ISSN 1512-1232

AMIR SCHOOR'S ALGORITHM REVISITED FOR BERNOULLI AND GEOMETRIC DISTRIBUTION INPUTS

Mita Pal, Soubhik Chakraborty and N.C. Mahanti

Department of Applied Mathematics
Birla Institute of Technology, Mesra, Ranchi-835215, India
Email address of the corresponding author: soubhikc@yahoo.co.in (S. Chakraborty)

Abstract

This paper makes a comparative study between Bernoulli distribution inputs and Geometric distribution inputs in Amir Schoor's matrix multiplication algorithm. For fixed order of the square matrices, the average number of multiplications is found to increase and decrease linearly with increasing probability of success for Bernoulli and Geometric inputs respectively. Given the opposite nature of these two probability distributions, the commonality of linearity confirms the robustness of the algorithm.

Key words: *Amir Schoor's algorithm, Bernoulli distribution, Geometric distribution, sparse matrix, dense matrix*

1. Introduction

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

The pseudo code for the computational version only is as follows [2]:

```
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
```

In the present work**,** we build the product matrix with all zero entries before starting the algorithm. Also, we would be using the code with the post factor matrix as a fully dense matrix and the pre-factor matrix with expected density p which may or may not be dense. 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 [3]). 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. We generate a uniform U[0, 1] variate and if this falls between 0 and p, the pre-factor matrix element is made 1 otherwise zero. Therefore the probability for an element of this matrix to be non zero is p and hence pn^2 is the expected number of non-zero elements. Dividing by n^2 we get the density as p. If a and b are the densities of pre and post factor matrices respectively, then the average case complexity is

ISSN 1512-1232

$O(abn^3)$ for uniform inputs [1]. We find the result holding for non-uniform Bernoulli inputs also because with a=p, b=1 (post factor matrix fully dense) and fixed n, our simulation results are confirming an O(p) complexity.

2. C++ code

2.1 C++ code depicting Amir Schoor's algorithm implemented with a pre-factor matrix generated from Bernoulli distribution with p as 0.2, 0.5 and 0.8. The post factor matrix is fully dense (discrete uniform inputs with non zero elements).

```
#include <conio.h>
#include <iostream.h>
#include <math.h>
#include <stdlib.h>
#include <iomanip.h> //defines setw()

void main()
{
clrscr();
randomize();
int n,s;
unsigned long int t;
float p=0.8,r,x;
cout<<"Enter n";
cin>>n;
float d=(float)(n+1)/(2*n);

int **a,**b,**c;
a=new int *[n];
b=new int *[n];
c=new int *[n];


for(int i=0;i<n;i++)
{
 *(a+i)=new int [n];
 *(b+i)=new int [n];
 *(c+i)=new int [n];
 if(!(a+i) && !(b+i) && !(c+i))
 {
   cout<<i<<"insufficient memory...";
   exit(1);
 }
}

for(i=0;i<n;i++)   //pre-factor matrix generation
{
 for(int j=0;j<n;j++)
 {
     r=(float)rand()/RAND_MAX;


     if(r<p)
```

ISSN 1512-1232

```
            *(*(a+i)+j)=1;
      else
            *(*(a+i)+j)=0;
  }
}

for( i=0;i<n;i++)   //post-factor matrix generation
{
  for(int j=0;j<n;j++)
  {

            int r1=rand();

     *(*(b+i)+j)=r1;
  }
}
for( i=0;i<n;i++)   //initialization of element of resultant matrix with zero elements
{
  for(int j=0;j<n;j++)
  {
    *(*(c+i)+j)=0;
  }
}

//Amir schoor's algorithm begins
t=0;
for( i=0;i<n;i++)
{
  for(int k=0;k<n;k++)
  {

    if (*(*(a+i)+k)!=0)
    {
          x=*(*(a+i)+k);
          for(int j=0;j<n;j++)
          {
            *(*(b+k)+j)=*(*(b+k)+j)+x;
            t=t+1;
          }
          for(j=0;j<n;j++)
          {
            *(*(c+i)+j)=*(*(c+i)+j)+(*(*(b+k)+j));
          }
    }
  }
}
cout<<"t="<<t;
delete a;
delete b;
delete c;
getch();
}
```

ISSN 1512-1232

2.2 C++ code depicting Amir Schoor's algorithm (we are only showing how the pre-factor matrix is generated) implemented with a pre-factor matrix generated from Geometric distribution with p as 0.2, 0.5 and 0.8. The post factor matrix is fully dense (discrete uniform inputs with non zero elements).

```
for(i=0;i<n;i++)   //pre-factor matrix generation
{
 for(int j=0;j<n;j++)
 {
    r=(float)rand()/RAND_MAX;
    *(*(a+i)+j)= log(r)/(log1-p)    ;
 }
}
```

3. Experimental Results

Table - 3.1 gives average (mean) number of multiplication t and standard deviation s (average taken over 100 trials) for different values of the arguments p for fixed order 100x100 of the square matrices for Bernoulli distribution inputs.

Table – 3.1 Mean number of multiplication t (averaged over 100 trials) and standard deviation s with varying p for Bernoulli distribution

p	**mean no. of multiplications t**	**standard deviation s**
0.1	127550	2216.06
0.2	223750	2696.38
0.3	320350	2441.41
0.4	420560	5973.14
0.5	516840	3070.24
0.6	609280	4389.72
0.7	706350	4268.55
0.8	804460	4193.33
0.9	904950	7729.33

ISSN 1512-1232

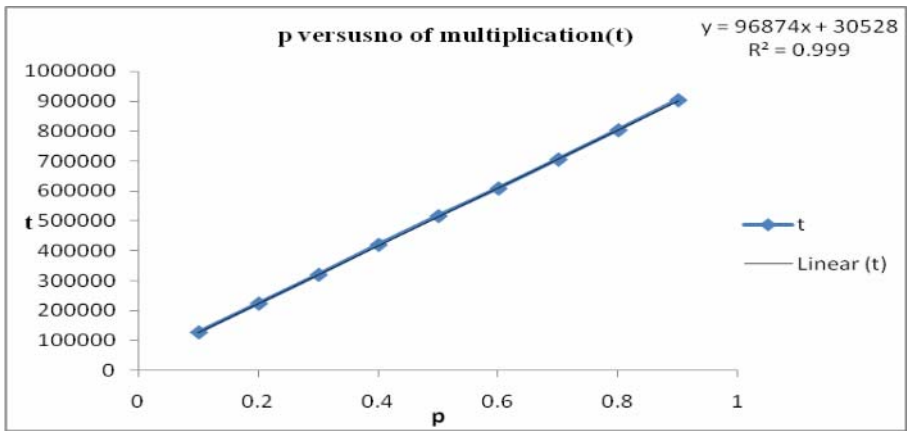

Fig. 1 Graph for p versus mean no of multiplication for Bernoulli distribution

Table - 3.2 gives average number of multiplication t and standard deviation (average taken over 100 trials) for different values of the arguments p for fixed order 100x100 of the square matrices for Geometric distribution inputs.

Table – 3.2 Mean number of multiplication t (averaged over 100 trials) and standard deviation s with varying p for Geometric distribution p mean no. of multiplications t standard deviation s

0.1	904300	2989.65
0.2	808830	3121.23
0.3	711720	7183.70
0.4	617380	4696.34
0.5	519210	4744.56
0.6	422680	3829.83
0.7	327110	3195.13
0.8	231930	4068.18
0.9	136460	3799.53

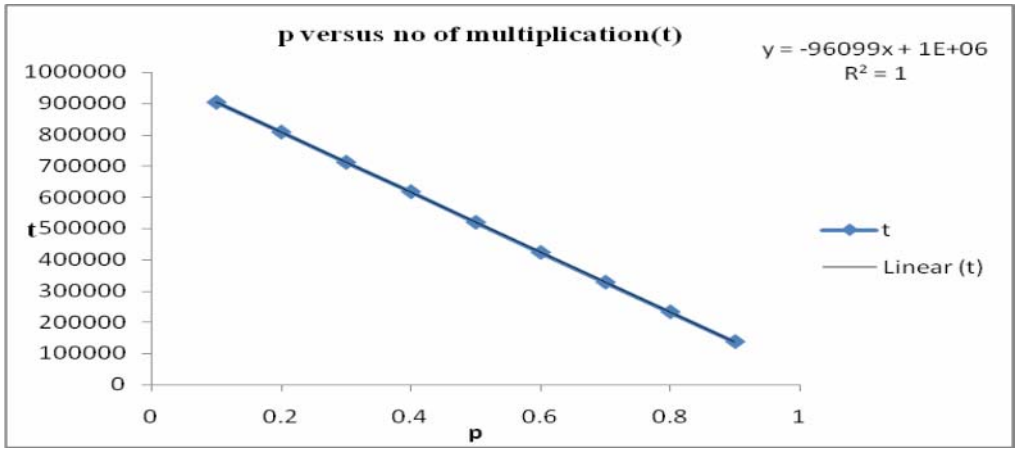

Fig. 2: Graph for p versus no of multiplication for Geometric distribution

ISSN 1512-1232

4. Discussion

It is easy to see that the number of multiplications increases linearly with p for fixed n for Bernoulli distribution inputs (fig. 1) while the same decreases linearly with p for fixed n for Geometric distribution inputs (fig. 2). We already know that Bernoulli distribution is opposite to Geometric distribution [4]. Bernoulli distribution is the special case of Binomial distribution for a single trial and Geometric distribution is the special case of Negative Binomial distribution when we want just one success and that Binomial distribution is opposite to Negative Binomial distribution because, in Binomial distribution, the number of trials is fixed and the number of successes is a random variable whereas in Negative Binomial distribution, reverse is the case. In Geometric distribution, if p is increased, we are more likely to get the desired success earlier so that the random variable giving the number of failures preceding the first success will decrease. In fact, the probability for this random variable to take zero value is exactly p. Hence increasing p increases the sparseness of the pre-factor matrix resulting in lesser multiplications. For the Bernoulli case, the random variable takes 0 value with probability 1-p and 1 with probability p. Hence if p is increased, the sparseness of the pre-factor matrix decreases resulting in more multiplications. As a final comment, given the opposite nature of these two probability distributions, the commonality of linearity confirms the robustness of the algorithm.

5. Conclusion

We have made a comparative study between Bernoulli distribution inputs and Geometric distribution inputs in Amir Schoor's matrix multiplication algorithm with the post factor matrix fully dense and with the density of the pre factor matrix depending on the probability p of success in Bernoulli and Geometric distributions. For fixed order n of the nxn square matrices, the mean number of multiplications is found to increase and decrease (as expected) linearly with increasing probability of success for Bernoulli and Geometric inputs respectively. Given the opposite nature of these two probability distributions, the commonality of linearity confirms the robustness of the algorithm so far as the operation multiplication is concerned. However, in [2] it is shown that if we work directly on program run time, a polynomial of degree two suffices to explain the average complexity because comparisons, which are n^2 in number (see the pseudo code in section 1), dominate over multiplication, an eye opener to the fact that there must be an alternative science that weighs computing operations rather than counting them, and further taking time of an operation as its corresponding weight, there should be a mixing of operations of different type conceptually into a bound which we agree to call a *statistical bound* [5].

References

1. A.Schoor, *Fast Algorithm for Sparse Matrix Multiplication*, Information Processing Letters 15, No.2, 1982, 87–89
2. S. Sahni, Data Structure and Algorithms in C++, Tata McGraw Hill, 2000.
3. S. Chakraborty and S. K. Sourabh, *On why an algorithmic time complexity measure can be system invariant rather than system independent*, Applied Mathematics and Computation 190 (1), 2007, 195–204
4. S. Ross, *A First Course in Probability*, Pearson Edu., 2006
5. S. Chakraborty, S. K. Sourabh, A Computer Experiment Oriented Approach to Algorithmic Complexity, Lambert Academic Publishing, 2010

Article received: 2011-03-03