Efficient Parallel Evaluation of Straight-line Code and Arithmetic Circuits

Gary L Miller\textsuperscript{1}
Mathematical Sciences Research Institute and
University of Southern California
Computer Science Department
Los Angeles, CA 90089-0782

Vijaya Ramachandran\textsuperscript{2}
Mathematical Sciences Research Institute and
University of Illinois
Coordinated Science Laboratory
Urbana, IL 61801-3082

Erich Kaltofen\textsuperscript{3}
Mathematical Sciences Research Institute and
Rensselaer Polytechnic Inst.
Computer Science Department
Troy, NY 12181

Abstract

A new parallel algorithm is given to evaluate a straight line program. The algorithm evaluates a program over a commutative semi-ring $R$ of degree $d$ and size $n$ in time $O(\log n(\log nd))$ using $M(n)$ processors, where $M(n)$ is the number of processors required for multiplying $n \times n$ matrices over the semi-ring $R$ in $O(\log n)$ time.

\textsuperscript{1}Research supported in part by National Science Foundation Grant MCS-800756 A01.

\textsuperscript{2}Research supported by NSF under ECS-8404866, the Semiconductor Research Corporation under RSCH 84-06-049-6, and by an IBM Faculty Development Award.

\textsuperscript{3}Research Supported in part by NSF Grant DCR-8504391 and by an IBM Faculty Development Award
1. Introduction

In this paper we consider the problem of dynamic evaluation of a straight line program in parallel. This is a generalization of the result of Valiant et al [7]. They consider the problem of taking a straight line program and transforming it into a program of "shallow" depth. Their transformation is performed by a sequential polynomial time algorithm. We show how to construct this "shallow" program with slightly smaller size and the same time bounds on-line, no preprocessing, as their off-line algorithm.

We consider two basically equivalent models of evaluation over a semi-ring: straight line programs and arithmetic circuits. In the introduction we will restrict our discussion to the former model while most of the rest of the paper will deal with the latter model. A straight line program over a commutative semi-ring \( R = (R, +, \times, 0, 1) \) is a sequence of assignment statements of the form \( a \leftarrow b + c \) or \( a \leftarrow b \times c \) where \( b \) and \( c \) are either elements of \( R \) or previously assigned variables. The value of a variable is the natural one. We will assume that the semi-ring operations can be performed in unit time. Let \( M(n) \) denote the number of processors required to multiply two \( n \times n \) matrices in \( \log n \) time over the semi-ring \( R \) [1, 3].

A special case of a straight line program is a Boolean circuit. Ladner has shown that the Boolean circuit evaluation problem is P-Complete [5]. It is therefore believed that this evaluation problem is not in NC [Co80]. In this paper, we show that circuits of degree \( d \) and size \( n \) (we define these terms in Definition 3) can be evaluated in time \( O(\log n(\log nd)) \) using \( M(n) \) processors. The crucial difference between this result and the result in Valiant et al. [7] is that our algorithm need not know the degree of the circuit in advance. As a nontrivial application of our procedure we can also compute the degree of a circuit in the above time and processor bounds. This follows because the operations of maximum and sum form a commutative semi-ring over the nonnegative integers. We know of no other parallel algorithm for computing the degree that satisfies the above time and processor bounds.

2. Preliminaries

We view a straight line program as a special case of a more general object, an arithmetic circuit. Our results are more easily applied to arithmetic circuits:

Definition 1: An arithmetic circuit is a edge-weighted directed acyclic graph (DAG) (where the weights on the edges are from the semi-ring \( R \)) satisfying the following conditions:

1. Each node is labeled as one of three types: a leaf, a multiplication node, or an addition node.

2. Leaves are assigned a value in \( R \), denoted \( \text{value}(v) \) for a leaf \( v \).

3. The indegree of a leaf node is zero, of a multiplication node is two, and of an addition