Report on: Efficient Parallel Evaluation of Straight-line Code and Arithmetic Circuits

Muhammad Mumtaz AHMAD

Article written by

Gary L. Miller, Vijaya Ramachandran and Erich Kaltofen

1.Introduction

Most of the fast algorithms now known seem to fall into a few general classes. The most common are ones based on repetition or iteration, classic examples being Euclid's algorithm for GCD, Newton's method for Find Root and the Gaussian elimination method for Linear Solve. Starting in the 1950s it began to be realized that fast algorithms could be based on nested or recursive processes, and such algorithms became increasingly popular in the 1980s. In most cases, the idea is recursively to divide data into parts, then to do operations on these parts, and finally reassemble the results.

Dynamic evaluation is a technique for producing multiple results according to a decision tree that evolves with program execution. Sometimes it is desired to produce results for all possible branches in the decision tree, while on other occasions it may be sufficient to compute a single result, which satisfies certain properties. This technique finds use in computer algebra where computing the correct result depends on recognising and properly handling special cases of parameters. The term straight-line reflects the fact that evaluating an SLP can be achieved by a program, which does not branch or loop so its execution is a straight-line. The question of how quickly arithmetic expressions can be evaluated on a computer with several independent arithmetic processors is of theoretical and practical interest. In this paper the author provide an efficient parallel algorithm to evaluate a straight-line program. Parallel computing closely resemble with well-known rule “divide and conquer”. Now I will discuss algorithm, its analyses, its mathematical proofs, context, results, conclusions and comments about the paper respectively starting from an important definition.

2. Definition:

An arithmetic circuit is an edge weighted directed acyclic graph (where the weights on the edges are from the semi-ring R) such that

1. Each node should be: a leaf, an addition node or a multiplication node.
2. The indegree of: a leaf is 0, an addition node is non-zero, and a multiplication node is 2.
3. Multiplication nodes are not connected with each other.
4. Each leave is assigned a value in R

2.1. Definition:

If v is an addition node with children $\mathrm{v}_1,\ldots,\mathrm{v}_\mathrm{k}$ then $\mathrm{value(v)} = \sum_{i=1}^{k} \mathrm{value}(\mathrm{v}_\mathrm{i}) \cdot \mathrm{U}(\mathrm{v}_\mathrm{i}, \mathrm{v})$ where $\mathrm{U}(\mathrm{v}_\mathrm{i}, \mathrm{v})$ is the weight on the edge from v_i to v. if v is a multiplication node with children $\mathrm{v}_1\, and\, \mathrm{v}_2$ then $\mathrm{value(v)} = \mathrm{value}(\mathrm{v}_1)\mathrm{value}(\mathrm{v}_2)\mathrm{U}(\mathrm{v}_1,\mathrm{v})\mathrm{U}(\mathrm{v}_2,\mathrm{v})$, but where any edge entering a multiplication circuit weight will be considered 1.Leaf will have value according to definition of Arithmetic circuit.

3. The Algorithm.

The main purpose of the algorithm is to evaluate the arithmetic circuits by removing edges and shrinking the circuits without changing the actual values.

An arithmetic circuit can be viewed as an upper-triangular matrix U with zero diagonal, where the entry U_ij is the weight on the edge from node v_i to node v_j if the edge exists and it is zero otherwise.

Three sub matrices are derived from U

$$\mathrm{U}(+,+)_\mathrm{ij} = \begin{cases} \mathrm{U}_\mathrm{ij} & \text{if } \mathrm{v}_\mathrm{i} \text{ and } \mathrm{v}_\mathrm{j} \text{ are addition nodes} \\ 0 & \text{otherwise} \end{cases}$$

$$\mathrm{U}(\mathrm{X},+)_\mathrm{ij} = \begin{cases} \mathrm{U}_\mathrm{ij} & \text{if } \mathrm{v}_\mathrm{j} \text{ an addition node} \\ 0 & \text{otherwise} \end{cases}$$

$$\mathrm{U}(\mathrm{X},\mathrm{X})_\mathrm{ij} = \begin{cases} \mathrm{U}_\mathrm{ij} & \text{if } \mathrm{v}_\mathrm{i} \text{ or } \mathrm{v}_\mathrm{j} \text{ is not an addition node} \\ 0 & \text{otherwise} \end{cases}$$

The matrix U (+, +) corresponds to the sub circuit containing only plus-plus edges and U (X, +) corresponds to the sub circuit containing any edge terminating at an addition node. While the matrix U (X, X) corresponds to the sub circuit containing only those edges such that at least one end node is not an addition node.

Thus U (+, +) + U (X, X) = U.

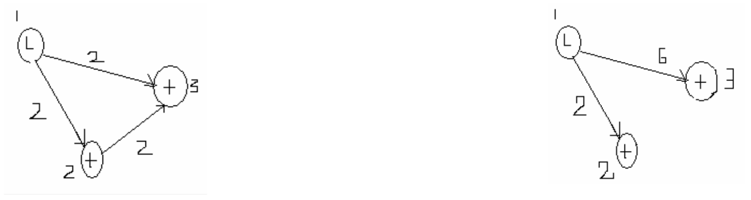
Three procedures are defined initially and then fourth procedure by combining three.

The procedure MM, $\mathrm{Eval}_{+}(\mathrm{U})$ and $\mathrm{Eval}_{\mathrm{x}}(\mathrm{U})$ applied to an arithmetic circuit return a new circuit with the same value.

3.1. Procedure MM (U)

$$\mathrm{U} \leftarrow \mathrm{U}(\mathrm{X},+) \bullet \mathrm{U}(+,+) + \mathrm{U}(\mathrm{X},\mathrm{X})$$

The procedure Matrix Multiply (MM) uses one matrix multiplication and one matrix addition over the semi-ring R. This procedure will remove the edges of the circuit and turn it into smaller one.



An arithmetic circuit before and after application of procedure MM(U)

3.2. Procedure $\mathrm{Eval}_{+}(\mathrm{U})$

```
for all addition nodes $v_j$ whose children are leaves do
    $\mathrm{value}(v_j) \leftarrow \sum_{i=1}^{n} \mathrm{value}(v_i) \cdot U_{ij}$
    Set $v_j$ to a leaf
    $U_{ij} \leftarrow 0 \text{ for } i \in \{1,...,n\}$
od
```

This procedure simply evaluates an addition node if all its children have been evaluated.

3.3. Procedure $\mathrm{Eval}_{\mathrm{x}}(\mathrm{U})$

```
    for all multiplication nodes $v_j$ with children $v_k$ and $v_l$ both of which are leaves
```

```
do
        $\mathrm{value}(v_j) \leftarrow \mathrm{value}(v_k) \cdot \mathrm{value}(v_i)$
        Set $v_j$ to a leaf
        $U_{kj} \leftarrow 0$ *and* $U_{lj} \leftarrow 0$
od
for all $U_{ji}$ *where* $v_j$ is a multiplication node with children $v_k$ and $v_l$
        and $v_k$ is a leaf and $v_l$ is not
do
$F_{lji} \leftarrow \mathrm{value}(v_k) \cdot U_{ji}$
od
for all pair (l, i) do
$W_{li} \leftarrow \sum_j F_{lji}$
$U_{li} \leftarrow U_{li} + W_{li}$
$U_{ji} \leftarrow 0$
od
```

The number of terms F_{lji} is at most the number of edges.

$\mathrm{Eval}_x(U)$ removes leaves in the first part and does a partial compress in the second part. A single procedure Phase is obtained by combining these three procedures. Repeated application of phase will eventually return the value of the circuit.

3.4. Procedure $\mathrm{Phase}(U)$

```
do
        $U \leftarrow \mathrm{MM}(U)$
        $U \leftarrow \mathrm{Eval}_+(U)$
        $U \leftarrow \mathrm{Eval}_x(U)$
```

od

4. Analysis of the algorithm:

The height of a circuit is used to analyse the number of applications of phase needed to evaluate a circuit of height h. The procedures can be viewed as maps of circuits to circuits.

The following lemma is the main tool to prove the result mathematically.

4.1. Lemma

If U and U' are arithmetic circuits as above and v' is a node of U' which is not a leaf and not an output node, then the height of v is at least twice the height of v'.

Where U is a circuit and U' its image under the transformation phase. Similarly v is a node of U and v' its image.

Proof: Let v' be a node of U' which is neither a leaf nor an output node. We use induction on the size of the sub circuit $U'_{v'}$ to prove the Lemma.

- Case 1

when all the children of v' are leaves.

There are two possibilities

Either v' is an addition node or

it is a multiplication node.

a) Suppose that v' is an addition node.

Then we have to prove that the height of v is at least 2, where v is the preimage v'.

Suppose that the height of v is less than 2.

But v cannot be of height 1 because a height 1 node must either be a leaf or all its children are leaves.

Hence one application of $\mathrm{Eval}_+(U)$ will transform v into a leaf, a contradiction.

Now suppose that the height is 3/2

then all the dominant children of v are addition nodes whose children are leaves.

Thus, after MM (U) and $\mathrm{Eval}_+(U)$ the node v will be a leaf, and hence v' will be a leaf. This proves the case when v' is an addition node of height 1.

b) Suppose that v' is a multiplication node with both its children leaves.

It is sufficient to show that both children of v have height at least 2.

Suppose that one child w has height less than 2.

Then after MM and Eval+ the node w will be a leaf.

Thus, after $\mathrm{Eval}_x(U)$ the vertex v will be either a leaf or an output node, depending on whether the other child of v is a leaf or not after Eval+, a contradiction.

This proves the initial cases of the induction.

The inductive case for multiplication nodes is rather straightforward.

- Case 2.

when one of the two children of v' is a leaf.

For the induction of multiplication node we use the result that if v' is a multiplication node which is not an output node and w' is a child of v' which is a leaf then the height of w is at least 2.

To prove the induction for addition nodes we proceed as follows

Suppose that v' is an addition node and w' be a dominant child of v'. If w' is a multiplication node then it is easy to follow the result.

If w' is an addition node. Then we use the result that h (w)$\leq$ h (v) –1.

5. Main Theorem

This theorem provides the actual result in mathematical form so that one can test the result mathematically

Statement: If U is an arithmetic circuit of degree d and size n then the value can be computed in parallel in time $O(\log n(\log nd))$ using at most M (n) processors.

Proof.

We use the result that If U is an arithmetic circuit with height h, and then after $\lfloor \log_2 h \rfloor + 1$ applications of Phase, all nodes of U are evaluated. The upper bounds are optimal for the procedure phase that is if h is the height of U then the procedure phase will be applied only $\lfloor \log_2 h \rfloor + 1$ times.

Now, if U is an arithmetic circuit of degree d and e is the number of plus-plus edges then the height of $\mathrm{U} \leq (1/2\mathrm{e} \cdot \mathrm{d} + \mathrm{d})$

Therefore $h = O(e \cdot d)$

Thus phase is applied at most O (log nd). That is each application of phase requires only log n parallel time. The matrix multiplication in MM (U) can be performed using O (M (n)) processors.

6.Context and Consequences:

The result of [1] was the best possible, up to small constant factors. It is compared with the well-known result of Valiant et al [2] that any multivariate polynomial of degree d that can be evaluated sequentially in n steps can be evaluated in O (log d)(log n+log d) parallel steps using $\mathrm{O}(\mathrm{n}^3\mathrm{d}^6)$ processors. The latter result is more general, but weaker, because of the log d terms in the time bound and the much larger number of processors. Both results apply to Boolean expressions, so they have implications for the design of circuits with small depth.

This paper shows that circuit of degree d and size n can be evaluated in time O (log n (log nd)) using M (n) processors. Where M (n) is the number of processors required for multiplying nxn matrices over the semi-ring R in O (log n). This is a generalization of the result of Valiant et al [2]. The crucial difference between this result and the result in Valiant et al [2] is that this algorithm need not know the degree of the circuit in advance. The degree of a circuit can also be computed in the above time and processor bounds.

7.Results:

It is noted that the result provided by the authors is tested and verified to be one of the best result. This result can be easily applied and verified both theoretically and practically. This is one of the interesting fields for new researchers as well as for general readers of interest in the field. Similar results for non-commutative rings are not available and also one can start his research for ring with division.

8.Conclusion and Comments:

Work exposed here falls under the optics of the efficient parallel algorithms to evaluate a straight-line program. The knowledge of the parallel computation made it possible to obtain fast calculations. Moreover, thanks to these authors, one knows now that how the results can be obtained by developing parallel algorithms. And how it can be verified mathematically. The researchers throughout in the paper tried their best to represent their result in a best way of expression. They have succeeded to draw the attention of the reader to read the whole paper. They explained the result using necessary definitions, figures and references. They focussed to produce the precise way rather than the quantity of the pages. They avoided discussing material irrelevant to the paper. They proved their claims using mathematical lemmas and theorem. In the paper the related algorithms are explained in very simple and easy ways. Procedures are represented in quite simple way for the readers. A simple language and its grammar are used in the paper to entertain the reader. Simple notations and mathematical symbols are used in the paper to make it easy. An equal interest is developed throughout the paper both for mathematicians and computer scientists. Interest related to graph theory and Ring theory draw the attention of related researchers in very simple way. The lemmas are provided along with their proofs using mathematical induction and contradiction. But there are some very serious technical mistakes in the paper. Perhaps that may occurred in printing but that are sufficient to put the reader in trouble. The authors must revise the paper.

References

[1] R. P. Brent, “The parallel evaluation of general arithmetic expressions”, Journal of the ACM 21 (1974), 201–206. CR 15#27055, MR 58#31996, Zbl 276.68010. rpb022

[2] L. G. Valiant, S. Skyum, S. Berkowitz and C. Rackoff, “Fast parallel computation of polynomials using few processors”, SIAM J. Computing 12, 1983, 641–644.

Computer Centre, Australian National University, Canberra, Australia

1991 Mathematics Subject Classification. Primary 68Q22; Secondary 65M15, 65Y05, 68Q25.