MinDistance DM

Basic installation and user guide

Nov 14, 2022

MinDistance DM is a software package with several fast parallel implementations for computing the minimum distance of a random linear code on distributed-memory architectures.

This guide complements the submitted paper *Parallel Implementations for Computing the Minimum Distance of a Random Linear Code on Distributed-memory Architectures* and includes basic setup and execution steps for the package.

The provided software package includes all the algorithms described in the paper (described in detail in Section 3 of the manuscript).

Basic installation steps

Configuration and testing

The configuration of *MinDistance DM* comprises only one steps:

```
# Step 1. Compilation
cd mpi
make
```

Testing correctness

After compilation, it is highly recommended to run a simple sanity check to validate the installation. A script with name `check_smallset_a.sh` is provided in the `mpi` folder to validate the correctness of the solutions using small matrices.

The expected output of a correct execution is:

```
./check_smallset_a

Testing file:  ../mat_smallset_a/atest001_4x4x0.in     Ok
Testing file:  ../mat_smallset_a/atest002_5x3x5.in     Ok
Testing file:  ../mat_smallset_a/atest003_7x3x4.in     Ok
Testing file:  ../mat_smallset_a/atest004_15x5x7.in     Ok
Testing file:  ../mat_smallset_a/atest005_9x4x2.in     Ok
Testing file:  ../mat_smallset_a/atest006_10x4x3.in     Ok
Testing file:  ../mat_smallset_a/atest007_11x4x3.in     Ok
Testing file:  ../mat_smallset_a/atest008_12x4x2.in     Ok
Testing file:  ../mat_smallset_a/atest009_14x7x2.in     Ok
Testing file:  ../mat_smallset_a/atest010_15x7x3.in     Ok
Testing file:  ../mat_smallset_a/atest011_16x7x4.in     Ok
Testing file:  ../mat_smallset_a/atest012_17x7x3.in     Ok
Testing file:  ../mat_smallset_a/atest013_18x7x5.in     Ok
Testing file:  ../mat_smallset_a/atest014_40x20x6.in     Ok
Testing file:  ../mat_smallset_a/atest015_100x70x7.in     Ok
Testing file:  ../mat_smallset_a/atest016_94x70x5.in     Ok
Testing file:  ../mat_smallset_a/atest017_90x63x6.in     Ok
Testing file:  ../mat_smallset_a/atest018_84x47x8.in     Ok
Testing file:  ../mat_smallset_a/atest019_73x47x7.in     Ok
Testing file:  ../mat_smallset_a/atest020_90x50x9.in     Ok
Testing file:  ../mat_smallset_a/atest021_50x20x8.in     Ok
Testing file:  ../mat_smallset_a/atest022_30x10x7.in     Ok
```

```
Testing file:  ../mat_smallset_a/atest023_23x10x6.in     Ok
Testing file:  ../mat_smallset_a/atest024_100x15x33.in     Ok
Testing file:  ../mat_smallset_a/atest025_277x15x110.in     Ok
Testing file:  ../mat_smallset_a/atest026_100x20x28.in     Ok
Testing file:  ../mat_smallset_a/atest027_115x20x33.in     Ok
Testing file:  ../mat_smallset_a/atest028_154x20x50.in     Ok
Testing file:  ../mat_smallset_a/atest029_153x20x50.in     Ok
Testing file:  ../mat_smallset_a/atest030_600x20x244.in     Ok
Testing file:  ../mat_smallset_a/atest031_20x10x1.in     Ok
```

Usage

The name of the generated executable is `test_crc.x`. It can be invoked as:

`mpirun -np NUM_PROCS ./test_crc.x <matrix_name>`

where `<matrix_name>` is the name of the generator to use.

Execution configuration

The configuration file with name `config.in` must reside in the current directory. Otherwise, the application will abort with an error message. This text file contains the parameters of the execution to launch.

A sample `config.in` is provided in the package. The following is a sample of the contents of the file:

```
1      Distributed alg. (1,2,3,4: dyn_abp; 5,6,7,8: sta_rep)
3 2 1  Prefix length for distributed algorithms
4 5    Node engine (1:bas, 2:opt, 3:sta, 4:sav, 5:sav+unr)
5 4    Number of saved generators stored in RAM (only for saved node engines)
1 2    Number of threads used in every process
10     Number of permutations to perform in the diagonalization (>=1).
0      Print matrices (0=no,1=yes)
```

For each row, only the first integer number is considered; the rest of the line is just ignored.

The following are some remarks about each line in the configuration file:

1. Distributed algorithm selection, each one corresponding to one of the algorithms describe in Sections 3.2 (for dynamic algorithms) and 3.3 (for static algorithms) in the manuscript:
   - **Dynamic algorithm. D-Lex (1)**.
   - **Dynamic algorithm. D-Lex-2cm (2)**.
   - **Dynamic algorithm. D-Lex-Lle (3)**.
   - **Dynamic algorithm. D-Lex-Lle-2cm (4)**.
   - **Static algorithm. S-Lex (5)**.
   - **Static algorithm. S-Lex-Snc (6)**.
   - **Static algorithm. S-Lle (7)**.
   - **Static algorithm. S-Lle-Snc (8)**.
2. Prefix length.
3. Node engine, described in detail in Section 2.2.1 of the manuscript:
   - **bas (1)**: Basic algorithm.
   - **opt (2)**: Optimized algorithm.
   - **sta (3)**: Stack-based algorithm.
   - **sav (4)**: Algorithm with saved additions.
   - **sav+unr (5)**: Algorithm with saved additions and unrollings.
4. Number of saved generation (if using saved node engines).
5. Number of threads used in every process.
6. Number of permutations to perform in the diagonalization (must be greater than 1).

Generators provided for testing purposes

- *mat_smallset_a*: A set of small code generators that can be employed for checking the installation. This set includes several generator dimensions and lengths. Depending on the architecture and the number of cores, the processing of all these generators usually takes a few minutes on the best algorithms.
- *mat_smallset_b*: Same as before, but the generators are a bit larger, and hence the processing can take longer.
- *mat_perf*: A set of large generators. The processing of one generator can take from several minutes to several weeks.

Input generator file format

Generators are provided as input data and follow a strict data format. Users must be sure they correctly prepare and format the input matrices following one of the supported formats, namely:

- Format 1:

```
3 7   matrix dimensions
 1 0 0 1 0 1 1
 0 1 0 1 1 0 1
 0 0 1 0 1 1 1
```

- Format 2:

```
3 7   matrix dimensions
[ [ 1, 0, 0, 1, 0, 1, 1 ],
  [ 0, 1, 0, 1, 1, 0, 1 ],
  [ 0, 0, 1, 0, 1, 1, 1 ] ]
```

The generators included in the package usually follow a common name scheme to easily identify them:

`CODENAME_NxKxDIST.in`

When DIST (distance) is unknown, a `_` character is used.

Sample execution and output

The output of a sample execution with the configuration file described above would include the following information:

Execution configuration information

Includes process information, matrix information and configuration details extracted from `config.in`. A sample is shown below:

```
Process 1 executing on machine: figual-asus
Process 0 executing on machine: figual-asus
Matrix dimensions:  7 x 17
Chars in file after reading matrix:
Trailing char: ' '
Trailing char: '
'
End of chars in file after reading matrix.
Read input matrix. Elapsed time (s.): 0.000388

% Distributed algorithm:                1
% Distributed prefix length:            3
% Node engine:                          4
% Number of saved generators:           5
% Number of threads per process:        1
% Number of permutations:               10
% Print matrices:                       0
```

Gamma matrices creation information

Includes, if configured, information about the creation of data structures for saving combinations. A sample is shown below:

```
Creating vector of Gamma matrices.
End of creating vector of Gamma matrices.
Creating data structures for saving combinations.
Generators:  1  Combinations:          7  Required accum.mem.(MB): 0.0
Generators:  2  Combinations:         21  Required accum.mem.(MB): 0.0
Generators:  3  Combinations:         35  Required accum.mem.(MB): 0.0
Generators:  4  Combinations:         35  Required accum.mem.(MB): 0.0
Generators:  5  Combinations:         21  Required accum.mem.(MB): 0.0
End of creating data structures for saving combinations.
```

The larger generators are processed, the larger the total accumulated memory will be. The user must check that this required space is not too large. The number of saved generators is defined in the `config.in` file.

Permutations and diagonalization

Includes information about the diagonalization stages. A sample is shown below:

```
Performing several matrix diagonalizations...
Creating vector of Gamma matrices.
End of creating vector of Gamma matrices.
Diagonalizing permuted matrix 0
Matrix diagonalized.
Proc: 0  Rank vector: 7 7 3

Diagonalizing permuted matrix 1
Matrix diagonalized.
Proc: 0  Rank vector: 7 7 3

Diagonalizing permuted matrix 2
Matrix diagonalized.
Proc: 0  Rank vector: 7 7 3

Diagonalizing permuted matrix 3
Matrix diagonalized.
Proc: 0  Rank vector: 7 7 3

Diagonalizing permuted matrix 4
Matrix diagonalized.
Proc: 0  Rank vector: 7 7 3

Diagonalizing permuted matrix 5
Matrix diagonalized.
Proc: 0  Rank vector: 7 7 3

Diagonalizing permuted matrix 6
Matrix diagonalized.
Proc: 0  Rank vector: 7 7 3

Diagonalizing permuted matrix 7
Matrix diagonalized.
Proc: 0  Rank vector: 7 7 3

Diagonalizing permuted matrix 8
Matrix diagonalized.
Proc: 0  Rank vector: 7 6 4

Diagonalizing permuted matrix 9
Matrix diagonalized.
Proc: 0  Rank vector: 7 6 3
```

```
Proc: 0 Best rank vector: 7 7 3

Erasing vector of Gamma matrices.
Finished matrix diagonalizations. Elapsed time (s.): 0.01
```

As can be seen, the application tests several permutations in order to keep the one with the largest ranks in the Gamma matrices.

Detailed distance calculation process

Includes information about the iterative process to find the minimum distance. A sample is shown below:

```
MEMORY_ALIGNMENT:                   16
Dimensions of input matrix:         7 x 17
Dimensions of compacted Gamma mats: 21 x 1
Distance Loop. lowerDist: 2  upperDist: 17
kInput:            7
numGammaMatrices:  3
  Generators: 1
  numGMatProcessed: 2
  numGMatContrib:   3
    Gamma Matrix: 1 of 3
    generate_with_alg_saved
    process_prefix_with_alg_saved
    fill_structures_for_saved_data_for_1_s
    generate_with_saved_recursive with cores: 1  0 1
    generate_with_saved_1_s
    End Gamma. lower:  3  upper:  3        Elapsed time (s.): 0.01

  End Combin.  lower:  3  upper:  3        Elapsed time (s.): 0.01

End Distance.  lower:  3  upper:  3        Elapsed time (s.): 0.01

Erasing vector of Gamma matrices.
Erasing data structures for saving addition of combinations.
Computed distance: 3     Elapsed time (s.): 0.01



Distance of input matrix:  3
% Process: 0  End of Program
% Process: 1  End of Program
```