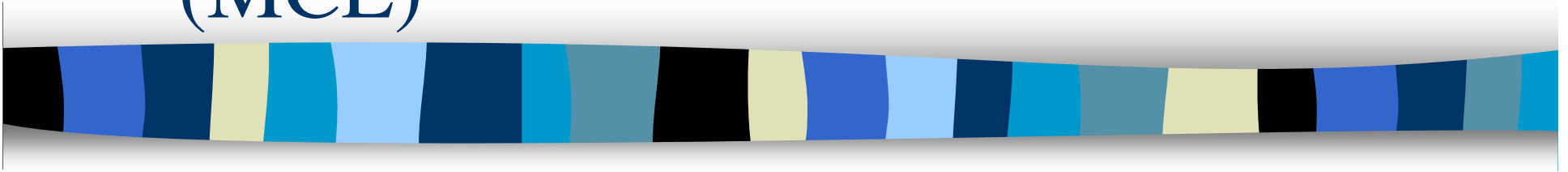


Clustering on Graphs: The Markov Cluster Algorithm (MCL)



CS 595D Presentation
By Kathy Macropol



MCL Algorithm

- Based on the PhD thesis by Stijn van Dongen

Van Dongen, S. (2000) *Graph Clustering by Flow Simulation*. PhD Thesis, University of Utrecht, The Netherlands.

- MCL is a graph clustering algorithm.
- MCL is freely available for download at <http://www.micans.org/mcl/>

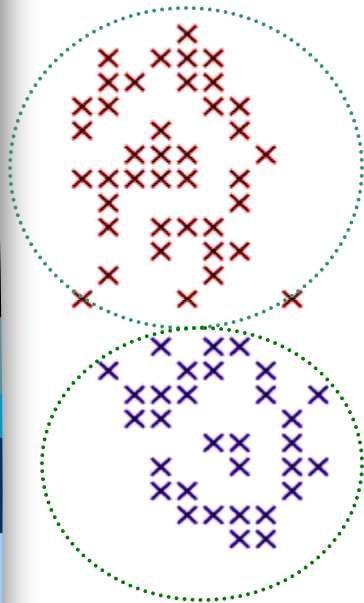


Outline

- Background
 - Clustering
 - Random Walks
 - Markov Chains
- MCL
 - Basis
 - Inflation Operator
 - Algorithm
 - Convergence
- MCL Analysis
 - Comparison to Other Graph Clustering Algorithms
 - RNSC, SPC, MCODE
 - RRW
- Conclusions

Graph Clustering

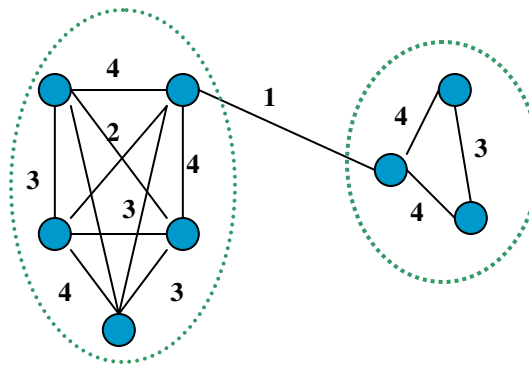
- Clustering – finding natural groupings of items.
- Vector Clustering



Each point has
a vector, i.e.

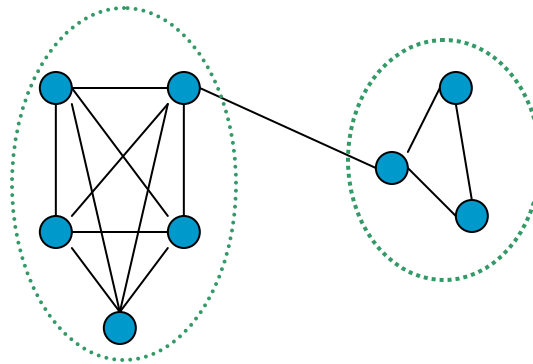
- x coordinate
- y coordinate
- color

Graph Clustering



Each vertex is
connected to
others by
(weighted or
unweighted)
edges.

Random Walks



- Considering a graph, there will be many links within a cluster, and fewer links between clusters.
- This means if you were to start at a node, and then randomly travel to a connected node, you're more likely to stay within a cluster than travel between.
- This is what MCL (and several other clustering algorithms) is based on.
 - Other ways to consider graph clustering may include, for example, looking for cliques. This tends to be sensitive to changes in node degree, however.

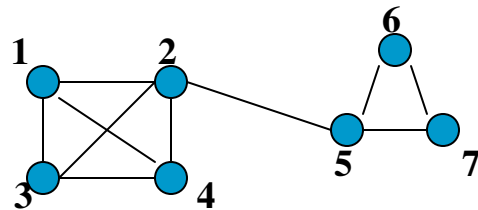


Random Walks

- By doing random walks upon the graph, it may be possible to discover where the flow tends to gather, and therefore, where clusters are.
- Random Walks on a graph are calculated using “Markov Chains”.

Markov Chains

- To see how this works, an example:



- In one time step, a random walker at node 1 has a 33% chance of going to node 2, 3, & 4, and 0% chance to nodes 5, 6, or 7.
- From node 2, 25% chance for 1, 3, 4, 5 and 0% for 6 and 7.
- Creating a transition matrix gives:

	1	2	3	4	5	6	7
1	0	.25	.33	.33	0	0	0
2	.33	0	.33	.33	.33	0	0
3	.33	.25	0	.33	0	0	0
4	.33	.25	.33	0	0	0	0
5	0	.25	0	0	0	.5	.5
6	0	0	0	0	.33	0	.5
7	0	0	0	0	.33	.5	0

(notice each
column sums
to one)

Also can be looked at as a probability matrix!

Markov Chains

■ A simpler example: $\begin{pmatrix} .6 & .2 \\ .4 & .8 \end{pmatrix}$

■ Next time step: $t_0 \rightarrow t_1 \rightarrow t_2$

$$1 \rightarrow 1 \rightarrow 1 + 1 \rightarrow 2 \rightarrow 1$$

$$.6 * .6 + .4 * .2 = .44$$

$$\begin{pmatrix} .6 & .2 \\ .4 & .8 \end{pmatrix} \begin{pmatrix} .6 & .2 \\ .4 & .8 \end{pmatrix} = \begin{pmatrix} .44 & .28 \\ .56 & .72 \end{pmatrix} \rightarrow \begin{pmatrix} .35 & .32 \\ .65 & .68 \end{pmatrix} \rightarrow \begin{pmatrix} .34 & .33 \\ .66 & .66 \end{pmatrix}$$

$$\xrightarrow{\text{eventually}} \begin{pmatrix} .33 & .33 \\ .66 & .66 \end{pmatrix}$$

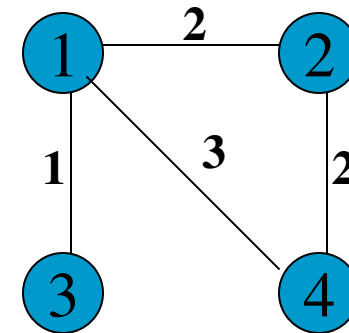


Markov Chain

- Markov Chain:
A sequence of variables X_1, X_2, X_3 , etc (in our case, the probability matrices) where, given the present state, the past and future states are independent.
- Probabilities for the next time step only depend on current probabilities (given the current probability).
- A random walk is an example of a Markov Chain, using the transition probability matrices.

Weighted Graphs

- To turn a weighted graph into a probability (transition) matrix, column normalize.



$$\begin{pmatrix} 0 & 2 & 1 & 3 \\ 2 & 0 & 0 & 2 \\ 1 & 0 & 0 & 0 \\ 3 & 2 & 0 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 1/2 & 1 & 3/5 \\ 1/3 & 0 & 0 & 2/5 \\ 1/6 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 \end{pmatrix}$$

Notice it's no longer symmetric.

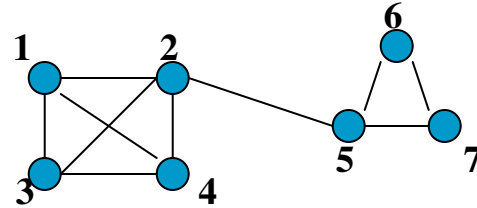
Adding Self Loops

- Small simple path loops can complicate things.
 - There is a strong effect that odd powers of expansion obtain their mass from simple paths of odd length, and likewise for even.
 - Adds a dependence to the transition probabilities on the parity of the simple path lengths.
- The addition of self looping edges on each node resolves this.
 - Adds a small path of length 1, so the mass does not only appear during odd powers of the matrix.

$$\begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix} \longrightarrow \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix}$$

Markov Chain Cluster Structure

■ Example:



0	.25	.33	.33	0	0	0	.15	.15	.15	.15	.15	.15	.15	.15	.15
.33	0	.33	.33	.33	0	0	.2	.2	.2	.2	.2	.2	.2	.2	.2
.33	.25	0	.33	0	0	0	.15	.15	.15	.15	.15	.15	.15	.15	.15
.33	.25	.33	0	0	0	0	.15	.15	.15	.15	.15	.15	.15	.15	.15
0	.25	0	0	0	0	.5	.15	.15	.15	.15	.15	.15	.15	.15	.15
0	0	0	0	.33	0	.5	.1	.1	.1	.1	.1	.1	.1	.1	.1
0	0	0	0	.33	.5	0	.1	.1	.1	.1	.1	.1	.1	.1	.1

eventually
→

Notice that, in the beginning time steps, before the flow really mixes, the cluster structure is pronounced in the matrix!

This is not a coincidence, and MCL uses this, modifying the random walk process to further emphasize the divide between clusters in the matrix.



MCL

- "Flow is easier within dense regions than across sparse boundaries, however, in the long run this effect disappears."
- During the earlier powers of the Markov Chain, the edge weights will be **higher** in links that are *within* clusters, and **lower** *between* the clusters.
- This means there is a correspondence between the distribution of weight over the columns and the clusterings.



MCL

- MCL deliberately boosts this affect by
 - Stopping partway in the Markov Chain
 - Then adjusting the transitions by columns.
 - For each vertex, the transition values are changed so that
 - Strong neighbors are further strengthened
 - Less popular neighbors are demoted.
- This adjusting can be done by raising a single column to a non-negative power, and then re-normalizing.
- This operation is named “Inflation”
- (Taking the Markov Chain powers is named “Expansion”)



MCL Inflation

- Example for inflation of 2 (squaring):

$$\begin{pmatrix} 0 \\ 1/2 \\ 0 \\ 1/6 \\ 1/3 \end{pmatrix} \quad \begin{pmatrix} 1/4 \\ 1/4 \\ 1/4 \\ 1/4 \\ 0 \end{pmatrix}$$

$$\begin{pmatrix} 0 \\ 9/14 \\ 0 \\ 1/14 \\ 4/14 \end{pmatrix} \quad \begin{pmatrix} 1/4 \\ 1/4 \\ 1/4 \\ 1/4 \\ 0 \end{pmatrix}$$

↓
Square, and
then normalize



MCL Inflation

DEFINITION 3. Given a matrix $M \in \mathbb{R}^{k \times l}$, $M \geq 0$, and a real nonnegative number r , the matrix resulting from rescaling each of the columns of M with power coefficient r is called $\Gamma_r M$, and Γ_r is called the **inflation** operator with power coefficient r . Formally, the action of $\Gamma_r : \mathbb{R}^{k \times l} \rightarrow \mathbb{R}^{k \times l}$ is defined by

$$(\Gamma_r M)_{pq} = (M_{pq})^r / \sum_{t=1}^k (M_{tq})^r$$

If the subscript is omitted, it is understood that the power coefficient equals 2. □



MCL Inflation

- The inflation operator is responsible for both strengthening and weakening of current. (Strengthens strong currents, and weakens already weak currents).
- The inflation parameter, r , controls the extent of this strengthening / weakening. (In the end, this influences the granularity of clusters.)



MCL Algorithm

- In MCL, the following two processes are alternated between repeatedly:
 - Expansion (taking the Markov Chain transition matrix powers)
 - Inflation
- The expansion operator is responsible for allowing flow to connect different regions of the graph.
- The inflation operator is responsible for both strengthening and weakening of current.

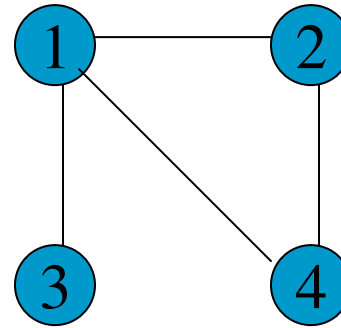


MCL Algorithm

1. Input is an un-directed graph, power parameter e , and inflation parameter r .
2. Create the associated matrix
3. Add self loops to each node (optional)
4. Normalize the matrix
5. Expand by taking the e^{th} power of the matrix
6. Inflate by taking inflation of the resulting matrix with parameter r
7. Repeat steps 5 and 6 until a steady state is reached (convergence).
8. Interpret resulting matrix to discover clusters.

MCL Algorithm

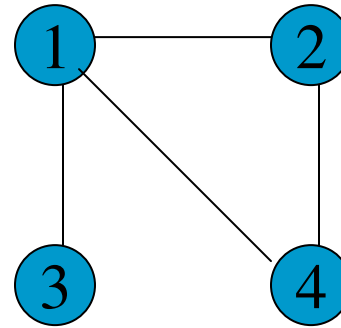
1. Input is an un-directed graph, power parameter e , and inflation parameter r .
2. Create the associated matrix
3. Add self loops to each node (optional)
4. Normalize the matrix
5. Expand by taking the e^{th} power of the matrix
6. Inflate by taking inflation of the resulting matrix with parameter r
7. Repeat steps 5 and 6 until a steady state is reached (convergence).
8. Interpret resulting matrix to discover clusters.



Power of 2
Inflation of 2

MCL Algorithm

1. Input is an un-directed graph, power parameter e , and inflation parameter r .
2. **Create the associated matrix**
3. Add self loops to each node (optional)
4. Normalize the matrix
5. Expand by taking the e^{th} power of the matrix
6. Inflate by taking inflation of the resulting matrix with parameter r
7. Repeat steps 5 and 6 until a steady state is reached (convergence).
8. Interpret resulting matrix to discover clusters.

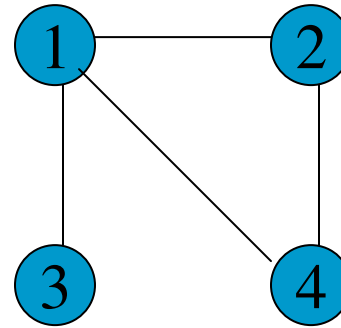


Power of 2
Inflation of 2

$$\begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix}$$

MCL Algorithm

1. Input is an un-directed graph, power parameter e , and inflation parameter r .
2. Create the associated matrix
3. Add self loops to each node (optional)
4. Normalize the matrix
5. Expand by taking the e^{th} power of the matrix
6. Inflate by taking inflation of the resulting matrix with parameter r
7. Repeat steps 5 and 6 until a steady state is reached (convergence).
8. Interpret resulting matrix to discover clusters.



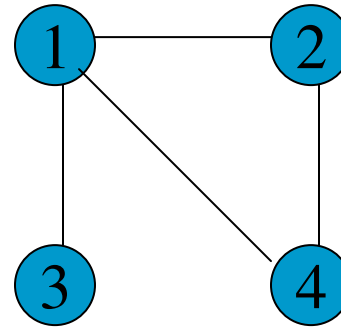
Power of 2
Inflation of 2

$$\begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix}$$

MCL Algorithm

1. Input is an un-directed graph, power parameter e , and inflation parameter r .
2. Create the associated matrix
3. Add self loops to each node (optional)
4. **Normalize the matrix**
5. Expand by taking the e^{th} power of the matrix
6. Inflate by taking inflation of the resulting matrix with parameter r
7. Repeat steps 5 and 6 until a steady state is reached (convergence).
8. Interpret resulting matrix to discover clusters.



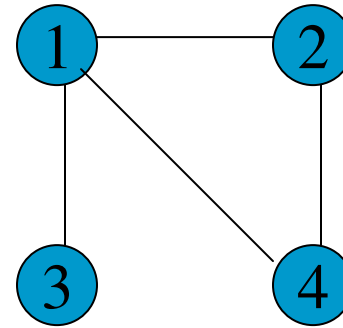
Power of 2
Inflation of 2

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1/4 & 1/3 & 1/2 & 1/3 \\ 1/4 & 1/3 & 0 & 1/3 \\ 1/4 & 0 & 1/2 & 0 \\ 1/4 & 1/3 & 0 & 1/3 \end{pmatrix}$$

MCL Algorithm

1. Input is an un-directed graph, power parameter e , and inflation parameter r .
2. Create the associated matrix
3. Add self loops to each node (optional)
4. Normalize the matrix
5. **Expand by taking the e^{th} power of the matrix**
6. Inflate by taking inflation of the resulting matrix with parameter r
7. Repeat steps 5 and 6 until a steady state is reached (convergence).
8. Interpret resulting matrix to discover clusters.



Power of 2
Inflation of 2

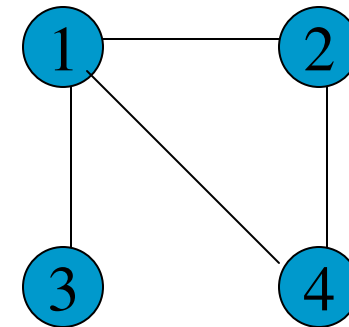
$$\begin{pmatrix} \frac{1}{4} & \frac{1}{3} & \frac{1}{2} & \frac{1}{3} \\ \frac{1}{4} & \frac{1}{3} & 0 & \frac{1}{3} \\ \frac{1}{4} & 0 & \frac{1}{2} & 0 \\ \frac{1}{4} & \frac{1}{3} & 0 & \frac{1}{3} \end{pmatrix} \begin{pmatrix} \frac{1}{4} & \frac{1}{3} & \frac{1}{2} & \frac{1}{3} \\ \frac{1}{4} & \frac{1}{3} & 0 & \frac{1}{3} \\ \frac{1}{4} & 0 & \frac{1}{2} & 0 \\ \frac{1}{4} & \frac{1}{3} & 0 & \frac{1}{3} \end{pmatrix}$$

=

$$\begin{pmatrix} .35 & .31 & .38 & .31 \\ .23 & .31 & .13 & .31 \\ .19 & .08 & .38 & .08 \\ .23 & .31 & .13 & .31 \end{pmatrix}$$

MCL Algorithm

1. Input is an un-directed graph, power parameter e , and inflation parameter r .
2. Create the associated matrix
3. Add self loops to each node (optional)
4. Normalize the matrix
5. Expand by taking the e^{th} power of the matrix
6. **Inflate by taking inflation of the resulting matrix with parameter r**
7. Repeat steps 5 and 6 until a steady state is reached (convergence).
8. Interpret resulting matrix to discover clusters.



Power of 2
Inflation of 2

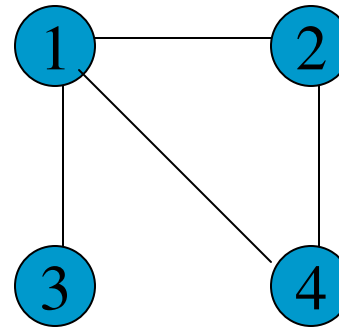
.35	.31	.38	.31
.23	.31	.13	.31
.19	.08	.38	.08
.23	.31	.13	.31

.13	.09	.14	.09
.05	.09	.02	.09
.04	.01	.14	.01
.05	.09	.02	.09

.47	.33	.45	.33
.20	.33	.05	.33
.13	.02	.45	.02
.20	.33	.05	.33

MCL Algorithm

1. Input is an un-directed graph, power parameter e , and inflation parameter r .
2. Create the associated matrix
3. Add self loops to each node (optional)
4. Normalize the matrix
5. Expand by taking the e^{th} power of the matrix
6. Inflate by taking inflation of the resulting matrix with parameter r
7. Repeat steps 5 and 6 until a steady state is reached (convergence).
8. Interpret resulting matrix to discover clusters.



Power of 2
Inflation of 2

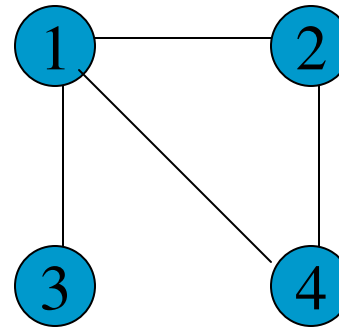
$$\begin{pmatrix} .70 & .33 & .49 & .33 \\ .12 & .33 & .01 & .33 \\ .05 & .02 & .49 & -- \\ .12 & .33 & .01 & .33 \end{pmatrix}$$

$$\begin{pmatrix} .94 & .33 & .50 & .33 \\ .03 & .33 & -- & .33 \\ .01 & -- & .50 & -- \\ .13 & .33 & -- & .33 \end{pmatrix}$$

$$\begin{pmatrix} 1 & .33 & .50 & .33 \\ -- & .33 & -- & .33 \\ -- & -- & .50 & -- \\ -- & .33 & -- & .33 \end{pmatrix}$$

MCL Algorithm

1. Input is an un-directed graph, power parameter e , and inflation parameter r .
2. Create the associated matrix
3. Add self loops to each node (optional)
4. Normalize the matrix
5. Expand by taking the e^{th} power of the matrix
6. Inflate by taking inflation of the resulting matrix with parameter r
7. Repeat steps 5 and 6 until a steady state is reached (convergence).
8. Interpret resulting matrix to discover clusters.



Power of 2
Inflation of 2

*Expand on in
just a minute.*

MCL Algorithm Convergence

- Not obvious that result will converge. Convergence is not proven in the thesis, however it is shown experimentally that it often does occur.
- In practice, the algorithm converges nearly always to a "doubly idempotent" matrix:
 1. It's at steady state.
 2. Every value in a single column has the same number (homogeneous).

$$\begin{pmatrix}
 1.000 & -- & -- & -- & -- & 1.000 & 1.000 & -- & -- & 1.000 & -- & -- \\
 -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- \\
 -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- \\
 -- & 1.000 & 1.000 & -- & 1.000 & -- & -- & -- & -- & -- & -- & -- \\
 -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- \\
 -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- \\
 -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- \\
 -- & -- & -- & 0.500 & -- & -- & -- & 0.500 & 0.500 & -- & 0.500 & 0.500 \\
 -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- \\
 -- & -- & -- & 0.500 & -- & -- & -- & 0.500 & 0.500 & -- & 0.500 & 0.500 \\
 -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & --
 \end{pmatrix}$$

M_{mcl}^{∞}



MCL Algorithm Convergence

- It *is* proven that when the matrix is in the neighborhood of being doubly idempotent, it converges quadratically.
- However, the final steady state may sometimes be cyclic and consist of a repeating series of matrices.
 - In certain cases, the expansion and inflation act as inverses of each other. However, a slight change of parameters and the equilibrium is broken.
 - Without self loops, it's possible on bipartite graphs because of odd path lengths. Adding self loops and slightly changing parameters fixes most of the problems.
 - Other graphs that may have periodic behavior are described, but will most likely be "a curiosity lacking cluster structure anyhow".

MCL Algorithm Convergence

0.200	0.250	--	--	--	0.333	0.250	--	--	0.250	--	--
0.200	0.250	0.250	--	0.200	--	--	--	--	--	--	--
--	0.250	0.250	0.200	0.200	--	--	--	--	--	--	--
--	--	0.250	0.200	--	--	--	0.200	0.200	--	0.200	--
--	0.250	0.250	--	0.200	--	0.250	0.200	--	--	--	--
0.200	--	--	--	--	0.333	--	--	--	0.250	--	--
0.200	--	--	--	0.200	--	0.250	--	--	0.250	--	--
--	--	--	0.200	0.200	--	--	0.200	0.200	--	0.200	--
--	--	--	0.200	--	--	--	0.200	0.200	--	0.200	0.333
0.200	--	--	--	--	0.333	0.250	--	--	0.250	--	--
--	--	--	0.200	--	--	--	0.200	0.200	--	0.200	0.333
--	--	--	--	--	--	--	--	0.200	--	0.200	0.333

M

0.380	0.087	0.027	--	0.077	0.295	0.201	--	--	0.320	--	--
0.047	0.347	0.210	0.017	0.150	0.019	0.066	0.012	--	0.012	--	--
0.014	0.210	0.347	0.056	0.150	--	0.016	0.046	0.009	--	0.009	--
--	0.027	0.087	0.302	0.062	--	--	0.184	0.143	--	0.143	0.083
0.058	0.210	0.210	0.056	0.406	--	0.083	0.046	0.009	0.019	0.009	--
0.142	0.017	--	--	--	0.295	0.083	--	--	0.184	--	--
0.113	0.069	0.017	--	0.062	0.097	0.333	0.012	--	0.147	--	--
--	0.017	0.069	0.175	0.049	--	0.016	0.287	0.143	--	0.143	0.083
--	--	0.017	0.175	0.012	--	--	0.184	0.288	--	0.288	0.278
0.246	0.017	--	--	0.019	0.295	0.201	--	--	0.320	--	--
--	--	0.017	0.175	0.012	--	--	0.184	0.288	--	0.288	0.278
--	--	--	0.044	--	--	--	0.046	0.120	--	0.120	0.278

$\Gamma_2 M^2,$

MCL Algorithm Convergence

0.448	0.080	0.023	--	0.068	0.426	0.359	--	--	0.432	--	--
0.018	0.285	0.228	0.007	0.176	0.006	0.033	0.005	--	0.007	--	--
0.005	0.223	0.290	0.022	0.173	--	0.010	0.017	0.003	0.001	0.003	0.001
--	0.018	0.059	0.222	0.040	--	0.001	0.187	0.139	--	0.139	0.099
0.027	0.312	0.314	0.028	0.439	0.005	0.054	0.022	0.003	0.010	0.003	0.001
0.116	0.007	0.001	--	0.004	0.157	0.085	--	--	0.131	--	--
0.096	0.040	0.013	--	0.037	0.083	0.197	0.001	--	0.104	--	--
--	0.012	0.042	0.172	0.029	--	0.002	0.198	0.133	--	0.133	0.096
--	0.001	0.015	0.256	0.009	--	--	0.266	0.326	--	0.326	0.346
0.290	0.021	0.002	--	0.017	0.323	0.260	--	--	0.316	--	--
--	0.001	0.015	0.256	0.009	--	--	0.266	0.326	--	0.326	0.346
--	--	0.001	0.037	0.001	--	--	0.039	0.069	--	0.069	0.112

$\Gamma_2(\Gamma_2 M^2 \cdot \Gamma_2 M^2)$

0.807	0.040	0.015	--	0.034	0.807	0.807	--	--	0.807	--	--
--	0.090	0.092	--	0.088	--	--	--	--	--	--	--
--	0.085	0.088	--	0.084	--	--	--	--	--	--	--
--	0.001	0.001	0.032	0.001	--	--	0.032	0.031	--	0.031	0.031
--	0.777	0.798	--	0.786	--	0.001	--	--	--	--	--
0.005	--	--	--	--	0.005	0.005	--	--	0.005	--	--
0.003	0.001	--	--	0.001	0.003	0.003	--	--	0.003	--	--
--	--	0.001	0.024	--	--	--	0.024	0.024	--	0.024	0.024
--	--	0.002	0.472	0.001	--	--	0.472	0.472	--	0.472	0.472
0.185	0.005	0.001	--	0.004	0.185	0.184	--	--	0.185	--	--
--	--	0.002	0.472	0.001	--	--	0.472	0.472	--	0.472	0.472
--	--	--	0.001	--	--	--	0.001	0.001	--	0.001	--

$(\Gamma_2 \circ \text{Squaring})$ iterated four times on M

MCL Algorithm Convergence

$$M_{mcl}^{\infty} = \begin{pmatrix} 1.000 & -- & -- & -- & -- & 1.000 & 1.000 & -- & -- & 1.000 & -- & -- \\ -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- \\ -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- \\ -- & 1.000 & 1.000 & -- & 1.000 & -- & -- & -- & -- & -- & -- & -- \\ -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- \\ -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- \\ -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- \\ -- & -- & -- & 0.500 & -- & -- & -- & 0.500 & 0.500 & -- & 0.500 & 0.500 \\ -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- \\ -- & -- & -- & 0.500 & -- & -- & -- & 0.500 & 0.500 & -- & 0.500 & 0.500 \\ -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- \end{pmatrix}$$

M_{mcl}^{∞}

[MCL Animated](#)

MCL Interpreting Clusters

- To interpret clusters, the vertices are split into two types. **Attractors**, which attract other vertices, and vertices that are *being* attracted by the attractors.
- Attractors have at least one positive flow value within their corresponding row (in the steady state matrix).
- Each attractor is attracting the vertices which have positive values within its row.

$$\begin{pmatrix} 1.000 & -- & -- & -- & -- & 1.000 & 1.000 & -- & -- & 1.000 & -- & -- \\ -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- \\ -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- \\ -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- \\ -- & 1.000 & 1.000 & -- & 1.000 & -- & -- & -- & -- & -- & -- & -- \\ -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- \\ -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- \\ -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- \\ -- & -- & -- & 0.500 & -- & -- & -- & 0.500 & 0.500 & -- & 0.500 & 0.500 \\ -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- \\ -- & -- & -- & 0.500 & -- & -- & -- & 0.500 & 0.500 & -- & 0.500 & 0.500 \\ -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- \end{pmatrix}$$

M_{mcl}^{∞}

MCL Interpreting Clusters

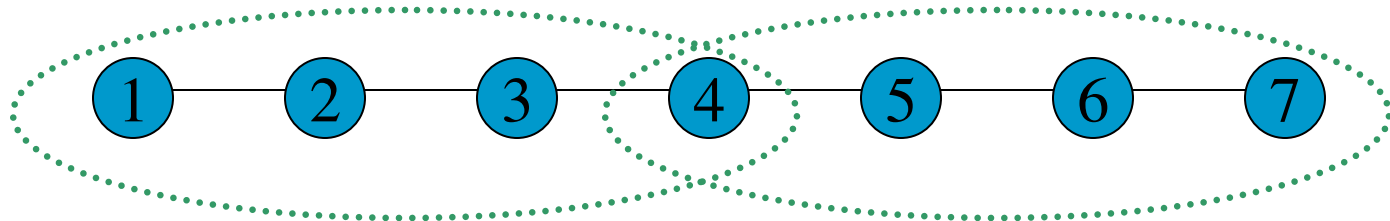
$$\begin{pmatrix}
 1.000 & -- & -- & -- & -- & 1.000 & 1.000 & -- & -- & 1.000 & -- & -- \\
 -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- \\
 -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- \\
 -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- \\
 -- & 1.000 & 1.000 & -- & 1.000 & -- & -- & -- & -- & -- & -- & -- \\
 -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- \\
 -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- \\
 -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- \\
 -- & -- & -- & 0.500 & -- & -- & -- & 0.500 & 0.500 & -- & 0.500 & 0.500 \\
 -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- \\
 -- & -- & -- & 0.500 & -- & -- & -- & 0.500 & 0.500 & -- & 0.500 & 0.500 \\
 -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & -- & --
 \end{pmatrix}$$

M_{mcl}^{∞}

- Attractors and the elements they attract are swept together into the same cluster.
- In this case, $\{1, 6, 7, 10\}$, $\{2, 3, 5\}$, $\{4, 8, 9, 11, 12\}$

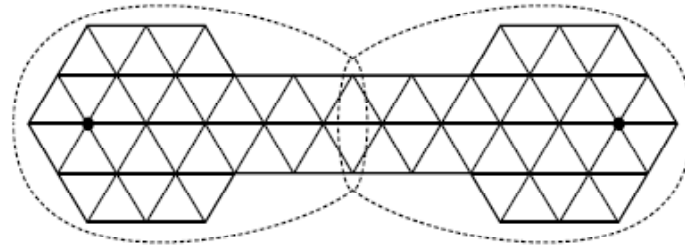
MCL Interpreting Clusters

- In general, overlapping clusters (where one node is contained in multiple clusters) are only found in very special cases of graph symmetry:
 - Only when a vertex is attracted *exactly* equally by more than one cluster
 - This occurs only when both clusters are isomorphic

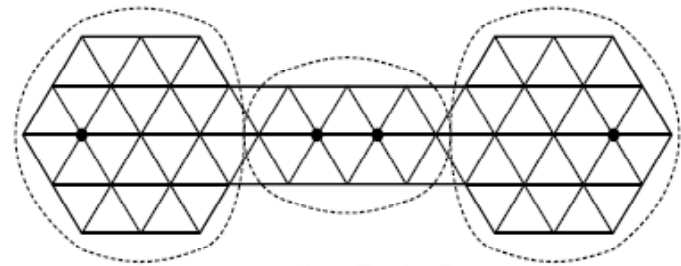


MCL Clusters

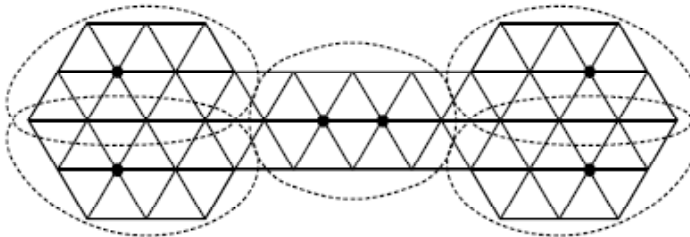
- The inflation parameter affects cluster granularity (a is the weight for added self loops)



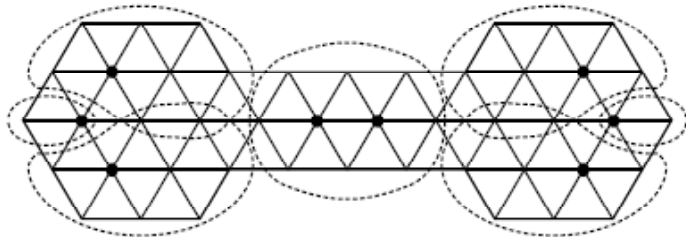
-a 1 -R 1.4



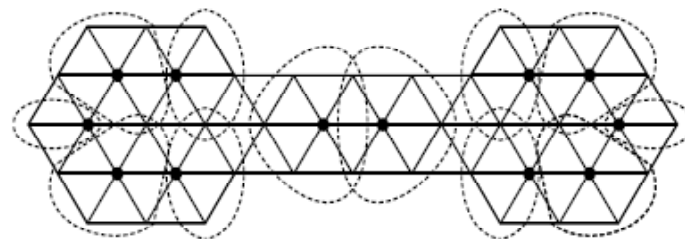
-a 1 -R 1.5



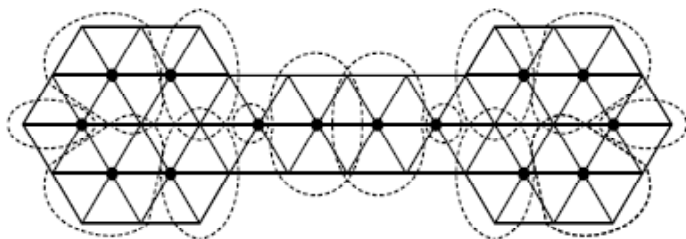
-a 1 -R 1.7



-a 1 -R 2.0



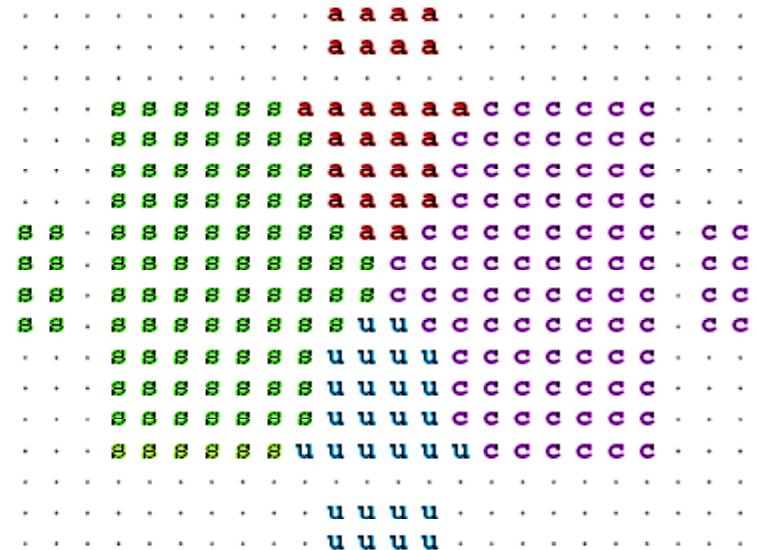
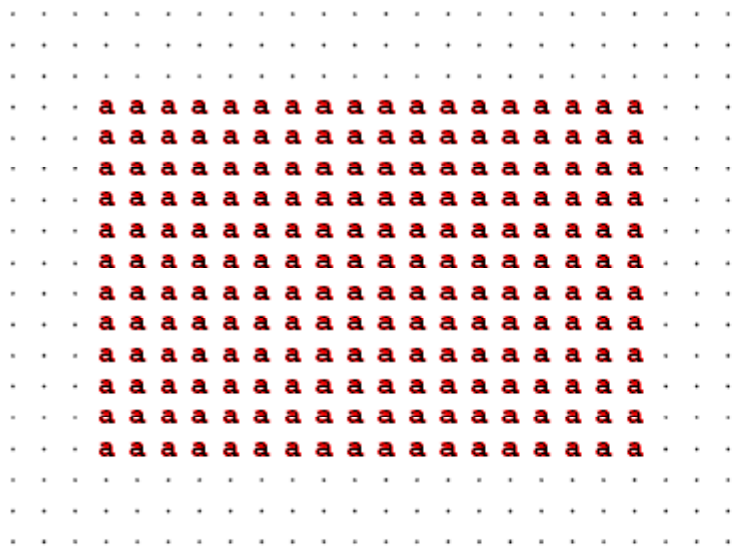
-a 1 -R 2.1



-a 1 -R 2.5

MCL Clusters

- For clusters with large diameter, MCL has problems
- Distributing flow across cluster needs long expansion and low inflation (otherwise the cluster will split).
- Takes many iterations and causes MCL to be sensitive to small perturbations in the graph.
 - The addition of small diameter clusters disturbs the clustering, since the low inflation parameter will cause them to disproportionately ‘inflate’ surrounding probabilities.





Analysis of MCL

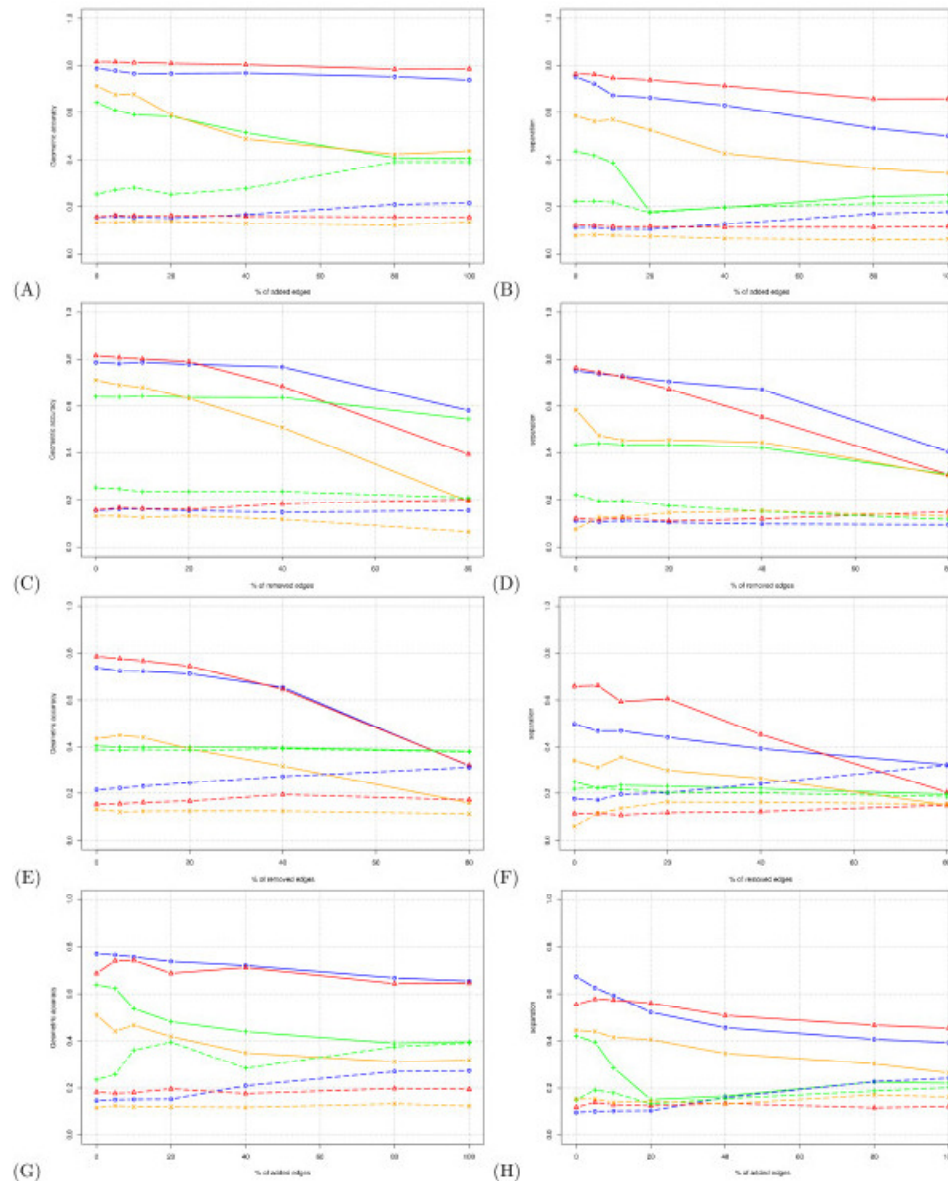
- $O(N^3)$, where N is the number of vertices.
 - N^3 cost of one matrix multiplication on two matrices of dimension N .
 - Inflation can be done in $O(N^2)$ time
 - The number of steps to converge is not proven, but experimentally shown to be ~ 10 to 100 steps, and mostly consist of sparse matrices after the first few steps.
- Speed may be improved through pruning.
 - Inspect matrix and set small values directly to zero (assume they would have reached there eventually anyways).
 - Works well when the diameter of the clusters is small. (Non-homogeneous distributions of weight)



Analysis of MCL

- In 2006, MCL fared well in a paper comparing it to three other clustering techniques
 - Brohee S, van Helden J (2006) Evaluation of clustering algorithms for protein–protein interaction networks. *BMC Bioinformatics* 7: 488
 - MCL vs. Restricted Neighborhood Search Clustering (RNSC) vs. Super Paramagnetic Clustering (SPC) vs. Molecular Complex Detection (MCODE)

Analysis of MCL



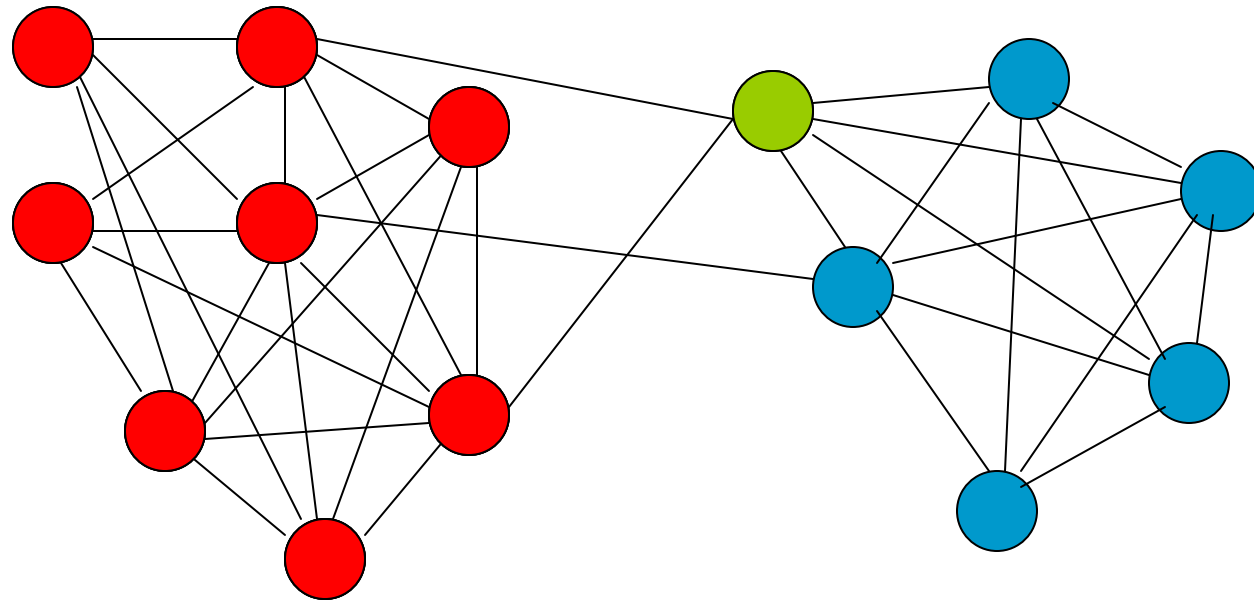
Each curve represents the value of accuracy (left panels) or separation (right panels).

(A-B) edge addition to the test graph. (C-D) edges removal from the test graph. (E-F) Edge removal from an altered graph with 100% of randomly added edges. (G-H) Edge addition to an altered test graph with 40% of randomly removed edges.

Color code: *blue* : MCL, *red* : RNSC, *orange* : MCODE, *green* : SPC. Dotted lines show the results obtained by permuting the clusters (negative control).

MCL Compared to RRW

- Comparison to Repeated Random Walks (RRW)
 - RRW is another Graph Clustering Method.



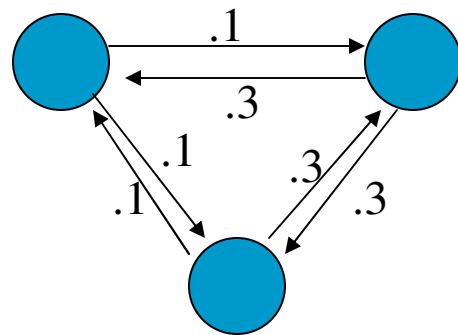
- Every cluster (including intermediate clusters) is stored.
- Clusters that overlap more than a threshold are later compared, and lower ranking clusters removed.

MCL Compared to RRW

- RRW Cluster p-value approximation:

$$\text{p-value} = 1 - (\text{score} \cdot \sqrt{n})$$

where n is the number of vertices in the cluster, and *score* is the average random walk distance between nodes in the cluster.



$$\text{score} = \frac{(.1 + .3 + .1 + .1 + .3 + .3)}{6} = .2$$

$$\text{p-value} = 1 - (.2 * \sqrt{3}) = 0.653$$



MCL Compared to RRW

- MCL, RRW, and a naïve nearest neighbor approach were run on a biological protein network for yeast (WI-PHI¹), as well as “noisy” versions of the network (edges added and deleted).
- Proteins with the same biological function should be clustered together
- The resulting clusters were compared to known protein groupings.

¹Kiemer L, Costa S, Ueffing M, Cesareni G: **WI-PHI: A weighted yeast interactome enriched for direct physical interactions.** *Proteomics* 2007, 7:932–943.

MCL Compared to RRW

Table 1: Results for the WI-PHI network

% in same MIPS category	RRW	MCL	Naïve
90+%	55%	17%	7.8%
80+%	69%	30%	19%
70+%	74%	42%	31%
60+%	84%	57%	44%
50+%	92%	77%	70%
25+%	100%	99%	98%

Table 2: Results for the FP40 network

% in same MIPS category	RRW	MCL	Naïve
90+%	56%	20%	6.3%
80+%	71%	34%	17%
70+%	78%	50%	26%
60+%	83%	63%	37%
50+%	92%	82%	63%
25+%	98%	99%	96%

Table 3: Results for the FN40 network

% in same MIPS category	RRW	MCL	Naïve
90+%	55%	10%	7.7%
80+%	67%	23%	19%
70+%	73%	32%	26%
60+%	80%	49%	44%
50+%	89%	79%	63%
25+%	99%	98%	95%

Table 4: Results for the Rewire40 network

% in same MIPS category	RRW	MCL	Naïve
90+%	33%	17%	4.8%
80+%	54%	32%	13%
70+%	58%	46%	21%
60+%	65%	65%	33%
50+%	76%	81%	57%
25+%	95%	98%	93%

Table 6: Precision, Recall, and Accuracy on preselected MIPS clusters

Network	Precision			Recall			Accuracy		
	RRW	MCL	Naïve	RRW	MCL	Naïve	RRW	MCL	Naïve
WI-PHI	.771	.512	.363	.672	.832	.791	.720	.657	.535
FP40	.791	.538	.362	.689	.768	.795	.738	.643	.537
FN40	.533	.423	.326	.536	.670	.699	.534	.532	.477
Rewire40	.654	.480	.370	.520	.565	.706	.583	.520	.511

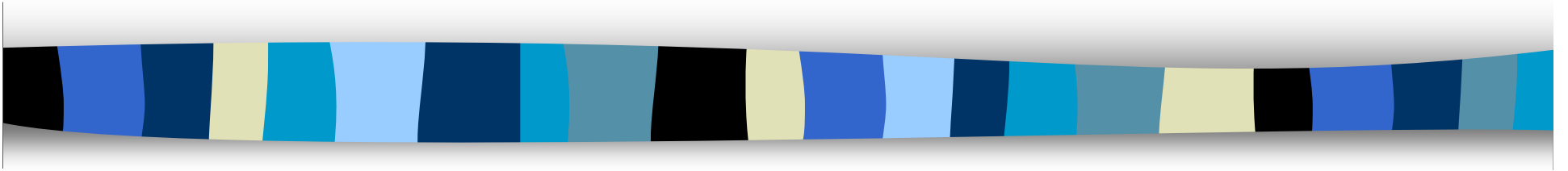
Average cluster size: RRW = 6, MCL = 12, Naïve = 9



Analysis of MCL

- Scales well with increasing graph size.
- Works with both weighted and unweighted graphs.
- Produces good clustering results.
- Robust against noise in graph data
- Number of clusters not specified ahead of time, but can adjust cluster granularity with parameters.
- Cannot find overlapping clusters (in general)
- Not suitable for clusters with large diameter.

Thank You!



Any Questions?