# Manual for MT-SDREM

Siddhartha Jain `sj1@cs.cmu.edu`

## Requirements and General instructions

- You must have Java 5.0 or above installed on your machine.
- Properties files are used to specify the paramaters and input data.
  See the DREM manual (`http://sb.cs.cmu.edu/drem/DREMmanual.pdf`) for details regarding the gene expression and protein-DNA binding file formats. The TF-gene priors have the same format as the DREM protein-DNA binding data file.
- Examples of how to run MT-SDREM on a cluster are included. The maximum Java heap size must be increased when working with large networks.
- Many intermediate output files are generated during each iteration of MT-SDREM. The M-TSDREM executables, code, input files, output produced, and example data are described below.

## Main steps

Throughout this manual, we assume that the MT-SDREM properties file is called `mtsdrem.props`.

1. Precompute and store paths as explained below
2. Configure the mtsdrem.props file. A description of how to configure it is provided below.
3. Run using the command. Note that running it might take a few hours. Intermediate results are stored in the `model` directory which is specified in `mtsdrem.props`.
   ```
   java -jar mtsdrem.jar <path to mtsdrem.props>
   ```

## Data needed

1. A protein protein interaction network in the format

   ```
   <protein> <interaction type> <protein> <probability>
   ⋮
   ```

   The columns have to be separated by the **tab** character. The `<interaction type>` can be one of `pp`(protein-protein), `ptm`(post-translational modification), and `pd`(protein-dna). The latter two are *directed* whereas `pp` is *undirected*. A sample protein-protein interaction network with gene names under the HGNC symbol naming scheme is provided in the file `ppi_ptm_pd_hgnc.txt`.
2. A list of transcription factors (TFs). A sample list containing 348 TFs is provided.

3. A list of TF-gene interactions in the format

   ```
   TF Gene Prior
   <TF> <Gene> <Prior>
   ⋮
   ```

   Again, the columns have to be tab separated. The prior can also be given the value `D` in which case, the value of the field `default.TF.prior` specified in `mtsdrem.props` is used.

4. Time series gene expression data for the conditions being examined. This should be in the same format as required by DREM. For DREM 2.0, the format is

   ```
   Gene tp1 tp2 ...
   <Gene> <expr for tp1> <expr for tp2> ...
   ⋮
   ```

5. A list of source proteins for each condition being examined in the format

   ```
   <protein>
   <protein>
   ⋮
   ```

## Precompute and store paths

Running MT-SDREM requires you to search for and store the top $k$ simple (non-cyclic) paths from the source proteins for a particular condition (infection by virus, heat stress, etc.) to all the transcription factors (TFs) in the database. This is a **mandatory** step. Below we describe the relevants files to configure and run the path search.

- `StorePaths.jar` - The executable for preprocessing the network data. It searches for paths from the sources to each TF and writes them all to disk, optionally filtering them to keep only the highest confidence paths.
- `allPaths.props` - An example properties file for StorePaths.jar. store/filter means the user wants to enumerate paths and remove low-confidence paths (recommended for large PPI networks). The next lines define the sources, targets, and networks. Node priors are used to give weights to vertices in the network. If the user stores and filters paths, then they can delete the intermediate output in stored.paths.dir once the filtering step is done (just give the final directory with the filtered paths to MT-SDREM as input).
- `allPaths.qsub`- An example showing how to call StorePaths.jar. This is a submission script for a PBS cluster and the last line shows the actual command.

## MT-SDREM

After computing the top $k$ paths, we're now ready to run the MT-SDREM algorithm.

- `mtsdrem.jar` - The MT-SDREM executable
- `mtsdrem.props` - The MT-SDREM properties file

– `edges.file`: Path to the protein protein interaction network file

– `sources.file`: Paths to the sources files for the conditions being examined. The paths to the different sources files should be tab separated. **Important:** From here on, the conditions are numbered starting from 0 **in the order** that the sources files are specified.

– `cond.class`: The "class" of the condition. There should be an integer value for each condition. Two conditions that are related should have the same integer value. The order of classes for the conditions should be same order the sources file have been specified.

– `tf.share.factor`: This is the $\alpha$ factor as specified in the paper.

– `model.dir`: The directory into which the output will be written. This is also the directory in which the TF-gene interaction files and the DREM config files for the different conditions are stored.

– `drem.settings.file`: The *prefix* to the DREM config file. If the prefix is `DREM_defaults`, for example, then the DREM config file for condition 0 will be `DREM_defaults_cond_0.txt`, for condition 1, it will be `DREM_defaults_cond_1.txt`, etc.

– `binding.priors.file`: The *prefix* to the TF-gene interaction file. If the prefix is `human_predicted100_`, for example, then the TF-gene interaction file for condition 0 will be `human_predicted100_cond_0.txt` and so on.

– `stored.paths.dir`: The path to the folder containing the stored paths as generated using `StorePaths.jar`. There should be only **1** path. If you have multiple conditions, you can simply concatenate the paths of all the different conditions together. The output file format from `StorePaths.jar` is a gunzipped `txt` file (`txt.gz`) for each TF so simple concatenating the files together using `cat` (in unix systems) or some other utility should work.

– `iterations`: The number of iterations to run the MT-SDREM algorithm for. We've found a value of 10 to work well though frequently a somewhat smaller value also gives good results.

– `path.enum.bound`: The number of top paths MT-SDREM should use (out of all the stored paths). A value of -1 means that all paths should be used.

– `orientation.restarts`: The number of times the orientation should be run with random TFs for the purposes of normalizing the TF scores. This value corresponds to the $L$ variable in the Supplemental Methods description fo the algorithm. We've found a value of 5 to work well.

– `drem.random.runs`: DREM needs to be run with randomized binding data multiple times to get an accurate assesment of which TFs are part of the regulatory network. We've found a value of 5 to work well. See the [1] for details.

– `dist.tfs`: This is the number of top-ranked TFs used to build the activity score distribution which is used to select TF targets for the orientation phase. See the [1] for details.

– `top.paths`: The number of top-ranked paths to consider when generating node scores as per the path flow measure specified in the paper.

– `dist.thres`:Only TFs that fall at or above this percentile in the activity score distribution are considered to be targets. A value of 0.50 should work. See the [1] for details.

– `target.thresh`: The TF score threshold in the oriented network for increasing or decreasing the binding priors. A value of 0.80 means the only TFs scoring above the 80th percentile of the score distribution are seleceted. See step 7 in the "Detailed description of the algorithm" section in Supplemental Methods of the accompanying paper for details. The value of 0.80 usually works.

– `node.thresh`: The node threshold parameter specified in step 7 in Supplemental Methods. Any TFs with percent path flow above the node threshold have their prior also increased for the next iteration.

  - `default.node.prior`: The default node prior for nodes in the path. The node prior for a node is multiplied with the path score. Thus the main effect of a smaller node prior is to give lower weight to longer paths. We've found a wide variety of node priors ranging from 0.50 to 0.95 to give good results.
  - `min.prior`: This is the minimum value the TF-gene binding prior can be reduced to. Generally set to 0.01.
  - `random.target.ratio`: The ratio to random TFs to real TFs when running network orientation with random TFs to generate TF scores. See step 4 of the algorithm.
  - `default.TF.prior`: The default TF binding prior
  - `startIteration`: The iteration to start at (used if you've already completed a few iterations and need to do a few more but don't want to start from the beginning). Thus if the startIteration is 2, then that means you've already finished 1 iteration and want to start the 2nd.
  - `opt.convergence.ratio`: The convergence ratio for the orientation procedure. 0.001 should work.
  - `DEBUG/XDEBUG/WarmDremStart`: DEBUG options. Should be set to false.

- `DREM_defaults_cond_i.txt`: The DREM config file for the *i*th condition. It's the same format that the original DREM software uses (as described in `http://sb.cs.cmu.edu/drem/DREMmanual.pdf`) except the `TF-gene_Interactions_File` will be generated dynamically and thus should be left blank, `TF-gene_Interaction_Source` should be set to `User Provided` and Active_TF_influence is an SDREM parameter.

- `mtsdrem.qsub`: An example showing how to call mtsdrem.jar on a PBS cluster.

## Output files

Intermediate output files are generated at every iteration of MT-SDREM, which can be used for debugging and analysis. At each iteration 'N', MT-SDREM writes the following files (some filenames may vary slightly based on the parameters used):

- `/N`: A subdirectory that contains output from the DREM runs that use randomized TF binding data. These can be ignored or deleted.

- `N.model`: The DREM model file for iteration N, which can be loaded by `drem.jar`.

- `N.model.activities`: Activity information for the TFs used to determine which TFs should be targets.

- `N.model.activitiesDynamic`: Activity information for the TFs used to determine which TFs should be targets.

- `N.model.activitiesStd`: Activity information for the TFs used to determine which TFs should be targets.

- `N.targets`: The TFs that were selected as targets at iteration N and their target weights. These are the proteins that were used for the network orientation.

- `N.targetsStd`: Like N.targets but contains information about the distribution of random TF activity scores.

- `itrN.out`: A log file

- `nodeScores_itrN_rankingMetric_orientation.restarts_top.paths_cond_i.txt`: The file containing the percent path flow through all the nodes for condition `i`. The path flow is recorded for both the entire set of paths being used, as well as for the number of top-ranked paths specified in the config parameter `top.paths`. The `rankingMetric` is usually `PathWeight`. `N` is the `N`th MT-SDREM iteration.
- `edgeDir_itrN.txt`: The inferred edge orientation for all the edges. It's in the format

  ```
  <protein A> <protein B> <0 or 1>
  ⋮
  ```

  If it is 1, that means the direction is from A to B otherwise from B to A. There is only one file for all conditions as the algorithm maintains a consistent edge direction across conditions.
- `scores_maxflow_itrN_r1.0_PathWeight_10_1000.txt`: Connectivity of the true target TFs in the network orientation versus randomly selected targets, which is used to determine which TFs' priors should be increased or decreased at the next iteration of MTSDREM.
- `human_predicted100_N_cond_i.txt`: The new TF-gene binding file that will be used as input at iteration N+1 for condition `i`. Here `human_predicted100_` is the *prefix* to the TF-gene binding files as specified in `mtsdrem.props`.

## Modified DREM

The DREM software (`http://sb.cs.cmu.edu/drem/`) that MTSDREM was built upon allows visualization of the active TFs and gene expression profiles after MTSDREM is run. There are several differences between the version distributed here and DREM 2.0: None of the new features descrbied in the DREM 2.0 manuscript are present yet (e.g. support for motif finding).

To view the final output (assuming 10 iterations), should load `10_cond_i.model` as the saved model (where i is the condition number starting at 0) and `<binding.priors>9_cond_i.txt` the TF-gene interactions where `<binding.priors>` is the binding priors file prefix as specified by in `mtsdrem.props`. The split table will show the activity score for each TF at that split and the max activity score across all splits. Key TF Labels includes options to display TFs at each split based on activity score. If the user uses activity scores to choose which TFs to show, the slider will be used to calculate $10^X$ (instead of $10^{-X}$) and all TFs with activity scores greater than $10^X$ at a split are shown. The output file `10.targetsStd` can be used to help choose the activity score threshold. The last column in this file gives the max activity score for each active TF. Therefore, the user can find the minimum of these values and use it (or a value slightly less than it) as the threshold for display purposes. There is also an option to show the TF only at the earliest split at which it exceeds this threshold on each path. When using activity score to display TFs, there will be [1] or [2] after the TF id if binary splits were used. [1] means the TF primarily controls the lower path out of the split and [2] is for the higher path. For higher order splits (3+), [1] is the lowest path out of the split and [3] is the highest.

`drem.jar` is the DREM executable, which can be run without command line parameters. Use the GUI to load the model file from MT-SDREM and the data. Optionally provide the `DREM_defaults_cond_i.txt` (for condition `i`) file at the command line to load the same settings that MT-SDREM used.

See `http://sb.cs.cmu.edu/drem/DREMmanual.pdf` for details.

## References

[1] Anthony Gitter, Judith Klein-Seetharaman, Anupam Gupta, and Ziv Bar-Joseph. Discovering pathways by orienting edges in protein interaction networks. *Nucleic acids research*, 39(4):e22–e22, 2011.