Available online at www.sciencedirect.com

ScienceDirect

Applied Mathematics and Computation 190 (2007) 195–204

APPLIED MATHEMATICS AND COMPUTATION

www.elsevier.com/locate/amc

On why an algorithmic time complexity measure can be system invariant rather than system independent

Soubhik Chakraborty [a,*], Suman Kumar Sourabh [b]

[a] *Department of Statistics, Marwari College, T.M. Bhagalpur University, Bhagalpur 812007, India*
[b] *University Department of Statistics and Computer Applications, T.M. Bhagalpur University, Bhagalpur 812007, India*

Abstract

The present paper argues that it suffices for an algorithmic time complexity measure to be system invariant rather than system independent (which means predicting from the desk).
© 2007 Elsevier Inc. All rights reserved.

Keywords: Amir Schoor's algorithm; Sparse matrices; Dense matrices; Average case complexity; Big oh

0. Introduction

> As far as the laws of mathematics are certain, they do not refer to reality. And as far as they refer to reality, they are not certain.
> Albert Einstein

This is an aggressive research paper which reflects a threefold philosophy on measuring time complexity of an algorithm:

Philosophy 1. Work directly on time interpreting the "weight" of a computing operation to be the corresponding time it consumes. Weighing permits collective considerations of all operations for analysis, which is not possible otherwise, rendering a realistic appeal thereby.

Philosophy 2. Conceive experimental side of algorithmic complexity as a special kind of a computer experiment (Ref. [7]) in which the response variable is a complexity (such as time).

Philosophy 3. Ask for an empirical estimate over a finite range of a non-trivial statistical l.u.b. of the algorithm (defined later in the paper after a "live" demonstration). As a "live" demonstration we will first show that empirical $O(n^2)$ Complexity is convincingly gettable with two dense matrices in $n \times n$ matrix multiplication.

[*] Corresponding author.
E-mail addresses: soubhikc@yahoo.co.in (S. Chakraborty), sourabh.suman@rediffmail.com (S.K. Sourabh).

0096-3003/$ - see front matter © 2007 Elsevier Inc. All rights reserved.
doi:10.1016/j.amc.2007.01.016

Next some applications of the underlying concept are suggested in general as well in matrix multiplication involving chain of matrices.

We already know that for square matrices of order n, the average case complexity of Amir Schoor's algorithm is $O(d_1 d_2 n^3)$ where d_1 and d_2 are the densities (fraction of non-zero elements) of the pre factor and the post factor matrices respectively. For a formal proof, see Ref. [1]. If the product of these densities is kept at $1/n$, we automatically have an $O(n^2)$ complexity trivially. For example, we may keep the pre factor matrix a sparse matrix with density $1/n$ and the post factor matrix fully dense with density unity. *Our aim is to get similar complexity empirically under more robust input conditions.* Therefore, in the present paper, we keep the pre factor matrix approximately as dense as triangular (see Ref. [2]) except that the zeroes are randomly allocated and the post factor matrix is fully dense. Using "smart statistics" we observe that fits to quadratic and cubic are equally good. Section 1 gives the code we used, Section 2 gives the observations followed by the statistical analysis. Section 3 gives conclusion and suggestions for future work.

Amir Schoor's algorithm: Let A, B, and C be pre-factor, post-factor and product matrices respectively. Amir Schoor's algorithm states that for every non-zero $a(i,k)$, multiply the kth row of B by $a(i,k)$ and add it to the ith row of C. See also Ref. [1].

The pseudocode for the computational version only is as follows:

```
for i = 1 to n
        for k = 1 to n
            while(a(i,k) <> 0)
                r = a(i,k)
                  for j = 1 to n
                    b(k,j) = b(k,j) * r
                  endfor
                for j = 1 to n
                  c(i,j) = c(i,j) + b(k,j)
                endfor
            endwhile
        endfor
endfor
```

Remark. We would build the product matrix with all zero entries before starting the algorithm. Also, we would be using the code for dense matrices only rather than sparse and hence Schoor's original data structure (the "row-column-value" structure) normally used for sparse matrices need not be adhered to. Recall that a triangular matrix is dense (see Ref. [2]) with density $(n+1)/(2n)$. Since the borderline between sparse and dense matrices is not well defined, we would agree to call the pre-factor matrix dense in which the fraction of zeroes is approximately equal to that in a triangular matrix.

1. Section I

C^{++} code prepared by Suman Kumar Sourabh depicting Amir Schoor's algorithm implemented with a pre-factor matrix approximately as dense as a triangular matrix (zeroes randomly allocated) and post factor matrix fully dense, executed in Pentium4

```
#include⟨iostream.h⟩
#include⟨conio.h⟩
#include⟨time.h⟩
#include⟨stdlib.h⟩
#include⟨iomanip.h⟩
int main()
```