Discrete Applied Mathematics 2 (1980) 167–170

COMMUNICATION

OPTIMAL MASS PRODUCTION

Arnon ROSENTHAL

Department of Computer and Communication Sciences, University of Michigan, Ann Arbor, MI 48109, USA

Communicated by R. M. Karp
Received 5 February 1980

It is often desirable to perform a "sensitivity analysis" by perturbing the data given with a problem. Motivated by serial and nonserial dynamic programming, we define a variety of sensitivity analysis problems, produce algorithms, and obtain lower bound results.

Consider an associative multiplication operation, initial values $Z_1, \ldots Z_N$, and alternate values $Z'_1, \ldots Z'_N$. Define $\text{Variant}_0 = \prod_1^N Z_j$, and $\text{Variant}_k = (\prod_1^{k-1} Z_j) Z'_k (\prod_{k+1}^N Z_j)$. This multiplication model is appropriate for many applications of serial dynamic programming (e.g. resource allocation, task assignment in distributed networks, equipment replacement, group knapsack) and for multiplying N matrices for convolving N functions. (In dynamic programming applications, the multiplication operation corresponds to optimally combining return functions of adjacent stages.)

1

We first define a very simple sensitivity analysis problem, namely the problem of computing *all* the variants. The practical contribution here is realizing that this form of sensitivity analysis is very attractive computationally. The actual algorithm is simple, and takes only a constant factor longer than just computing $\prod_1^N Z_j$. This form of sensitivity analysis does not seem to appear in the dynamic programming textbooks, perhaps because it is less obvious in specific applications than in this more abstract setting.

Define $L_j = \prod_1^j Z_i$ and $R_j = \prod_j^N Z_i$. L_0 and R_{N+1} are the identity.

Algorithm 1. (Mass produce variants in an arbitrary semigroup).
Produce $\{R_j \mid j = N-1, N-2, \ldots 1\}$ by multiplying right to left.
Produce $\{L_j \mid j = 2, 3, \ldots N-1\}$ by multiplying left to right.
For $k = 1, 2, \ldots N$: $\text{Variant}_k = L_{k-1} Z'_k R_{k+1}$.

Analysis. The algorithm requires exactly $4N$-5 multiplications.

Theorem 1. *Every algorithm which computes* $\prod_1^N Z_j$ *and all its variants requires at least* $4N$-5 *multiplications. Furthermore, every optimal algorithm performs essentially the same set of multiplications as algorithm* 1.

Proof method [3]. The computation is modelled as a directed acyclic graph with multipliers at the nodes. The graph is transformed step by step until it contains the graph for algorithm 1.

For a problem similar to the above, [3] also includes an algorithm valid in all semigroups which is (essentially) the only optimal algorithm, even in commutative semigroups.

Open problem. For each s $(1 \leq s \leq N)$ find the best algorithm which stores no more than s intermediate results. What bounds can be obtained for the time-storage product? J. Bentley and N. Swami have shown that $N \log_2 N$ operations suffice for $\log_2 N$ intermediate storage locations.

2. Treelike associations

In algorithm 1, multiplications were performed left-to-right or right-to-left. We now consider other parenthesizations (i.e. associations), to be represented by binary trees. Each leaf corresponds to an individual term Z_j, and each internal node represents the product of the terms represented by its two sons.

Notation. T_v denotes the terms represented by node v. $\prod(T_v)$ denotes the product of those terms. $\bar{T}_v$ denotes the set of terms not in T_v. v' and v'' denote the sons of v.

2.1. *On-line changes*

Suppose that instead of calculating all the variants, we receive and must immediately execute commands to change the value of a term and then compute the product. If the relative frequencies with which each term is changed are known, and the intermediate results corresponding to one tree may be stored, what tree (i.e. association) should be chosen? To change Z_j requires (path-length from Z_j to root) multiplications. The model immediately yields the results: The expected cost of making a change is minimized in an optimum alphabetic tree or a Huffman tree, depending on whether multiplication commutes [2, 3]. The worst case is minimized by a uniform binary tree.

2.2. *Unequal multiplication costs*

If different multiplications can take drastically different amounts of time, so "number of multiplications" is an inadequate complexity measure, then the

problem becomes much harder. Some examples are the nonserial dynamic programming problem of [1] (which includes many classical problems like set cover and satisfiability), and probabilistic circuit and fault tree problems [5]. A "good" association (i.e. a tree for which $\prod_1^N Z_i$ is obtained quickly) is used below to obtain all the variants. Algorithm 2 below is a simplification and generalization of algorithms in [4, 5]. We assume that a good association tree is known, and that multiplication commutes.

Algorithm 2. (1) (Find $\prod Z_i$ using given association.) For all internal nodes of the tree, ordered bottom-up

$$\prod(T_v) = \prod(T_{v'}) * \prod(T_{v''}).$$

(2) For all internal nodes v, ordered top-down

$$\prod(\bar{T}_{v'}) = \prod(\bar{T}_v) * \prod(T_{v''}),$$
$$\prod(\bar{T}_{v''}) = \prod(\bar{T}_v) * \prod(T_{v'}). \qquad \text{(produce complements)}$$

(3) For all leaves w, $\text{Variant}_w = \prod(\bar{T}_w) * Z_w$.

Now suppose each term Z is associated with a set V_z of "variables". A set T of terms is then associated with variables $\bigcup_{Z \in T} V_Z$. The *active set* (denoted $\text{Ac}(T)$) is {Variables $x \mid x$ belongs to a term in T and a term in $\bar{T}$}.

Assume also that for all sets S_1 and S_2 of terms, the time to multiply $\prod(S_1) * \prod(S_2)$ is a nondecreasing function of $[\text{Ac}(S_1) \cup \text{Ac}(S_2)]$. Our motivating examples obey all these conditions.

Theorem 3. *Under the above assumptions, the entire algorithm takes at most a constant factor longer than step* 1. *In this sense, algorithm* 2 *is within a constant factor of optimal.*

Sketch of proof

Lemma. (i) $\text{Ac}(T_v) = \text{Ac}(\bar{T}_v)$.

(ii) $\text{AC}(T_{v'}) \cup \text{AC}(T_{v''}) \supseteq \text{Ac}(T_v) \cup \text{Ac}(T_{v'})$.

By the lemma, the hypothesis of nondecreasing time, and symmetry, each step 2 multiplication at node v takes no longer than the multiplication at v in step 1. Step 3 takes no longer than step 2.

References

[1] U. Bertele and F. Brioschi, Nonserial Dynamic Programming (Academic Press, New York, 1972).

[2] D. Knuth, The Art of Computer Programming, Vol. 3 (Addison-Wesley, Reading, MA, 1973).

[3] A. Rosenthal, Optimal algorithms for sensitivity analysis in associative multiplication problems, to appear in Theoret. Comput. Sci.
[4] A. Rosenthal, An optimal algorithm for sensitivity analysis in nonserial optimization problems, to appear.
[5] A. Rosenthal, Decomposition algorithms and sensitivity analysis for probabilistic circuits and fault trees, submitted to SIAM J. Appl. Math.