facts mentioned above.

4. An example

In this section we elaborate on the Rectangular algorithm by solving an example. Given the network *N* in Fig. 4, we would like to derive the shortest paths between every pair of nodes of this network. First we solve the example using the

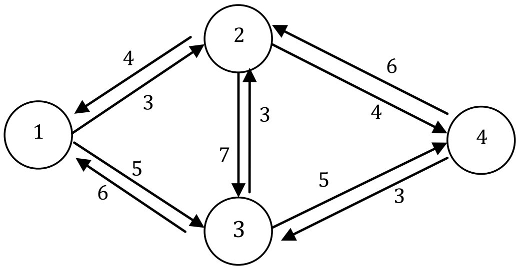

Fig. 4. The network of the example.

Floyd–Warshall algorithm, and then we apply the Rectangular algorithm to solve the example. This example clarifies how the Rectangular algorithm is more efficient than the Floyd–Warshall algorithm.

It is trivial to show that following the Floyd–Warshall algorithm, D_4 and R_4 would be

$$D_4 = \begin{bmatrix} 0 & 3 & 5 & 7 \\ 4 & 0 & 7 & 4 \\ 6 & 3 & 0 & 5 \\ 9 & 6 & 3 & 0 \end{bmatrix} \quad \text{and} \quad R_4 = \begin{bmatrix} - & 2 & 3 & 2 \\ 1 & - & 3 & 4 \\ 1 & 2 & - & 4 \\ 3 & 2 & 3 & - \end{bmatrix}.$$

However, a complete calculation for deriving only matrices D_1 and R_1 is as follows.

Applying step 3 of the Floyd–Warshall algorithm to derive D_1 would result in the following values. Note that the first row and the first column ($j = 1$) would be the same. Furthermore, the diagonal also remains the same.

$$\begin{aligned}
d_{23} &= \min(d_{23}, d_{21} + d_{13}) = \min(7, 4 + 5) = 7 \\
d_{24} &= \min(d_{24}, d_{21} + d_{14}) = \min(4, 4 + \infty) = 4 \\
d_{32} &= \min(d_{32}, d_{31} + d_{12}) = \min(3, 6 + 3) = 3 \\
d_{34} &= \min(d_{34}, d_{31} + d_{14}) = \min(5, 6 + \infty) = 5 \\
d_{42} &= \min(d_{42}, d_{41} + d_{14}) = \min(6, \infty + \infty) = 6 \\
d_{43} &= \min(d_{43}, d_{41} + d_{13}) = \min(3, \infty + 5) = 3.
\end{aligned}$$

Thus D_1 is $\begin{bmatrix} 0 & 3 & 5 & \infty \\ 4 & 0 & 7 & 4 \\ 6 & 3 & 0 & 5 \\ \infty & 6 & 3 & 0 \end{bmatrix}$. R_1 is $\begin{bmatrix} - & 2 & 3 & - \\ 1 & - & 3 & 4 \\ 1 & 2 & - & 4 \\ - & 2 & 3 & - \end{bmatrix}$ and is calculated by deriving the following values:

$$\begin{aligned}
r_{12} &= k = 2 \\
r_{13} &= k = 3 \\
r_{21} &= k = 1 \\
r_{23} &= k = 3 \\
r_{24} &= k = 4 \\
r_{31} &= k = 1 \\
r_{32} &= k = 2 \\
r_{34} &= k = 4 \\
r_{42} &= k = 2 \\
r_{43} &= k = 3.
\end{aligned}$$

We showed how the Floyd–Warshall algorithm works. This was required, as we believe our Rectangular algorithm generates optimal solutions, of course in a fewer steps. Now we continue with the example by explaining the Rectangular algorithm.

As $n = 4$, we need two square 4×4 matrices D_j and R_j for five stages, starting from the stage '0' and going to the stage '4'. The matrices D_0 and R_0 can be derived by applying step 2 of the Rectangular algorithm:

$$D_0 = \begin{bmatrix} 0 & 3 & 5 & \infty \\ 4 & 0 & 7 & 4 \\ 6 & 3 & 0 & 5 \\ \infty & 6 & 3 & 0 \end{bmatrix} \quad \text{and} \quad R_0 = \begin{bmatrix} - & 2 & 3 & - \\ 1 & - & 3 & 4 \\ 1 & 2 & - & 4 \\ - & 2 & 3 & - \end{bmatrix}.$$

Following the Rectangular algorithm to derive the D_1 and the R_1 matrices, since $j = 1$ the first row and the first column of the D_1 matrix are exactly same as the first row and the first column of the D_0 matrix (remember that the diagonal values remain intact during all stages). Thus we have to recalculate only d_{23}, d_{24}, d_{32}, d_{34}, d_{42} and d_{43}. Following Speed-Up Procedure 1, since there is ∞ in the first row and also in the first column of the D_0 matrix, the entities of the fourth row and the fourth

$$D_0 = \begin{bmatrix} 0 & 3 & 5 & \infty \\ 4 & 0 & 7 & 4 \\ 6 & 3 & 0 & 5 \\ \infty & 6 & 3 & 0 \end{bmatrix}$$

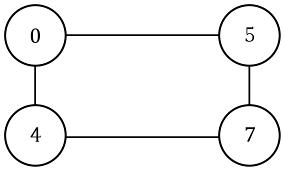

Fig. 5. The constructed rectangle for calculating entity d_{23}.

$$D_0 = \begin{bmatrix} 0 & 3 & 5 & \infty \\ 4 & 0 & 7 & 4 \\ 6 & 3 & 0 & 5 \\ \infty & 6 & 3 & 0 \end{bmatrix}$$

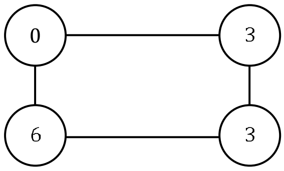

Fig. 6. The constructed rectangle for calculating entity d_{32}.

column of the D_0 matrix will appear intact in the D_1 matrix (see Speed-Up Procedure 1). Hence, we need to recalculate only d_{23} and d_{32}.

To calculate d_{23}, we construct a rectangle in D_0 starting at d_{11} as we are in stage 1 ($j = 1$). The opposite corner would be d_{23} (see Figs. 4 and 6). Having these two corners we can construct the rectangle as shown in Fig. 5.

Thus in D_1, $d_{23} = \min(7, 4 + 5)$. Similarly $d_{32} = \min(3, 6 + 3)$ (Fig. 6).

Thus $D_1 = \begin{bmatrix} 0 & 3 & 5 & \infty \\ 4 & 0 & 7 & 4 \\ 6 & 3 & 0 & 5 \\ \infty & 6 & 3 & 0 \end{bmatrix}$. It is trivial to show that $R_1 = \begin{bmatrix} - & 2 & 3 & - \\ 1 & - & 3 & 4 \\ 1 & 2 & - & 4 \\ - & 2 & 3 & - \end{bmatrix}$. It is clear that this procedure (the Rectangular algorithm) has reduced the amount of calculation substantially. Continuing this, we will stop at $D_4 = \begin{bmatrix} 0 & 3 & 5 & 7 \\ 4 & 0 & 7 & 4 \\ 6 & 3 & 0 & 5 \\ 9 & 6 & 3 & 0 \end{bmatrix}$ and $R_4 = \begin{bmatrix} - & 2 & 3 & 2 \\ 1 & - & 3 & 4 \\ 1 & 2 & - & 4 \\ 3 & 2 & 3 & - \end{bmatrix}$ (which is the same as the result from the Floyd–Warshall algorithm) which enables us to find the shortest path costs and routes between any two arbitrary nodes $(i, k) \in N$. From this simple example which contains only four nodes, it is clear that the Rectangular algorithm developed is faster and more efficient than the Floyd–Warshall algorithm. Again we emphasize that the Rectangular algorithm will reduce the amount of calculation substantially.

5. Conclusion

In this work we introduced a novel approach for calculating the shortest path in networks with cycles. The proposed approach, the Rectangular algorithm, improves on the Floyd–Warshall algorithm, one of the best available algorithms for treating this problem, in a number of ways. The Floyd–Warshall algorithm and the Rectangular algorithm have exactly the same performance in deriving the D_0 and the R_0 matrices. For the stages $j \geq 1$, however, the Rectangular algorithm derives the associated matrices much more quickly due to the reduced amount of calculation. This has been explained graphically using simulated data. As future research directions, the authors are investigating further improvements of the Rectangular algorithm, to reduce the amount of calculation required when deriving the D_j and the R_j matrices.

References

[1] R.E. Bellman, On a routing problem, Quarterly of Applied Mathematics 16 (1) (1958) 87–90.
[2] E.W. Dijkstra, A note on two problems in connection with graphs, Numeriskche Mathematik 1 (1959) 269–271.
[3] L.R. Ford, D.R. Fulkerson, Flows in Networks, Princeton University Press, Princeton, NJ, 1962.
[4] G. Gallo, S. Pallotino, Shortest paths algorithms, Annals of Operations Research 13 (1988) 3–79.
[5] B.V. Cherkassky, A.V. Goldberg, T. Radzik, Shortest paths algorithms: theory and experimental evaluation, Mathematical Programming 73 (2) (1996) 129–174.
[6] R.W. Floyd, Algorithm 97, Communications of the ACM 5–6 (1962) 345.
[7] S. Warshall, A theorem on boolean matrices, Journal of the ACM 9 (1962) 11–12.