Parallel & Distibuted Computing: Lecture 11

Alberto Paoluzzi

November 20, 2018

Concepts and Terminology

1. Sparse Arrays Representations
2. Special structure
3. Sparse Arrays in Julia
4. Special Sparse Arrays in Julia
5. Sparse array linear algebra software
6. Examples

Sparse Arrays Representations

Sparse array definition

Section Source: [From Wikipedia, the free encyclopedia](https://en.wikipedia.org/wiki/Sparse_matrix)

- In numerical analysis and computer science, a sparse matrix or sparse array is a matrix in which most of the elements are zero (dominant element)
- By contrast, if most of the elements are nonzero, then the matrix is dense.
- The number of zero-valued elements divided by the total number of elements (e.g., `m x n` for an $m \times n$ matrix) is called the sparsity of the matrix
- which is equal to 1 minus the density of the matrix.
- Conceptually, sparsity corresponds to systems that are loosely coupled.
- Large sparse matrices often appear in scientific or engineering applications when solving partial differential equations

Sparse array representation

1. Coordinate list (COO)
2. Compressed sparse row (CSR, CRS or Yale format)
3. Compressed sparse column (CSC or CCS)
4. List of lists (LIL)
5. Dictionary of keys (DOK)

Coordinate list (COO)

- COO stores a list of (row, column, value) tuples.
- Entries are often sorted by row index and then by column index, to improve random access
- Format useful for incremental matrix construction.

Coordinate list (COO) Example

```
julia> M = [0 0 3 0 4;
            0 0 5 7 0;
            0 0 0 0 0;
            0 2 6 0 0]

45 Array{Int64,2}:
 0  0  3  0  4
 0  0  5  7  0
 0  0  0  0  0
 0  2  6  0  0

julia> sparse(M)
45 SparseMatrixCSC{Int64,Int64} with 6 stored entries:
...
```

Coordinate list (COO) Example

```
julia> s = sparse(M)
45 SparseMatrixCSC{Int64,Int64} with 6 stored entries:
  [4, 2]  =  2
  [1, 3]  =  3
  [2, 3]  =  5
  [4, 3]  =  6
  [2, 4]  =  7
  [1, 5]  =  4

julia> Matrix(s)
45 Array{Int64,2}:
 0  0  3  0  4
 0  0  5  7  0
 0  0  0  0  0
 0  2  6  0  0
```

Coordinate list (COO) Example

```
julia> findnz(s)
([4, 1, 2, 4, 2, 1], [2, 3, 3, 3, 4, 5], [2, 3, 5, 6, 7, 4])

julia> findnz(s[1,:])
([3, 5], [3, 4])

julia> Matrix(s)
45 Array{Int64,2}:
 0  0  3  0  4
 0  0  5  7  0
 0  0  0  0  0
 0  2  6  0  0
```

Compressed sparse row (CSR, CRS or Yale format)

The compressed sparse row (CSR) or compressed row storage (CRS) format represents a matrix M by three (one-dimensional) arrays, that respectively contain

```
*   nonzero values,
*   the extents of rows, and
*   column indices
```

- It is similar to COO, but compresses the row indices, hence the name
- This format allows fast row access and matrix-vector multiplications (Mv)

CSR Example

Compressed sparse column (CSC or CCS)

CSC is similar to CSR except that values are read first by column, a row index is stored for each value, and column pointers are stored

- CSC is (val, row_ind, col_ptr), where
  - val is an array of the (top-to-bottom, then left-to-right) non-zero values of the matrix;
  - row_ind is the row indices corresponding to the values; and,
  - col_ptr is the list of val indexes where each column starts
- The name is based on the fact that column index information is compressed relative to the COO format
- One typically uses another format (LIL, DOK, COO) for construction
- This format is efficient for arithmetic operations, column slicing, and matrix-vector products
- This is the traditional format for specifying a sparse matrix in MATLAB (via the `sparse` function) and Julia

CSC Example

List of lists (LIL)

LIL stores one list per row, with each entry containing the column index and the value

- Typically, these entries are kept sorted by column index for faster lookup
- This is another format good for incremental matrix construction.[2]

LIL Example

Dictionary of keys (DOK)

DOK consists of a dictionary that maps (row, column)-pairs to the value of the elements

- Elements that are missing from the dictionary are taken to be zero
- The format is good for incrementally constructing a sparse matrix in random order, but poor for iterating over non-zero values in lexicographical order
- One typically constructs a matrix in this format and then converts to another more efficient format for processing

DOK Example

Special structure

Types of special structure

1. Banded
2. Diagonal
3. Symmetric
4. Images

1

[1] see also [*Analysis of Using Sparse Matrix Storage Formats in Image Compression Techniques*](https://acadpubl.eu/jsi/2017-117-15/articles/15/41.pdf)

Banded

The lower bandwidth of a matrix A is the smallest number p such that the entry $a_{i,j}$ vanishes whenever $i > j + p$.

Similarly, the upper bandwidth is the smallest number p such that $a_{i,j} = 0$ whenever $i < jp$ (Golub & Van Loan 1996).

For example, a tridiagonal matrix has lower bandwidth -1 and upper bandwidth 1

```
julia> [1 0 2 0 0; 3 4 0 5 0; 0 6 7 0 8; 0 0 9 10 0; 0 0 0 11
5×5 Array{Int64,2}:
 1  0  2   0   0
 3  4  0   5   0
 0  6  7   0   8
 0  0  9  10   0
 0  0  0  11  12
```

Diagonal

A very efficient structure for an extreme case of band matrices, the diagonal matrix, stores just the entries in the main diagonal as a one-dimensional array,

Hence a diagonal $n \times n$ matrix requires only n entries.

Symmetric

A symmetric sparse matrix arises as the adjacency matrix of an undirected graph

it can be stored efficiently as an adjacency list

Sparse Arrays in Julia

Sparse Arrays

*Sparse Arrays**

Sparse Arrays Documentation

*Docs**

Examples

https://github.com/JuliaStdlibs/SparseArrays.jl

Special Sparse Arrays in Julia

Sparse array linear algebra software

Sparse array linear algebra software

SuiteSparse, a suite of sparse matrix algorithms: GREAT !!

ALGLIB is a C++ and C# library with sparse linear algebra support (no Julia)

PETSc, Portable, Extensible Toolkit for Scientific Computation. A huge C library, contains many different matrix solvers.

Eigen3 is a C++ library that contains several sparse matrix solvers. However, none of them are parallelized.

MUMPS (MUltifrontal Massively Parallel sparse direct Solver), written in Fortran90, is a frontal solver

PaStix PaStiX (Parallel Sparse matriX package) is a scientific library that provides a high performance parallel solver for very large sparse linear systems based on direct methods.

SuperLU General purpose library for the direct solution of large, sparse, nonsymmetric systems of linear equations. Written in C and callable from either C or Fortran. It uses MPI, OpenMP and CUDA to support various forms of parallelism.

aaaaa

Examples

Cholesky Factorization

Gaussian Elimination on Sparse Matrices