Appendix A: Shortest Path Algorithms

A.1 INTRODUCTION

In this appendix, the Bellman-Ford algorithm and the Floyd-Warshall algorithm shortest path algorithms are described. These algorithms can be used to solve several problems in VLSI signal processing such as computing the iteration bound of recursive DFGs (see Chapter 2), retiming (see Chapter 4), and retiming for folding (see Chapter 6).

The level of detail included here is intended to be enough so that the reader can use these algorithms to solve the problems mentioned above. The theory behind these algorithms has been omitted; however, for the interested reader, there is a reference that discusses the theory behind these algorithms (e.g., [1], [2]).

The concepts of shortest paths and negative cycles are demonstrated using the DFG in Figure A.1. The length of a path is the sum of the weights on the edges of the path. The path $2 \rightarrow 3 \rightarrow 4$ has length $1 + 2 = 3$. There are 2 paths from node 2 to node 4, namely the path $2 \rightarrow 4$ with length 1 and the path $2 \rightarrow 3 \rightarrow 4$ with length 3. The shortest path from node 2 to node 4 is $\min\{1, 3\} = 1$. A negative cycle in a directed graph is a directed cycle such that the sum of the lengths on the edges of the cycle is negative. Figure A.1

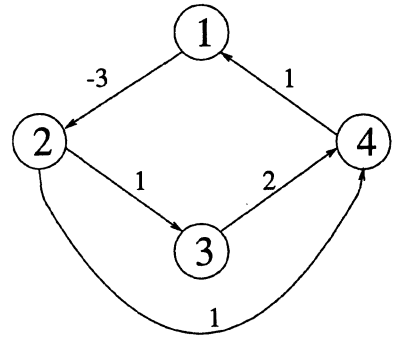

Fig. A.1 A graph containing 1 negative cycle.

contains the 2 cycles $2 \to 4 \to 1 \to 2$ and $2 \to 3 \to 4 \to 1 \to 2$. The cycle $2 \to 4 \to 1 \to 2$ is a negative cycle because the sum of the edge weights is $1 + 1 + (-3) = -1$. The other cycle, $2 \to 3 \to 4 \to 1 \to 2$, is not a negative cycle because the sum of the edge weights is $1 + 2 + 1 + (-3) = 1$. With this in mind, each of the 2 shortest path algorithms works as follows. If there are no negative cycles in the graph, the algorithm returns TRUE and provides some information about the lengths of the shortest path between the nodes (the amount of information varies between the two algorithms, as we will see in the following sections). If there exists at least 1 negative cycle in the graph, the algorithm returns FALSE. To summarize, each of the algorithms searches for a negative cycle and provides shortest path information if no negative cycle exists.

The following assumptions and notation are used. We assume that no parallel edges exist in the graph, i.e., for 2 nodes U and V, there is at most 1 edge from U to V ($U \xrightarrow{e} V$) and at most 1 edge from V to U ($V \xrightarrow{e} U$). Let

$$w(U \xrightarrow{e} V) = \begin{cases} \text{the length of the edge } U \xrightarrow{e} V, & \text{if } U \xrightarrow{e} V \text{ exists} \\ \infty, & \text{otherwise} \end{cases},$$

where the length of the edge is simply a number associated with the edge. Let n be the number of nodes in the graph, and assume that the n nodes are numbered $1, 2, \ldots, n$. The shortest path from the node U to the node V is denoted as S_{UV}.

A.2 THE BELLMAN-FORD ALGORITHM

The Bellman-Ford algorithm is a single-point shortest path algorithm. If no negative cycles exist in the graph, it finds the shortest path from an arbitrarily chosen node U (called the origin) to each node in the graph. For a graph with n nodes, the algorithm constructs $n-1$ vectors $\mathbf{r}^{(k)}$, $k = 1, 2, \ldots, n-1$, which

1. $r^{(1)}(U) = 0$
2. *For* $k = 1$ *to* n
3. *If* $k \neq U$
4. $r^{(1)}(k) = w(U \xrightarrow{e} k)$
5. *For* $k = 1$ *to* $n - 2$
6. *For* $V = 1$ *to* n
7. $r^{(k+1)}(V) = r^{(k)}(V)$
8. *For* $W = 1$ *to* n
9. *If* $r^{(k+1)}(V) > r^{(k)}(W) + w(W \xrightarrow{e} V)$
10. $r^{(k+1)}(V) = r^{(k)}(W) + w(W \xrightarrow{e} V)$
11. *For* $V = 1$ *to* n
12. *For* $W = 1$ *to* n
13. *If* $r^{(n-1)}(V) > r^{(n-1)}(W) + w(W \xrightarrow{e} V)$
14. *return FALSE and exit*
15. *return TRUE and exit*

Fig. A.2 The Bellman-Ford algorithm.

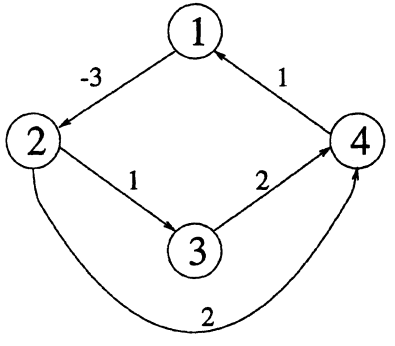

Fig. A.3 A graph containing no negative cycles.

are each of size $n \times 1$, as described in Figure A.2. Note that $\mathbf{r}^{(k)}(U)$ represents the U-th entry in the column vector $\mathbf{r}^{(k)}$.

If TRUE is returned, then $r^{(n-1)}(V)$ is S_{UV}, which is the shortest path from the node U to the node V. If FALSE is returned, then the graph contains at least one negative cycle.

Example A.2.1 *In this example, the Bellman-Ford algorithm in Figure A.2 is used to find the shortest path from the node* $U = 2$ *to the nodes in the graph in Figure A.3. The values of* $r^{(k)}(V)$ *for* $k = 1, 2, 3$ *and* $V = 1, 2, 3, 4$ *are shown in Table A.1. The Bellman-Ford algorithm returns TRUE in this example, so the value of* $r^{(3)}(V)$ *is the shortest path from node 2 to node V for* $V = 1, 2, 3, 4$. ■

Table A.1 Values of $r^{(k)}(V)$ for $k = 1, 2, 3$ and $V = 1, 2, 3, 4$ from Example A.2.1

$r^{(k)}(V)$	$k = 1$	$k = 2$	$k = 3$
$V = 1$	∞	3	3
$V = 2$	0	0	0
$V = 3$	1	1	1
$V = 4$	2	2	2

Table A.2 Values of $r^{(k)}(V)$ for $k = 1, 2, 3$ and $V = 1, 2, 3, 4$ from Example A.2.2

$r^{(k)}(V)$	$k = 1$	$k = 2$	$k = 3$
$V = 1$	∞	2	2
$V = 2$	0	0	-1
$V = 3$	1	1	1
$V = 4$	1	1	1

Example A.2.2 *In this example, the Bellman-Ford algorithm is used to find the shortest path from the origin $U = 2$ to the nodes in the graph in Figure A.1. The values of $r^{(k)}(V)$ for $k = 1, 2, 3$ and $V = 1, 2, 3, 4$ are shown in Table A.2. In this example, the Bellman-Ford algorithm returns FALSE because $r^{(3)}(3) > r^{(3)}(2) + w(2 \xrightarrow{e} 3)$, so a negative cycle is detected. This negative cycle can be seen in Figure A.1, where the cycle $2 \to 4 \to 1 \to 2$ has length $1 + 1 - 3 = -1$.* ■

A.3 THE FLOYD-WARSHALL ALGORITHM

The Floyd-Warshall algorithm is an all-points shortest path algorithm. If no negative cycles exist in the graph, the algorithm finds the shortest path between all possible pairs of nodes in the graph. The algorithm constructs $n + 1$ matrices $\mathbf{R}^{(k)}$, $k = 1, 2, \ldots, n + 1$, which are each of size $n \times n$, as described in Figure A.4.

If TRUE is returned, then $r^{(n+1)}(U, V)$ is S_{UV}, which is the shortest path from the node U to the node V. If FALSE is returned, then the graph contains a negative cycle.

Example A.3.1 *In this example, the Floyd-Warshall algorithm in Figure A.4 is used to solve the all-pairs shortest path problem for the graph in Figure A.3. The values of $r^{(k)}(U, V)$ for $U, V \in \{1, 2, 3, 4\}$ and $k = 1, 2, 3, 4, 5$ are given in Table A.3, where $r^{(k)}(U, V)$ is the element U, V in the matrix $\mathbf{R}^{(k)}$. The*

```
1.   For V = 1 to n
2.       For U = 1 to n
3.           r^(1)(U,V) = w(U -e-> V)
4.   For k = 1 to n
5.       For V = 1 to n
6.           For U = 1 to n
7.               r^(k+1)(U,V) = r^(k)(U,V)
8.               If r^(k+1)(U,V) > r^(k)(U,k) + r^(k)(k,V)
9.                 r^(k+1)(U,V) = r^(k)(U,k) + r^(k)(k,V)
10.  For k = 1 to n
11.      For U = 1 to n
12.          If r^(k)(U,U) < 0
13.            return FALSE and exit
14.  return TRUE and exit
```

Fig. A.4 The Floyd-Warshall algorithm.

Floyd-Warshall algorithm returns TRUE because all of the diagonal elements in the matrices $\mathbf{R}^{(k)}$, $k = 1, 2, 3, 4$, *are nonnegative, so* $r^{(5)}(U, V)$ *is the shortest path from the node* U *to the node* V. ■

Example A.3.2 *In this example, the Floyd-Warshall algorithm is used to solve the all-pairs shortest path problem for the graph in Figure A.1. The values of* $r^{(k)}(U, V)$ *for* $U, V \in \{1, 2, 3, 4\}$ *and* $k = 1, 2, 3, 4, 5$ *are given in Table A.4, where* $r^{(k)}(U, V)$ *is the element* U, V *in the matrix* $\mathbf{R}^{(k)}$. *The Floyd-Warshall algorithm returns FALSE because some of the diagonal elements in the matrices* $\mathbf{R}^{(k)}$, $k = 3, 4$ *and* 5, *are negative. Therefore, the graph contains a negative cycle.* ■

A.4 COMPUTATIONAL COMPLEXITIES

The Bellman-Ford algorithm detects negative cycles and solves the *single-source* shortest path problem if no negative cycles exist in $\mathcal{O}(n^3)$ time, while the Floyd-Warshall algorithm detects negative cycles and solves the *all-pairs* shortest path problem if no negative cycles exist in $\mathcal{O}(n^3)$ time. The Bellman-Ford algorithm requires $\mathcal{O}(n^4)$ time to solve the all-pairs shortest path problem since it needs to be run once for each of the n nodes. The Floyd-Warshall algorithm is preferred for applications that require all-pairs shortest path solution, and the Bellman-Ford algorithm and Floyd-Warshall algorithm can be used interchangeably for applications that require a single-source shortest path solution.

Modifications to the Bellman-Ford algorithm and early exit conditions for both algorithms can be utilized to improve computational efficiency. The

Table A.3 Values of $r^{(k)}(U, V)$ for Example A.3.1

$$\mathbf{R}^{(1)} = \begin{bmatrix} \infty & -3 & \infty & \infty \\ \infty & \infty & 1 & 2 \\ \infty & \infty & \infty & 2 \\ 1 & \infty & \infty & \infty \end{bmatrix} \quad \mathbf{R}^{(2)} = \begin{bmatrix} \infty & -3 & \infty & \infty \\ \infty & \infty & 1 & 2 \\ \infty & \infty & \infty & 2 \\ 1 & -2 & \infty & \infty \end{bmatrix} \quad \mathbf{R}^{(3)} = \begin{bmatrix} \infty & -3 & -2 & -1 \\ \infty & \infty & 1 & 2 \\ \infty & \infty & \infty & 2 \\ 1 & -2 & -1 & 0 \end{bmatrix}$$

$$\mathbf{R}^{(4)} = \begin{bmatrix} \infty & -3 & -2 & -1 \\ \infty & \infty & 1 & 2 \\ \infty & \infty & \infty & 2 \\ 1 & -2 & -1 & 0 \end{bmatrix} \quad \mathbf{R}^{(5)} = \begin{bmatrix} 0 & -3 & -2 & -1 \\ 3 & 0 & 1 & 2 \\ 3 & 0 & 1 & 2 \\ 1 & -2 & -1 & 0 \end{bmatrix}$$

Table A.4 Values of $r^{(k)}(U, V)$ for Example A.3.2

$$\mathbf{R}^{(1)} = \begin{bmatrix} \infty & -3 & \infty & \infty \\ \infty & \infty & 1 & 1 \\ \infty & \infty & \infty & 2 \\ 1 & \infty & \infty & \infty \end{bmatrix} \quad \mathbf{R}^{(2)} = \begin{bmatrix} \infty & -3 & \infty & \infty \\ \infty & \infty & 1 & 1 \\ \infty & \infty & \infty & 2 \\ 1 & -2 & \infty & \infty \end{bmatrix} \quad \mathbf{R}^{(3)} = \begin{bmatrix} \infty & -3 & -2 & -2 \\ \infty & \infty & 1 & 1 \\ \infty & \infty & \infty & 2 \\ 1 & -2 & -1 & -1 \end{bmatrix}$$

$$\mathbf{R}^{(4)} = \begin{bmatrix} \infty & -3 & -2 & -2 \\ \infty & \infty & 1 & 1 \\ \infty & \infty & \infty & 2 \\ 1 & -2 & -1 & -1 \end{bmatrix} \quad \mathbf{R}^{(5)} = \begin{bmatrix} -1 & -4 & -3 & -3 \\ 2 & -1 & 0 & 0 \\ 3 & 0 & 1 & 1 \\ 0 & -3 & -2 & -2 \end{bmatrix}$$

interested reader can find these details in [1].

REFERENCES

1. E. L. Lawler, *Combinatorial Optimization: Networks and Matriods.* Rinehart and Winston, 1976.

2. T. H. Cormen, C. E. Leiserson and R. C. Rivest, *Introduction to Algorithms*, MIT Press, 1990.