NCGIA Core Curriculum in Geographic Information Science

Unit 064 - Representing Networks

by F. Benjamin Zhan
Department of Geography, Southwest Texas State University, San Marcos, Texas

---

Advanced Organizer

Topics covered in this unit

- What are the basic elements of a network?
- How are networks represented?
- What are the commonly used shortest path algorithms
- What are the common application areas in which networks are used?

Intended learning outcomes

After reading this unit, you should be able to:

- formally define a network and illustrate the definition with examples
- understand the advantages and disadvantages of different ways of representing a network
- describe the forward star representation of a network
- choose a shortest path algorithm for computing shortest paths on a network based on the characteristics of the problem
- cite examples of typial network application areas

Full Table of Contents

Metadata and Revision History

Unit 064 - Representing Networks

1. What are the basic elements of a network?

- A *network* is a system of linear features connected at intersections and interchanges.
  - These intersections and interchanges are called nodes.
  - The linear feature connecting any given pair of nodes is called an arc.
- Formally, a *network* is defined as a directed graph $G = (N, A)$ consisting of an indexed set of nodes N with $n = |N|$ and a spanning set of directed arcs A with $m = |A|$, where n is the number of nodes and m is the number of arcs.
  - Each arc on a network is represented as an ordered pair of nodes, in the form from node i to node j, denoted by (i, j).
- In the GIS literature, a *network arc* is often called a *network link*.

- In addition to nodes and arcs, other basic elements of a network commonly used in GIS include paths, tours, stops, centers, and turns.
  - A shortest *path* is the shortest or least cost path from a source node (origin) to a destination node. In practice, pathfinding is to find the shortest or most efficient way to visit a sequence of locations.
  - A *tour* is similar to a path, but a tour is closed path, meaning that the first node and the final node on the path are the same node on the network.
  - A *stop* is a location visited in a path or a tour.
  - A *center* is a location where certain resources are supplied.
  - A *turn* on a network is the transition from one arc to another arc on a network.

- Examples of real world networks include road networks, telecommunication networks, and river networks, among others.
  - An example network is given in Figure 1.
    - In this example, nodes are represented by the small circles and the arcs are represented by the lines connecting the nodes.
    - The number in each circle is the node identifier (ID).
    - The numerical value next to each arc can be considered the distance (time) of traversing that arc.
    - A direction is also given for each arc.
    - For convenience of discussion, we will use this example network throughout this unit.

2. How are networks represented?

- How a network is represented in a computer is vital to the performance of a particular

network analysis software package.

- The tradeoff of choosing a particular data structure is often between speed and storage space.
- In network analysis, commonly used representations of a network include
  - Node-Arc Incidence Matrix
  - Node-Node Adjacency Matrix
  - Adjacency Lists
  - Forward and Reverse Star Representation
- Past research has demonstrated that the Forward and Reverse Star Prepresentation is the most efficient among all existing network data structures for representing a network (Ahuja et al. 1993, pp. 31-37; Cherkassky et al. 1993).

- We only discuss Node-Node Adjacency Matrix and Forward and Reverse Star Prepresentation in this unit.
  - The reason for this choice is that the Node-Node Adjacency Matrix is the most basic form of representing network topology and the Forward and Reverse Star Prepresentation is the most efficient.
- Detailed discussions about other representations can be found in, for example, Ahuja et al. (1993, pp. 31-37).

2.1. Node-Node Adjacency Matrix

- Below is the Node-Node Adjacency Matrix representation of the network given in Figure 1.
  - The rows and columns in the matrix correspond to the nodes on the network.
  - A non-zero element in the *i*th row and *j*th column in the matrix represents the numerical value associated with arc *(i, j)*.
  - A zero element in the matrix in the ith row and jth column in the matrix indicates that there exists no arc going from node *i* to node *j*.

Table 1. Node-Node Adjacency Matrix of the example network

0	4	5	0	0	0
0	0	0	8	3	0
6	0	0	7	0	6
9	0	0	0	2	0
0	5	0	0	0	0
0	0	8	0	0	0

- The storage space required for the Node-Node Adjacency Matrix representation is *an^2* for a network with *n* nodes, where *a* is a constant.

- The advantages of the Node-Node Adjacency Matrix representation are that it is very easy to implement and is suitable for dense networks.

2.2. Forward and Reverse Star Representation

- The Forward and Reverse Star representation stores the arcs eminating from the nodes in a single array.
- In constructing the Forward Star representation of a network, a unique sequence number is assigned to each arc to obtain the ordering of the arc list.
    - Arcs are numbered in the following order:
        - first arcs eminating from node 1 are numbered, then those from node 2, and so forth.
        - Arcs eminating from the same node are numbered in an arbitrary fashion.
    - Once this list of ordered arcs are obtained, data associated with the arcs are stored in single arrays sequentially.
        - For example, for any arc (i, j), if it is numbered arc 10, then the starting-node, ending-node, and length of this arc are stored in the array positions starting-node(10), ending-node(10), and length(10).
- In addition to the list of ordered arcs, a pointer is also maintained for each node *i*, denoted by *pointer(i)*.
    - The numerical value associated with *pointer(i)* is the smallest-numbered arc eminating from node *i*.
    - If there exists no arc going out from node *i*, then *pointer(i)* is set to be equal to *pointer(i+1)*.
    - For consistency, we set *pointer(1) = 1* and *pointer(n+1) = m+1*.
- Tables 2 and 3 are the Forward Star representation of the network example given in Figure 1.
    - In the arrays below, the first row and first column are added for illustrative purposes only.

Table 2. A list of order arcs in the Forward Star Representation

Arc No.	Starting-node	Ending-node	Arc-length
1	1	2	4
2	1	3	5
3	2	4	8
4	2	5	3
5	3	1	6
6	3	4	7
7	3	6	6
8	4	5	2

9	4	1	9
10	5	2	5
11	6	3	8

Table 3. Pointer to each node in the Forward Star Representation

Corresponding node	Element value
1	1
2	3
3	5
4	8
5	10
6	11
(7)	12

- The Forward Star representation is a data structure that can be used to efficiently determine the set of arcs outgoing from any node.
- On the flip side, the Reverse Star representation is a data structure that provides an efficient means to determine the set of incoming arcs for any node.
    - The Reverse Star representation of a network can be constructed in a manner similar to the Forward Star representation.
    - The only difference is that *incoming* arcs at each node are numbered sequentially.

- Tables 4 and 5 are the Reverse Star Representation of the example network shown in Figure 1.

Table 4. A list of order arcs in the Reverse Star Representation

Arc No.	Starting-node	Ending-node	Arc-length
1	3	1	6
2	4	1	9
3	1	2	4

4	5	2	5
5	1	3	5
6	6	3	8
7	2	4	8
8	3	4	7
9	4	5	2
10	2	5	3
11	3	6	6

Table 5. Pointer to each node in the Reverse Star Representation

Corresponding node	Element value
1	1
2	3
3	5
4	7
5	9
6	11
(7)	12

- There is a significant amount of duplicate information when both the forward star and reverse star representations are stored in a computer.
  - To avoid the duplication, we only maintain a single array called *trace* which stores the arc numbers in the forward star representation.
  - The sequence in which the arc numbers are stored corresponds to the sequence in the reverse star representation.
    - For example, the first arc in the reverse star representation is arc (3, 1) whose arc number in the forward star representation is arc number 5. Therefore, we have *trace(1) = 5*.
    - Similarly, we have *trace(8) = 6*, and so on.
    - Of course, the size of the single array *trace* is *m.*

- A compact forward and reverse star representation of the example network is given in Figure 2.

- The storage space required for the Forward and Reverse Star representation is $an+bm$ for a network with n nodes and m arcs, where a and b are constants, .

- The advantages of the Forward and Reverse Star representation are that it saves space, it is efficient to manipulate, and it is suited for dense as well as sparse networks.

2.3. Representation of Network Attributes

- The key to network representation is to represent nodes, arcs and network topology efficiently.

- Once the nodes, arcs, and network topology are efficiently represented, other data and information associated with nodes, arcs, stops, centers, and turns can be represented as attributes either associated with nodes or arcs.

- Different GIS packages may have different network data models.

---

3. Computation of shortest paths on a network

- The computation of shortest path algorithms is a vital component of any network analysis task.
  - For many network analysis tasks, the computation of shortest (fastest, least cost) paths is almost always the first step because shortest paths are often needed as input to "higher level" models.
  - When the network involved is large, the computation of shortest paths is a computationally intensive process.
  - Therefore, the choice of the fastest and most efficient shortest path algorithm is a very important task in network analysis.

- Existing shortest path algorithms can be categorized into two groups: *label-setting* and *label-correcting*.
  - Both groups of algorithms are iterative and both employ the *labeling method* in computing one-to-all (one node to all other nodes) shortest paths.
  - These two groups of algorithms differ in the ways in which they update the estimate (i.e., upper bound) of the shortest path distance associated with each node at each iteration and in the ways in which they converge to the final optimal one-to-all shortest paths.

In *label-setting* algorithms, the final optimal shortest path distance from the source node to the destination node is determined once the destination node is scanned (Gallo and Pallottino 1988; Ahuja *et al*. 1993).
    - Hence, if it is only necessary to compute a one-to-one shortest path, then a label-setting algorithm can be terminated as soon as the destination node is scanned, and there is no need to exhaust all nodes on the entire network.
- In contrast, a *label-correcting* algorithm treats the shortest path distance estimates of all nodes as temporary and converges to the final one-to-all optimal shortest path distances until its final step when the shortest paths from the source node to all other nodes are determined.

- A key operation in many shortest path algorithms is the *labeling method* (Gallo and Pallottino 1988; Ahuja *et al*. 1993).
    - The labeling method employs a *scanning operation* to scan each node progressively until all nodes on a network are scanned.
    - During the scanning process, the distance labels associated with nodes to which there exist an outgoing arc from the node being scanned is updated.
    - A formal discussion of the labeling method and related scanning process can be found in Ahuja *et al*. (1993), Cherkassky *et al*. (1993), and Zhan (1997).
- Zhan and Noon (1998) conducted an extensive computational evaluation of the 15 algorithms using real road networks.
    - These 15 algorithms, along their complexity and references, are summarized in Figure 3.
    - Based on Zhan and Noon's evaluation, the fastest algorithms for computing shortest paths on real road networks are:
        - the Pallottino's graph growth algorithm implemented with two queues (TWO-Q) and
        - the Dijkstra's algorithm implemented with approximate buckets (DIKBA).
    - Zhan and Noon also suggest avoiding the Bellman-Ford-Moore implementations (BF and BFP) and the naïve implementation of Dijkstra's algorithm (DIKQ).
- In addition to the algorithms mentioned above, another commonly used shortest path algorithm is the Floyd-Warshall algorithm.
    - This algorithm is normally used for computing all-to-all (from every node to every other node) shortest paths.
    - This algorithm is very easy to implement, but a distance matrix has to be maintained during the execution of this algorithm.
    - Thus, the memory requirement can be fairly large when large network is involved. The complexity of this algorithm is O(n^3).
    - A detailed description of this algorithm can be found in Ahuja *et al*. (1993, pp.147-150)

---

4. Common network operations and applications

- Most of the topics described below deserve a separate unit for a better explanation. However, to limit the length of this unit, these topics are only briefly defined and

described below.

4.1. Common network operations

- *Pathfinding* is the process to find the shortest, least cost, or most efficient path or tour on a network.
- *Tracing* is the process to determine a connected portion of a network that are either flow from this connected portion of the network to a given node or flow from a given node to this connected portion of the network.
- *Allocation* is the process to assign portions of a network to a location (e.g., a center) based on some given criteria.

4.2. Common network applications

4.2.1. Geocoding

- *Geocoding* is the process for building a relationship between locational data in a database and street address data that are normally in a tabular format.
- In many applications, there are only tabular address data available. Thus, geocoding provides a very convenient mechanism to estabish a database relationship between geographic locations and addresses.
- There are many examples of geocoding. For example, in retail analysis, customers' addresses can be used to create maps showing locations of different customers with different shopping behaviors. In crime mapping and analysis, addresses of crime incidents can be used to create maps for identifying "hot spots".

4.2.2. Location-allocation

- *Location-allocation* is the process of determing the optimal locations for a given number of facilities based on some criteria and simultaneously assigning the population to the facilities.
- Location-allocation analysis is commonly used in both the public and private sectors. The determination of locations for retail stores, restaurants, banks, factories, and warehouses is used in the private sector. In the public sector, the choice of locations for libraries, hospitals, post offices, and schools can be supported by analysis results from location-allocation models.

4.2.3. Business logistics

- The optimization of vehicle routing and delivery scheduling is vital for many business operations. *Business logistics* is concerned with such an optimization. The combined power of GIS and network analysis makes GIS an ideal environment for analyses related to business logistics.

4.2.4. Spatial interaction and gravity modeling

- The interaction between different locations in geographic space and the mathematical modeling of the interaction are important in application areas such as transportation and retail analyses. Gravity models are commonly used to support these analyses.
- Gravity modeling can be conveniently supported through network analysis in a GIS environment.

4.2.5. Dynamic segmentation

- Dynamic segmentation is a particular network model used to represent, analyze, query, and display linear features. The basic difference between dynamic segmentation and the network representations discussed above is that dynamic segmentation has the flexibility to associate an attribute to a portion of an arc or several arcs (e.g., through the definition of a route).
- Dynamic segmentation is commonly used to model linear features such as highways, river networks, power lines, city streets, and telephone lines.

---

5. References

- Ahuja, R. K., Magnanti, T. L., and Orlin, J. B. (1993) *Network Flows: Theory, Algorithms and Applications*. Englewood Cliffs, NJ: Prentice Hall.

- Ahuja, R. K., Mehlhorn, K., Orlin, J. B., and Tarjan, R. E. (1990) Faster Algorithms for the Shortest Path Problem. *Journal of Association of Computing Machinery* 37, 213-223.

- Bellman, R. E. (1958) On a Routing Problem. *Quarterly of Appllied Mathmatic*s 16, 87-90.

- Cherkassky, B. V., Goldberg, A. V., and Radzik, T. (1993) Shortest Paths Algorithms: Theory and Experimental Evaluation. Technical Report 93-1480, Computer Science Department, Stanford University.

- Dial, R. B. (1969) Algortihm 360: Shortest Path Forest with Topological Ordering. *Communications of the ACM* 12, 632-633.

- Dial, R. B., Glover, F., Karney, D., and Klingman, D. (1979) A Computational Analysis of Alternative Algorithms and Labeling Techniques for Finding Shortest Path Trees. *Networks* 9, 215-248.

- Dijkstra, E. W. (1959) A Note on Two Problems in Connection with Graphs. *Numeriche*

Mathematik 1, 269-271.

- Fredman, M. L., and Tarjan, R. E. (1987) Fibonacci Heaps and Their Uses in Improved Network Optimization Algorithms. *Journal of the ACM* 34, 596-615.

- Gallo, G., and Pallottino, S. (1988) Shortest Paths Algorithms. *Annals of Operations Research*, 13, 3-79.

- Glover, F., Glover, R., and Klingman, D. (1984) Computational Study of an Improved Shortest Path Algorithm. *Networks* 14, 25-37.

- Glover, F., Klingman, D., and Philips, N. (1985) A New Polynomially Bounded Shortest Paths Algortihm. *Operations Research* 33, 65-73.

- Goldberg, A. V., and Radzik, T. (1993) A Heuristic Improvement of the Bellman-Ford Algorithm. *Applied Mathematics Letters* 6, 3-6.

- Hung, M. H., and Divoky, J. J. (1988) A Computational Study of Efficient Shortest Path Algorithms. *Computers & Operations Research* 15, 567-576.

- Mondou, J.-F., Crainic, T. G., and Nguyen, S. (1991) Shortest Path Algorithms: A Computational Study with the C Programming Language. *Computers & Operations Research* 18, 767-786.

- Pallottino, S. (1984) Shortest-Path Methods: Complexity, Interrelations and New Propositions. *Networks* 14, 257-267.

- Pape, U. (1974) Implementation and Efficiency of Moore Algorithms for the Shortest Root Problem. *Mathematical Programming* 7, 212-222.

- Zhan, F. B. (1997) Three Fastest Shortest Path Algorithms on Real Road Networks: Data Structures and Procedures. *Journal of Geographic Information and Decision Analysis.* 1(1).

- Zhan, F. B., and Noon, C. E. (1996) Shortest Path Algorithms: An Evaluation Using Real Road Networks. *Transportation Science,* 32(1):65-73.

6. Exam and discussion questions

1. For a given network, compute the arc to node ratio.

2. Construct the forward star and reverse star representation of a given network.

3. What are the commonly used shortest path algorithms?

4. Give five different examples of network application.

Citation

To reference this material use the appropriate variation of the following format:

F. Benjamin Zhan. (1998) Representing Networks, *NCGIA Core Curriculum in GIScience*, http://www.ncgia.ucsb.edu/giscc/units/u064/u064.html, created November 5, 1998.

Created: November 5 , 1998. Last revised: December 23, 1998.

Unit 064 - Representing Networks

Table of Contents

Unit 064 - Representing Networks

Metadata and Revision History

Page contents

---

1. About the main contributor

- author: F. Benjamin Zhan, Department of Geography, Southwest Texas State University, San Marcos, Texas

2. Details about the file

- unit title: Representing Networks
- unit key number: Unit 064

3. Key words

4. Index words

5. Prerequisite units

6. Subsequent units

7. Revision history

- November 5, 1998 - original draft created
- December 23, 1998 - original draft posted

- Back to the Unit

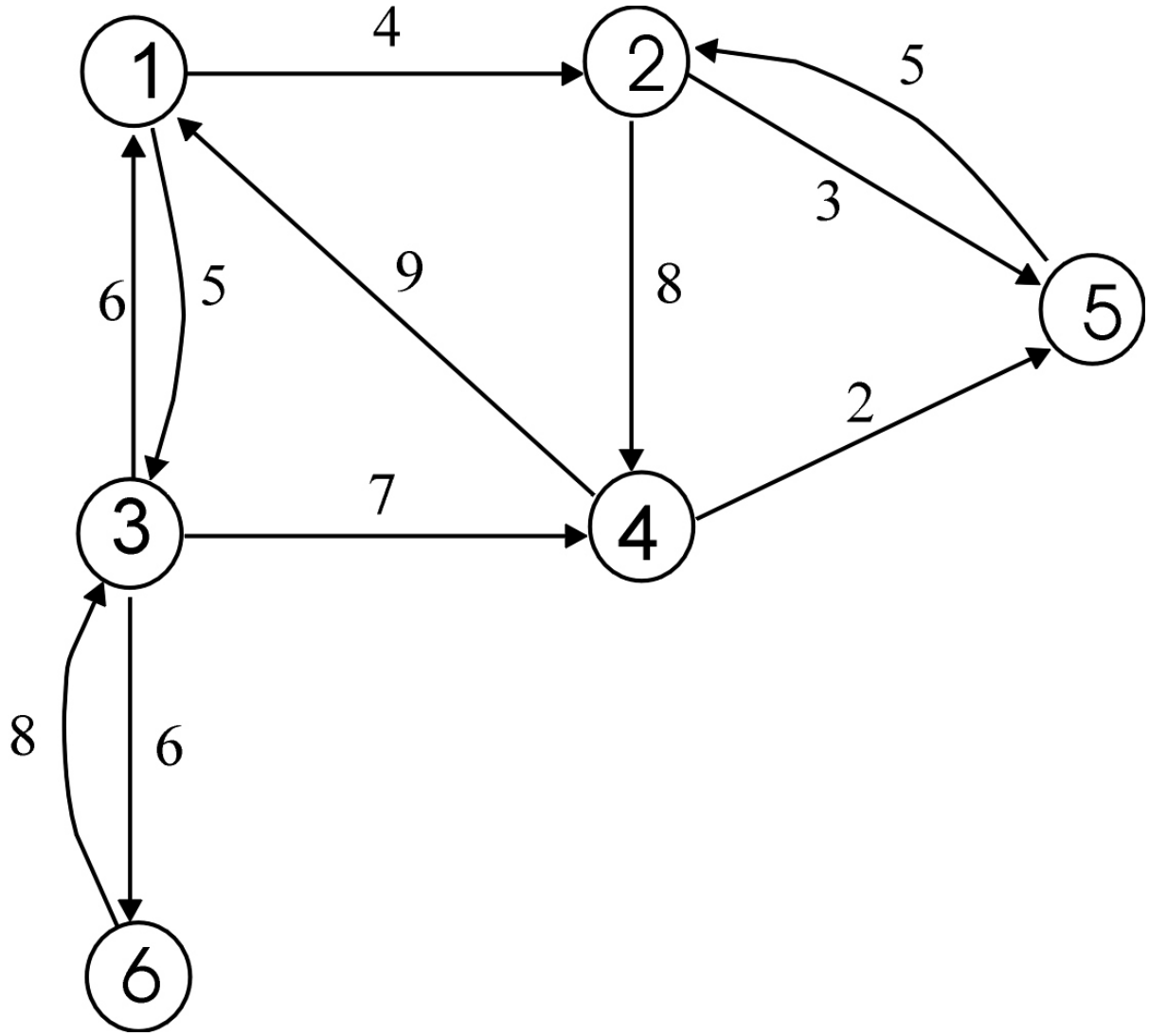
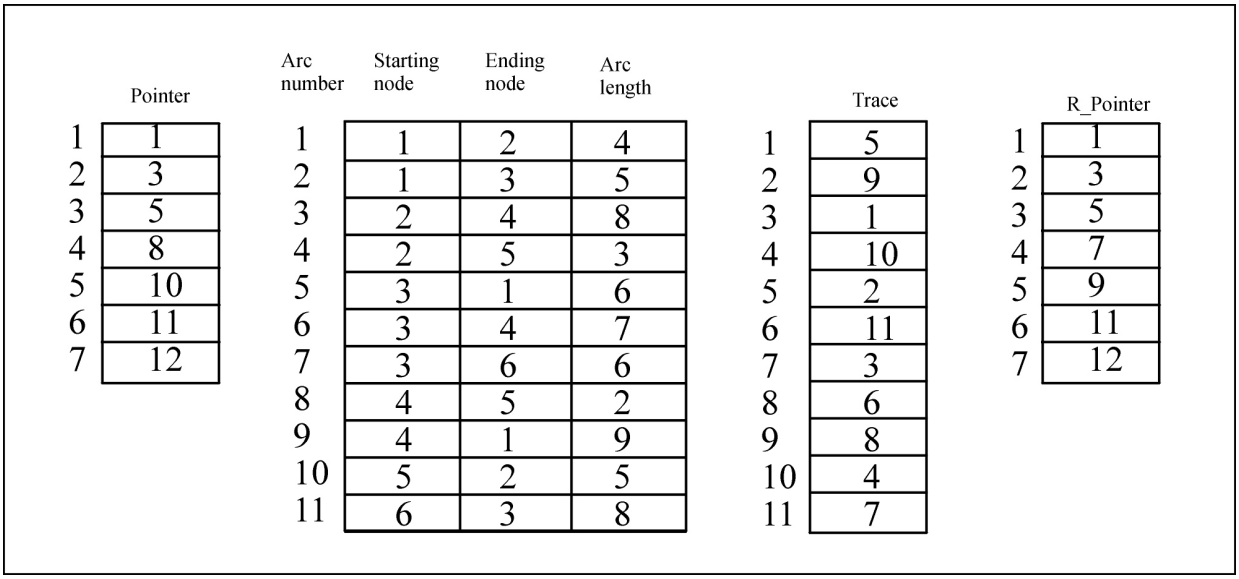


Figure 1. An Example Network.



	Pointer
1	1
2	3
3	5
4	8
5	10
6	11
7	12

	Arc number	Starting node	Ending node	Arc length
1	1	2	4
2	1	3	5
3	2	4	8
4	2	5	3
5	3	1	6
6	3	4	7
7	3	6	6
8	4	5	2
9	4	1	9
10	5	2	5
11	6	3	8

	Trace
1	5
2	9
3	1
4	10
5	2
6	11
7	3
8	6
9	8
10	4
11	7

	R_Pointer
1	1
2	3
3	5
4	7
5	9
6	11
7	12

Figure 2. Compact Forward and Reverse Star Representation of the Example Network.

Abbrevation	Implementation Description	Complexity	Additional References
BF	Bellman-Ford-Moore basic implementation	$O(nm)$	Bellman (1958)
BFP	Bellman-Ford-Moore with parent–checking	$O(nm)$	
DIKQ	Dijkstra's – naive implementation	$O(n^2)$	Dijkstra (1959)
DIKB	Dijkstra's using buckets structure – basic implementation	$O(m + nC)$	Dial (1969)
DIKBM	Dijkstra's using buckets structure – with overflow bag	$O(m + n(C/\alpha + \alpha))$	Cherkassky *et al.* (1993)
DIKBA	Dijkstra's using buckets structure – approximate buckets	$O(m\beta + n(\beta + C/\beta))$	
DIKBD	Dijkstra's using buckets structure – double buckets	$O(m + n(\beta + C/\beta))$	
DIKF	Dijkstra's using heap structure – Fibonacci heap	$O(m + n\log(n))$	Fredman and Tarjan (1987)
DIKH	Dijkstra's using heap structure – k–array heap	$O(m\log(n))$	Corman *et al.* (1990)
DIKR	Dijkstra's using heap structure – R–heap	$O(m + n\log(C))$	Ahuja *et al.* (1990)
PAPE	Incremental Graph – Pape–Levit implementation	$O(n2^n)$	Pape (1974)
TWO_Q	Incremental Graph – Pallottino implementation	$O(n^2m)$	Pallottino (1984)
THRESH	Threshold Algorithm	$O(nm)$	Glover *et al.* (1984, 1985)
GOR	Topological Ordering – basic implementation	$O(nm)$	Goldberg and Radzik (1993)
GOR1	Topological Ordering – with distance updates	$O(nm)$	

Notation:
n is the number of network nodes. m is the number of network arcs.
C is the maximum arc length in a network. α and β are input parameters.

Figure 3. Summary of 15 Shortest Path Algorithms.