JOURNAL OF RESEARCH of the National Bureau of Standards
Vol. 86, No. 3, May-June 1981

Properties of Labeling Methods for Determining Shortest Path Trees*

Douglas R. Shier† and Christoph Witzgall†

National Bureau of Standards, Washington, DC 20234

January 21, 1981

A number of labeling procedures for determining shortest paths in a network employ a sequence list in order to carry out the required steps systematically. This paper studies certain formal properties of such sequence lists. It is shown that the desirable property of branching out from nodes whose labels represent actual in-tree distances is assured for certain ways of managing the sequence list, but not for others. The relationship of this property to the computational complexity of various labeling procedures is also investigated.

Key Words: Complexity; labeling; network; sequence list; shortest path; tree

1. Introduction

A number of methods for finding shortest paths in networks have been proposed during the past 30 years. Stimulated to a great extent by the wealth of application areas in which shortest path calculations arise (notably in transportation planning models), considerable effort has been directed toward the efficient implementation of such methods for large-scale problems. Recent evidence [2, 11][1] indicates that a method proposed by Pape [11] is remarkably successful in practice. However, compelling reasons for the observed efficiency of Pape's method are lacking. One motivation for the present work is to find some formal justification for the success of Pape's method.

To begin, some necessary terminology and notation will be introduced. Consider a *directed network* (N, A) with *node set* N and *arc set* A, and let

$$I(a) \varepsilon N, J(a) \varepsilon N$$

denote the *origin* and *destination* of arc $a \varepsilon A$. Given a *length* $\ell(a)$ for each arc $a \varepsilon A$, the length $\ell(P)$ of any path P is defined to be the sum of its constituent arc lengths. A frequently encountered problem is that of finding, among all paths (if any) extending from $i \varepsilon N$ to $j \varepsilon N$, a *shortest path:* i.e., a path from i to j having minimum length. It is assumed that the network contains no closed paths with negative length, in order to guarantee that such shortest paths always exist.

If r is a given node of N, then shortest paths from r to all nodes j accessible (by a path) from r can be selected to form a *shortest path tree* with *root* r. That is, the unique path in this tree from node r to node j is in fact a shortest path between the nodes.

A tree T, rooted at node r, can be uniquely specified by a *predecessor map* q that assigns to each node $j \neq r$ in T its *predecessor arc* $q(j)$ in the tree. Similarly, each node $j \neq r$ in the tree has a unique *predecessor node* $p(j) = I(q(j))$ in the tree. The nodes $p(j), p(p(j)), \ldots, r$ constitute the *ancestors* of node j in the tree. The *branch* at node i of tree T, or $B(i, T)$, is the largest subtree of T rooted at i. Thus, $B(i, T)$ contains node i and all nodes j which have i as an ancestor.

*AMS Subject Classification: 05C35, 05C05

†Center for Applied Mathematics, National Engineering Laboratory

[1] Numbers in brackets indicate literature references at the end of this paper.

A number of methods for determining a shortest path tree with root r are based on providing a *labeling* (T,d), where T is a tree rooted at r in the network and $d\colon N \to R \cup \{\infty\}$ assigns a *label* $d(j)$ to each node j such that

$$d(j)=\infty \text{ for nodes } j \text{ not in } T. \tag{1}$$
$$d(i) + \ell(a) \leqslant d(j) \text{ for all arcs a } = (i,j) \text{ in } T. \tag{2}$$

If i is an ancestor of j in T, then $P(i,j)$ denotes the unique simple path in T from node i to node j. If (T,d) is a labeling, then repeated application of property (2) produces

$$d(i) + \ell(P(i,j)) \leqslant d(j) \tag{3}$$

for all nodes i,j in T with i an ancestor of j. In particular, if node j is in T then r is an ancestor of j, whence

$$d(r) + \ell(P(r,j)) \leqslant d(j). \tag{4}$$

Accordingly, $d(j) - d(r)$ is an upper bound on the length of the tree path from r to j. As a consequence, $d(j) - d(r)$ is also an upper bound on the length of a shortest path in the network from r to j.

A labeling (T,d) is said to be *optimal* if

$$d(r)=0, \tag{5}$$
$$d(I(a)) + \ell(a) \geqslant d(J(a)) \text{ for all } a \in A. \tag{6}$$

A suitable interpretation of (6) is assumed in the case of infinite labels. If a labeling (T,d) is optimal, then T is a shortest path tree rooted at node r [5,14]. On the other hand, if T is a shortest path tree rooted at r, then the distances $d(j)$ from the root to node j define an optimal labeling (T,d).

If (T,d) is any labeling, and (6) is violated for some arc $a \in$A, then redefining

$$d(J(a)) = d(I(a)) + \ell(a)$$

while leaving all other labels unchanged,

$$d(j) = d(j) \text{ for } j \in N - \{J(a)\}$$

and modifying the tree T in an obvious fashion to contain the arc a, will produce a new labeling $(\hat{T},\hat{d})$. This observation forms the basis of the so-called *labeling methods* for finding shortest path trees. These methods employ successive *label corrections* of the above kind to construct an optimal labeling. The various labeling methods differ in their strategies for selecting arcs which are to be examined for possible label corrections.

A large class of labeling methods *branch out* from nodes, that is, they examine successively and in fixed order all arcs in the *forward star*

$$F(k) = \{a \in A\colon I(a) = K\}$$

of the node k from which to branch out. Candidate nodes for branching out are kept and prioritized on a *sequence list*. Employing a predecessor map q for characterizing trees, these methods follow the pattern:

```
Step 1: Put $d(r) = 0$ and $d(j) = \infty$ for $j \in N - \{r\}$.
Step 2: Enter node $r$ into the top position of the sequence list.
Step 3: Remove the top node $k$ from the sequence list.
Step 4: For each arc a in the forward star $F(k)$: if a violates (6), then put
            $d(J(a)) = d(I(a)) + \ell(a), \quad q(J(a)) = a,$
        and enter node $J(a)$ into the sequence list.
Step 5: If the sequence list is empty, then STOP; else return to Step 3.
```

Such *sequence-list driven labeling methods* are the subject of this paper. These methods differ from one another essentially in the way the nodes $J(a)$, whose labels have been corrected, are entered into the sequence list. Commonly used *sequence disciplines* that prescribe how nodes are introduced on the list include FIFO (nodes enter at the bottom), and LIFO (nodes enter at the top). There are various disciplines that are 2-WAY (nodes enter at either the bottom or top) such as the sequence discipline proposed by Pape [11]. Finally, there is the well-known discipline of DIJKSTRA (keep the sequence list sorted by labels increasing toward the bottom). We are interested in certain formal properties of such sequence disciplines.

For instance, the label $d(j)$ of some node j is called *sharp*, with respect to (T,d), if equality holds in (4). Since $d(r) = 0$ always holds for a labeling algorithm, a sharp label $d(j)$ represents the actual path length from r to j in T, and not just an upper bound. It is undesirable to branch out from a node j whose label is not sharp, since this condition guarantees that all labels directly and indirectly corrected from node j will have to be corrected again. It will be shown that LIFO and certain 2-WAY sequence disciplines branch out only from nodes having sharp labels, whereas this does not necessarily hold for a FIFO discipline. This property turns out to be closely connected to the question of how the order of nodes on the sequence list relates to the natural order of nodes in the associated tree.

2. Active Nodes

We call nodes appearing on the sequence list *active*. Since labeling methods based on sequence lists terminate when there are no active nodes remaining, the following fact is necessary for the proper functioning of such methods.

LEMMA 1: *For any sequence-list driven labeling method, the active nodes include the origins of all arcs which violate the optimality condition (6).*

PROOF: The lemma holds initially, when only the root r is active. Assume it holds at some intermediate stage. Branching out from some active node k will assure that all arcs in the forward star $F(k)$ satisfy the optimality condition. Therefore, removing node k from the sequence list will not cause the lemma to be violated. Also, the only arcs that previously satisfied (6) but do no longer must originate at those nodes whose label has been reduced by branching out from the node k. However, these nodes have just been entered in the sequence list.

LEMMA 2: *Any node* j *in* T *having a non-sharp label* d(j) *with respect to* (T,d) *must have an active ancestor in* T.

PROOF: Consider the path $P(r,j)$ in T from r to j. If all arcs in $P(r,j)$ satisfied the optimality condition (6), then

$$d(r) + \ell(P(r,j)) \geqslant d(j).$$

However, using property (4) of the labeling (T,d) yields

$$d(r) + \ell(P(r,j)) = d(j),$$

whence label $d(j)$ would in fact be sharp. Thus, at least one arc of $P(r,j)$ must violate (6), and its origin must be active by Lemma 1.

We can now derive a general condition on sequence disciplines which assures that only nodes having sharp labels are branched out from. To this end, observe that a natural order relation exists among the nodes of the tree T associated with the labeling (T,d). We write

$$i < j\,(T)$$

if i is an ancestor of j in T. A sequence discipline is called *order-compatible* if the sequence list never contains ancestors of the top node k in the list. In other words, the top node k does not have an active ancestor.

Examples of such sequence disciplines will be examined later in the paper. The following general condition on sequence disciplines is a direct restatement of Lemma 2.

THEOREM 1: *Under an order-compatible sequence discipline, a labeling method only branches out from nodes having sharp labels.*

3. Sequence Disciplines

As demonstrated in [2,7], the particular choice of sequence discipline employed in labeling procedures can profoundly affect the efficiency of the resulting shortest path algorithms. In this section, then, we will discuss several commonly-used sequence disciplines for determining shortest paths.

Recall that in sequence-list driven labeling methods a node is removed from the *top* of the sequence list (if the list is nonempty) and its forward star is then scanned. Any node i whose label is changed is entered, in some fashion, on the sequence list. For example, entering node i may always be placed at the *top* of the list, at the *bottom* of the list, or at *either the top or the bottom* of the list, depending on certain other information associated with node i.

In a LIFO (Last-In-First-Out) sequence discipline, any node i not appearing already on the sequence list is inserted at the top of the list. In case node i already appears on the list, either (1) the node is moved from its present list position to the top of the list, or (2) the node remains in its current position. We refer to these two variants as (1) LIFO/MOVE and (2) LIFO/NO MOVE, respectively. Shortest path algorithms based on LIFO/NO MOVE [7] make use of a "flag" to signify whether or not a node is currently on the sequence list. A reasonable implementation of the LIFO/MOVE version appears to require in addition the use of a doubly-linked list.

In a FIFO (First-In-First-Out) sequence discipline, any node i not appearing already on the list is inserted at the bottom of the list. In case node i already appears on the list, either (1) the node is moved to the bottom of the list, or (2) the node remains in its current position. Thus, in the latter case, nodes are branched out from in the order in which they are placed on the sequence list. These variants are referred to as FIFO/MOVE and FIFO/NO MOVE, respectively. The computational behavior of the second of these two variants has been studied in [2,7,8].

In the 2-WAY sequence discipline described by Pape [11], nodes i that have their label $d(i)$ corrected for the first time are placed at the bottom of the list. Nodes i that have previously been on the list (but are not currently) are placed at the top of the list when $d(i)$ is corrected. If node i already appears on the list, either (1) the node is moved from its present list position to the top of the list, or (2) the node remains in its current position. Again, these variants are referred to as PAPE/MOVE and PAPE/NO MOVE, respectively. Pape's description of this algorithm [11] leaves open which variant he has in mind. Dial, Glover, Karney, and Klingman [2] have implemented the second variant.

(In the sequence disciplines described above, the sequence list is considered to be linearly ordered, with the top node being the "first" node with respect to this linear order. If no new nodes are added to the top of the list, then the "second" node in the order thus becomes the new top node. It is easy to think of disciplines in which the associated list does not maintain a linear order structure; in these cases, the succession problem is regulated in some other manner.)

It should be noted that both the LIFO and PAPE sequence disciplines are special cases of another conceptually useful sequence discipline. Indeed, suppose f is a *tree function* defined with respect to the tree T of labeling(T,d). Namely, $f\colon N \to R \cup \{\infty\}$ is such that

$$i < j(T) \text{ implies that } f(i) \leqslant f(j). \tag{7}$$

If strict inequality holds above then f is called a *strict tree* function. Examples of tree functions abound. For example, if $f(i)$ denotes the number of nodes in the path $P(r,i)$ from r to i in T, then f is a (strict) tree function. Or, if $g(i)$ denotes the number of nodes in $B(i,T)$, then $-g$ is a (strict) tree function. If the network arc lengths are all nonnegative and (T,d) is a labeling, then the labels $d(i)$ define a tree function, in view of property (3). Tree functions find application in the efficient tracing of cycles in networks [1,12,13].

Consider now the *tree-derived sequence discipline*, based on a tree function f, that adds node i to the top of the list if

$$f(i) \leqslant \max \{f(u)\colon u \text{ is active}\} \tag{8}$$

and to the bottom of the list otherwise. Here the tree function f is defined with respect to the "old" tree T prior to update by the branching out that has just corrected the label on node i. Also, we suppose that the active nodes u in (8) are those which are active in the "old" sequence list. Of course, this tree-derived sequence discipline also has two variants (MOVE/NO MOVE) depending on whether a node i already on the list is moved in the prescribed manner or remains in its current position.

Using the tree function

$$f(i) = 0 \text{ for all } i \in N$$

clearly produces the LIFO discipline. Using the tree function

$$f(i) = \begin{cases} 0 \text{ if } i \in T \\ \infty \text{ if } i \in T \end{cases}$$

produces the PAPE discipline. It will become clear later that FIFO cannot be derived from a tree function. The more general notion of a tree-derived sequence discipline has been introduced because it will be shown in the next section that every such discipline possesses the desirable property of being order-compatible.

Strict tree functions can be used to define sequence disciplines in which the top element is an active node for which the strict tree function assumes its minimum value. Such sequence disciplines are trivially order-compatible, since any active ancestor j of the current top node k would have $f(j) < f(k)$; this contradicts the fact that k was chosen to have minimum value of $f(i)$ over active nodes i. Thus, the corresponding labeling methods will automatically branch out from sharp labels, by Theorem 1.

By using the labels $d(i)$ as a tree function and selecting the top node as an active node of minimum label, one obtains the well-known "label-setting" method of Dijkstra [3], for networks with positive arc lengths. The requirement of positive arc lengths ensures that d is a strict tree function (yielding order-compatibility) and that once a node is removed from the sequence list, it will never reappear on the list (whence the label can be permanently set). Dijkstra's method also works, perhaps with minor modification, in the presence of negative arc lengths [4,10], even though the above two properties are not assured. Somewhat surprisingly, the DIJKSTRA sequence discipline is order-compatible even in the presence of negative arc lengths, as will be shown in the next section.

4. Branching Out From Sharp Labels

In this section, we present some major results that indicate which of the sequence disciplines discussed in section 3 do in fact guarantee that only nodes with sharp labels are used for branching out. This desirable property will be assured for order-compatible sequence disciplines, by Theorem 1. The first major result of this section shows that tree-derived sequence disciplines always possess this property.

THEOREM 2: *Every tree-derived sequence discipline is order-compatible.*

PROOF: Suppose the sequence discipline is not order-compatible. Then there is a first time that order-compatibility is violated. Let k be the corresponding top node of the list Λ at that time and let j be an active node ($j \in \Lambda$) such that $j < k$ (T) in the associated tree T. Since j is currently active, but has not always been active, there exists a progression of lists $\Lambda_1, \Lambda_2, \ldots, \Lambda_s$ (with associated trees $T_1, T_2, \ldots, T_s$) such that $\Lambda_s = \Lambda$, $j \in \Lambda_1$, and $j \in \Lambda_m$ for $1 < m \leqslant s$. Also, $T_s = T$.

Since node j has not been branched out from while on lists $\Lambda_1, \ldots, \Lambda_s$, it follows that

$$B(j,T_1) \supseteq B(j,T_2) \supseteq \ldots \supseteq B(j,T_s). \tag{9}$$

Indeed, the only way the branch $B(j,T_m)$ can gain new arcs in $B(j,T_{m+1})$, $1 \leqslant m < s$, is if some node $i \varepsilon B(j,T_m)$ has been branched out from. The existence of such a node $i \varepsilon \Lambda_m$ with $j < i\,(T_m)$ and $m < s$ contradicts the fact that (T_s, Λ_s) was the first instance when order-compatibility was violated.

In view of (9), the assumed relation $j < k\,(T)$ implies

$$j < k\,(T_m), \text{ for } 1 \leqslant m \leqslant s. \tag{10}$$

Moreover, it is claimed that

$$d(k) \text{ remains the same in all trees } T_1, \ldots, T_s. \tag{11}$$

Suppose, to the contrary, that $d(k)$ was corrected in branching out from node $i \varepsilon \Lambda_v$. If $i \varepsilon B(j,T_v)$, then (10) would not hold in T_{v+1}. Thus, $i \varepsilon B(j,T_v)$ with $v < s$, but this contradicts the fact that the first violation of order-compatibility occurred for (T_s, Λ_s).

Consider now the manner in which node j was added to the sequence list Λ_1.

CASE I: Node j was added to the top of Λ_1. Thus, the only way node k can precede j on list Λ_s is for $d(k)$ to have been corrected in some T_m, $1 \leqslant m \leqslant s$, contradicting (11).

CASE II: Node j was added to the bottom of Λ_1. Since k is in $\Lambda_s = \Lambda$, it must be in all Λ_m ($1 \leqslant m \leqslant s$). Otherwise, in changing from inactive to active status at some list Λ_m, its label must have been corrected; but this is prohibited by (11). In particular, $k \varepsilon \Lambda_1$. Recall that a tree-derived sequence discipline adds node j to the bottom of list Λ_1 only if (8) fails to hold. Since $k \varepsilon \Lambda_1$, this means that $f(j) > f(k)$ in T_1. However, by (10) we have $j < k\,(T_1)$, and using property (7) of a tree-function produces $f(j) \leqslant f(k)$, a contradiction.

Since either case yields a contradiction, the sequence discipline is in fact order-compatible.

Because LIFO and PAPE sequence disciplines are special cases of tree-derived sequence disciplines, Theorems 1 and 2 produce the following results.

RESULT 1. Under LIFO/MOVE or LIFO/NO MOVE sequence disciplines, a labeling method always branches out from nodes having sharp labels.

RESULT 2. Under PAPE/MOVE or PAPE/NO MOVE sequence disciplines, a labeling method always branches out from nodes having sharp labels.

While the definition of a tree-derived sequence discipline in section 3 used a tree function f_T on the "old" tree T (before branching out has occurred), it is also possible to employ instead a tree function $f_{\hat{T}}$ based on the "new" tree $\hat{T}$ (which possibly incorporates new arcs emanating from the node just used for branching out). The proof of Theorem 2 shows that this second type of tree-derived sequence discipline is order-compatible as well. Specifically, in case II we would have $k \varepsilon \Lambda_2$, and $f(j) > f(k)$ would hold in $T_2 = \hat{T}$; again, a contradiction is reached to the fact that $j < k\,(T_2)$.

In summary, it does not really matter whether the old tree function values $f_T(i)$ or the newly-updated tree function values $f_{\hat{T}}(i)$ are used in defining the 2-WAY sequence discipline based on (8). In either case, the sequence discipline is order-compatible, and so only nodes k with sharp labels will be used for branching out. This property still holds whether we view all nodes updated by branching out as entering the sequence list simultaneously or sequentially.

However, under a FIFO sequence discipline, a node with a non-sharp label can be branched out from. For example, consider the network of figure 1, together with the associated sequence lists and trees produced an appropriate FIFO discipline. At the fourth step, node c is the top node of the list but it has an ancestor currently on the list. Thus, the label of node c is not sharp, and c will be used for branching out at the next step.

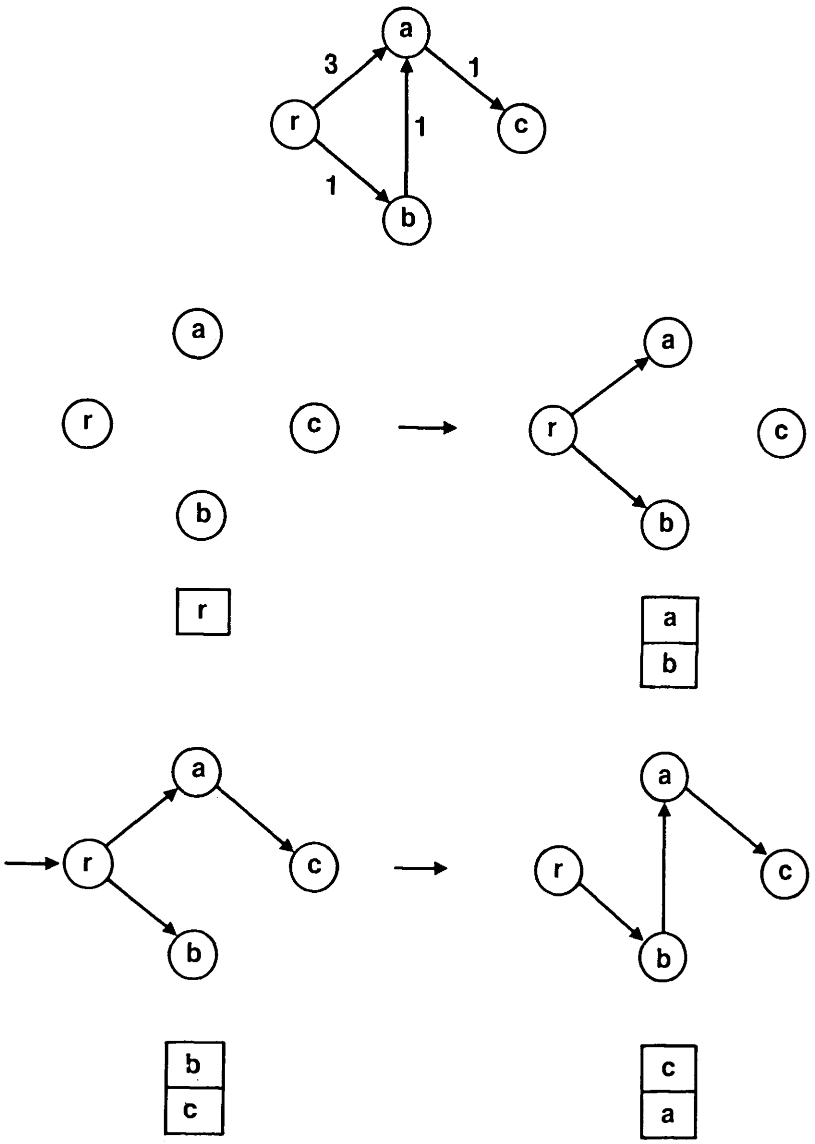


FIGURE 1.

RESULT 3. Under FIFO/MOVE or FIFO/NO MOVE sequence disciplines, a labeling method will not necessarily branch out from nodes having sharp labels.

A few additional observations are warranted. In the proof of Theorem 2, the fact that k was the top node of Λ was not used in any essential way. As a result, any tree-derived sequence discipline (including LIFO and PAPE) possesses the *strong-compatibility property:*
(SCP) If the sequence list Λ is linearly ordered and if $i, j \varepsilon \Lambda$ then

$$i < j(T) \qquad i \text{ precedes } j \text{ in } \Lambda.$$

Clearly, by Theorem 1, any discipline having the SCP will always branch out from nodes with sharp labels.

Also, the LIFO/MOVE sequence discipline creates a progression of trees having a very special property. Namely, any tree generated by such a discipline must have a form like that shown in figure 2, where the associated (linearly-ordered) sequence list Λ has entries $a, b, c, \ldots, y, z$ with a the top entry and z the bottom entry. Active nodes are indicated by squares in this figure, and inactive nodes are indicated by circles. There can be any number of active nodes (possibly none) adjacent from one of the "central" inactive nodes. The circled "I" configuration signifies an arbitrary collection of subtrees, possibly empty, of inactive nodes. This special property of LIFO/MOVE can be established in a straightforward manner by induction. Note that from the structure of the tree in figure 2, it is clear that nodes $a, b, c, \ldots; y, z$ on the sequence list are always *incomparable* in the associated tree T: i.e., neither $i < j\,(T)$ nor $j < i\,(T)$ holds. This property is not guaranteed to hold, however, for other sequence disciplines.

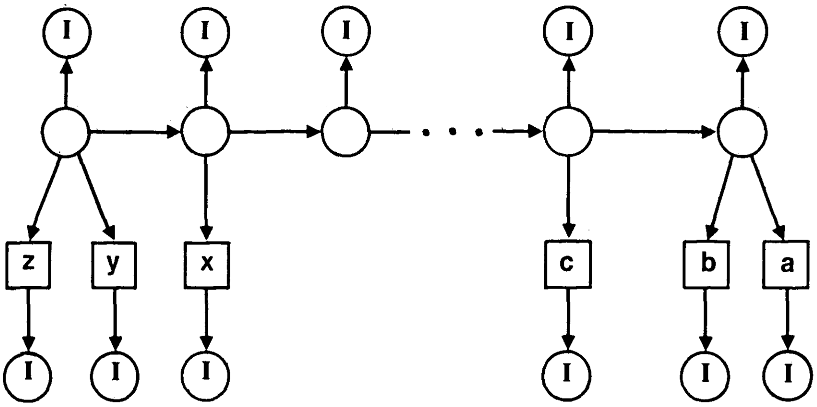


FIGURE 2.

The second major result of this section establishes that the Dijkstra sequence discipline is also order-compatible, even though arcs of negative length may be present. This result is somewhat surprising in that the labels $d(i)$ no longer form a tree function in the presence of negative arc lengths. However, it will be shown that the labels do define a tree function when restricted to active nodes (Theorem 3).

Notice that under the Dijkstra sequence discipline branching out from node k along a negative length arc creates an active label which is smaller than all labels of active nodes already on the list. Thus, the newly-labeled node becomes the top node of the list, giving the procedure somewhat the flavor of a LIFO-based procedure. To formalize this statement we define, for each node i in tree T,

$$L(i) = \min\{d(u)\colon u \text{ is active}, u \notin B(i,T)\}.$$

By convention, $L(i) = \infty$ if there are no active nodes outside B(i,T). Branch B(i,T) is said to be *saturated* if

$$d(j) \geqslant d(i)$$

holds for all active nodes j in $B(i,T)$. Our key observation is

$$L(i) \geqslant d(i) \text{ for nonsaturated } B(i,T). \tag{12}$$

Note that (12) implies

Each nonsaturated branch contains all active nodes of minimum label. (13)

We now proceed to prove (12). The statement is trivially satisfied for the initial labeling on the tree consisting of the root r alone. Assume it is true for subsequent labelings, including the present one (T,d,S); here S denotes the set of active nodes. After branching out from node k, which according to the modified Dijkstra procedure satisfies

$$d(k) = \min\{d(j) : j \varepsilon S\},$$

a new labeling $(\hat{T}, \hat{d}, \hat{S})$ results. The nodes with actual label changes are

$$h_1, \ldots, h_m \varepsilon F(k)$$

so that

$$\hat{d}(h_t) < d(h_t), t = 1, \ldots, m,$$

$$\hat{S} = S \cup \{h_1, \ldots, h_m\} - \{k\}.$$

Note that $m = 0$ is possible. In this case, the only change is a reduction in the number of active nodes: a previously nonsaturated branch may now be saturated; $L(i)$ may increase. Statement (12) remains true regardless. We assume henceforth that $m > 0$.

CASE I: Suppose node i is not an ancestor of node k in $\hat{T}$; then it is not an ancestor of node k in T. By (13), since $B(i,T)$ does not contain the minimum node k, $B(i,T)$ is saturated in (T,d,S). Note that

$$B(i,\hat{T}) \subseteq B(i,T).$$

Moreover, any active node $j \neq i$ in $B(i,\hat{T})$ is different from $h_1, \ldots, h_m$. Thus,

$$\hat{d}(j) = d(j) \geqslant d(i) \geqslant \hat{d}(i)$$

for all active nodes $j \neq i$ in $B(i,\hat{T})$. Thus, $B(i,\hat{T})$ is saturated for all nodes $i \neq k$ which are not ancestors of k in $\hat{T}$. This implies (12) holds in $\hat{T}$ for these nodes.

CASE II: Suppose $i = k$. Then we have

$$\hat{L}(k) \geqslant L(k) \geqslant d(k) = \hat{d}(k)$$

since $B(k,T) \subseteq B(k,\hat{T})$, since there are no label changes outside $B(k,\hat{T})$, and since $d(k)$ is a minimum label in (T,d,S). This implies statement (12) holds in $\hat{T}$ for node k.

CASE III: Suppose node i is an ancestor of k in $\hat{T}$, and therefore in T. Note again that

$$B(i,\hat{T}) = B(i,T) \cup B(k,\hat{T}) \supseteq B(i,T),$$

and that there are no label changes outside $B(i,\hat{T})$. Thus,

$$\hat{L}(i) \geqslant L(i).$$

In addition,

$$\hat{d}(i) = d(i) \leqslant L(i) \leqslant \hat{L}(i)$$

unless $B(i,T)$ is saturated. In the latter case, since $d(k)$ is a minimum label of (T,d,S),

$$\hat{d}(i) = d(i) \leqslant d(k) \leqslant L(i) \leqslant \hat{L}(i).$$

Thus, (12) holds in $\hat{T}$ for all nodes.

The required result will now be shown to follow from statement (13).

THEOREM 3: *Under the DIJKSTRA sequence discipline, the labels* d(i) *form a strict tree function when restricted to the active nodes.*

REMARK. If this theorem is proved, then the DIJKSTRA discipline is order-compatible, since an active ancestor j of node k in T would by Theorem 3 satisfy $d(j) < d(k)$, contradicting the fact that node k is an active node of minimum label. We proceed therefore to a proof of the theorem.

PROOF: The theorem clearly holds for the initial labeling on the tree consisting of node r alone. Suppose the theorem holds for (T,d) and that node k is used for branching out. By induction, node k does not have an active ancestor. The only possible violation of the theorem for the next labeling $(\hat{T},\hat{d})$ occurs because a newly-active node h_t has a label that is too large. However, $B(h_t,T)$ does not contain node k and so it does not contain all active nodes with minimum label. By (13), branch $B(h_t,T)$ is saturated, whence

$$d(h_t) \leqslant d(j)$$

holds for all active nodes j in $B(h_t,T)$. Since any active node j in $B(h_t,\hat{T})$ is active in $B(h_t,T)$, and since

$$\hat{d}(h_t) < d(h_t) \leqslant d(j) = \hat{d}(j),$$

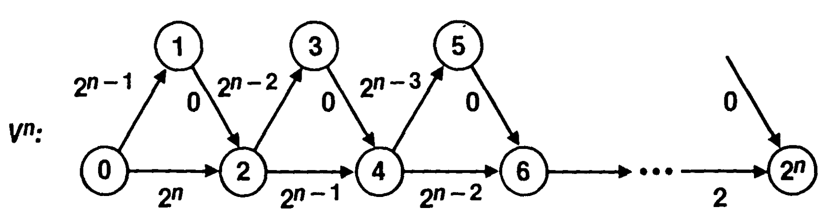
all active nodes in $B(h_t,T)$ satisfy the requirements of the theorem.

RESULT 4. Under the DIJKSTRA sequence discipline, a labeling method always branches out from nodes having sharp labels.

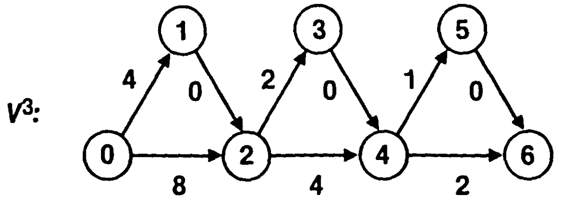
5. Computational Complexity of Labeling Methods

In this section the (worst-case) computational complexity of labeling procedures based on various sequence disciplines will be established. We consider the effort of calculating a shortest path tree by a labeling procedure to be the number of arcs examined (i.e., used in branching out). This definition is optimistic in that it does not include the work inherent in data-structure manipulations or in finding suitable arcs to examine. For NO MOVE variants, however, the latter constitute only an insignificant portion of the total work involved. (An alternative measure of effort is the total number of nodes entered onto the sequence list. Since each such node, apart from the root, gets placed on the sequence list as a result of examining some arc, this alternative measure is a lower bound for the first.)

We first study the effort required, in the worst case, to solve the shortest path problem using a LIFO sequence discipline. Consider the networks V^n, $n = 0, 1, 2, \ldots$ defined by



By convention, V^0 consists simply of the single node 0. Generally, V^n consists of $2n+1$ nodes and $3n$ arcs, where n designates the number of *segments*. For $n = 3$, we would have



We assume further that the networks V^n are represented by forward stars and that the arcs in these forward stars are scanned in order of increasing length.

The following propositions concerning the application of LIFO to V^n follow readily by induction.

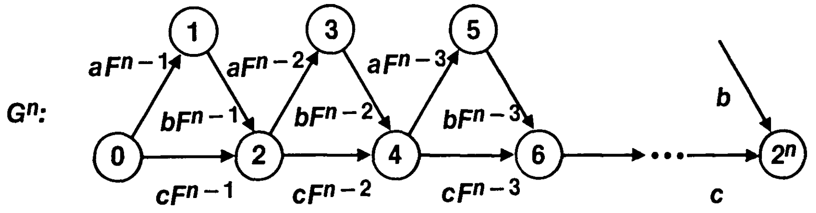
If $E(n)$ denotes the effort of solving V^n then $E(0) = 0$ and $E(n+1) = 3 + 2E(n)$ for $n = 0, 1, \ldots$ (14)

Every examination of an arc results in a node being added to the sequence list. (15)

Proposition (14) shows that $E(n) = 3(2^n-1)$ and thus the effort is exponential. Proposition (15) shows that LIFO/NO MOVE and LIFO/MOVE perform identically when applied to networks V^n. These facts are summarized in Result 5 below.

RESULT 5. Under LIFO/MOVE or LIFO/NO MOVE sequence disciplines, a labeling method has exponential computational complexity.

The idea behind the construction of the networks V^n is to build sequences of segments identical in topology, but with arc lengths for any segment being larger by a factor F than the corresponding arc lengths of the segment immediately to the right. For example, consider the networks G^n defined by



Suppose that arcs in forward stars are again scanned in order of increasing length.

LEMMA 3: *If* F $\geqslant 2$, $c > a + b$, *and* $c > a$ *then LIFO is exponential on the networks* G^n.

To show this, we first demonstrate a crucial fact about the 2^n paths from node 0 to node $2n$: $P_n(1), P_n(2), \ldots, P_n(2^n)$ which are generated by the LIFO discipline.

If paths $P_n(1), P_n(2), \ldots, P_n(2^n)$ are the paths generated in order by the LIFO discipline, then (16)

$$\ell(P_n(1)) > \ell(P_n(2)) > \ldots > \ell(P_n(2^n)).$$

PROOF: When $n = 1$, the paths are $P_1(1) = [0,2]$, $P_1(2) = [0,1,2]$ and $\ell(P_1(1)) = c > a + b = \ell(P_1(2))$, by assumption.

Suppose the assertion is true for $n = k - 1$. Then the set of paths produced in G^k by LIFO have lengths

$$\ell(P_k(1)) = F^{k-1}c + \ell(P_{k-1}(1))$$
$$\ell(P_k(2)) = F^{k-1}c + \ell(P_{k-1}(2))$$

.

.

.

$$\ell(P_k(2^{k-1})) = F^{k-1}c + \ell(P_{k-1}(2^{k-1}))$$
$$\ell(P_k(2^{k-1}+1)) = F^{k-1}a + F^{k-1}b + \ell(P_{k-1}(1))$$
$$\ell(P_k(2^{k-1}+2)) = F^{k-1}a + F^{k-1}b + \ell(P_{k-1}(2))$$

.

.

.

$$\ell(P_k(2^k)) = F^{k-1}a + F^{k-1}b + \ell(P_{k-1}(2^{k-1})).$$

By induction

$$\ell(P_{k-1}(1)) > \ell(P_{k-1}(2)) > \ldots > \ell(P_{k-1}(2^{k-1}))$$

and so

$$\ell(P_k(1)) > \ell(P_k(2)) > \ldots > \ell(P_k(2^{k-1})),$$

$$\ell(P_k(2^{k-1}+1)) > \ell(P_k(2^{k-1}+2) > \ldots > \ell(P_k(2^k))$$

It suffices then to show that

$$D = \ell(P_k(2^{k-1})) - \ell(P_k(2^{k-1}+1)) > 0$$

Now,

$$D = F^{k-1}c + \ell(P_{k-1}(2^{k-1})) - (F^{k-1}a + F^{k-1}b + \ell(P_{k-1}(1)))$$

$$= F^{k-1}c + (F^{k-2}a + F^{k-2}b) + \ldots + (Fa + Fb) + (a+b) - (F^{k-1}a + F^{k-1}b + F^{k-2}c + \ldots + Fc + c)$$

$$= F^{k-1}(c-a-b) - (F^{k-2}(c-a-b) + \ldots + F(c-a-b) + (c-a-b)).$$

Since $c-a-b > 0$, $D > 0$ if and only if

$$F^{k-1} - (F^{k-2} + \ldots + F + 1) > 0$$

or, since $F > 1$,

$$F^k > 2F^{k-1} - 1\,.$$

Now since $F \geqslant 2$, $F^k \geqslant 2F^{k-1} > 2F^{k-1} - 1$ and so (16) is established. (In fact, $D > 0$ for all k if and only if $F \geqslant 2$.)

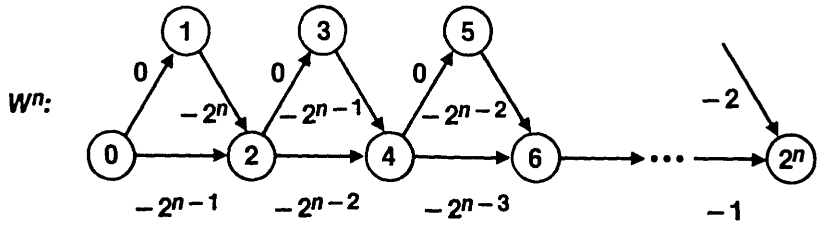
From (16) it follows that every arc incident to node 2n causes an update of d(2n), and similarly for nodes 2j ($1 \leqslant j < n$) and nodes 2j−1 ($1 \leqslant j < n$). This means all potential label corrections are made, and the exponential behavior follows with

$$E(n) = 3(2^n - 1)\,, n = 0, 1, \ldots$$

A similar fact holds when forward stars are ordered by decreasing length.

LEMMA 4: *If* $F \geqslant 2$, $c > a + b$, *and* $a > c$ *then LIFO is exponential on the networks* G^n.

Notice that V^n is the special case of G^n with $a = 1$, $b = 0$, $c = 2$, and $F + 2$. Lemma 3 thus guarantees exponential behavior on V^n, assuming that arcs in $F(k)$ are ordered by increasing length. The following networks W^n (corresponding to $a = 0$, $b = -2$, $c = -1$, $F = 2$) require exponential effort by LIFO when arcs in $F(k)$ are ordered by decreasing length (see Lemma 4).



We now show that

RESULT 6. Under PAPE/MOVE or PAPE/NO MOVE sequence disciplines, a labeling method has exponential computational complexity.

This result is not surprising inasmuch as Pape's method becomes a LIFO method once all nodes have been entered on the sequence list. To exhibit an actual example, we modify networks V^n by adding arcs from the root node 0 to all nodes not already connected to it. These arcs are given a very large arc length M ($>2^{n+1}$). The forward star of the root is arranged by nonincreasing arc length, whereas all other forward stars are arranged as before by increasing arc length. Pape's method, when applied to these modified networks $\hat{V}^n$, will enter all non-root nodes into the sequence list with nodes 2 and 1 being next-to-last and last, respectively. The nodes j having label $d(j) = M$ will be branched out from first. This will not produce any label changes, so that finally all active nodes will have disappeared except nodes 2 and 1, in this order. From this point on, Pape's method will reproduce LIFO as applied to the original networks V^n. Again, there is no difference between PAPE/MOVE and PAPE/NO MOVE.

RESULT 7. Under the DIJKSTRA sequence discipline, a labeling method has exponential computational complexity if arcs of negative lengths are admitted.

This result has been previously obtained by Johnson [9]. We show that it follows rather easily from our present results. If all arc lengths are nonpositive, and if the forward stars of the network are arranged in order of decreasing arc length, then the LIFO/MOVE discipline will be equivalent to the DIJKSTRA discipline, since branching out from a node of smallest label will place another node of smallest label in the top position of the sequence list. The networks W^n described earlier then provide the necessary evidence.

RESULT 8. Under FIFO/MOVE or FIFO/NO MOVE sequence disciplines, a labeling method has computational complexity $O(n^3)$, where $n = |N|$.

To demonstrate this result, let us define sets S_t, $t = 1, 2, \ldots$, as containing those nodes added in a FIFO manner to the sequence list during branching out of nodes in S_{t-1}. We set $S_0 = \{r\}$. In the case of FIFO/MOVE, a node j whose label is updated by branching out from $k \varepsilon S_{t-1}$ is always moved to S_t, even if node j is currently on the bottom of the sequence list.

A label $d(j)$ is said to have *cardinality* $c(j)$ if the path P corresponding to the path length $d(j)$ has precisely $c(j)$ arcs. It can be readily shown by induction that for a FIFO discipline (MOVE or NO MOVE)

$$c(j) \geqslant t \text{ for all nodes } j \varepsilon S_t. \tag{17}$$

As a matter of fact, $c(j) = t$ for the case of FIFO/MOVE. It follows from (17) that $S_n = \phi$ inasmuch as no shortest path length $d(j)$ on the sequence list need have $c(j) > n - 1$. Thus, all FIFO methods require at most n sets S_t before terminating. Since each set S_t can contain at most n − 1 nodes (node r cannot re-enter the sequence list) and since branching out from a node entails at most $n - 1$ arc examinations, the effort required is no more than $O(n^3)$. It is easy to give examples where this bound is achieved and thus a FIFO-based method has worst-case complexity $O(n^3)$, as stated in Result 8.

6. Conclusions

This paper has investigated two properties of sequence disciplines for labeling procedures: branching out from sharp labels, and worst-case computational complexity. Of the disciplines studied, only FIFO fails to branch out from sharp labels, yet only FIFO has polynomial complexity. Clearly, either of these properties alone is not sufficient to guarantee good performance in practice. For example, the LIFO discipline has been observed [2,7] to be inefficient in practice, even though it branches out from sharp labels. Also, the PAPE discipline has proven to be surprisingly successful [2] in sparse networks, even though it can require exponential effort for certain sparse networks ($\hat{V}^n$ of sect. 5).

In the highly structured sparse networks used for assessing the relative efficacy of Pape's method [2], this method does indeed act similar to a FIFO method while maintaining the sharp labeling property. We conjecture that Pape's method, as well as other 2-WAY methods, can achieve success by combining in a certain sense these two desirable, but apparently conflicting, properties. At the present writing it is not known whether there exists a polynomial labeling method that branches out from sharp labels. Even if such a method cannot be found, the 2-WAY tree-derived disciplines appear to be a promising area of further investigation. For example, it is not difficult to show that a 2-WAY method, based on using the new labels d as a tree function, performs polynomially for the networks $\hat{V}^n$ on which Pape's method is exponential.

7. References

[1] Barr, R.; Glover, F.; Klingman, D. Enhancements of spanning tree procedures for network optimization. INFOR **17**: 16–34: 1979 January.

[2] Dial, R.; Glover, F.; Karney, D.; Klingman, D. A computational analysis of alternative algorithms and labeling techniques for finding shortest path trees. Networks **9**: 215–248; 1979.

[3] Dijkstra, E. W. A note on two problems in connexion with graphs. Numer. Math. **1**: 269–271; 1959.

[4] Edmonds, J.; Karp, R. M. Theoretical improvements in algorithmic efficiency for network flow problems. J. Assoc. Comput. Mach. **19**: 248–264; 1972 April.

[5] Ford, L. R. Jr.; Fulkerson, D. R. *Flows in Networks.* Princeton, N. J.: Princeton University Press; 1962. 193 pp.

[6] Frank, H.; Frisch, I. T. *Communication, Transmission, and Transportation Networks.* Reading, Mass.: Addison-Wesley; 1971. 479 pp.

[7] Gilsinn, J.; Witzgall, C. A performance comparison of labeling algorithms for calculating shortest path trees. Nat. Bur. Stand. (U.S.) Tech. Note 772; 1973 April. 87 pp.

[8] Golden, B. Shortest-path algorithms: a comparison, Operations Research **24**, Technical Note: 1164–1168; 1976 November-December.

[9] Johnson, D. B. A note on Dijkstra's shortest path algorithm, J. Assoc. Comput. Mach. **20**: 385–388; 1973 July.

[10] Murchland, J. D. The 'once-through' method of finding all shortest distances in a graph from a single origin, London School of Economics, LBS–TNT–56.1; 1969.

[11] Pape, U. Implementation and efficiency of Moore-algorithms for the shortest route problems, Math. Programming **7**: 212–222; 1974.

[12] Shier, D. R.; Witzgall, C. Arc tolerances in shortest path and network flow problems, Networks **10**: 277–291; 1980.

[13] Srinivasan, V.; Thompson, G. L. Accelerated algorithms for labeling and relabeling of trees, with applications to distribution problems, J. Assoc. Comput. Mach. **19**: 712–726; 1972 October.

[14] Witzgall, C.; Gilsinn, J. F.; Shier, D. R. Shortest paths in graphs, in *Case Studies in Mathematical Modeling,* London: Pitman Publishing (in pess).