MOGAMUN - A Multi-Objective Genetic Algorithm to Find Active Modules in Multiplex Biological Networks

Elva Ma. Novoa del Toro

24 août 2020

Package

MOGAMUN 0.99.1

Contents

1 Introduction

This document describes the use of MOGAMUN, the type and format of the input data, and the post-processing and visualization of the output data. MOGAMUN is a package to find active modules (i.e., highly connected subnetworks with an overall deregulation) in multiplex biological networks. For a detailed
description of MOGAMUN check out the preprint https://www.biorxiv.org/content/10.1101/2020.05.25.114215v1. All the expression datasets and networks that we used to obtain the results reported in our preprint are available in the GitHub repository https://github.com/elvanov/MOGAMUN-data.

2 Workflow

The workflow of MOGAMUN is composed of 3 main steps: initialization of parameters, providing input data and running the algorithm.

2.1 Initialization of parameters

We set here the values for the evolution parameters and other general parameters, such as the minimum and maximun sizes of the subnetworks, via `mogamun.init`. In total, there are 11 customizable parameters:

- `PopSize` is an integer that determines the size of the population (*default = 100*).
- `MinSize` and `MaxSize` are integers that determine the minimun and maximum number of nodes that are allowed in each subnetwork (i.e. individual), respectively (*default min. = 15, max. = 50*).
- `Generations` is an integer that determines the total number of generations of the evolution process (*default = 500*).
- `TournamentSize` is an integer that determines the size of the tournament, i.e. the number of individuals that will participate in the tournament for the selection of parents (*default = 2*).
- `CrossoverRate` is a real value in the [0-1] range that determines the crossover rate, where 0 means that crossover will never be performed and 1 means that crossover will always be performed (*default = 0.8*).
- `MutationRate` is a real value in the [0-1] range that determines the mutation rate, where 0 means that mutation will never be performed and 1 means that mutation will always be performed (*default = 0.1*).
- `JaccardSimilarityThreshold` is an integer that determines the maximum Jaccard similarity coefficient to consider two subnetworks as different (*default = 30*). Subnetworks with a Jaccard similarity coefficient higher than this value are considered as duplicates. Only one of them (the best one, according to the Pareto dominance criteria) will remain in the population, and the rest will be replaced by random subnetworks.
- `MaxNumberOfAttempts` is an integer that determines the maximum number of attempts allowed to try to find compatible parents for the crossover (*default = 3*).
- `Measure` is a string that can take any of two values: **FDR** or **PValue**. It determines which statistical value will be taken into account to obtain the list of significantly differentially expressed genes (*default = FDR*).
- `ThresholdDEG` is a real value that determines the maximum value of the `Measure` to consider a gene as significantly differentially expressed (*default = 0.05*).

If MOGAMUN is to be run with the default values, execute `EvolutionParameters <- mogamun.init()`. Otherwise, specify the parameters to change, separated by commas. Please be aware than although we recommend to let the evolution run for 500 generations, this process can be long. For instance, using a multiplex networks with the three layers that we provide in *MOGAMUN/ExampleFiles.zip* and the default values for all the parameters, the process took approximately 12 hours in a desktop computer with Intel processor i7 at 3.60GHz and 32GB of RAM.

```
parameters <- mogamun.init(Generations = 1, PopSize = 10)
```

2.2 Providing input data

MOGAMUN uses two sources of information: one or more biological networks, and the statistical values resulting from a differential expression analysis or any other test that gives as result *p*-values or False Discovery Rates (FDR) associated to genes. The second step of the workflow is to provide the input data using the `mogamun.load.data` function, which has 5 parameters:

- `EvolutionParameters` is the list returned by `mogamun.init`.
- `DifferentialExpressionPath` is a string with the full path to the **comma separated** file containing the results of the differential expression analysis. Please note that such file must be composed of at least two columns: "gene" and `Measure`. The column "gene" must contain **gene names** and the column `Measure` must be called either "FDR" or "PValue" (depending on which of the two was seleced in `mogamun.init`) and it should contain the results of the statistical test. We strongly recommend to have a third columm ("logFC") with the $log_2(fold\_change)$, i.e. the ratio of difference of the expression between the two groups under study (e.g. patients vs control). There is a sample file in *MOGAMUN/ExampleFiles/DE/Sample_DE.csv*.
- `NodesScoresPath` is a string with the full path to the **comma separated** file containing the nodes scores for every node in the network. Please note that if this file does not exist, MOGAMUN will create it.
- `NetworkLayersDir` is a string with the path to the directory (i.e. folder) that contains the networks that will compose the multiplex network. Please note that each biological network must be in a separate file with 2-column **tab separated** format. The name of each file must start with a different character, which will be used in the `Layers` parameter. There are two example files in *MOGAMUN/ExampleFiles/LayersMultiplex*.
- `Layers` is a string composed of a chain of characters that determine which files from the `NetworkLayersDir` are to be used to build the multiplex network (e.g. "123" to use the networks which names start with "1", "2", and "3").

```
dePath <- system.file("ExampleFiles/DE/Sample_DE.csv", package = "MOGAMUN")
scoresPath <-
    system.file("ExampleFiles/DE/Sample_NodesScore.csv", package = "MOGAMUN")
layersPath <-
    system.file("ExampleFiles/LayersMultiplex/", package = "MOGAMUN")

loadedData <-
    mogamun.load.data(
        EvolutionParameters = parameters,
        DifferentialExpressionPath = dePath,
        NodesScoresPath = scoresPath,
        NetworkLayersDir = layersPath,
```

```
            Layers = "23"
        )
```

2.3 Running the algorithm

Once we have defined all the parameters and provided the input data, we are ready to run MOGAMUN using the `mogamun.run` function, which has 4 parameters:

- `LoadedData` is the list returned by `mogamun.load.data`
- `Cores` is an integer that determines the number of cores that will be used to run MOGAMUN. This number must be in line with the number of physical processor cores (*default = 1*). One core is needed per run.
- `NumberOfRunsToExecute` is an integer that determines the number of runs to be performed (*default = 1*). If you have enough resources, we strongly recommend to run MOGAMUN 30 times. Please note that if you wish to run MOGAMUN a higher number of times than cores you have, the pending runs will be executed sequentially.
- `ResultsDir` is a string with the full path to the directory where the results must be stored (*default* = '.', i.e. current working directory).

```
mogamun.run(LoadedData = loadedData, ResultsDir = '.')
## Warning in dir.create(ResultsPath, recursive = TRUE): './Experiment_2020-08-24'
## already exists
## [1] "Run 1. Gen. 1 completed"
## [1] "FINISH TIME, RUN 1: 2020-08-24 14:00:41"
## [[1]]
##           used (Mb) gc trigger  (Mb) max used  (Mb)
## Ncells 1114486 59.6    2115904 113.1  2115904 113.1
## Vcells 2017489 15.4    8388608  64.0  4301254  32.9
```

In the results directory (`ResultsDir`) you will find a subfolder which name contains the date when you executed the experiment. Inside, there will be two files per run (**MOGAMUN_Results_StatisticsPerGeneration_RunN.csv** and **MOGAMUN_Results__Run_N.txt**). The file** MOGAMUN_Results_StatisticsPerGeneration_RunN.csv** contains the best values for the two objectives (average nodes score and density) per generation, which you can use to check the convergence, for instance. The file **MOGAMUN_Results__Run_N.txt** contains the complete final population of size `PopSize` (i.e. all the subnetworks from the last generation), one per row. The number of elements in every row is variable because the size of each subnetwork can vary between `MinSize` and `MaxSize`. If **X_n** is the number of elements in the **n**-th row: the nodes of the subnetwork are the first **X_n**-4 elements. The last four elements correspond to the average nodes score, density, rank, and crowding distance, respectively. The best (non-dominated) subnetworks have are those with rank = 1.

3 Postprocessing of the results

In our preprint (https://www.biorxiv.org/content/10.1101/2020.05.25.114215v1), we ran MOGAMUN 30 times for each experiment. This increases the chances to find the global maxima. Given that the result of every run is the set of subnetworks with rank = 1, if you execute MOGAMUN multiple times the final result is the union of the results of all the individual runs. But considering that in such set there might be subnetworks that are better

than others (according to the Pareto dominance), to obtain the final result we calculate the accumulated Pareto front. To this goal, we re-rank the set composed by the union of all the results, and leave only those subnetworks with rank $= 1$. In addition, we propose to merge the subnetworks that are very similar, in order to avoid having two different networks if they only differ for one node, for instance (see `JaccardSimilarityThreshold`).

Depending on the number of runs you execute, there are other plots that might be generated during the postprocessing, such as scatter plots (always) and boxplots (only if `NumberOfRunsToExecute` in `mogamun.run` > 1).

Finally, it is possible to visualize the subnetworks from the accumulated Pareto front in Cytoscape. Please note that the network you used to build the multiplex will be filtered to leave only those interactions among the genes that are included in the result. The nodes will be colored according to their *logFC* value, where green stands for downregulated, red means upregulated, and white means no deregulation. Nodes corresponding to significantly differentially expressed genes will have black border, and the color of the edges will be different for each layer of the multiplex network.

The postprocessing is done with the function `mogamun.postprocess`, which has 4 parameters:

- `ExperimentDir` is a string with the full path to the directory where the results are stored. It must coincide with the `ResultsDir` folder from `mogamun.run`.
- `LoadedData` is the list returned by `mogamun.load.data`.
- `JaccardSimilarityThreshold` is an integer that determines the minimum Jaccard similarity coefficient that a pair of subnetworks must have to be merged in a single subnetwork (default $= 70$).
- `VisualizeInCytoscape` is a boolean value (TRUE/FALSE) that determines whether the accumulated Pareto front is to be displayed in Cytoscape or not. NOTE. When `VisualizeInCytoscape` is TRUE, Cytoscape needs to be up and running in the computer (default $=$ TRUE).

Please note that you can make use of `mogamun.postprocess` with any number of runs. If the experiment to be postprocessed contains a single run, the result will be the set of subnetworks in the first Pareto front. Otherwise, the result will be the accumulated Pareto front, as we already explained.