RESEARCH ARTICLE OPEN ACCESS

Sparse Matrix to Decimal Coding (SMDC) Algorithm

Afsal K[#1], Sainul Abideen[#2], Dr. V kabeer[#3]
[#]*Department of Computer Science Farook College, Calicut, India*
Corresponding Author: Afsal K

ABSTRACT
We recently introduced a new method for Sparse matrix storage[1] which will considerably reduce the storage space by storing only nonzero elements along with the weight of each row(or column) and the number of rows(or column). This paper discusses two algorithms, SMDC Algorithm to convert a sparse matrix into decimal coding format and Reverse SMDC Algorithm to convert a decimally coded matrix back into the normal sparse matrix format. SMDC is a space optimized storage method for storing sparse matrices. It can store a sparse matrix with m rows and n columns and nnz nonzero elements, with smaller (m or n) + nnz +1 storage space, which is very much space efficient storage compared to most of the sparse matrix storage methods.

Keywords**:** *Sparse matrix, decimal coding matrix storage, Dense matrix storage, Decimal coding algorithm, sparse matrix storage algorithm.*

Date of Submission: 17-07-2017 Date of acceptance: 26-07-2017

I. INTRODUCTION

Sparse matrix storage is a very important area of mathematics and computer science. Sparse matrices are matrices with a considerable number of zero value elements and the storage methods mainly developed to reduce storage space[8] by storing information only related to nonzero elements. Most of such methods used in optimizing data storage size, perform data processing, increasing computation performance etc in computer science[3][4][5].

The basic idea of sparse matrix storage is to store only nonzero elements in memory with its address (positions) and avoid zero elements to reduce memory space[7][8]. So, if N is the number of nonzero elements in a sparse matrix, V will be an array of nonzero elements and R will be the row reference array which will be stored the raw reference and C will be the column reference array which will be containing column reference. So in the normal method, if N is the number of nonzero elements, 3N will be the storage space needed to store the matrix. By getting the nonzero values and references, we can regenerate the sparse matrix anytime in need. We are able to do basic matrix operations also by using the values stored in memory.

There are dozens of sparse matrix storage methods used in the industry according to the situations. few applications need time efficient storage methods and few will need space efficient methods, storage methods are selected according to the purpose and applications[5][2][6].

Here, we discuss the algorithms used to convert the sparse matrix into decimal coded representation and the decimally coded representation back into the sparse matrix. We already proposed the method in the previous paper which found more efficient in memory utilization. Most of the test cases show considerably less memory space to store a sparse matrix by using this method instead of using conventional methods. Moreover sparse matrices, we tried the method with a slight modification for a normal dense matrix which is having any particular non-zero element repeating instead of zero, and we were able to store that matrix too with considerably reduced storage space. The same method can be applied for storing a dense matrix but the only difference is that we have to consider repeated value too instead of zero in the system and it will consume one more storage space. Then the total memory space used to store a dense matrix will be small(m / n) + nnr +2, where m is the number of rows, n is the number of columns and nnr is the number of elements other than the repeated most repeated element.

II. METHOD

Consider a m x n sparse matrix A

a11	a12	a13	…….	a1n
a21	a22	a23	…….	a2n
a31	a32	a33	…….	a3n
.	.	.	…….	.
.	.	.	…….	.
.	.	.	…….	.
am1	am2	am3	…….	amn

Where m is the number of rows and n is the number of columns and nnz is the number of non-zero elements in the matrix A. The sparse matrix A can be represented as two one dimensional arrays P, V as follows. Array P stores the horizontal/vertical weight of each row/column. The first position of P will be stored with the number of row/column of the given matrix A. Its needed to store the number of row/column to understand the actual number and any one of the above we can identify from the number of weights.

P = [m/n, W1, W2, W3, …….Wm/n]

Array V stores all nonzero elements in the given matrix A in row/column major order.

V = [nz1, nz2, nz3, …..…. nzk]

III. WEIGHT CALCULATION

Consider row major weight calculation, let the number of columns, n be 4. We take the first row r1=[a11 a12 a13 a14] where a11, a12, a14 are zeros and a13 is a nonzero element. Then we consider the nonzero element as 1. So the row r1 will be [0 0 1 0], we consider this row as a binary number with four bits and convert it to decimal equivalent Hence the weight of the row will be 2 in decimal value from the above example. The decimal value represents the entire row and we will be able to convert it to binary and to create the actual row by replacing 1s with actual numbers stored in the array V. For calculating weight in column major, we have to do the same procedure with each column in the matrix.

Let m be 4

We take the first column c1= [a11 a21 a31 a41] where a11, a31, a41 are zeros and a21 is a nonzero element.

Then we consider the nonzero element as 1. So the row c1 will be [0 1 0 0], we consider it as a binary number and convert it to decimal equivalent as we done in the row major weight calculation. Hence the weight of the column will be 4 in decimal value for the given example. The decimal value represents the entire column and we will be able to convert it to binary and to create the actual row by replacing 1s with actual numbers stored in the array V. We can save space by selecting smallest from the number of rows and number of columns. If the matrix has 10 rows and 3 columns, it is better to select row major as P will be having only 3 elements and another value to store the number of columns.

IV. SMDC ALGORITHM

The SMDC algorithm can be easily constructed by two simple iterations through row elements and column elements. if the element is zero, just avoid the element, if it is a nonzero element, push the element to the value array V. After the iteration, the size of the array V will be the number of nonzero elements in the matrix. Let A be an m x n matrix with nnz non-zero elements. Let V be a value array which stores non-zero values and P is an array to store weight of each row or column. The first element of P will be the number of rows or columns according to the weight calculation method selected. Here we have selected row major weight calculation method and we can also implement color major algorithm with minor differences. We need to initialise two Arrays V and P which will hold non-zero elements and weight of each row. We are traversing through each and every cell in the matrix using two loops. On each cell, we are checking the value whether it is zero or not. If it is a non-zero element, we store it in value array V, and the non-zero elements of the matrix A will be replaced by 1. By finishing the process, V will be holding all nonzero elements and the matrix A will be having only zeros and ones as cell values.

```
INITIALISE Array V, P
INITIALISE k=1
FOR i= 1 TILL m
  FOR j= 1 TILL n
     IF A[i][j] != 0
        PUSH A[i][j] TO V[k]
        PUSH 1 TO A[i][j]

PUSH Number of Column to P[1]

FOR i = 1 TO m
   FOR j= n TO 1 STEP -1
      P[i+1] = P[i+1]+ A[i][j] * 2 ^ (n-j)
```

Next step is the Store number of columns in the first position of array P, this will help to reconstruct the Sparse Matrix. Other cells of P will be filled with the weight of each column. Weight can be calculated by two nested loops which will parse through each element in the matrix A and cover the binary sequence to Decimal using normal binary to decimal conversion method. Decimally coded matrix will be stored in arrays V, P, where V contains non-zero values and P is the position matrix, which contains weights where each value represents the weight of the row. The first value in the array P will be the number of columns.

V. REVERSE SMDC ALGORITHM

Reverse SMDC Algorithm is used to convert a decimally coded matrix to its original sparse matrix format. Here, we scan through each and every element in P, the first element will be treated as the number of row and rest of the elements are decimal values of positional weights. After converting each decimal value into an n-bit binary number, a new matrix of size m x n will be created and the binary values act as the m rows of the matrix. Then we replace each binary 1 with values of the array V. By repeating this process till the last decimal value in P, the original Sparse matrix A can be constructed.

```
INITIALIZE Matrix A
ASSIGN P[1] to n
FOR i = 2 TO Size of P
  FOR j= 1 TO n
    IF P[i]-2^(n-j)>=0
 ASSIGN  P[i] MOD(2^n-j) TO A[i-1][j]
      ASSIGN P[i]-2^(n-j) to P[i]
     ELSE
A[i-1][j]=0
INITIALISE k =1
FOR i= 2 to Size of P
   FOR j= 1 to n
      IF A[i-1][j]== 1
         REPLACE A[i-1][j] with V[k]
         INCREMENT k by 1
```

Here, we are initialising matrix A to store Sparse matrix constructed using the given decimally coded matrix in the form of value array V and positional weights of rows in array P. We are aware that first element of P is holding the number of columns in the sparse matrix. So we copy the first element of P, P[1] to n. The number of rows will be the (size of P) -1. By using two nested loops up to the number of rows and columns, we create the matrix with zeros and ones as the first step. Each value stored in P will be converted into binary format and it will be stored in the matrix. Now we know positions of nonzero values as they holding one and other cells holding zero. By parsing through each and every cell of the created matrix with zeros and ones, we replace all ones with nonzero values in value array V.

VI. CONCLUSIONS

The algorithm is implemented using computer environment and different sparse matrices are tested in both row major and column major scenarios. Sparse matrices with different size and cell values are tested and both SMDC and Reverse SMDC algorithms are working smoothly. Tried basic matrix operations like addition, subtraction, transpose etc with the Decimally coded matrix and the results are obtained.

REFERENCES

[1] Sparse Matrix using Decimal Coding, V.Kabeer, Afsal K, Sainul Abideen. International Journal of research in engineering and technology. eISSN : 2319-1163, pISSN : 2321-7308. Vol : 5. Special Issue : 22

[2] Sparse Matrix Storage Format, Fethulah Smailbegovic, Georgi N. Gaydadjiev, Stamatis Vassiliadis Computer Engineering Laboratory, Electrical Engineering Mathematics and Computer Science Mekelweg 4, 2628CD Delft TU Delft fethulah@computer.org stamatis, georgi@ce.et.tudelft.nl

[3] Matrix computing coprocessor for an embedded system.Bin Zhang , Kuizhi Mei, Jizhong Zhao Xian Jiaotong University, Xian 710049, China.

[4] A hardwaresoftware co-design approach for implementing sparse matrix vector multiplication on FPGAs Shweta Jain-Mendon , Ron Sass Reconfigurable Computing Systems Lab, Department of Electrical Computer Engineering, University of North Carolina at Charlotte, United States.

[5] Optimization of sparse matrix-vector multiplication using reordering techniques on GPUs Volume 36, Issue 2, March 2012, Pages 65- 77. Juan C. Pichel, Francisco F. Rivera, Marcos Fernndez, Aurelio Rodrguez.

[6] Information storage and effective data retrieval in sparse matrices. Hans J. Bentz, Michael Hagstroem University of Osnabrueck Germany. Guenther Palm University of Duesseldorf Germany.

[7] Sparse Matrix Technology. von: Sergio Pissanetzky Elsevier Reference Monographs, 1984 ISBN: 9781483270401 , 336 Seiten.

[8] Sparse Matrices and their Applications. ISBN13 9781461586777, Publisher :-Springer-Verlag New York Inc. [11] Iterative Methods for Sparse Linear Systems. by Yousef Saad, Second edition with corrections. January 3rd, 2000.

International Journal of Engineering Research and Applications (IJERA) is **UGC approved** Journal with Sl. No. 4525, Journal no. 47088. Indexed in Cross Ref, Index Copernicus (ICV 80.82), NASA, Ads, Researcher Id Thomson Reuters, DOAJ.

Afsal K. "Sparse Matrix to Decimal Coding (SMDC) Algorithm." International Journal of Engineering Research and Applications (IJERA) 7.7 (2017): 92-94.