PMORSy – PARALLEL SPARSE MATRIX ORDERING SOFTWARE FOR SHARED-MEMORY SYSTEMS

User Guide

Anna Yu. Pirova[1]

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

Keywords: multilevel nested dissection, sparse matrix ordering, reducing fill-in, shared-memory systems.

Abstract. *PMORSy is a parallel library for reordering symmetric sparse matrix to reduce its fill-in for shared-memory systems. It is based on the multilevel nested dissection algorithm with modifications for vertex separators.*

1. What is PMORSy

PMORSy [2] is an open source library for computing fill-reducing ordering of sparse symmetric matrices. It is a parallel version of MORSy library [3] for shared-memory systems. MORSy and PMORSy are based on the multilevel nested dissection algorithm with modifications for using vertex separator at all steps of multilevel scheme. Parallel processing is done in a task-based fashion and implemented using OpenMP 3.0 technology.

PMORSy is freely available from the web-page [1]. It is cross-platform and can be used under Linux and Windows operating systems.

2. PMORSy installation

PMORSy 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 "PMORSy_1.x.y" contains files for building the library on Linux and Windows operating systems.

To link PMORSy to your program you need include file *pmorsy.h* and add dependencies to libPMORSy.a (Linux) or pmorsy.lib (Windows).

2.1 Building on Linux

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

2.2 Building on Windows

In order to build PMORSy using Microsoft Visual Studio you need Visual Studio version 2010 or later. The folder *PMORSy / Windows* contains projects *PMORSy.vcproj* and *PMORSy.icproj* for Intel C++ Composer.

Compiled binaries pmorsy.lib will appear at *PMORSy / libpmorsy / $(PlatformName) / $(ConfigurationName)* folder.

2.3 Configuration options

PMORSy 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 PMORSy' internal use for random numbers generation. To take advantage of MKL, you need to define `MORSY_USE_MKL` at *pmorsy.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 significantly.

Printing

To allow PMORSy to print information about its parameters and compression, define `MORSY_PRINT` at the *pmorsy.h* file or *Makefile.inc* file.

Time information

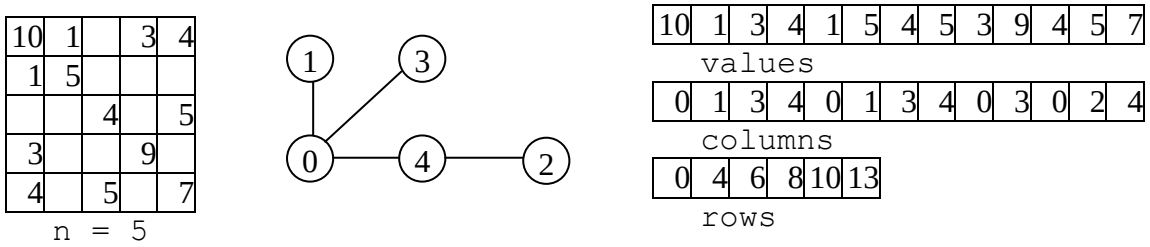
To allow PMORSy to print ordering routines time, define `MORSY_TIMERS` at the *pmorsy.h* file or *Makefile.inc* file.

3. PMORSy interface

3.1 Input data structure

Let $A = (a_{ij})$ be a sparse symmetric n by n matrix. It is assigned tothe graph $G = (V, E)$ withthesetofvertexes$V$andthesetofedges$E$where each vertex $v_i$is associated withmatrix 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) ∈Eif 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 matrixhave been written row by row, `columns` array contains numbers of column of non-zero elements from array `values`, and elementsof 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 adjacencylists 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 CRS storage structure.

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

3.2 Ordering routine

Synopsis:

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

Description:

This function computes the ordering of sparse matrix that minimizes 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.

Parameters:

- `n` – a number of matrix rows or number of vertexes in the graph;
- `xadj`, `adjncy` – a structure of sparse matrix or adjacency lists of the graph as they were described in Section 3.1.`xadj` is an array of size `n + 1`, `adjncy` is an array of size `xadj[n]`.
- `perm` – an array of size `n` that contains the inverse fill reducing ordering. If*A*is the initial matrix and A^* is A after reorderingwhenrow `I` of matrix $A^*$is row `perm[i]`of matrix A.
- `iperm` – an array of size `n` that contains 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 asthey described in Section3.3. 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_ERROR` – an error occurred during reordering.

3.3 **Options array**

All parameters of ordering routine are 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_DEFAULT]`– specifies if the default options would be used.
  0 – use user defined options,
  1 – use the default options.
- `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),
  2 – heavy clique matching.
- `options[MORSY_PARTITIONING_TYPE]` – specifies the algorithm for initial partitioning stage.

0 – rooted level structure from pseudoperipheral vertex (the default value),
1 – growing partition,
2 – greedy growing partition,
3 – Kernigan-Lin partition started from growing partition.

If values 1, 2 is used, PMORSy will construct 10 different partitions and select the best one. If value 3 is used, PMORSy will construct 5 different partitions and select the best one.

- `options [MORSY_REFINEMENT_TYPE]` - specifies the type of refinement procedure that will be used.

  0 – prosedure whith additional constraint for the refinement stage (the default value). In this case, `options[MORSY_REFINEMENT_STOP]` is used to limit the number of passes during the iterative refinement algorithm.

  1 – the basic refinement algorithm. In this case, `options[MORSY_REFINEMENT_STOP]` is not used.

  Setting this parameter to 0 reduces reordering time but potentially increases matrix fill-in.
- `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 100.
- `options[MORSY_COARSERNING_STEPS]` – specifies maximum number of steps during the coarsening stage.
  The value must be nonnegative. The default value is 10.
- `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 needed to be set to 20.
  The value must be greater than 10 and less than 90. Our experiments show that the best orderings were obtained with the 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 (the default value),
  2 – at every other intermediate graph.
  Setting this parameter to 2 reduces reordering time but potentially increases matrix fill-in.
- `options[MORSY_IS_CONNECTED]` – specifies ifa number of connected components of the graph is known. If graph is not connected, the components of the graph would be determined and ordered separately.
  0 – the graph is not connected (the default value),
  1 – the graph is connected.
- `options[MORSY_IS_REFINEMENT_LIMITED]` – specifies a limit to the number of outer loop iterations of the separator refinement algorithm.
  0 – no limits (the default value),
  1 – limited, not more than one iteration.
  Setting this parameter to 1 reduces reordering time but potentially increases matrix fill-in.
- `options[MORSY_REFINEMENT_STOP]` – specifies a maximum disbalance ratio that is reached during the refinement procedure. The value must be greater than 10 and less than 50.The parameter value means $max(V_1, V_2) / min(V_1, V_2) * 10$. For example, the value 12 allows the ratio 1.2. The larger values tend to the orderings with better qualities. The smaller values allow to reduce reordering time. The default value is 15.

- `options[MORSY_REFINEMENT_CUT]` - specifies a maximum allowed negative gain value in the refinement algorithm. The parameter value determines k in the formula $(1 - \max_{v \in V} \sum_{(u,v) \in E} w(u))/k$. The value must be between 1 and 3. This parameter influences to the number of passes of the inner loop of the refinement algorithm. The larger values of the parameter reduce number of passes, and, thus reduce reordering run-time. The default value is 3.
- `options[MORSY_NUM_THREADS]` – specifies the number of OpenMP threads that will be used for the ordering. The default value is 4.
- `options[MORSY_BARRIER]` – specifies the maximum size of graph that would be ordered sequentially. The default value is 1000.

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()` previously:

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

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 with matrices from The University of Florida Sparse Matrix Collection [3]. 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-in.

Parameters:

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

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 3.3.

4. Examples

```
#include "pmorsy.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 = PMORSY_NestedDissection(n, rows, columns, *perm, *iperm, NULL);
  return error;
}

// running MORSy with the "quality-priority" options
// perm, iperm were allocated before this function
intMatrixReorderingBestQuality(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 = PMORSY_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] = 15; // not more than 15 steps at the
coarsening stage
  error = PMORSY_NestedDissection(n, rows, columns, perm, iperm, options);
  return error;
}

// running MORSy with modified time-priority parameters
// perm permutation isn't necessary
int MatrixReorderingTimePriorityOptions(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 is connected
  options[MORSY_DISBALANCE_COEFF] = 30;  // set disbalance coefficient to 0.3
  // for refinement stage
  error = PMORSY_NestedDissection(n, rows, columns, NULL, iperm, options);
  return error;
}
```

5. License and contact information

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

6. References

[1] MORSy web-page: http://hpc-education.unn.ru/en/research/overview/sparse-algebra/morsy

[2] Pirova A., Meyerov I., Kozinov E., Lebedev S. PMORSy: parallel sparse matrix ordering software for fill-in minimization // Optimization Methods and Software. – 2016.

[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).

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