IMPLEMENTATION AND EFFICIENCY OF MOORE-ALGORITHMS
FOR THE SHORTEST ROUTE PROBLEM

U. PÄPE

Technische Universität Berlin, W. Berlin, Germany

Received 7 February 1972
Revised manuscript received 29 June 1974

In the last 15 years, a good deal of effort has been devoted to the study of the shortest route problem. More than 200 publications are known but little has been reported concerning relative efficiencies. For a long time the Dijkstra method was considered the most efficient one. Programming work, using different data structures and implementation techniques for several algorithms, has shown that a variant of Moore's method seems to be most efficient for different types of graph structures.

The main objective of this paper is to show the strong relationship between an algorithm and its implementation.

1. Problem

Consider a network with node set $X$, $|X| = n$, and directed arcs $(i, j)$ having distances $d_{ij}$. Let $J = \{j_1, \ldots, j_t\} \subset X$, and $K = X \setminus J$, where $t < n$.

The problem is to determine the lengths $mj_k$ of the shortest routes to each $k \in K$ from that $j \in J$ ($g \in \{1, \ldots, t\}$), which is nearest to $k$.

The case $t = 1$ is the well-known problem for determining the lengths of the shortest routes from one node to all other nodes. The case of several roots can be solved by adding to the network one node $j_{n+1}$ connected to all roots by arcs of length zero. The multiple root problem may then be regarded as a single root problem; however, this trick is inefficient when parameters in the network are subsequently altered and the new problem must be solved [3].

2. The algorithms of Moore and d'Esopo

The algorithm of Moore [2]

The algorithm is an iterative process with the following general
step $\nu$:

$$S_\nu := \Gamma(S_{\nu-1}) ;$$

$$m_{j_k}^{(\nu)} := \min \{m_{j_k}^{(\nu-1)}, m_{j_i}^{(\nu-1)} + d_{ik} : i \in S_{\nu-1} \}, \quad k \in S_\nu ;$$

$$S_\nu := \{k : k \in S_\nu, m_{j_k}^{(\nu)} \neq m_{j_k}^{(\nu-1)} \} ;$$

$S_{\nu-1}$ denotes the set of current nodes, and $S_\nu$ the set of successors of nodes in $S_{\nu-1}$.

The initial condition of this algorithm is:

$$S_0 := \{j_1, ..., j_t \} ;$$

$$m_{j_k}^{(0)} := \infty, \quad k \neq j_g, \quad g = 1, ..., t ;$$

$$m_{j_g}^{(0)} := 0, \quad g = 1, ..., t .$$

The algorithm is applied for $\nu = 1, 2, ...$ and terminates if $S_\nu = \emptyset$.

The algorithm of d'Esopo [2]

The general step $\nu$ is the following

$$S_\nu := \Gamma(S_{\nu-1}) ;$$

$$m_{j_k}^{(\nu)} := \min \{m_{j_k}^{(\nu-1)}, m_{j_i}^{(\nu-1)} + d_{ik} : i \in S_{\nu-1} \}, \quad k \in S_\nu ;$$

$$R_\nu := R_{\nu-1} \cup S_{\nu-1} ;$$

$$S_\nu := \{k : k \in S_\nu, m_{j_k}^{(\nu)} \neq m_{j_k}^{(\nu-1)} \} ;$$

if $S_\nu \cap H \neq \emptyset$  then $(S_\nu := S_\nu \cup H ; H := \emptyset)$

else

if $S_\nu \cap R_\nu \neq \emptyset$  then $(H := H \cup (S_\nu \setminus (S_\nu \cap R_\nu));\quad S_\nu := S_\nu \cap R_\nu)$

else

if $S_\nu = \emptyset \land H \neq \emptyset$ then $(S_\nu := H ; H := \emptyset)$

The initial condition is