JOURNAL OF MATHEMATICAL ANALYSIS AND APPLICATIONS 73, 134–137 (1980)

Dynamic Programming Is Optimal for Certain Sequential Decision Processes

ARNON ROSENTHAL*

Department of Computer and Communication Sciences, University of Michigan, Ann Arbor, Michigan 48104

Submitted by L. A. Zadeh

A lower bound on the work to find a minimum-cost path in a monotone loop-free sequential decision process is proved. We show that dynamic programming always performs the smallest possible number of function evaluations. This is no more than the number required simply to prove that the chosen path is of minimum cost.

1. A *loop-free monotone sequential decision process*, considered in [1], can be defined as an acyclic directed graph, $G = (V, E)$, where each edge e has an associated nondecreasing function h_e: $R \to R$. It will be convenient to assume that G has only a single vertex (denoted s) with no entering edges, and only a single vertex (denoted t) with no edges leaving. Parallel edges with different functions may exist between the same pair of vertices.

The functions h_e can be extended to give path costs by the rule:

$$\text{cost}(P) = \begin{Bmatrix} 0 \text{ if } P \text{ is the path containing no edges} \\ h_e(\text{cost}(P')) \text{ if } P \text{ consists of } P' \text{ followed by } e \end{Bmatrix}.$$

This is a generalization of the usual shortest path problem, in which $\text{cost}(P) = \text{cost}(P') + \text{length}(e)$.

The above definition is equivalent to the definition in [1], and the minimum-cost path problem is to find a minimum cost path from s to t. (This is identical to the optimal policy problem of [1]). The problem can be solved by an algorithm essentially the same as the familiar algorithm for solving ordinary shortest path problems on acyclic graphs [2, Section 6]. (For brevity, the algorithm will not be presented here.)

* This work was partially supported by a grant from Horace Rackham Graduate School, and by NSF Grant MCS77-01753.

0022-247X/80/010134-04$02.00/0

2. Computational Effort

The most appropriate measure of the work done by the algorithm is the number of evaluations of functions h_e. This measure is especially appropriate if h_e represents a complicated formula as is found in queueing or fluid flow applications. The dynamic programming algorithm is known to require $|E|$ such function evaluations, regardless of the values obtained for the cost functions. Clearly, we would not expect to do very much better than this. Our purpose in this note is to study the sense in which this dynamic programming algorithm is indeed optimal.

An edge $e = (u, v)$ is called *interesting* if

(i) there is a path from s to t which includes e; and

(ii) there is a path from s to t which does not include e.

(If (i) is violated, e may be ignored. If (ii) is violated, then e must be on the optimal path, and the problem splits into the independent subproblems of determining optimal paths from s to u, and from v to t.) We are concerned only with interesting edges — uninteresting edges can be handled without regard to cost functions.

There are several senses in which a given algorithm can be said to use an "optimal" number of operations. [1, Theorem 3.3] states essentially: Given any n = desired number of vertices, and p = desired degree for vertices, there exists a graph (V, E) with $|V| = n$, each vertex except having P outward edges, such that the algorithm will require $|E|$ evaluations. Thus, dynamic programming, which requires $|E|$ evaluations is optimal on these graphs.

3. Results

Dynamic programming is optimal in a much stronger sense than the above.

Theorem. *Consider any graph all of whose edges are "interesting". Then every valid algorithm for every collection of edge-functions, will require as many evaluations as dynamic programming.*

Proof. To determine that a path P^* is of minimum cost, an algorithm must perform enough edge-function evaluations $h_{e_1}(y_1)$, $h_{e_2}(y_2)$,.. to eliminate the possibility that some other path has lower cost. The algorithm has knwoledge only of edge-functions at evaluated points, and the fact that all edge-functions are nondecreasing. We formally express the requirements for a proof:

Proof criterion. A set of function evaluations $h_{e_1}(y_1)$, $h_{e_2}(y_2)$,... is sufficient to prove that P^* has minimum cost if and only if P^* has minimum cost for every

collection of nondecreasing edge-functions $\{h_e(\cdot) \mid e \in E\}$ such that $\bar{h}_{e_1}(y_1) = h_{e_2}(y_1)$, $\bar{h}_{e_1}(y_2) = h_{e_2}(y_2)$,....

Now any valid algorithm must perform enough evaluations to prove that the chosen path P^* is of minimum cost. Thus, the lemma below will imply the theorem:

LEMMA. *Given a loop-free monotone sequential decision process with a graph $G = (V, E)$, such that E contains only "interesting" edges. Then any proof that a given path is minimum-cost contains at least $\mid E \mid$ evaluations.*

Proof of Lemma. Assume to the contrary, that for some edge $\bar{e} \subset E$, there is no number y such that $h_{\bar{e}}(y)$ has been evaluated. Let M be some positive number far larger than any number in the problem.

Case 1. $\bar{e}$ is on P^*. Consider edge-functions $\{\bar{h}_e \mid e \in E\}$ identical to the given ones except $\bar{h}_{\bar{e}}(y) \equiv M$ (for all y). For the new cost functions, $\bar{e}$ cannot be on the optimal path, so P^* is not optimal. Therefore, the proof was invalid, according to the proof criterion.

Case 2. $\bar{e}$ is not on P^*. Define a new edge-function $\{\bar{h}_e \mid e \in E\}$ identical to the given one, except $\bar{h}_{\bar{e}}(y) \equiv -M$. Now, $\bar{e}$ must be on the optimal path. Therefore, the proof was insufficient Q.E.D.

It is interesting to note that the argument for the second case fails if h_e is required to be nonnegative. This failure is irreparable, because for some cost functions, there exist proofs which do not need $\mid E \mid$ evaluations, but instead exploit the knowledge that all h_e are nonnegative.

4. DISCUSSION

Moravek [4] used a dimensionality argument to obtain a bound on the number of comparisons needed to prove optimality in any ordinary shortest path problem on an acyclic diagraph. The extension of his result to *all* loop-free monotone sequential decision processes does not seem straight-forward.

Computational complexity theorists have extensively studied procedures which somehow "guess" the solution to a problem and then "guess" a shortest proof that the solution is correct. Such procedures are known as nondeterministic algorithms [3], and are the foundation for the famous "$P \neq NP$" conjecture. The lemma implies that no algorithm, even a nondeterministic algorithm, will be superior for any input to dynamic programming, because dynamic programming always produces a shortest proof.

In [5] we show that nonserial dynamic programming is "optimal" for a more difficult optimization problem. That result establishes an exponential lower

bound which is valid for a large class of algorithms. There too, a nonadaptive deterministic algorithm is as good as even nondeterministic algorithms.

REFERENCES

1. T. IBARAKI, On the optimality of algorithms for finite state sequential decision processes, *J. Math. Anal. Appl.* **53** (1976), 618–643.
2. T. IBARAKI, Solvable classes of discrete dynamic programming, *J. Math. Anal. Appl.* **43** (1973), 642–693.
3. A. AHO, J. HOPCRAFT, AND J. ULLMAN, "The Design and Analysis of Computer Algorithms," Addison–Wesley, Reading, Mass., 1974.
4. J. MORAVEK, A note upon the minimal path problem, *J. Math. Anal. Appl.* **30** (1970), 702–717.
5. A. ROSENTHAL, Dynamic programming is optimal for nonserial optimization problems, submitted for publication.