ISSN 0105-8517

A Note on the Complexity of the Transpose of a Matrix

Philip Matthews
Carl Sturtivant

DAIMI PB – 265
September 1988

AARHUS UNIVERSITY
COMPUTER SCIENCE DEPARTMENT
Ny Munkegade 116 – DK 8000 Aarhus C – DENMARK
Telephone: + 45 6 12 71 88 *Telex: 64767 aausci dk*

Abstract

Let $\mathbf{x}$ be a column vector of indeterminates. We show that the complexity of computing the linear forms $A\mathbf{x}$ for a fixed matrix A is essentially the same as that of computing the linear forms $A'\mathbf{x}$ where the prime denotes transpose. Our result also holds for non-square matrices, under a simple restriction.

Introduction

Let A be a matrix over the field K. Given an appropriately sized vector of indeterminates $\mathbf{x}$, A defines the linear forms $A\mathbf{x}$. We informally define the complexity of A (denoted $\kappa(A)$) to be the complexity of computing the linear forms $A\mathbf{x}$. Of course, κ is only fully defined with respect to a particular model of computation.

Preliminaries

We use definitions similar to those in [Valiant, 77]. We define a *binary linear combination program* (or BLC program) to be a straight-line program in which each line computes an arbitrary linear combination of any two previously computed sub-results (by sub-results here we include the indeterminates $x_1 \ldots x_n$). Clearly, such a program can only compute linear forms.

Corresponding to a BLC program is a BLC circuit. This is just a fan-in 2 circuit (DAG) where for convenience, there are n input vertices with in-degree 0 which correspond to the indeterminates $x_1 \ldots x_n$, m output vertices for the resulting linear forms, and where each intermediate vertex corresponds to a linear combination and has in-degree 2. We refer to all vertices that are not input vertices as *computation vertices.* The output vertices are simply distinguished computation vertices. We also label the edges of the circuit with elements of the field K as follows: if an edge e is an ingoing edge to some computation vertex at which the sub-result at e is included in a linear combination with weight λ, then e is labeled by λ. Thus an alternative interpretation of the action of the circuit is that when a value is transmitted down an edge labeled λ, then it is scaled by λ, and when values enter a vertex, they are added.

We define a *linear program* to be a straight-line program in which each line computes an arbitrary linear combination of any number of previously computed sub-results and indeterminates. Corresponding to a linear program is a linear circuit. This is precisely the same as a BLC circuit, except that the fan-in to a vertex is not restricted to 2.

Complexity Measures

We define various complexity measures for linear forms and note their interrelationships.

Definition 1/1 $\kappa_{BLC}(A)$ equals the minimum number of lines in any BLC program computing the linear forms $A\mathbf{x}$ over the field K, where output scaling is free.

Definition 2/2 $\kappa_L(A)$ equals the minimum, over all linear circuits computing the linear forms $A\mathbf{x}$ over K, of the number of edges minus the number of computation vertices.

Definition 3/3 $\kappa_{NSA}(A)$ equals the minimum number of *non-scalar arithmetic* operations in an arithmetic straight-line program to compute the linear forms $A\mathbf{x}$ over K. Here we include the addition of two non-constant quantities in the count of non-scalar operations.

Definition 4/4 $\kappa_A(A)$ is defined in the same way as κ_{NSA}, except with an all-operation count.

Proposition 1/5 κ_{NSA} differs from κ_{BLC} by at most a constant factor, as does κ_A, provided the field K is infinite.

Proof See e.g. [Borodin 75].

Remark: For finite fields, this is open.

Proposition 2/6 For all matrices A, $\kappa_{BLC}(A) = \kappa_L(A)$.

Proof A BLC circuit is a linear circuit. Furthermore, if there is a BLC circuit of size C computing $A\mathbf{x}$, then the number of edges minus the number of computation vertices in that circuit is C, since each computation vertex has in-degree 2. Thus $\kappa_{BLC}(A) \geq \kappa_L(A)$.

Conversely, suppose there is a linear circuit computing $A\mathbf{x}$ of complexity C (i.e. the number of edges minus the number of computation vertices equals C). Each computation vertex in the circuit with in-degree d contributes $d-1$ to the complexity. However, such a node may be simulated by a tree of $d-1$ fan-in 2 nodes. The only problem is with $d = 1$, but this problem may be eliminated by assuming this to be an output scaling. The result of replacing all vertices with fan-in greater that 2 with such simulating trees, clearly gives a BLC circuit that has C computation vertices. Thus $\kappa_{BLC}(A) \leq \kappa_L(A)$.

□

Transposition

In this section, we consider linear circuits with n inputs and m outputs over a field K. We assume the input vertices are indexed $1 \ldots n$, and the output vertices are indexed $1 \ldots m$. The linear circuit is then simply an edge-weighted DAG computing linear forms $A\mathbf{x}$ where A is $m \times n$. We assume there are no isolated input or output vertices.

It is easy to see that A_{ij} is then just the sum of the weights of all paths in the circuit from input j to output i. (Here the weight of a path is simply the product of the weights on its edges.)

Proposition 3/7 Let G be a linear circuit computing linear forms defined by a matrix A. Let $rev(G)$ be the linear circuit defined by reversing the sense of all the edges of G and regarding the outputs as inputs and vice-versa. Then $rev(G)$ computes linear forms corresponding to the matrix A'.

Proof Let B be the matrix associated with $rev(G)$; thus B_{ij} equals the sum of the weights of all paths from input j to output i in $rev(G)$. But this is just the sum of the weights over all paths from input i to output

j in G itself, which, by definition, is A_{ji}. Thus $B_{ij} = A_{ji}$ and $B = A'$.

$\square$

Proposition 4/8 For any $m \times n$ matrix A, provided A has no zero columns or zero rows, $\kappa_L(A) - n = \kappa_L(A') - m$.

Proof Let G be a minimal linear circuit for $A\mathbf{x}$. Then by proposition 3/7, $rev(G)$ computes the linear forms associated with A'. Therefore, $\kappa_L(A)$ equals the number of edges in G minus the number of computation vertices in G. Thus $\kappa_L(A) - n$ is the number of edges of G minus the number of nodes of G, which is the same as the number of edges of $rev(G)$ minus the number of nodes of $rev(G)$. Since $rev(G)$ corresponds to A', $\kappa_L(A') - m \leq \kappa_L(A) - n$. Repeating the argument beginning with an optimal circuit for A' yields the opposite inequality.

$\square$

Corollary Let A be a square matrix with no zero rows or columns. Then $\kappa_L(A) = \kappa_L(A')$.

Proof Follows immediately.

$\square$

References

Borodin Borodin, A. and Munro, I., *The Computational Complexity of Algebraic and Numeric Problems.* American Elsevier, 1975.

Valiant Valiant, Leslie G., "Graph-Theoretic Arguments in Low-Level Complexity", *Mathematical Foundations of Computer Science, 1977. Lecture Notes in Computer Science # 53, Springer Verlag.*