A DIJKSTRA-LIKE SHORTEST PATH ALGORITHM FOR CERTAIN CASES OF NEGATIVE ARC LENGTHS

SUKHAMAY KUNDU

Abstract.
If each negative length arc of a digraph $G$ is acyclic, i.e., does not belong to any cycle, then we show that the shortest paths from a given node to all other nodes can be computed in $O(V^2)$ time, where $V$ is the number of nodes in $G$.

Keywords: shortest path, digraph, algorithm.

CR category: 5.25, 5.32.

1. Introduction.
There are basically two algorithms for computing the shortest length paths from a given node to all other nodes of a digraph $G$: Dijkstra's algorithm and Ford's (or Floyd's) algorithm [1, 4]. These algorithms have computation bounds $O(V^2)$, and $O(V^3)$, respectively, where $V$ is the number of nodes in $G$. Although Dijkstra's algorithm is more efficient than Ford's algorithm, the former is applicable only if the arc lengths are non-negative. We show that a Dijkstra-like algorithm can be obtained if the negative length arcs are acyclic, i.e., do not belong to any cycle. The computation bound of the new algorithm is $O(V^2)$, which is an improvement from Ford's $O(V^3)$ algorithm. For dense digraphs, where $E \geq c \cdot V^{1+\epsilon}$ for some $c, \epsilon > 0$, an implementation technique developed by Johnson [2, 3] can be used to further improve the computation bound to $O(E)$, where $E$ is the number of arcs. An extensive bibliography on the shortest path algorithms is given in [2].

2. Results.
We assume that $G$ has a distinguished node 1 from which we wish to find the shortest path-lengths to the other nodes $i, 2 \leq i \leq V$. The lengths of a path is defined to be the sum of its arc lengths. The distance $d(x, y)$ of node $y$ from node $x$ is the shortest length of a path from $x$ to $y$. We assume that all nodes of $G$ can be reached from the distinguished node. The following theorem is a basis for Dijkstra's algorithm.

Received July 7, 1980. Revised September 10, 1980.
THEOREM 1. Let $G$ be a digraph with non-negative arc lengths. There exists a labeling $x(1), x(2), \ldots, x(V)$ of the nodes of $G$ in non-decreasing order of their distances from node 1 such that there exists a shortest path to each $x(i)$ which uses only the nodes in \{x(1), x(2), \ldots, x(i-1)\} as intermediate nodes.

The labeling \{x(i)\} and the shortest path lengths are determined, one by one, in the order $x(1), x(2), \ldots, x(V)$ by repeated application of the steps (2.1) and (2.2) below. Initially, the "current" distance $f(y)$ to a node $y$ is set to $\infty$ for each $y \neq 1$, $f(1)$ to zero and $i$ equals 1. After each iteration, $i$ is increased by one and the current distance to a node gives the shortest path length to it using only the labeled nodes as intermediate nodes. The non-negativity of the arc lengths $c(x,y)$ justifies equation (2.1), and the smallest $f(y)$ equals the distance $d(1,y)$.

(2.1) $x(i)$ is the label of a node which has smallest $f(y)$ among the unlabeled nodes,

(2.2) $f(y) = \min \left[ f(y), f(x(i)) + c(x(i), y) \right]$.

We make an important observation here. Suppose the non-negative arc length condition is removed for the arcs originating at the starting node 1, other arc lengths are still being non-negative. Then the steps (2.1) and (2.2) remain valid for computing the shortest path lengths. The labeling \{x(i)\} now corresponds to the non-decreasing distances, as before, except perhaps for the node 1.

Now assume that the negative length arcs in the digraph $G$ satisfy the condition (2.3) below. The condition is clearly more restrictive than the assumption in Ford's algorithm, namely, that $G$ has no negative length cycle.

(2.3) No directed cycle of $G$ contains a negative length arc.

Consider the decomposition of $G$ into its strong components $C_1, C_2, \ldots, C_k$; by (2.3), each $C_i$ consists entirely of non-negative arcs. Let $C_1 < C_2 < \ldots < C_k$ be a topological ordering of the components so that all negative arcs go from a node in $C_i$ to a node in $C_j$ for some $j > i$. The original shortest path problem for $G$ can now be solved by solving one shortest path problem for each component, in the order $C_1, C_2, \ldots$. More precisely, corresponding to the component $C_i, i \geq 2$, define the digraph $C'_i$ by adding a new source node $z_i$ and the arcs $(z_i, w)$ for each arc $(x, w)$ in $G$, where $x \notin C_i$ and $w \in C_i$. The length of $(z_i, w)$ equals the shortest path length from node 1 to node $x$ plus the length of the arc $(x, w)$ in case of multiple arcs to a node $w$, remove all but the smallest length arc to that node. The arcs of $C'_i$ are non-negative, except perhaps those originating from the node $z_i$. It is easy to see that the length of a shortest path to a node $w \in C_i$ is the same as the distance $d(z_i, w)$ computed for the digraph $C'_i$. To incorporate the stepwise shortest path computations on $C_1, C_2, \ldots$ without explicitly constructing these digraphs and without increasing the computation time bound, one simply needs to restrict the node set in (2.1) suitably in taking the minimum. The minima are taken initially