ON A NETWORK ALGORITHM OF BUTRIMENKO

D. E. Bell

February 1974 WP-74-8

Abstract

This paper considers an algorithm suggested by A. Butrimenko for a network communications problem. It is shown here that the algorithm converges, and to the optimal solution, when started from a particular initial solution.

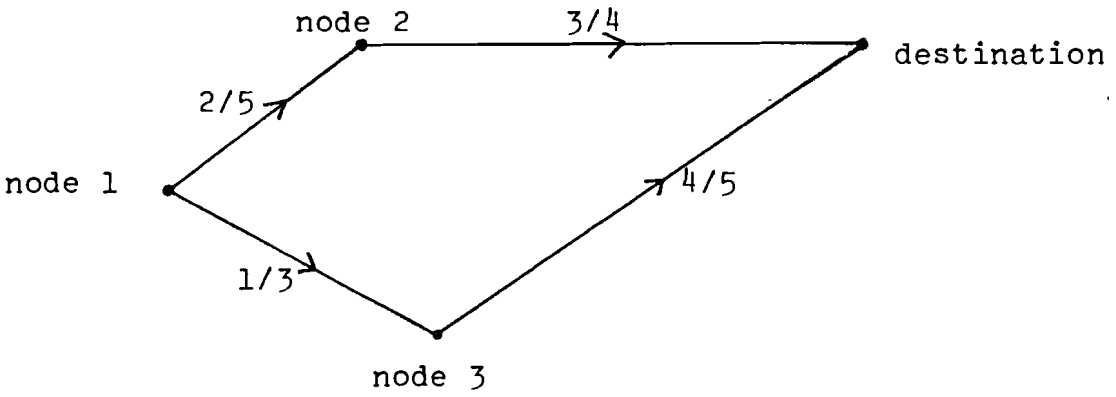
Contents

On a Network Algorithm of Butrimenko

1. Description of the Problem

Given a directed network with n nodes, a distinguished node called the destination, a set of probabilities $\{p_{ij}\}$ where arc (i,j) is "open" with probability p_{ij} and "closed" otherwise, the object is to find the probability at any given node of reaching the destination. Travel in the network is sequential, after transferring from one node to another the arc probabilities are independent. When at a node, it is known which arcs directed out of the node are open and which are closed.

For example, consider the following network:



Starting at node 1, suppose first that the policy is to move to node 2 if possible, node 3 otherwise. Then the probability of reaching the destination is

$$\frac{2}{5} \cdot \frac{3}{4} + \frac{3}{5} \cdot \frac{1}{3} \cdot \frac{4}{5} = \frac{23}{50} \quad .$$

The reverse policy (node 3 first) would be slightly better

$$\frac{1}{3} \cdot \frac{4}{5} + \frac{2}{3} \cdot \frac{2}{5} \cdot \frac{3}{4} = \frac{7}{15} \quad .$$

The questions are, what is the best policy and how may the associated probabilities be calculated.

2. An Algorithm

Dr. Butrimenko suggests the following iterative algorithm. Start with an initial guess $P^o = (P_1^o,\ldots,P_n^o)$ of the node probabilities, with $P^o_{\text{destination}} = 1$. Suppose the nodes are indexed so that

$$P_1^o \geq P_2^o \geq P_3^o \geq \cdots \geq P_n^o \quad .$$

Then

$$P_i^t = p_{i1}P_1^{t-1} + (1-p_{i1})\, p_{i2}P_2^{t-1} + (1-p_{i1})(1-p_{i2})\, p_{i3}P_3^{t-1} + \ldots$$

where the term in P_i^{t-1} is omitted. $P^t_{\text{destination}} = 1$ for all values of t. Butrimenko asks whether this iterative scheme will converge, and if it does, whether the solution is meaningful. It will be shown here that for the starting solutions

(i) $P_i^o = 1$ for all nodes i

(ii) $P_i^o = 0$ for all nodes except the destination

the algorithm converges and in case (i) to the optimal solution for the given problem.

3. Solutions to the Recursion

It will be useful to have a compact notation for the recursion of section 2. Let

$$p_i = (p_{i1},\ldots,p_{in})$$

where $p_{ii} = 0$ and

$$P^t = (P_1^t,\ldots,P_n^t)$$

where P_i^t is the estimate at iteration t of the probability of reaching the destination from node i. Define a function f by the relation

$$f(p_i, P) = p_{i\sigma(1)}P_{\sigma(1)} + (1-p_{i\sigma(1)})p_{i\sigma(2)}P_{\sigma(2)} + \cdots$$

where σ is a permutation chosen so that

$$P_{\sigma(1)} \geq P_{\sigma(2)} \geq P_{\sigma(3)} \geq \cdots \geq P_{\sigma(n)} \quad .$$

Note that we can always take $\sigma(1)$ = destination.

This notation reduces the recursion of section 2 to

$$P_i^t = f(p_i, P^{t-1})$$

which, if it converges, leads to the relation

$$P_i = f(p_i, P) \quad . \qquad (*)$$

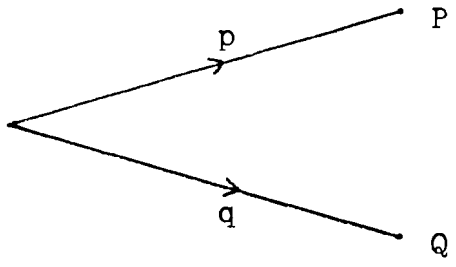
It will be shown later that this has at least one solution, but at present it is not clear that it need have exactly one solution. If it has more than one solution, then even if the algorithm converges, it may not be known whether it has converged to the optimal solution.

4. The Optimal Policy

Suppose that, situated at node i, it is known for each $j \neq i$ the maximum probability P_j of reaching the destination from j. Since, at node i, we know which arcs are open and which not, how should we proceed? It is clear that we should choose the open arc which leads to the node having maximum probability. If that is not clear, the following proof should help.

Lemma. The optimal policy at a node is to move to the node which, amongst those available, affords the best chance of reaching the destination.

Proof. Consider the case where only two arcs lead out of a node. The first, open with probability p



leads to a node from which the probability of reaching the destination is P and the second with probabilities q, Q respectively. Suppose P > Q. If the policy of the lemma is adopted, there is a probability of

$$pP + (1 - p)\, qQ \tag{1}$$

of gaining the destination. Otherwise the probability is

$$qQ + (1 - q)\, pP \quad . \tag{2}$$

But (1) is larger than (2) if and only if P > Q. Now consider a situation with three arcs with parameters P, Q, R and assume P > Q > R.

The ordering (P,Q,R) gives a probability

$$pP + (1 - p)[qQ + (1 - q)\, rR]$$

which is greater than

$$pP + (1 - p)\,[rR + (1 - r)\,qQ]$$

since $Q > R$.

Hence $(P,Q,R) > (P,R,Q)$ and by repetition of such pairwise interchanges

$$(P,Q,R) > (P,R,Q) > (R,P,Q) > (R,Q,P)$$

and

$$(P,Q,R) > (Q,P,R) > (Q,R,P) > (R,Q,P)$$

showing that (P,Q,R) is the best policy. This process may be repeated for any number of arcs. ||

The lemma was somewhat labored but it is important to establish the optimality of that policy. Note then that this policy gives the maximum probability of reaching the destination from i, P_i, as

$$P_i = p_{i1}P_1 + (1-p_{i1})\,p_{i2}P_2 + (1-p_{i1})(1-p_{i2})\,p_{i3}P_3 + \ldots \qquad (3)$$

which is the limiting situation for the recurrence relation given in section 2, that is, equation (*).

Since this relation holds for all nodes i, excepting only the destination node, it is demonstrated that the optimal solution to the given problem satisfies recursion (3).

5. Proof of Convergence to the Optimal Solution

The approach here will be to show that a particular sequence $\{P^t\}$ has the properties

(i) each P^t is an upper bound for all solutions to (*),

(ii) it converges to a solution of (*).

Thus the algorithm will have converged to the optimal solution of the problem.

Theorem 1. If P satisfies

$$P_i = f(p_i, P) \quad \text{for all } i$$

and $P \leq P^t$, then

$$P_i \leq P_i^{t+1} = f(p_i, P^t) \quad .$$

Proof. If

$$f(p_i, P) = \sum_j a_{ij} P_j$$

and

$$f(p_i, P^t) = \sum_j b_{ij} P_j^t$$

then $\Sigma\, a_{ij}P_j \leq \Sigma\, a_{ij}P_j^t$ by assumption

and $\Sigma\, a_{ij}P_j^t \leq \Sigma\, b_{ij}P_j^t$ by the lemma

so that $P_i \leq P_i^{t+1}$ as required. ||

Hence if an initial value $P_i^o = 1$ for all i is chosen, this is evidently an upper bound for the optimal solution; hence, the algorithm produces a sequence of upper bounds.

Theorem 2. If $P^{t-1} \geq P^t$, then $P^t \geq P^{t+1}$.

Proof. $P_i^{t+1} = \Sigma\, a_{ij}P_j^t$ say,

and $\Sigma\, a_{ij}P_j^t \leq \Sigma\, a_{ij}P_j^{t-1}$ by assumption,

but $\Sigma\, a_{ij}P_j^{t-1} \leq f(p_i, P^{t-1})$ by the lemma

and so $P_i^{t+1} \leq P_i^t$. ||

Main Theorem. From the starting solution $P_i = 1$ for all i, the algorithm converges and does so to the optimal solution.

Proof. By Theorem 2, the sequence $\{P^t\}$ is monotonically decreasing and thus converges to some solution P^∞ since it is bounded below by zero. By Theorem 1, P^∞ is an upper bound for all solutions to the recursion (*). By the lemma, the optimal solution satisfies the recursion. Hence, P^∞ is the optimal solution. ||

6. Conjecture

Starting from an initial solution $P_i^o = 0$ for all i, it may be shown that the resulting sequence $\{P^t\}$ is a sequence of lower bounds for all solutions to (*) and converges to one such solution. Hence, amongst the set of such solutions one is an upper bound and one is a lower bound. This state of affairs suggests that the two bounding solutions may indeed be the same solution and that there is exactly one solution to (*). This would show that if the algorithm converged, it would do so to the optimal solution. It remains to show whether the algorithm converges for any starting solution.

7. Finite Convergence for Acyclic Networks

For a certain class of networks, the conjecture is true and, in addition, the algorithm converges finitely.

Theorem 3. For an acyclic network, the algorithm converges in at most n-1 steps and to the optimal solution, from any initial values of P_j^o.

Proof. The idea is that after the kth iteration, $P_j^k = P_j^\infty$ for at least k+1 values of j.

Let S_0 = {destination node}

S_1 = {set of nodes whose only outgoing arcs go to the destination node}

and in general,

$$S_k = \{\text{set of nodes whose outgoing arcs are only incident with nodes in } \bigcup_{i=0}^{k-1} S_i\}.$$

The claim is that $P_j^k = P_j^\infty$ for all $j \in S_k$. The case $k = 0$ is automatic since $P^t_{\text{destination}}$ is constantly 1.

Assume, using induction, that it is true for $k = 0,1,\ldots,t$. But for $i \in S_{t+1}$, the recurrence relation

$$P_i^{t+1} = P_{il}P_l^t + \ldots$$

is only in terms of those P_j^t with fixed values by definition of the sets S_k; hence, $P_i^{t+1} = P_i^{t+2} = P_i^\infty$, and thus $P_i^{t+1} = P_i^\infty$ for all $i \in S_{t+1}$.

It now only remains to show that

$$\left|S_{k+1} - S_k\right| \geq 1 \quad \text{for all} \quad \left|S_k\right| \leq n-1 \ .$$

Suppose that for some k, $S_{k+1} = S_k$ and $\left|S_k\right| < n$. Let $\bar{S}_k$ be those nodes not in S_k. Since $\bar{S}_k \cap S_{k+1}$ is empty, each node of $\bar{S}_k$ has an outgoing arc to another node in $\bar{S}_k$. Starting at any node $i \in \bar{S}_k$ construct a sequence $(i,i_1,i_2,\ldots)$ where $p_{i_s i_{s+1}} > 0$ and $i_s \in \bar{S}_k$ for all j. Since $\bar{S}_k$ is finite, some $i_s = i_t$ for $s \neq t$ and $\bar{S}_k$ contains a cycle, which is contradictory.

Hence the algorithm converges finitely to a solution which is independent of the initial values. ||

Note that the algorithm for the acyclic case is a one stage process since each P_j^{∞} may be obtained sequentially immediately.