Floyd-Warshall's algorithm

A. Agnetis*

Given a network $G = (N, A)$, and a set of weights c_{ij} for each $(i, j) \in A$, the (s, t)-shortest path problem can be formulated as:

$$\min \sum_{(i,j)\in A} c_{ij} x_{ij} \tag{1}$$

$$\sum_{(s,j)\in\delta^+(s)} x_{sj} = 1 \tag{2}$$

$$\sum_{(i,t)\in\delta^-(t)} x_{it} = 1 \tag{3}$$

$$\sum_{(h,j)\in\delta^+(h)} x_{hj} - \sum_{(i,h)\in\delta^-(h)} x_{ih} = 0, \quad h \in N - \{s, t\} \tag{4}$$

$$x_{ij} \geq 0 \quad (i, j) \in A \tag{5}$$

where $x_{ij} = 1$ if (i, j) belongs to the path and $x_{ij} = 0$ otherwise. Note that this is equivalent to the problem of sending one unit of flow from s to t through an uncapacitated network, at minimum cost.

It is well known that formulation (1)–(5) solves the problem as long as there are no negative cycles (hence, in particular, if $c_{ij} \geq 0$ for all $(i, j) \in A$). In fact, if negative cycles exist, the concept of shortest path becomes controversial, since it would be convenient to pass through one such cycle an unlimited number of times. One can then formulate the problem as that of finding the shortest *simple* path (i.e., a path which does not pass twice through the same node) from s to t, but this is by no means an easy problem.

However, in general there *can* be negative cycles, and the problem arises of detecting their existence. Floyd-Warshall's algorithm is a simple, though effective algorithm that allows to:

- detect a negative cycle, if it exists
- compute the shortest path from i to j, for all node pairs i, j, if no negative cycles exist.

*Dipartimento di Ingegneria dell'Informazione e Scienze Matematiche - Università di Siena

This algorithm computes in parallel the shortest path for all node pairs, and stops either when all shortest paths are computed or a negative cycle is found. In what follows, C denotes the $n \times n$ adjacency matrix, where we let $c_{ii} = 0$ for $i = 1, \ldots, n$ and $c_{ij} = +\infty$ if $(i, j) \notin A$. Also, recall that, given a path from i to j, we call *internal nodes* of the path all the nodes of the path except i and j. For the sake of clarity, we first illustrate the algorithm supposing that no negative cycle exists.

The central concept in Floyd-Warshall's algorithm is the following. Suppose that the nodes are arbitrarily numbered as 1, 2, ..., n. We let $\pi_{ij}^{(k)}$ denote the shortest path from i to j, *such that its internal nodes belong to the subset* $\{1, 2 \ldots, k\}$.

The algorithm has n iterations. At each iteration k, an $n \times n$ matrix $C^{(k)}$ is computed in which, for each pair i, j, entry $c_{ij}^{(k)}$ is the length of $\pi_{ij}^{(k)}$. At the beginning of the algorithm, we initialize $C^{(0)}$ as C. We next show that matrix $C^{(k+1)}$ can be easily computed from $C^{(k)}$.

Let us consider two nodes i and j, and suppose we have computed matrix $C^{(k)}$. We want to determine the value of $c_{ij}^{(k+1)}$, i.e., the length of the shortest path $\pi_{ij}^{(k+1)}$ from i to j such that its internal nodes belong to $\{1, \ldots, k+1\}$. Actually, two cases may hold.

- Path $\pi_{ij}^{(k+1)}$ does not pass through node $k+1$. In this case, having extended the set of candidate internal nodes to $k+1$ does not produce any benefit. Hence, $\pi_{ij}^{(k+1)} \equiv \pi_{ij}^{(k)}$ and therefore
$$c_{ij}^{(k+1)} = c_{ij}^{(k)}.$$
- Path $\pi_{ij}^{(k+1)}$ *does* pass through node $k+1$. Now consider the two subpaths of $\pi_{ij}^{(k+1)}$, from i to $k+1$ (say, π') and from $k+1$ to j (say, π''). The internal nodes of both π' and π'' are all contained in the subset $\{1, 2, \ldots, k\}$. Therefore, π' and π'' are indeed the shortest paths from i to $k+1$ and, respectively, from $k+1$ to j, such that their internal nodes belong to $\{1, 2, \ldots, k\}$, i.e., $\pi' \equiv \pi_{i,k+1}^{(k)}$ and $\pi'' \equiv \pi_{k+1,j}^{(k)}$. Therefore, in this case
$$c_{ij}^{(k+1)} = c_{i,k+1}^{(k)} + c_{k+1,j}^{(k)}.$$

In conclusion, the above considerations allow one to compute the generic element of $C^{(k+1)}$ as

$$c_{ij}^{(k+1)} = \min\{c_{ij}^{(k)}; c_{i,k+1}^{(k)} + c_{k+1,j}^{(k)}\}. \tag{6}$$

Hence, the algorithm proceeds by subsequently computing $C^{(1)}$,$C^{(2)}$,...,$C^{(n)}$. Clearly, the entries of $C^{(n)}$ are the lengths of the shortest paths between any two pairs of nodes.

A few comments are in order.

- *Complexity.* The computational complexity of Floyd-Warshall's algorithm can be easily computed. In fact, for each value $c_{ij}^{(k)}$ can be computed in constant time, being the minimum between two quantities. Since i, j and k all span from 1 to n, the overall complexity is $O(n^3)$.

- *Path reconstruction.* Besides computing the values of the shortest paths, one must also be able to reconstruct such paths, for each node pair i, j. Floyd and Warshall proposed a very compact and clever way of storing all the information to reconstruct the path. In order to be able to backtrack each path, all we need to store, for each shortest path from i to j, is the last internal node, i.e., the predecessor of node j. In fact, if the last internal node of the shortest path from i to j is, say, node h, we can continue backtracking the path finding the last internal node of the shortest path from i to h... and so on. To this aim, while computing the matrix $C^{(k)}$, we also construct the *predecessor matrix* $P^{(k)}$, in which the entry $p_{ij}^{(k)}$ is defined as the last internal node of the path $\pi_{ij}^{(k)}$, i.e., the predecessor of j. (We suppose that $p_{ij}^{(k)}$ is undefined if $c_{ij}^{(k)} = +\infty$.) Actually, $p_{ij}^{(k)}$ can be computed very easily. In fact, with reference to the two cases considered in (6), one has that if $\pi_{ij}^{(k+1)} \equiv \pi_{ij}^{(k)}$, then of course $p_{ij}^{(k+1)} = p_{ij}^{(k)}$, while in the other case, being $\pi_{ij}^{(k+1)}$ the concatenation of $\pi_{i,k+1}^{(k)}$ and $\pi_{k+1,j}^{(k)}$, one has $p_{ij}^{(k+1)} = p_{k+1,j}^{(k)}$. The entries of $P^{(n)}$ then allow one to reconstruct all shortest paths.

  Notice that $P^{(n)}$ gives a compact representation of all shortest paths. In fact, since there are $O(n^2)$ shortest paths, listing them explicitly would require $O(n^3)$ memory space. Instead, $P^{(n)}$ contains all relevant information in $O(n^2)$.

- *Negative cycles.* From the viewpoint of computational complexity, one may observe that all shortest paths can also be computed applying n times Dijkstra's algorithm, every time from a different source node. Since the complexity of Dijkstra's algorithm is $O(n^2)$, also in this way one would get an overall complexity of $O(n^3)$. However, unlike Dijkstra's algorithm, Floyd-Warshall's allows dealing with negative cycles too. Actually, we only need apply (6) even when $i = j$. In fact, suppose that a negative cycle exists, and let i the highest-numbered node belonging to the cycle. Then, by construction, one has that $c_{ii}^{(i-1)} < 0$, indicating that there is a way to go from i to i passing only through lower-numbered nodes, such that the total length of the arcs is negative. When this occurs, of course the algorithm stops.

As an example, consider a graph having the following adjacency matrix:

$$C = \begin{pmatrix} 0 & 5 & 1 & +\infty \\ +\infty & 0 & +\infty & 3 \\ 10 & 2 & 0 & 6 \\ 4 & +\infty & +\infty & 0 \end{pmatrix}$$

applying the algorithm, one gets the following matrices, in which at each step the entries corresponding to a new path (i.e., whenever $c_{ij}^{(k+1)} = c_{i,k+1}^{(k)} + c_{k+1,j}^{(k)}$) are circled.

$$C^{(0)} = \begin{pmatrix} 0 & 5 & 1 & +\infty \\ +\infty & 0 & +\infty & 3 \\ 10 & 2 & 0 & 6 \\ 4 & +\infty & +\infty & 0 \end{pmatrix} \qquad P^{(0)} = \begin{pmatrix} 1 & 1 & 1 & - \\ - & 2 & - & 2 \\ 3 & 3 & 3 & 3 \\ 4 & - & - & 4 \end{pmatrix}$$

$$C^{(1)} = \begin{pmatrix} 0 & 5 & 1 & +\infty \\ +\infty & 0 & +\infty & 3 \\ 10 & 2 & 0 & 6 \\ 4 & \textcircled{9} & \textcircled{5} & 0 \end{pmatrix} \qquad P^{(1)} = \begin{pmatrix} 1 & 1 & 1 & - \\ - & 2 & - & 2 \\ 3 & 3 & 3 & 3 \\ 4 & \textcircled{1} & \textcircled{1} & 4 \end{pmatrix}$$

$$C^{(2)} = \begin{pmatrix} 0 & 5 & 1 & \textcircled{8} \\ +\infty & 0 & +\infty & 3 \\ 10 & 2 & 0 & \textcircled{5} \\ 4 & 9 & 5 & 0 \end{pmatrix} \qquad P^{(2)} = \begin{pmatrix} 1 & 1 & 1 & \textcircled{2} \\ - & 2 & - & 2 \\ 3 & 3 & 3 & \textcircled{2} \\ 4 & 1 & 1 & 4 \end{pmatrix}$$

$$C^{(3)} = \begin{pmatrix} 0 & \textcircled{3} & 1 & \textcircled{6} \\ +\infty & 0 & +\infty & 3 \\ 10 & 2 & 0 & 5 \\ 4 & \textcircled{7} & 5 & 0 \end{pmatrix} \qquad P^{(3)} = \begin{pmatrix} 1 & \textcircled{3} & 1 & \textcircled{2} \\ - & 2 & - & 2 \\ 3 & 3 & 3 & 2 \\ 4 & \textcircled{3} & 1 & 4 \end{pmatrix}$$

$$C^{(4)} = \begin{pmatrix} 0 & 3 & 1 & 6 \\ \textcircled{7} & 0 & \textcircled{8} & 3 \\ \textcircled{9} & 2 & 0 & 5 \\ 4 & 7 & 5 & 0 \end{pmatrix} \qquad P^{(4)} = \begin{pmatrix} 1 & 3 & 1 & 2 \\ \textcircled{4} & 2 & \textcircled{1} & 2 \\ \textcircled{4} & 3 & 3 & 2 \\ 4 & 3 & 1 & 4 \end{pmatrix}.$$

These two final matrices contain the length of all shortest paths and all the information needed for their reconstruction.