MORSy – SPARSE MATRIX ORDERING SOFTWARE

User Guide

Anna Yu. Pirova[1]

[1]Lobachevsky State University of Nizhni Novgorod
23 Gagarin av., 603950, Nizhni Novgorod, Russia
pirova@vmk.unn.ru

Keywords: multilevel nested dissection, sparse matrix ordering, reducing fill-in, vertex separator.

Abstract. *MORSy – a tool for reordering symmetric sparse matrix to reduce its fill-in. It is based on multilevel nested dissection algorithm with modifications for vertex separators.*

1. What is MORSy

MORSy is an open source library for computing fill-reducing ordering of sparse matrix. It can be used when direct methods is used to solve system of linear equations with sparse symmetric matrix. MORSy is based on the multilevel nested dissection algorithm with modifications for using vertex separator at all steps of multilevel scheme [1]. MORSy has been developed at the Department of Computational Mathematics and Cybernetics at the State University of Nizhny Novgorod. It is used in the High Performance Computing Center of the State University of Nizhny Novgorod [2] for solving sparse systems of linear equations in the process of finite element simulation of heart activity.

MORSy is freely available [1]. It is cross-platform and can be used under Linux and Windows operating systems.

2. MORSy installation

MORSy is written in C so it required a C compiler. To use MORSy in your application you need to build it as the library file and to link it to your application. Archive "MORSy_1.0" contains files for building the library on Linux and Windows operating systems.

To link MORSy to your program you need include file *morsy.h* and add dependencies to libMORSy.a (Linux) or morsy.lib (Windows).

2.1 Building on Linux

In order to build MORSy under Linux, you need a GNU make and a C compiler. The folder *MORSy/Linux* contains *Makefile.txt* and *Makefile.inc*. To build MORSy, type

```
$ make
```

To remove compiled binaries, type

```
$ make clean
```

To set configuration options such as compiler and defines, edit *Makefile.inc*. The default compiler is gcc (http://gcc.gnu.org/). For the defines description see Section 2.3.

Compiled binaries libMORSy.a will appear at *MORSy / libmorsy* folder.

2.2 Building on Windows

In order to build MORSy using Microsoft Visual Studio you need Visual Studio version 2008 or later. The folder *MORSy / Windows* contains projects *MORSy.vcproj* and *MORSy.icproj* for Intel C++ Composer.

Compiled binaries MORSy.lib will appear at *MORSy / libmorsy / $(PlatformName) / $(ConfigurationName)* folder.

2.3 Configuration options

MORSy has configuration properties that can be set before compiling the library.

MKL random generator

If you have Intel MKL library or Intel C++ Composer, you can link it for MORSy' internal use for random numbers generation. To take advantage of MKL, you need to define USE_MKL at *morsy.h* file or *Makefile.inc* file and then link MKL to your application.

Our experiments have shown that MKL random numbers generator provides sometimes better orderings then standard `rand()`. It also reduces the run time of reordering.

Printing

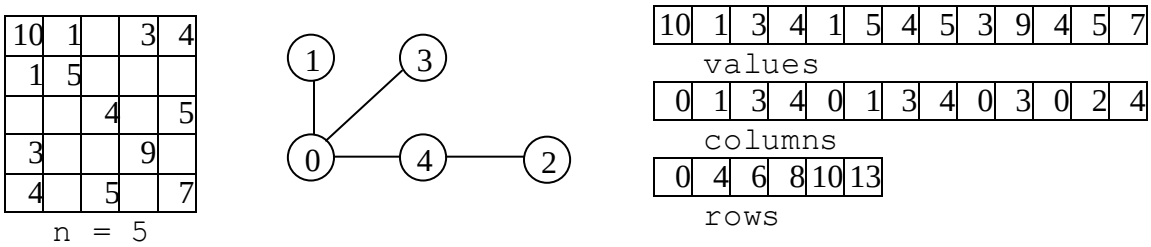
To allow MORSy to print information about its parameters and compression, define PRINT at the *morsy.h* file or *Makefile.inc* file.

3. MORSy interface

3.1 Input data structure

Let $A = (a_{ij})$ be the sparse symmetric n by n matrix. It is assigned to the graph $G = (V, E)$ with the set of vertexes V and the set of edges E where each vertex v_i is associated with matrix row i (i = 1, 1, ..., n), and each edge(v_i, v_j) is associated with non-zero element of matrix, i.e. (v_i, v_j) $\in E$ if and only if $a_{ij} \neq 0$ (i, j = 1, 2, ..., n; $i \neq j$) (Figure 1, *a*).

We assume that matrix A is stored using the Compress Rows Storage (CRS) format and numbering starts from 0. Matrix of size `n` is represented via three arrays: `rows` of size `n + 1`, `columns` of size `rows[n]` and `values` of size `rows[n]`. `Values` array contains all non-zero elements of the matrix have been written row by row, `columns` array contains numbers of column of non-zero elements from array `values`, and elements of row `i` is stored in the arrays `columns` and `values` from index `rows[i]` to index `rows[i + 1] - 1` for `i = 0, 1, ..., n - 1` (Figure 1, *b*). The adjacency lists of the graph G is the same as a pair (`rows`, `columns`) where vertices adjacent to the vertex i is stored in array columns from indexes `rows[i]` to index `rows[i + 1] - 1` for `i = 0, 1, ..., n - 1`.



a) Sparse matrix and its correspondent graph b) Sparse matrix storage

Firure 1. Sample matrix and its storage structure

Ordering routine input is graph represented by adjacency lists were each list is sorted in increasing order. Numbering starts from 0.

3.2 Ordering routine

Synopsis:

```
int MORSY_NestedDissection(int   n,   int*   xadj,   int*   adjncy,
                           int* perm, int* iperm, int* options)
```

Description:

This function computes the ordering of sparse matrix that minimize its fill-in. It is based on multilevel nested dissection algorithm [1].

Function interface is similar to that on METIS_NodeND function from METIS library [4].

Parameters:

- `n` - number of matrix rows or number of vertexes in the graph;
- `xadj`, `adjncy` – structure of sparse matrix or adjacency lists of the graph as they were described in Section 3.1.`xadj` is array of size `n + 1`, `adjncy` is array of size `xadj[n]`.
- `perm` – array of size `n` that contain the inverse fill reducing ordering. If A is the initial matrix and A^* is A after reordering then row `i` of matrix A^* is row `perm[i]` of matrix A.
- `iperm` – array of size `n` that contain fill reducing ordering. If A is the initial matrix and A^* is A after reordering when row `iperm[i]` of matrix A^* is row `i` of matrix A.

Arrays `perm` and `iperm` should have been allocated before calling `MORSY_NestedDissection()`. One of these arrays may be set to NULL, but not both.

- `options` – array of size `MORSY_NUM_OPTIONS` that contains parameters of ordering as they described in Section 0. If `options` is set to NULL, the default parameters will be used.

Return value:

Error code of reordering:
`MORSY_OK` – no errors occurred during reordering,
`MORSY_MEMORY_ERROR` – error occurred while memory was allocated,
`MORSY_INTERNAL_ERROR` – error occurred while reordering was performed,
`MORSY_OPTIONS_ERROR` – input parameters error.

3.3 **Options array**

All parameters of ordering routine is combined into options array. The size of array is specified by `MORSY_NUM_OPTIONS`:

```
int options[MORSY_NUM_OPTIONS];
```

The meanings of its entries are as follows:

- `options[MORSY_IS_COMPRESS]` – specifies that graph structure should be compressed by contracting vertices with the same adjacency lists before applying multilevel nested dissection algorithm. This preprocessing allows to reduce reordering time.
  0 – does not compress graph,
  1 – try to compress graph (the default value).
- `options[MORSY_MATCHING_TYPE]` – specifies the matching algorithm to be used at the coarsening stage.
  0 – random matching,
  1 – heavy edge matching (the default value).
- `options[MORSY_COARSE_TO]` – specifies the minimum number of vertices in the coarsest graph at the coarsening stage and the maximum number of vertices in the graph that will be ordered by automatic nested dissection algorithm instead of multilevel scheme.
  The minimum value is 3, the default value is 20.
- `options[MORSY_COARSERNING_STEPS]` – specifies maximum number of steps during the coarsening stage.

The value must be nonnegative.

- `options[MORSY_DISBALANCE_COEFF]` – specifies the influence of imbalance for the refinement algorithm. Let S, V_1, V_2 be the separator size and the sizes of produced disconnected parts of the graph after removing this separator, respectively. The parameter value determines $k * 100$ in the formula $S * (1 + k * \max(V_1, V_2) / \min(V_1, V_2))$. For example, if $k = 0.2$, then parameter is set to 20.
  The value must be greater than 10 and less than 90. Our experiments shows that the best orderings were obtained with values from 20 to 30. The default value is 20.
- `options[MORSY_REFINEMENT_STEP]` – specifies the step with whom the separator refinement at the uncoarsening stage should be processed.
  1 – at every intermediate graph during,
  2 – every other intermediate graph (the default value).
  Setting this parameter to 2 reduces reordering time but potentially increases matrix fill-in.
- `options[MORSY_IS_CONNECTED]` – specifies if a number of connected components of the graph is known.If there are more than one components, they would be determined and ordered separately.
  0 – graph is not connected (the default value),
  1 – graph is connected.
- `options[MORSY_IS_REFINEMENT_LIMITED]` – specifies limit to the number of iterations of the separator refinement algorithm.
  0 – no limits (the default value),
  1 – limited, not more than one iteration.

If one of the `options` entries is set to unsupported value, it will be reset to the default value.

To use the ordering with the default parameters, you can set `options` array to NULL or use `MORSY_SetDefaultOptions()` function (Section 3.4) . To modify a certain parameters of the algorithm, we recommend you to call `MORSY_SetDefaultOptions()` previosly:

```
int options[MORSY_NUM_OPTIONS];
MORSY_SetDefaultOptions(options);
options[MORSY_IS_CONNECTED] = 1;
options[MORSY_MATCHING_TYPE] = 0;
...
```

3.4 **Parameter setting routines**

MORSy provides functions for defining specific sets of parameters: default set, parameters set for maximizing the quality of ordering, parameters set for minimizing reordering time when the quality of ordering is acceptable. All configurations were adjusted during our experiments [2] with matrixes from The University of Florida Sparse Matrix Collection [5]. Thus, these configurations are not universal but appropriate for the most of matrices. You can use an arbitrary parameter setting function of this kind.

Synopsis:

```
void MORSY_SetDefaultOptions(int* options)
```

Description:

This function set the default values to the reordering parameters.

Parameters:

- `options` – array of size `MORSY_NUM_OPTIONS` that contains parameters of ordering as they described in Section 3.3.

Synopsis:

```
void MORSY_SetTimePriorityOptions(int* options)
```

Description:

This function set reordering parameters for minimizing the time of finding the ordering with the acceptable fill.

Parameters:

- `options` – array of size `MORSY_NUM_OPTIONS` that contains parameters of ordering as they described in Section 0.

Synopsis:

```
void MORSY_SetQualityPriorityOptions(int* options)
```

Description:

This function set reordering parameters for maximizing the quality of ordering, i.e. for minimizing number of non-zero elements in the matrix factor.

Parameters:

- `options` – array of size `MORSY_NUM_OPTIONS` that contains parameters of ordering as they described in Section 0.

4. Examples

```
#include "morsy.h"

/* Input parameters:
* n - matrix size
* values, columns, rows represent CRS structure of the matrix
* values  - non-zero elements have been written sequentially row by row
* columns - number of column of each element from values array
* rows    - indexes indicated starting index of elements of row i
*           in the arrays values and columns
* iperm   - pointer to array contains a new permutation
*  perm   - pointer to array contains inverse permutation
*/

int MatrixReorderingDefault(int n, int* columns, int* rows, double* values,
                            int** iperm, int** perm)
{
  int error; // error code
  // iperm must be allocated before reordering
  *iperm = (int*)malloc(sizeof(int)*n);
  // perm must be allocated before reordering
  *perm  = (int*)malloc(sizeof(int)*n);

  // running MORSy with the default options
  error = MORSY_NestedDissection(n, rows, columns, *perm, *iperm, NULL);
  return error;
}
```

```
// running MORSy with the "quality-priority" options
// perm, ipers were allocated before this function
int MatrixReorderingBestQuality(int n, int* columns, int* rows,
                                double* values, int* iperm, int* perm)
{
  int options[MORSY_NUM_OPTIONS];
  int error; // error code

  MORSY_SetQualityPriorityOptions(options);
  error = MORSY_NestedDissection(n, rows, columns, perm, iperm, options);
  return error;
}

// running MORSy with user parameters
int MatrixReorderingUserOptions(int n, int* columns, int* rows,
                                double* values, int* iperm, int* perm)
{
  int options[MORSY_NUM_OPTIONS];
  int error; // error code

  MORSY_SetDefaultOptions(options);     // set options by default values
  options[MORSY_IS_CONNECTED] = 1;      // graph of matrix is connected
  options[MORSY_COARSERNING_STEPS] = 5; // not more then 5 steps at the
coarsening stage
  error = MORSY_NestedDissection(n, rows, columns, perm, iperm, options);
  return error;
}

// running MORSy with modified time-priority parameters
// perm permutation isn't necessary
int MatrixReordering TimePriorityOptions(int n, int* columns, int* rows,
                                double* values, int* iperm)
{
  int options[MORSY_NUM_OPTIONS];
  int error; // error code

  MORSY_SetTimePriorityOptions(options); // set time-priority options
  options[MORSY_IS_CONNECTED] = 1;       // graph of matrix is connected
  options[MORSY_DISBALANCE_COEFF] = 30;  // set disbalance coefficient to 0.3
                                         // for refinement stage
  error = MORSY_NestedDissection(n, rows, columns, NULL, iperm, options);
  return error;
}
```

5. License and contact information

The MORSy library may be used under the terms of the GNU Lesser General Public License version 2.1 as published by the Free Software Foundation. Please review the following information to ensure the GNU Lesser General Public License version 2.1 requirements will be met: http://www.gnu.org/licenses/old-licenses/lgpl-2.1.html.

This software is provided as is, with absolutely no warranty expressed or implied. Any use is at your own risk.

If you have any problems, send email to pirova@vmk.unn.ru with the brief description of the problem.

6. References

[1] MORSy - Sparse Matrix Ordering Software for minimizing fill-in: http://hpc-education.unn.ru/research/overview/sparse-algebra/morsy

[2] Bastrakov S. , Meyerov I. , Gergel V. et al. High Performance Computing in Biomedical Applications. Procedia Computer Science, 18, 10–19, 2013.

[3] Pirova A., Meyerov I. MORSy – a new tool for sparse matrix reordering // An International Conference on Engineering and Applied Sciences Optimization (Kos Island, Greece, 4-6 June 2014). (accepted).

[4] Karipis G. METIS. A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices. Version 5.0. Technical report, University of Minnesota, Department of Computer Science and Engineering, 2011. URL: [http://glaros.dtc.umn.edu/gkhome/fetch/sw/metis/manual.pdf].

[5] The University of Florida Sparse Matrix Collection: http://cise.ufl.edu/research/sparse/matrices/.