The Single-Vector Compression Storage Research for Uniform Adjacent Block Diagonal Matrix

Hou Yongyan[1,a] ,Ren Zhiguo[2,b]

[1.] Zhixing College, Northwest Normal University, Lanzhou, China

[2.] School of Electronics and Information Engineering, Lanzhou City University, Lanzhou, China

[a.]498264224@qq.com [b.]ren_zhiguo@qq.com

Keywords: Compression storage, uniform adjacent block diagonal matrix, the row priority single-vector compression storage, the column priority single-vector compression storage

Abstract: Matrix is a mathematical object. We are not interested in data itself in the data structure, but how to store the elements in the matrix, so that various operations can be performed effectively. We often use a two-dimensional array to store the elements in the matrix sequentially. If adopt this method of storage, when there is large number of zero elements and have regular distribution a particular element will consume large amounts of storage unit. For high order matrix, the storage method is not only waste storage unit, but also takes a lot of time for invalid computation, it is obviously not desirable. In order to save the storage space, we need to compress storage for such matrix. The main purpose of the compressed storage is to make more of the same nonzero elements share the same storage unit according to the distribution of matrix element, while the zero elements don't allocate storage space. In this paper, we studied the row priority single-vector compressed storage and the column priority single-vector compressed storage of the uniform adjacent block diagonal matrix, and obtained the corresponding storage address mapping function, so as to help the scientific research worker.

1. Introduction

Matrix is a mathematical object, commonly used in scientific computing and engineering calculation. We are not interested in data itself in the data structure, but how to store the elements in the matrix, and make the various operations can run effectively. When programming in a high-level language, often use a two-dimensional array to store the elements in the matrix sequentially. If adopt this method of storage, we can random access each data element, and thus can easily realize operations of the matrix. But, when there is large number of zero elements in the matrix and have regular distribution, if still use a two-dimensional array to store the matrix, a particular element will consume a large of storage unit. For high order matrix, the storage method is not only waste storage unit, but also takes a lot of time for Invalid computation, it is obviously not desirable. In order to save the storage space, we need to compress storage for such matrix. The main purpose of the compressed storage is to make more of the same nonzero elements share the same storage unit according to the distribution of matrix element, while the zero elements don't allocate storage space. In this paper, we studied the uniform adjacent block diagonal matrix, and obtained the corresponding storage address mapping functions. The tags and terms used in the paper please reference [1-6].

In recent years, relevant research has produced some new results [7-16]. In this paper, we studied the uniform adjacent block diagonal matrix, and obtained the row priority single-vector compression storage address functions of uniform adjacent block diagonal matrix, and also obtained the column priority single-vector compression storage address functions of uniform adjacent block diagonal matrix for the first time, so as to help the scientific research workers.

2. The definition of uniform adjacent block diagonal matrix

This type of special matrix is often encountered in engineering calculations, as shown in Figure 1. If an n-order square matrix A satisfies the following conditions, it is called a uniform adjacent block diagonal matrix:

1) Matrix A is an n-order square matrix.

2) m is a positive integer, where m represents the order in the adjacency sub-matrix Λ in square A. The adjacency sub-matrix: The bottom-right corner element of the i-th adjacency sub-matrix is the top-leftmost element of the i+1th adjacency sub-matrix, and they share one element. The range of i is: $1 \leq i < (n-1)/(m-1)$

3) Each adjacent sub-matrix Λ is a square matrix with m^2 elements. The main diagonal of each adjacent sub-matrix Λ is on the main diagonal of matrix A.

4) When m=2, uniformly adjacent block diagonal matrix can also be considered as a banded matrix, n is a positive integer greater than 3.When m>2, m and n need to satisfy the relation:(n-1) mod (m-1)=0

5) When r=(n-1)/(m-1), The sum of the elements in all adjacent submatrices is $(m^2-1) \times r+1$. In addition to the $(m^2-1) \times r+1$ elements in the adjacency sub-matrix, the remaining elements in matrix A are all zero or the same constant

The 6-order uniform block diagonal matrix and sub-matrix of m=2, as shown in figure 1.

$$A = \begin{bmatrix} a_{11} & a_{12} & 0 & 0 & 0 & 0 \\ a_{21} & a_{22} & a_{23} & 0 & 0 & 0 \\ 0 & a_{32} & a_{33} & a_{34} & 0 & 0 \\ 0 & 0 & a_{43} & a_{44} & a_{45} & 0 \\ 0 & 0 & 0 & a_{54} & a_{55} & a_{56} \\ 0 & 0 & 0 & 0 & a_{65} & a_{66} \end{bmatrix}$$

$$\Lambda_1 = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \quad \Lambda_2 = \begin{bmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{bmatrix}$$

$$\Lambda_3 = \begin{bmatrix} a_{33} & a_{34} \\ a_{43} & a_{44} \end{bmatrix} \quad \Lambda_4 = \begin{bmatrix} a_{44} & a_{45} \\ a_{54} & a_{55} \end{bmatrix}$$

$$\Lambda_5 = \begin{bmatrix} a_{55} & a_{56} \\ a_{65} & a_{66} \end{bmatrix}$$

Fig.1. 6 order uniform adjacent block diagonal matrix and sub-matrix of m=2

3. The row priority single-vector compression storage of uniform adjacent block diagonal matrix.

If the data elements of a uniform block diagonal matrix are compressed and stored in a single-vector storage space B, the size of the single-vector B is $(m^2-1) \times r+1$.

If the row priority storage is used for these r sub-matrices----when compressing and storing, the top-leftmost sub-matrix in square matrix A is stored at first and the sub-matrix at the lower right corner of the square A is stored at last. When storing each sub-matrix, the elements of the first row are stored first and the elements of the last row are stored last. For the elements in each row, the first element is stored at first, then store the second one... Finally, the last element in this row is stored.

Then, when n=6, m=2, r=(n-1)/(m-1)=5, the uniform adjacent block diagonal matrix A can be compressed and stored in a single-vector storage space B.

The size of B can be obtained by calculation, it is $(m^2-1) \times r+1=(2^2-1) \times 5+1=16$.As shown in figure 2.

$$A=\begin{bmatrix} a_{11} & a_{12} & 0 & 0 & 0 & 0 \\ a_{21} & a_{22} & a_{23} & 0 & 0 & 0 \\ 0 & a_{32} & a_{33} & a_{34} & 0 & 0 \\ 0 & 0 & a_{43} & a_{44} & a_{45} & 0 \\ 0 & 0 & 0 & a_{54} & a_{55} & a_{56} \\ 0 & 0 & 0 & 0 & a_{65} & a_{66} \end{bmatrix}$$

B=

a_{11}	a_{12}	a_{21}	a_{22}	a_{23}	a_{32}	a_{33}	a_{34}
0	1	2	3	4	5	6	7

a_{43}	a_{44}	$a_{\cdot 45}$	a_{54}	a_{55}	a_{56}	a_{65}	a_{66}
8	9	10	11	12	13	14	15

Fig. 2. The results of compression storage at n=6,m=2,r=5

For a data element $a_{i,j}$ in the uniform adjacent block diagonal matrix A, it can be compressed storage into the single direction storage space B, if the element $a_{i,j}$ in A corresponding element is b_k. It can be seen that k is a function of (i,j,m), k is a function that can be computed:

$$k=f\,(i,j,m)=(i-1)*2+(j-1) \qquad (1\text{-}1)$$

When i=5,j=6 can be calculated to be k=(i-1)*2+(j-1)=(5-1)*2+(6-1)=13.You can see that the element $a_{5,6}$ in the matrix A is just in the position b_{13} in the single-vector B. So the conclusion 1-1 is true.

When n=5, m=3, r=(n-1)/(m-1)=2, the corresponding uniform adjacent block diagonal matrix A can be compressed storage into a single-vector storage space B.

The size of B can be obtained by calculation, it is $(m^2-1)\times r+1=(3^2-1)\times 2+1=17$.

As shown in figure 3.

$$A=\begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 & 0 \\ a_{21} & a_{22} & a_{23} & 0 & 0 \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \\ 0 & 0 & a_{43} & a_{44} & a_{45} \\ 0 & 0 & a_{53} & a_{54} & a_{55} \end{bmatrix}$$

B=

a_{11}	a_{12}	a_{13}	a_{21}	a_{22}	a_{23}	a_{31}	a_{32}	a_{33}
0	1	2	3	4	5	6	7	8

a_{34}	a_{35}	a_{43}	a_{44}	a_{45}	a_{53}	a_{54}	a_{55}
9	10	11	12	13	14	15	16

Fig.3. The results of compression storage at n=5, m=3,r=2

For a data element $a_{i,j}$ in the uniform adjacent block diagonal matrix A, it can be compressed storage into the single direction storage space B, if the element $a_{i,j}$ in A corresponding element is b_k. It can be seen that k is a function of (i,j,m), k is a function that can be computed:

$$k=f\,(i,j,m)=(i-1)*3+(j-1) \qquad (1\text{-}2)$$

When i=5,j=4 can be calculated to be k=(i-1)*3+(j-1)=(5-1)*3+(4-1)=15.You can see that the element $a_{5,4}$ in the matrix A is just in the position b_{15} in the single-vector B. So the conclusion 1-2 is true.

Similarly, when n=7, m=4, r=(n-1)/(m-1)=2, the corresponding uniform adjacent block diagonal matrix A can be compressed into a single-vector storage space B.

The size of B can be obtained by calculation, it is $(m^2-1)\times r+1=(4^2-1)\times 2+1=31$.

As shown in figure 4.

For a data element $a_{i,j}$ in the uniform adjacent block diagonal matrix A, it can be compressed storage into the single direction storage space B, if the element $a_{i,j}$ in A corresponding element is b_k. It can be seen that k is a function of (i,j,m), k is a function that can be computed:

$$k=f(i,j,m)=(i-1)*4+(j-1) \qquad (1\text{-}3)$$

When i=7,j=6 can be calculated to be k=(i-1)*4+(j-1)=(7-1)*4+(6-1)=29.You can see that the element $a_{7,6}$ in the matrix A is just in the position b_{29} in the single-vector B. So the conclusion 1-3 is true.

$$A=\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & 0 & 0 & 0 \\ a_{21} & a_{22} & a_{23} & a_{24} & 0 & 0 & 0 \\ a_{31} & a_{32} & a_{33} & a_{34} & 0 & 0 & 0 \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} & a_{46} & a_{47} \\ 0 & 0 & 0 & a_{54} & a_{55} & a_{56} & a_{57} \\ 0 & 0 & 0 & a_{64} & a_{65} & a_{66} & a_{67} \\ 0 & 0 & 0 & a_{74} & a_{75} & a_{76} & a_{77} \end{bmatrix}$$

B=

a_{11}	a_{12}	a_{13}	a_{14}	...	a_{74}	a_{75}	a_{76}	a_{77}
0	1	2	3	...	27	28	29	30

Fig.4. The results of compression storage at n=7, m=4,r=2

In summary, for a data element $a_{i,j}$ in a uniform adjacent block diagonal matrix A, it compressed storage into the single direction storage space B, if the element $a_{i,j}$ in A corresponding element is b_k. It can be seen that k is a function of (i,j,m).

When m=2, k is a function of (i,j,m) , formula1-1 was obtained k=(i-1)×2+(j-1).When m=3, k is a function of (i,j,m) , formula1-2 was obtained k=(i-1)×3+(j-1).When m=4, k is a function of (i,j,m) , formula1-3 was obtained k=(i-1)×4+(j-1).

In a uniform adjacent block diagonal matrix A, each sub-block matrix is an m-order matrix. In addition to the elements of the sub-matrix on the diagonal, the rest are some constants. The element $a_{i,j}$ in the sub-matrix is stored into a single direction B now. It has stored i-1 complete row's elements before the element $a_{i,j}$, obviously. So the elements number is m×(i-1). The number of elements at this column (j column) before $a_{i,j}$ is (j-1).

Therefore, when the elements in the matrix A are stored in a single-vector, k is a function of (i,j,m): k=(i-1)×m+(j -1).

Then the row priority address mapping function of the uniform adjacent block diagonal matrix we can be obtained. (formula 1):

$$k= f(i,j,m)= (i-1)\times m+(j -1) \qquad (1)$$

4. The Column priority single-vector compression storage of uniform Adjacent block diagonal matrix.

If the column priority storage is used for these r sub-matrices---- when compressing and storing, the top-leftmost sub-matrix in square matrix A is stored at first and the sub-matrix at the lower right corner of the square A is stored at last.

When storing each sub-matrix, the elements of the first column are stored first and the elements of the last column are stored last. For the elements in each column, the first element is stored at first, then store the second one... Finally, the last element in this column is stored.

Then, when n=6, m=2, the corresponding uniform adjacent block diagonal matrix A can be compressed and stored in the single-vector storage space B. As shown in figure 5.

Similar to row priority compressed storage, in column priority storage, for a data element $a_{i,j}$ in uniform adjacent block diagonal matrix A, it compressed storage into the single-vector storage space B, if the element $a_{i,j}$ in A corresponding element is b_k. It can be seen that k is a function of (i,j,m), k is a function that can be computed:

$$k=f(i,j,m)=(j-1)\times 2+(i-1) \qquad (2)$$

$$A=\begin{bmatrix} a_{11} & a_{12} & 0 & 0 & 0 & 0 \\ a_{21} & a_{22} & a_{23} & 0 & 0 & 0 \\ 0 & a_{32} & a_{33} & a_{34} & 0 & 0 \\ 0 & 0 & a_{43} & a_{44} & a_{45} & 0 \\ 0 & 0 & 0 & a_{54} & a_{55} & a_{56} \\ 0 & 0 & 0 & 0 & a_{65} & a_{66} \end{bmatrix}$$

B=

a_{11}	a_{21}	a_{12}	a_{22}	a_{32}	a_{23}	a_{33}	a_{43}
0	1	2	3	4	5	6	7

a_{34}	a_{44}	a_{54}	a_{45}	a_{55}	a_{65}	a_{56}	a_{66}
8	9	10	11	12	13	14	15

Fig. 5. The results of compression storage at n=6,m=2 and r=5

So when n=6 and m=3, the corresponding uniform block diagonal matrix A can be compressed into a single vector storage space B. As shown in figure 6.

$$A=\begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 & 0 \\ a_{21} & a_{22} & a_{23} & 0 & 0 \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \\ 0 & 0 & a_{43} & a_{44} & a_{45} \\ 0 & 0 & a_{53} & a_{54} & a_{55} \end{bmatrix}$$

B=

a_{11}	a_{21}	a_{31}	a_{12}	a_{22}	a_{32}	a_{13}	a_{23}	a_{33}
0	1	2	3	4	5	6	7	8

a_{43}	a_{53}	a_{34}	a_{44}	a_{54}	a_{35}	a_{45}	a_{55}
9	10	11	12	13	14	15	16

Fig. 6. The results of compression storage at n=6, m=3 and r=2

For a data element $a_{i,j}$ in the uniform adjacent block diagonal matrix A, it can be compressed storage into the single direction storage space B, if the element $a_{i,j}$ in A corresponding element is b_k. It can be seen that k is a function of (i,j,m), k is a function that can be computed:

$$k=f\,(i,j,m)=(j-1)*3+(i-1) \qquad (3)$$

When i=5,j=4 can be calculated to be k=(j-1)*3+(i-1)=(4-1)*3+(5-1)=13.You can see that the element $a_{5,4}$ in the matrix A is just in the position b_{13} in the single-vector B. So the conclusion 2-2 is true.

When n=7,m=4,r=(n-1)/(m-1)=2, the corresponding uniform adjacent block diagonal matrix A can be compressed into a single-vector storage space B.

The size of B can be obtained by calculation, it is $(m^2-1)\times r+1=(4^2-1)\times 2+1=31$.

As shown in figure 7.

For a data element $a_{i,j}$ in the uniform adjacent block diagonal matrix A, it can be compressed storage into the single direction storage space B, if the element $a_{i,j}$ in A corresponding element is b_k. It can be seen that k is a function of (i,j,m), k is a function that can be computed:

$$k=f\,(i,j,m)=(j-1)*4+(i-1) \qquad (4)$$

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & 0 & 0 & 0 \\ a_{21} & a_{22} & a_{23} & a_{24} & 0 & 0 & 0 \\ a_{31} & a_{32} & a_{33} & a_{34} & 0 & 0 & 0 \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} & a_{46} & a_{47} \\ 0 & 0 & 0 & a_{54} & a_{55} & a_{56} & a_{57} \\ 0 & 0 & 0 & a_{64} & a_{65} & a_{66} & a_{67} \\ 0 & 0 & 0 & a_{74} & a_{75} & a_{76} & a_{77} \end{bmatrix}$$

B=

a_{11}	a_{21}	a_{31}	a_{41}	. . .	a_{47}	a_{57}	a_{67}	a_{77}
0	1	2	3	. . .	27	28	29	30

Fig.7 The results of compression storage at n=7, m=4,r=2

When i=5,j=7 can be calculated to be k=(j-1)*4+(i-1)=(7-1)*4+(5-1)=28.You can see that the element $a_{7,6}$ in the matrix A is just in the position b_{28} in the single-vector B. So the conclusion 2-3 is true.

For a data element $a_{i,j}$ in the uniform block diagonal matrix A, it compressed storage into the single-vector storage space B, if the element of $a_{i,j}$ in A corresponding element is b_k. It can be seen that k is a function of (i,j,m), the function that can be computed:

When m=5, k is a function of (i,j,m):

$$k=f(i,j,m)=(j-1)\times 5+(i-1) \qquad (5)$$

Similarly, in the uniform adjacent block diagonal matrix A, each sub-block matrix is m-order matrix. In addition to the elements of the sub-matrix on the diagonal, the rest are some constants. The element $a_{i,j}$ in the sub-matrix is stored into a single direction now. It has stored j-1 complete columns before the element $a_{i,j}$ stored, obviously. So the element number is m×(j-1). The number of elements at this columns (j raw) before $a_{i,j}$ is (i-1).

Therefore, when the elements in the matrix A are stored in a single vector B, k is a function of (i,j,m): k=(j-1)×m+(i-1).

Then the column priority address mapping function of the uniform adjacent block diagonal matrix we can be obtained. (Formula 6):

$$k= f(i,j,m)= (j-1)\times m+(i -1) \qquad (6)$$

5. Conclusion

In this paper, we studied the compressed storage of uniform adjacent block diagonal matrix for the first time.

We obtained the row priority single-vector compressed storage address mapping function of the uniform adjacent block diagonal matrix formula 1, $k= f(i,j,m)= (i-1)\times m+(j -1)$. We also obtained the column priority single-vector compressed storage address mapping function of uniform adjacent block diagonal matrix formula 2, $k= f(i,j,m)= (j-1)\times m+(i -1)$.

In an n-order matrix with r sub-matrices (m order), it is visible that the data elements are compressed into a single-vector, and the compression ratio CR is:

$$CR=(1-((m^2-1)*r+1)/n^2))*100\%$$

In a 100 order square matrix, there are 10 order square matrices whose number are 11, the compression ratio is:

$$CR= (1-(((100-1)*11+1))/100^2))*100\%=89.1\%$$

These conclusions have a high compression ratio, and we hope to the results provide the basis theoretical for the data compression and storage of the algorithms design for data processing and scientific computing.

Acknowledgement

This paper is a part of the project: Study on compressed storage of the matrixes (NO.2018A-105). The project is financial supported by scientific research projects of colleges and universities in Gansu Province.

If there are have any problems in this paper, please point out and correct the problems.

References

[1] Donald E.Knuth. Fundamental Algorithms, volume 1 of The Art of Computer Programming. Addison- Wesley, 1968.Third edition, 1997.

[2] Donald E.Knuth. Seminumerical Algorithms, volume 2 of The Art of Computer Programming. Addison- Wesley,1969. Third edition, 1997.

[3] Donald E.Knuth. Sorting and Searching, volume 3 of The Art of Computer Programming. Addison- Wesley, 1973.Second edition,1998.

[4] Thomas H.Cormen, Charles E.Leiserson, Ronald L.Rivest, Clifford Stein. Introduction to Algorithm, the third edition. The MIT Press,2009.

[5] Bondy J A and Marty U S R, Graph Theory with Applications, The Macmillan Press Ltd, New York, 1976.

[6] Alfred V.Aho,John E.Hopcroft,and Jeffrey D.Ullman.Data structures and Algorithms. Addison-Wesley, 1983.

[7] Ren Zhiguo, Data Structure(C Language Description), Science Press, Peking China, 2016.

[8] T.C.Hu and M.T.Shing. Computation of Matrix chian products.Part 1, SIAM Journal on Computing, 1982 ,11(2):362-373.

[9] T.C.Hu and M.T.Shing. Computation of Matrix chian products.Part 2, SIAM Journal on Computing, 1984,13(2):228-251.

[10] Mark Allen Weiss. Data Structures and Algorithm analysis in Java. Addison-Wesley, third edition,2007.

[11] Don Coppersmith and Shmuel Winograd. Matrix Multiplication via arithmetic progression. Journal of Symbolic Computation, 1990 ,9(3):251-280.

[12] Langr D, Tvrdik P. Evaluation Criteria for Sparse Matrix Storage Formats[J]. IEEE Transactions on Parallel & Distributed Systems, 2016, 27(2):428-440.

[13] Ordonez C, Zhang Y, Cabrera W. The Gamma Matrix to Summarize Dense and Sparse Data Sets for Big Data Analytics[J]. IEEE Transactions on Knowledge & Data Engineering, 2016 , 28 (7) :1905-1918

[14] Wilson Rodrí, guezCalderó. Fortran application to solve systems from NLA using compact storage[J]. Dyna, 2015, 82(192):249-256.

[15] Nakatsukasa, Y., Soma, T., & Uschmajew, A. (2017). Finding a low-rank basis in a matrix subspace. Mathematical Programming, 162(1-2), 325-361.

[16] Varajão, D., Rui, E. A., Miranda, L. M., & Lopes, J. A. P. (2018). Modulation strategy for a single-stage bidirectional and isolated ac–dc matrix converter for energy storage systems. IEEE Transactions on Industrial Electronics, 65(4), 3458-3468.