Opl Res. Q., Pergamon Press 1977. Vol. 28, 2, i, pp. 339 to 346. Printed in Great Britain

Minimum Paths in Directed Graphs

A. FRIEZE

Department of Computer Science and Statistics, Queen Mary College, University of London

This paper considers the problem of finding paths from a fixed node to all other nodes of a directed graph which minimise a function defined on the paths. Under certain assumptions a characterisation of optimal paths is derived. Two algorithms which are generalisations of standard shortest path methods are then given.

INTRODUCTION

THE SHORTEST route problem is of considerable importance in operational research. This paper considers a generalisation of this problem where path length is defined as a real valued function defined on paths and satisfying certain conditions.

The Dijkstra[1] shortest route algorithm for graphs with non-negative arc lengths has already been generalised in Hu.[2] The methods described here are generalisations of methods applicable to graphs without negative circuits.

The plan of the paper is as follows: we first give a precise description of the problem. A less precise description might be that given a distinguished starting node we try to find routes to all other nodes of a directed graph which minimise some measure of length. This measure is assumed to be such that the 'length' of a path depends only on the length up to the penultimate node and the last arc. This length increases with that to the penultimate node and to make the problem meaningful we must assume that traversing a cycle cannot reduce the length of a path.

We show that minimum paths have the characteristic that 'a minimum path has the property that the path up to its penultimate node can be assumed to be minimum also'.

We then use this characterisation to prove the validity of his algorithms. The first is a generalisation of an algorithm due to Pollatschek and Avi-Itzhak[3] and the second generalises an algorithm due to Moore[4] and Murchland.[5]

A MINIMUM PATH PROBLEM

Let G be a finite directed graph with nodes N and arcs A. Let ϕ be a real valued function defined on the paths of G and s be a distinguished 'source' node of G. The problem we discuss is that of finding for each node $j \neq s$ a path from s to j which minimises ϕ over all paths from s to j.

Where arc length is defined and for path P, $\phi(P)$ = the sum of the lengths of the arcs of P, we have a normal shortest path problem.

For an arbitrary function ϕ there is clearly not much to say and so we next deal with the assumptions made in the paper.

(1) We denote by Δ the value of $\phi((s))$ where (s) denotes the trivial path consisting of s alone.
(2) For each arc (i,j) there is a real function ψ_{ij} such that for a path P consisting of a path Q terminating at i followed by arc (i,j) we have

$$\phi(P) = \psi_{ij}(\phi(Q))$$

i.e. the ϕ value of P depends only on the ϕ value of Q and not on its node sequence.
(3) ψ_{ij} is a monotonic increasing function for each arc (i,j).

To simplify expressions we shall write the function value $\psi_{ij}(x)$ in the form $x * (i,j)$ ($*$ is now a real valued binary operation defined on $R \times A$). This allows us to write for path $P = (i_1,i_2,\ldots i_p)$ the expression

$$\phi(P) = \Delta * (i_1,i_2) * (i_2,i_3) * \ldots (i_{p-1},i_p)$$

where the order of evaluation of the expression is only meaningful from left to right.

This next assumption reduces in the shortest path case to the non-existence of negative cycles.
(4) For arbitrary real x and cycle $(i_1,i_2,\ldots i_p,i_1)$ in G we assume that

$$x * (i_1,i_2) * \ldots * (i_p,i_1) \geq x.$$

Examples

(i) Shortest paths: $x*(i,j) = x + c_{ij}$ where c_{ij} is the lengths of arc (i,j)
(ii) Minimax paths: $x*(i,j) = \max(x,c_{ij})$
(iii) Time dependent arc lengths: $x*(i,j) = x + c_{ij}(x)$ where $c_{ij}(x)$ is the length of arc (i,j) if a traveller arrives at node i at time x.
(iv) This approach deals satisfactorily with the case when there is more than one arc joining node i to node j and $\psi_{ij}(x) = \min_k(\lambda_{ijk}(x))$ where λ_{ijk} corresponds to the k^{th} arc from i to j e.g. in the context of (iii) having arrived at node i at time x a traveller selects the currently shortest arc for going from i to j.

The effect of these assumptions is to eliminate path problems where sub-paths of minimum paths cannot be assumed minimal e.g. the travelling salesman problem.

It follows from (4) that we can restrict our attention to elementary paths and further from (3) that these paths can always be selected so that they make up the structure of a directed tree rooted at s. Such a tree can be defined uniquely by a predecessor function $\pi: N \rightarrow N$ such that $\pi(s) = s$ and

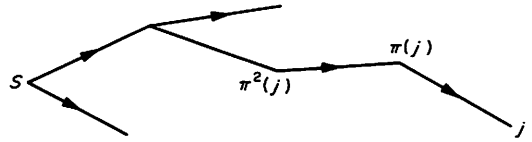

FIG. 1.

such that for each $j \neq s$ there exists $t > 0$ such that $\pi^t(j) = s$ [See fig. 1].

Given such a tree or equivalently a function π we define y_j to be the ϕ value of the unique tree path from s to j. We have $y_s = \Delta$ and a simple inductive argument on the number of arcs in a path shows that

$$y_j = y_{\pi(j)} * (\pi(j),j) \quad \text{for } j \neq s. \tag{5}$$

Theorem 1. The predecessor function π defines a set of optimal paths if and only if

$$y_j = \min(y_i * (i,j) | (i,j) \in A). \tag{6}$$

Proof. If: assume that (6) holds and that π' is any other predecessor function and y'_j are its associated path values. Then we have

$$y'_j \quad y'_{\pi(j)} * (\pi'(j),j) \tag{7}$$

and

$$y_j \leq y_{\pi'(j)} * (\pi'(j),j) \text{ by (6)} \tag{8}.$$

It follows from (3) that $y_{\pi'(j)} \leq y'_{\pi'(j)} \rightarrow y_j \leq y'_j$. Since $y_s = y'_s = \Delta$ a simple inductive argument on the number of arcs in paths defined by π' will give the result. Only if: suppose next that (6) does not hold. Define the function π' by $\pi'(s) = s$ and

$$y_{\pi'(j)} * (\pi'(j),j) = \min(y_i * (i,j) | (i,j) \in A) \tag{9}$$

and $\pi(j) \neq \pi'(j)$ only if $\pi(j)$ does not give the minimum on the RHS of (9).

We show first that π' defines a directed tree. This is equivalent to showing that the graph H with nodes N and arcs $(\pi'(j),j)$ does not contain a circuit. Suppose that to the contrary it does contain a circuit C which without loss of generality we denote by $(1,2,\ldots p,1)$. Since π does not define a circuit there must exist a node j of C such that $\pi(j) \neq \pi'(j)$ and for simplicity assume $j = 1$. This implies that

$$y_1 > y_p * (p,1) \tag{10}$$

we also have that by definition of π' that

$$y_j \geq y_{j-1} * (j-1,j) \quad \text{for } j \geq 2. \tag{11}$$

Successively substituting (11) into (10) and using (3) gives

$$y_1 > y_1 * (1,2) * (2,3) * \ldots * (p,1)$$

which contradicts (4). Thus π' defines a tree. Let y'_j be the associated path values for paths in this tree. Now the y'_j satisfy (7) and the y_j satisfy (8) with the inequality sign reversed. The argument following (8) can then be adapted to show that $y'_j \leq y_j$ for $j \in N$. By assumption there is a k such that $\pi'(k) \neq \pi(k)$ and so

$$\begin{aligned} y_k &> y_{\pi'(k)} * (\pi'(k), k) \\ &\geq y'_{\pi'(k)} * (\pi'(k), k) \\ &= y'_k. \end{aligned}$$

This shows that π is not optimal and completes the proof of the theorem.

AN ALGORITHM

The second part of theorem 1 suggests the following algorithm for solving our problem.

Step 1

Select an arbitrary predecessor function π.

Step 2

Calculate the values y_j associated with π. This can be achieved by using the following procedure until all y_j have been determined: choose j for which y_j is not known and compute $\pi(j)$, $\pi^2(j)$,... until one reaches a node i for which y_i is known. Note that $y_s = \Delta$ always. One then calculates the y value for each node on the path i to t using (5). Failure to find such a node i after a number of tries one less than the number of nodes with unknown y value indicates the existence of a circuit and implies that (4) is not satisfied.

Step 3

Test π for optimality by calculating π' as in (9). If $\pi = \pi'$ we terminate, otherwise replace π by π' and go to step 2.

This algorithm must converge after a finite number of iterations because there are only a finite number of predecessor functions π and the algorithm only repeats a function immediately prior to termination.

Example

To illustrate the method we consider the network of Fig. 2 and take $x * (i,j) = a_{ij}x + b_{ij}$ where $a_{ij} \geq 0$. The values a_{ij}, b_{ij} are indicated on the arcs ($\Delta = 0$).

To start the algorithm we construct a tree using the adaptation of the Dijkstra algorithm suggested in Hu. The important point in this case is that we cannot be sure that this solution is optimal.

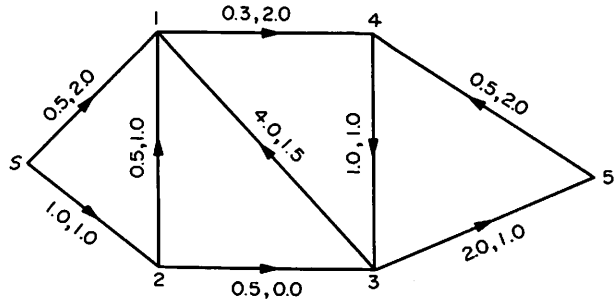

FIG. 2.

Step 1

j	1	2	3	4	5
$\pi(j)$	s	s	2	1	3

Step 2

j	1	2	3	4	5
y_j	2.0	1.0	0.5	2.6	2.0

Step 3

j	1	2	3	4	5
$\pi'(j)$	s	s	2	1	3

and so optimality of the 'Dijkstra' start is proven.

COMPARISON WITH DYNAMIC PROGRAMMING

The structure of this problem is such that dynamic programming could be used. If we define $f_r(j)$ = the minimum ϕ value of a path from s to j using r arcs or less then as in Bellman's shortest path algorithm we can write

$$f_{r+1}(j) = \min(f_r(i) * (i,j) | (i,j) \in A).$$

The above equation is used iteratively until $f_{r+1} = f_r$. One can show without difficulty that if at some stage of the above algorithm we have a predecessor function π such that its associated values y_j satisfy $y_j = f_r(j)$ for some r then the y_j after the next iteration satisfy $y_j \leq f_{r+1}(j)$. This is a direct consequence of the fact that

$$y'_j \leq y_{\pi'(j)} * (\pi'(j), j)$$

in the notation of the second part of the above theorem.

Thus if in step 1 we take $\pi(j) = s$ for all j with $\Delta * (s,j) = \infty$ for $(s,j) \notin A$—then the algorithm will terminate after no more than $n - 1$ iterations.

The experience of Pollatschek and Avi-Itzhak in the shortest path case suggests that convergence is actually more rapid than the dynamic programming method.

AN ALTERNATIVE ALGORITHM

At each stage of this algorithm we have a tree and we select a *candidate* node with which we try to find a better path to its neighbours.

Step 1

$C = \{s\}$ = initial set of candidate nodes.
$y_s = \Delta$ and $y_j = \infty$ for $j \neq s$.
$\pi(j) = s$ for all j.

Step 2

If $C = \phi$ the problem is solved, otherwise choose $j \in C$ and remove j from C.

Step 3

Process node j i.e. for each node k such that $(j,k) \in A$ compute $y_j * (j,k)$. If this quantity is less than the current value of y_k put $y_k = y_j * (j,k)$ and $\pi(k) = j$ and add k to C if it is not already there.

Go to step 2.

In the theorem below we assume that when choosing a candidate from C we may use the rule first in–first out.

Theorem 2

If assumptions (1)–(4) hold then the algorithm converges after a finite number of iterations and the paths defined by the algorithm are optimal. If the maximum number of arcs in a path defined by the algorithm is r then there can have been no more than $1 + t(n - (r + 1)/2)$ implementations of step 3.

Proof

We note first that throughout the algorithm the following inequality holds

$$y_j \geq y_{\pi(j)} * (\pi(j),j). \tag{12}$$

This holds initially and assume inductively that it holds prior to processing some node k and that $(k,t) \in A$. If $y_t \leq y_k * (k,t)$ then step 3 leaves (12) unaltered and if $y_t > y_k * (k,t)$ then step 3 makes (12) an equation. Thus (12) holds after processing k and hence throughout.

We now show that throughout the algorithm π defines a tree. This is true initially and so assume inductively that it holds prior to processing node k and this property is lost when for $(k,t) \in A$ we put $\pi(t) = k$. Assume that a circuit C is created and as in theorem 1 (only if part) we assume $C = (1,2,\ldots p,1)$ where $p = k$ and $1 = t$. Then (10) holds else we would not have changed $\pi(t)$ and (11) holds because of (12). Thus C is a negative circuit which is a contradiction. Thus π defines a tree throughout. Once a tree

is changed in step 3 it cannot re-appear as the y values never increase. We cannot keep the same tree for an infinite number of iterations since if there is no change the number of candidates is reduced by one and the process must stop eventually.

Suppose next that after termination there is a node k and a path $P = (i_0, i_1, \ldots i_p)$ from s to k such that $y_k > \phi(P)$ let i_t be the first node of P such that $y_{i_t} > \phi(i_0, i_1, \ldots i_t)$. However after i_{t-1} was last processed we must have

$$y_{i_t} \leq y_{i_{t-1}} * (i_{t-1}, i_t) = \phi(i_0, i_1, \ldots i_t)$$

which is a contradiction. A similar independent argument shows that i_{t-1} must be processed.

To prove the last part let $S_t \subseteq N$ consist of those nodes j for which the number of arcs in the path defined by the algorithm is no more than t. We note that once all the nodes in S_t have been processed for the last time a node in $S_{t+1} - S_t$ will be processed once more only. Further because of the order in which nodes are processed these will be finally processed after at most $n - |S_t|$ further implementations of step 2.

Thus the total number of iterations is no more than

$$|S_0| + n - |S_0| + n - |S_1| + \ldots + n - |S_{r-1}|. \tag{13}$$

Using the fact that $|S_t| \geq t + 1$ for $t = 1, \ldots r - 1$ in (13) completes the argument.

Now the dynamic programming algorithm requires $r'(n-1) + 1$ processings to prove convergence where r' is the maximum number of arcs in a path. Now in general $r' \leq r$ but a small perturbation in $*$ to $x*(i,j) + \epsilon$ is sufficient to ensure that $r' = r$. Alternatively if in step 3 there is equality between y_k and $y_j*(j,k)$ we can relabel if the path via j has fewer arcs. Thus this algorithm is superior to dynamic programming and it is surprising therefore that it is not mentioned more in text-books and papers e.g. it is not mentioned for example in Dreyfus[6]. Evidence of the quality of this algorithm applied to shortest path problems is given in Pape[7].

Though we have as yet no direct computational experience we would think that the first algorithm was preferable to the second when the number of arcs in at least one minimum path is large and conversely when there are relatively few arcs.

A FURTHER GENERALISATION

The above results can be generalised to cases where the function ϕ is not real valued.

As an example suppose that there are two real valued functions ϕ_1, ϕ_2 e.g. $\phi_1(P)$ = path length and $\phi_2(P)$ = length of longest arc in P. The problem is to

find paths which minimise $\lambda_1\phi_1 + \lambda_2\phi_2$ where $\lambda_1, \lambda_2 \geq 0$. We can approach this problem by denoting $y_i = (y_{i1}, y_{i2})$ where y_{ij} = the ϕ_j value of a path from s to i and taking $y_i \leq y_i'$ to mean $\lambda_1 y_{i1} + \lambda_2 y_{i2} \leq \lambda_1 y_{i1}' + \lambda_2 y_{i2}'$. The algorithms can then be applied as above.

In general then we assume that ϕ: PATHS $\rightarrow X$ where PATHS denotes the set of paths of our graph and X is a set together with a binary relation $\leq$ which has the following properties.

$$\text{For } x, y \in X \text{ either } x \leq y \text{ or } y \leq x \text{ or both.} \tag{13}$$

$$x \leq x \text{ for all } x \in X \text{ (reflexivity).} \tag{14}$$

$$x \leq y \text{ and } y \leq z \text{ implies } x \leq z \text{ (transitivity).} \tag{15}$$

Note that if $x \leq y$ and $y \leq x$ we cannot assume $x = y$ and so $\leq$ is not necessarily an order relation. We write $x < y$ to mean $x \leq y$ but $y \not\leq x$. (Note that $x < y \leq z \rightarrow x < z$).

The functions $\psi_{ij}: X \rightarrow X$ are monotonic in the sense that $x \leq y \rightarrow \psi_{ij}(x) \leq \psi_{ij}(y)$. For a finite subset $S \subseteq X$ we write $\min(x \in S)$ to mean an element $y \in S$ such that $y \leq x$ for $x \in S$. By (13) the min of a set is well defined. A path P minimises ϕ over a set of paths if $\phi(P) \leq \phi(P')$ for any other path P' in the set.

By retracing the proofs in the previous sections and interpreting entities as in this section we can show that the algorithms described will find paths from s to all other nodes which minimise ϕ.

REFERENCES

[1] E. W. DIJKSTRA (1959) A note on two problems in connection with graphs. *Numerische Mathematik* **1**, 269–271.

[2] T. C. HU (1969) *Integer programming and network flow.* Addison Wesley, New York.

[3] M. A. POLLATSCHEK and B. AVI-ITZHAK (1974) An efficient shortest route algorithm. *Angewandte Informatik* **16**, 477–482.

[4] E. F. MOORE (1957) *The shortest path through a maze. Proceedings of the international symposium on switching part II.* The Annals of the Computation Laboratory of Harvard University. Harvard University Press.

[5] J. D. MURCHLAND (1967) *The "One-Through" method of finding all shortest distances in a graph from a single origin.* Transport Network Theory Unit, London Graduate School of Business Studies. Report LBS-TNT-56.

[6] S. E. DREYFUS (1969) An appraisal of some shortest path algorithms. *Opns. Res.* **17**, 395–412.

[7] U. PAPE (1974) Implementation and Efficiency of Moore Algorithms for the Shortest Route Problem. *Math. Prog.* **7**, 212–222.

J. Opl Res. Soc. Pergamon Press 1978. Vol. 29, 1, p. 95. Printed in Great Britain

Addendum

to

"*Goal-seeking Behaviour in Queueing Systems*" (*Opl Res. Q.* 1976 **27**, 605–614)

GEORGE W. TYLER

No EXACT criterion was given previously for states of accommodation, although they were linked generally to maximum waiting times. Following Beneš,[1] we can now propose that the energy of our goal-seeking queueing system is represented by the number of customers in the system, and states of accommodation correspond to states of maximum entropy. In our particular case, the situation is complicated slightly by customer discouragement, because a proportion of customers queueing at any instant will be selected out before service commences. However, since these particular customers contribute nothing to the effectiveness of the system we can simply discount them and define energy on the effective number of customers in the system, i.e. those that are subsequently accepted for service. States of accommodation then correspond to maximum values of the entropy based on these effective customers alone. To estimate the effective number of customers in the system, we simply divide the waiting times of serviced customers by the mean service time. So for the probabilities of finding n effective customers in the system we get:

$$p_{n\,n<\sigma T} = A \exp\{-[(\sigma - \alpha)/\sigma]n\}$$

$$p_{n\,n\geq\sigma T} = A \exp(\alpha T - n).$$

The entropy is given by:

$$H = -\int_0^\infty p_n \log p_n \, dn.$$

Taking natural logs, this gives:

$$H = 1 - \ln A - \alpha A T \exp[-(\sigma - \alpha)T].$$

GEORGE W. TYLER

REFERENCE

[1] V. E. BENEŠ (1963) A "thermodynamic" theory of traffic in connecting networks. *Bell Syst. tech. J.* **42**, 567–607.

J. Opl Res. Soc. Pergamon Press 1978. Vol. 29, 1, p. 96. Printed in Great Britain

Corrigendum

Minimum Paths in Directed Graphs, Opl Res. Q. **28,** pp. 339–346

IN THE proof of theorem 2 on pp. 344–345 it states "Once a tree is changed in step 3 it cannot re-appear as the y values never increase". This statement would be valid if y_i was always the ϕ value of the path from s to i defined by the tree. This is not always true as the y values "lag behind" in this algorithm. However, it is always true that y_i is the ϕ value of some path from s to i and the number of such paths is finite. Thus one can see that a tree could only re-appear a finite number of times. The proof then continues as before.

A. M. FRIEZE