Introduction to MCL

Natural clusters in a graph are characterised by the presence of many edges between the members of that cluster, and one expects that the number of higher-length (longer) paths between two arbitrary nodes in the cluster is high. In particular, this number should be high, relative to node pairs lying in different natural clusters. A different angle on this is that random walks on the graph will infrequently go from one natural cluster to another.

The MCL algorithm finds cluster structure in graphs by a mathematical bootstrapping procedure. The process deterministically computes (the probabilities of) random walks through the graph, and uses two operators transforming one set of probabilities into another. It does so using the language of stochastic matrices (also called Markov matrices) which capture the mathematical concept of random walks on a graph.

The MCL algorithm simulates random walks within a graph by alternation of two operators called expansion and inflation. Expansion coincides with taking the power of a stochastic matrix using the normal matrix product (i.e. matrix squaring). Inflation corresponds with taking the Hadamard power of a matrix (taking powers entrywise), followed by a scaling step, such that the resulting matrix is stochastic again, i.e. the matrix elements (on each column) correspond to probability values.

A column stochastic matrix is a non-negative matrix with the property that each of its columns sums to 1. Given such a matrix M and a real number $r > 1$, the column stochastic matrix resulting from inflating each of the columns of M with power coefficient r is written $\Gamma_r(M)$, and Γ_r is called the inflation operator with power coefficient r. Write $\Sigma_{r,j}(M)$ for the summation of all the entries in column j of M raised to the power r (sum after taking powers). Then $\Gamma_r(M)$ is defined in an entrywise manner by setting

$$\Gamma_r(M_{ij}) = M_{ij}^r \,/\, \Sigma_{r,j}(M)$$

Each column j of a stochastic matrix M corresponds with node j of the stochastic graph associated with M. Row entry i in column j (i.e. the matrix entry M_{ij}) corresponds with the probability of going from node j to node i. It is observed that for values of $r > 1$, inflation changes

the probabilities associated with the collection of random walks departing from one particular node (corresponding with a matrix column) by favouring more probable walks over less probable walks.

Expansion corresponds to computing random walks of higher length, which means random walks with many steps. It associates new probabilities with all pairs of nodes, where one node is the point of departure and the other is the destination. Since higher length paths are more common within clusters than between different clusters, the probabilities associated with node pairs lying in the same cluster will, in general, be relatively large as there are many ways of going from one to the other. Inflation will then have the effect of boosting the probabilities of intra-cluster walks and will demote inter-cluster walks. This is achieved without any a priori knowledge of cluster structure. It is simply the result of cluster structure being present.

Eventually, iterating expansion and inflation results in the separation of the graph into different segments. There are no longer any paths between these segments and the collection of resulting segments is simply interpreted as a clustering. The inflation operator can be altered using the parameter *r*. Increasing this parameter has the effect of making the inflation operator stronger, and this increases the granularity or tightness of clusters.

With this, the MCL algorithm can be written as

```
G is a graph
add loops to G                              # see below
set Γ to some value                         # affects granularity
set M_1 to be the matrix of random walks on G

while (change) {
   M_2 =  M_1 * M_1                         #  expansion
   M_1 =  Γ(M_2)                            #  inflation
   change   =  difference(M_1, M_2)
}

set CLUSTERING as the components of M_1     #  see below
```

It is possible to compute `change` in a simpler way, directly from **M_2** and using the characteristics of the limits of the MCL process. This is beyond the scope of this introduction however.

Informally, cast in the language of stochastic flow, we can state that expansion causes flow to dissipate within clusters whereas inflation eliminates flow between different clusters. Expansion and inflation represent different tidal forces which are alternated until an

equilibrium state is reached. An equilibrium state takes the form of a so-called doubly idempotent matrix, i.e. a matrix that does not change with further expansion or inflation steps. The graph associated with such a matrix consists of different connected directed components. Each component is interpreted as a cluster, and has a star-like form, with one attractor in the centre and arcs going from all nodes of that component to the attractor. In theory, attractor systems with more than one attractor may occur (these do not change the cluster interpretation). Also, nodes may exist that are connected to different stars, which is canonically interpreted as cluster overlap, or in other words nodes may belong to multiple clusters.

With respect to convergence, it can be proven that the process simulated by the algorithm converges quadratically around the equilibrium states. In practice, the algorithm starts to converge noticeably after 3-10 iterations. Global convergence is very hard to prove; it is conjectured that the process always converges if the input graph is symmetric. This conjecture is supported by results concerning the matrix iterands. For symmetric input graphs, it is true that all iterands have real spectrum (the set of eigenvalues), and that all iterands resulting from expansion have non-negative spectrum and are diagonally symmetric to a positive semi-definite matrix. It can be shown that these matrices have a structural property which associates a directed acyclic graph (DAG) with each of them. It turns out that inflation strengthens (in a quantitative sense) this structural property and will never change the associated DAG, whereas expansion is in fact able to change the associated DAG. This is a more mathematical view on the tidal forces analogy mentioned earlier. DAGs generalise the star graphs associated with MCL limits, and the spectral properties of MCL iterands and MCL limits can be related via the inflation operator. These results imply that the equilibrium states can be viewed as a set of extreme points of the set of matrices that are diagonally similar to a positive semi-definite matrix. This establishes a close relationship between the MCL iterands, MCL limits, and cluster (and DAG) structure in graphs.

The MCL algorithm also associates return probabilities (or loops) with each node in the initial input graph. The flow paradigm underlying MCL naturally requires this, and it can be motivated in terms of the spectral and structural properties mentioned earlier. As for the loop weights that are chosen, experience shows that a neutral value works well. It is possible to choose larger weights, and this will increase cluster granularity. The effect is secondary however to that of varying

the inflation parameter, and the algorithm is not very sensitive to changes in the loop weights.

A very important asset of the algorithm is its bootstrapping nature, retrieving cluster structure via the imprint made by this structure on the flow process. Further key benefits of the algorithm are (i) it is not misled by edges linking different clusters; (ii) it is very fast and very scalable; (iii) it has a natural parameter for influencing cluster granularity; (iv) the mathematics associated with the algorithm shows that there is an intrinsic relationship between the process it simulates and cluster structure in the input graph; and (v) its formulation is simple and elegant. From the definition of the MCL algorithm it is seen that it is based on a very different paradigm than any linkage-based algorithm. One possible view of this is that MCL, although based on similarities between pairs, recombines these similarities (via expansion) and is thus affected by similarities on the level of sets (as generalising pairs). Alternating expansion with inflation turns out to be an appropriate way of exploiting this recombination property.

An introduction to the mathematics associated with the MCL algorithm.