Performance Criteria for Graph Clustering and Markov Cluster Experiments

S. van Dongen

Information Systems (INS)

INS-R0012 May 31, 2000

Report INS-R0012
ISSN 1386-3681

CWI
P.O. Box 94079
1090 GB Amsterdam
The Netherlands

Performance Criteria for Graph Clustering and Markov Cluster Experiments

Stijn van Dongen
CWI
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

ABSTRACT
In [6] a cluster algorithm for graphs was introduced called the Markov cluster algorithm or MCL algorithm. The algorithm is based on simulation of (stochastic) flow in graphs by means of alternation of two operators, expansion and inflation. The results in [8] establish an intrinsic relationship between the corresponding algebraic process (MCL process) and cluster structure in the iterands and the limits of the process. Several kinds of experiments conducted with the MCL algorithm are described here. Test cases with varying homogeneity characteristics are used to establish some of the particular strengths and weaknesses of the algorithm. In general the algorithm performs well, except for graphs which are very homogeneous (such as weakly connected grids) and for which the natural cluster diameter (i.e. the diameter of a subgraph induced by a natural cluster) is large. This can be understood in terms of the flow characteristics of the MCL algorithm and the heuristic on which the algorithm is grounded.

A generic performance criterion for clusterings of weighted graphs is derived, by a stepwise refinement of a simple and appealing criterion for simple graphs. The most refined criterion uses a particular Schur convex function, several properties of which are established. A metric is defined on the space of partitions, which is useful for comparing different clusterings of the same graph. The metric is compared with the metric known as the equivalence mismatch coefficient.

The performance criterion and the metric are used for the quantitative measurement of experiments conducted with the MCL algorithm on randomly generated test graphs with 10000 nodes. Scaling the MCL algorithm requires a regime of pruning the stochastic matrices which need to be computed. The effect of pruning on the quality of the retrieved clusterings is also investigated.

2000 Mathematics Subject Classification: 05A18, 05B20, 15A48, 15A51, 62H30, 68R10, 68T10, 90C35.
Keywords and Phrases: Clustering, graph clustering, flow simulation, partition metric.
Note: Mathematical aspects of the MCL process are described in [8]. The work was carried out under project INS–3.2, Concept Building from Key–Phrases in Scientific Documents and Bottom Up Classification Methods in Mathematics.

1. Introduction

The Markov cluster algorithm or MCL algorithm was introduced in [6]. Its definition is repeated here and a summary of results from [6] and [8] is given. The report corresponds with Chapters 9, 10, and 12 in the PhD thesis [7]. The MCL algorithm is basically a shell around an algebraic process defined for stochastic matrices, called the MCL process. The process consists of alternation of two operators, called *expansion* and *inflation*. The image of a stochastic matrix under expansion is just some finite power of the matrix, according to the normal matrix product. The image of a (column) stochastic matrix under inflation is formed by raising each entry of the matrix to the same positive power, followed by a rescaling of the columns such that the result is stochastic again. An MCL process is defined by a column stochastic input matrix T_1 and two rows[1] of exponents $e_{(i)}$, $r_{(i)}$, resulting in a row of stochastic matrices $M_{(i)}$, defined by

$$T_{2i} = \mathrm{Exp}_{e_i}(T_{2i-1}) \qquad i = 1, \ldots \tag{1.1}$$

$$T_{2i+1} = \; ,_{r_i}(T_{2i}) \qquad i = 1, \ldots \tag{1.2}$$

[1] The notation $e_{(i)}$ is shorthand for $\{e_i\}_{i \in I\!N}$ and likewise $r_{(i)}$ for $\{r_i\}_{i \in I\!N}$.

```
                                      # $G$ is a voidfree graph.
                                      # $e_i \in I\!N, e_i > 1, i = 1, \ldots$.
MCL $(G, \Delta, e_{(i)}, r_{(i)})$ {          # $r_i \in I\!R, r_i > 0, i = 1, \ldots$.

    $G = G + \Delta$;                         # Possibly add (weighted) loops.
    $T_1 = \mathcal{T}_G$;                    # Create associated Markov graph
                                      # according to Definition 1.
    for $k = 1, \ldots, \infty$ {
        $T_{2k} = \mathrm{Exp}_{e_k}(T_{2k-1})$;
        $T_{2k+1} = \,,_{r_k}(T_{2k})$;
        if ($T_{2k+1}$ is (near–) idempotent)    break;
    }
    Interpret $T_{2k+1}$ as clustering according to Definition 3;
}
```

Figure 1: The basic *MCL* algorithm.

where $\mathrm{Exp}_{e_i}(T)$ is simply T^{e_i}, and $e_i \in I\!N$ and $r_i \in I\!R$, $r > 0$. The image under the *inflation operator* $,_r$ of a nonnegative matrix $M \in I\!R^{k \times l}$ having no zero columns is defined by

$$(\,,_r M)_{pq} \;=\; (M_{pq})^r \Big/ \sum_{i=1}^{k} (M_{iq})^r$$

The heuristic behind the *MCL* process is its expected behaviour for stochastic graphs[2] possessing cluster structure. It is to be expected that if a graph possesses cluster structure, random walks will tend to stay in the same cluster relatively long before moving to another cluster. The largest transition probabilities will generally correspond with nodes lying in the same 'natural cluster' (in the presence of cluster structure). The effect is boosted in the *MCL* process via the inflation operator. Via expansion, transition probabilities corresponding with random walks of higher length are computed. The inflation operator promotes larger transition probabilities at the cost of small probabilities (if $r > 1$). It will thus promote random walks between nodes lying in the same cluster, and demote random walks between nodes lying in different clusters. In practice the default is that all expansion values e_i are chosen equal to two, and the inflation values r_i are chosen greater than one. For some extreme cases which are mainly of theoretical interest it is beneficial to use inflation values smaller than one (see Section 6).

In Figure 1 the basic layout of the *MCL* algorithm is given. In this setup the algorithm has four parameters. The first is the graph to be clustered, the second is a diagonal matrix Δ representing loop weights[3], and the third and fourth are the parameter rows for respectively expansion and inflation. Given an input graph $G = (V, w)$, the *MCL* algorithm adds for each node q a loop of weight Δ_{qq} to G by increasing $w(q, q)$ with the amount Δ_{qq}. Subsequently the columns of the associated matrix are rescaled such that the result T_1 is stochastic (Definition 1). This is the standard way of defining transition probabilities inducing random walks on a graph [14]. It is argued in [6] that this localization step is natural, as global fluctuations of the weight function over the graph are difficult to interpret for the purpose of clustering, and may cause heavy–weight

[2]I.e. weighted graphs for which the associated incidence matrix is stochastic.

[3]Experiments show that adding loops to a graph is in many cases a beneficial manoeuvre. Some remarks on this are made on page 7.

regions to dominate and gobble up neighbouring regions of lower weight. Such is the case for k–path clustering, a variant of single link clustering defined in terms of *path numbers* or *capacities* [6]. In the setting of the *MCL* algorithm a node q can still be moved far away from *all* its neighbours by increasing the weight of its loop (making Δ_{qq} large). The *MCL* algorithm proceeds by computing an *MCL* process for the matrix T_1. In the experiments described in this report the expansion row $e_{(i)}$ is taken constant equal to two everywhere; the inflation row $r_{(i)}$ assumes a constant on a prefix of some finite length, and assumes another constant on the postfix of infinite length (see the legend on page 8).

In practice the process converges very fast towards a matrix which is idempotent under both expansion and inflation. Such a matrix is called **doubly idempotent**. The limits of an *MCL* process are in general extremely sparse. The process furthermore has the power to separate an irreducible component of a matrix (a connected component of the associated graph) into several such components. By this it is meant that the limit may have structural properties which are different from the structural properties of the input matrix and any of the iterands. The structural properties of the limit are interpreted as a clustering of the input graph G, via the mapping of nonnegative idempotent matrices onto clusterings in Definition 3. Additionally, the iterands of the *MCL* process also posses a structural property which allows a mapping onto directed acyclic graphs generalizing the mapping given in Definition 3. This is stated in Theorem 2.

The *MCL* process converges quadratically in the neighbourhood of doubly idempotent stochastic matrices for which all columns have precisely one nonzero entry. It converges quadratically on a macroscopic scale (in terms of block structure) for doubly idempotent stochastic matrices in general. If a limit column has more than one nonzero entry, this implies either cluster overlap (Definition 3) or the presence of a so-called attractor system of cardinality greater than one (Definition 2). The clustering associated with such a matrix is stable under perturbations of the *MCL* process (that is, it is essentially defined by the block structure), except for the phenomenon of overlap [6]. Limits having columns with more than one nonzero entry seem to imply the existence of an automorphism of the input graph[4]. Examples are given in Sections 3 and 4.

The following concepts are useful for associating clusterings with different types of matrices. A **strongly connected component** of a directed graph is (the subgraph induced by) a maximal set of nodes such that there is a path from every node to every other node in the subgraph induced by the nodes. A **weakly connected component** of a directed graph G is a maximal set of nodes T such that at least one strongly connected component S is contained in T and such that all nodes from which there is a path in G to S are contained in T as well. In the set of all weakly connected components of G each one of them is guaranteed to contain at least one strongly connected component not contained in any of the other weakly connected components.

The graph notation used in this report follows standard conventions, with one exception. Given a directed graph G with node set V and nonnegative weight function w defined on V^2, there is said to be *an arc from q to p with weight $w(p,q)$, $p,q \in V$*, iff $w(p,q) > 0$. This way of mapping the ordered pair (p,q) onto an arc was chosen in [6] because the *MCL* algorithm was introduced using column stochastic matrices.

Definition 1 *Let $G = (V, w)$ be a directed graph, where V is a set of n nodes and w is a nonnegative weight function defined on V^2. The sum of the weights of all arcs originating from a node q is written W_q, thus $W_q = \sum_p w(p,q)$. The associated Markov matrix of G is written $\mathcal{T}_G$. lies in $\mathbb{R}_{\geq 0}^{n\times n}$, and is defined by*

$$\left[\mathcal{T}_G\right]_{pq} = \frac{w(p,q)}{W_q}$$

[4]This does not imply that the *MCL* process is useful to detect graph automorphisms, since these limits are instable under inflation. It means that in practice *MCL* limits tend to be extremely sparse, which is useful for scaling the *MCL* algorithm.

The theorem below is preparatory to the mapping of nonnegative column allowable idempotent matrices onto overlapping clusterings[5]. The qualifier **column allowable** means that a matrix has no zero columns.

Theorem 1 [6] *Let M be a nonnegative column allowable idempotent matrix of dimension N, let G be its associated graph. For s, t, nodes in G, write $s \rightarrow t$ if there is an arc in G from s to t. By definition, $s \rightarrow t \iff M_{ts} \neq 0$. Let α, β, γ be nodes in G. The following implications hold.*

$$(\alpha \rightarrow \beta) \wedge (\beta \rightarrow \gamma) \implies \alpha \rightarrow \gamma \tag{1.3}$$
$$(\alpha \rightarrow \alpha) \wedge (\alpha \rightarrow \beta) \implies \beta \rightarrow \alpha \tag{1.4}$$
$$\alpha \rightarrow \beta \implies \beta \rightarrow \beta \tag{1.5}$$

Definition 2 *Let G be the associated graph of a nonnegative column allowable idempotent matrix M of dimension n, with nodes labeled $1, \ldots, n$. The node α is called an* **attractor** *if $M_{\alpha\alpha} \neq 0$. If α is an attractor then the set of its neighbours is called an* **attractor system**. □

By Theorem 1, each attractor system in G induces a weighted subgraph in G which is complete. These form the unique cores of the clustering associated with (nonnegative idempotent) M as stated below.

Definition 3 *Let M be a nonnegative column allowable idempotent matrix of dimension n, let G be its associated graph on the node set $V = \{1, \ldots, n\}$. Let $E_i, i = 1, \ldots, k$ be the different attractor systems of G. For $v \in V$ write $v \rightarrow E_i$ if there exists $e \in E_i$ with $v \rightarrow e$. Theorem 1 then implies that $v \rightarrow f$ for all $f \in E_i$. The (possibly) overlapping clustering $\mathcal{C} = \{C_1, \ldots, C_k\}$, associated with M, is defined by*

$$C_i = E_i \cup \big\{v \in V \mid v \rightarrow E_i\big\} \tag{1.6}$$

Alternatively, $\mathcal{CL}$ is defined as the set of weakly connected components of the graph associated with M.

A matrix A is called **diagonally positive semi-definite** or **dpsd**, if it is diagonally similar to a positive semi-definite matrix, that is, if there is a diagonal matrix d such that $d^{-1}Ad$ is positive semi-definite. It was shown in [8] that if a symmetric matrix is input to the *MCL* process by replacing it with its associated Markov matrix according to Definition 1, then all even-labeled iterands are guaranteed to be *dpsd*, assuming that the expansion parameter is constant equal to two. This means precisely that the even-labeled matrices computed by the *MCL* algorithm are *dpsd* (if the expansion parameter is constant equal to two). A *dpsd* matrix has an interesting property relating its diagonal entries to the off-diagonal entries. The result in [8] is proven for the most general case of matrices defined over the complex numbers, but the result is stated here simply for nonnegative real matrices.

Theorem 2 [8] *Let A be* dpsd *and nonnegative real square of dimension n. For p, q indices in $1, \ldots, n$, write $q \looparrowright p$ iff $A_{pq} \geq A_{qq}$. Write $p \sim q$ iff columns p and q are identical.*

The arc $\looparrowright$ defines a directed acyclic graph (DAG) on $\{1, \ldots, n\}/\sim$.

The theorem can be used to associate a clustering with a *dpsd* matrix by taking as unique cores of the clustering all the endclasses of the associated *DAG*, and by joining each core with all the nodes that reach it in the *DAG*. It was shown in [8] that there is a rather precise sense in which this mapping from *dpsd* matrices onto clusterings is related to the mapping of nonnegative doubly idempotent matrices onto clusterings. In

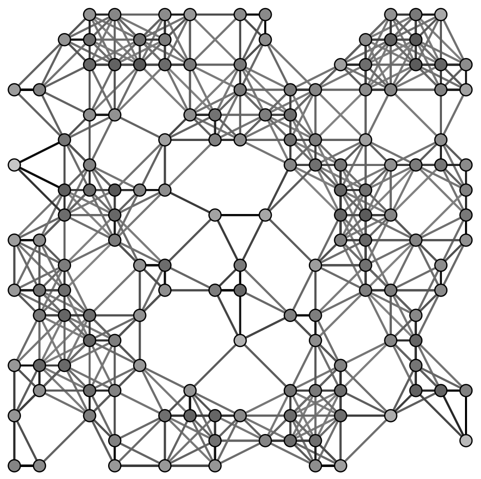


Figure 2: A geometric graph.

practice, mapping an *MCL* iterand onto an overlapping clustering is best done by defining a graph H by $\looparrowright$ as in the theorem, and subsequently taking as a cluster all the weakly connected components of H.

The *MCL* process is first illustrated for the graph in Figure 2, taken from the article *A computational study of graph partitioning* [10]. Two nodes are connected if the distance between them is at most $\sqrt{8}$ units. The strength of a bond is inversely related to the distance, and corresponds with its grey level. Note that this is a peculiar type of graph because the similarity function on the nodes is derived from their depicted location (which means that a simple transformation step makes the graph embeddable in Euclidean space). The *MCL* process is first illustrated for this graph in Figure 3. Four iterands are depicted from left to right and top to bottom. For each node, at most sixteen neighbours are shown. The bottom right graph corresponds with the limit from this particular *MCL* process. The degree of shading of a bond between two nodes indicates the maximum value of flow, taken over the two directions: the darker the bond the larger the maximum. The degree of shading of a node indicates the total amount of incoming flow. Thus, a dark bond between a white node and a black node indicates that the maximum flow value is found in the direction of the dark node, and that hardly any flow is going in the other direction.

The bottom right graph L represents a situation where flow is constant; the dark nodes are attractors and the bonds indicate which nodes are attracted to them. The corresponding matrix is idempotent. The picture contains all necessary information needed to reconstruct the matrix — the limit of an *MCL* process is in general highly structured. The graph L generically induces a clustering of the input graph by taking as clusters all the weakly connected components (Definition 3). It can be seen from L that all the attractor systems have cardinality one and that no overlap occurs. The limit matrix is thus extremely sparse, as all columns have precisely one nonzero entry.

Scaling the MCL algorithm

The key to scaling the *MCL* algorithm lies in the observation just made. The limit of an *MCL* algorithm is in general extremely sparse, and the iterands are sparse in a weighted sense. That is, the iterands fill very rapidly, but the large majority of the entries stays very small throughout. The nonzero entries of a column are usually

[5]For this theorem it is not necessary to assume that the matrices are stochastic, nor that they are idempotent under inflation.

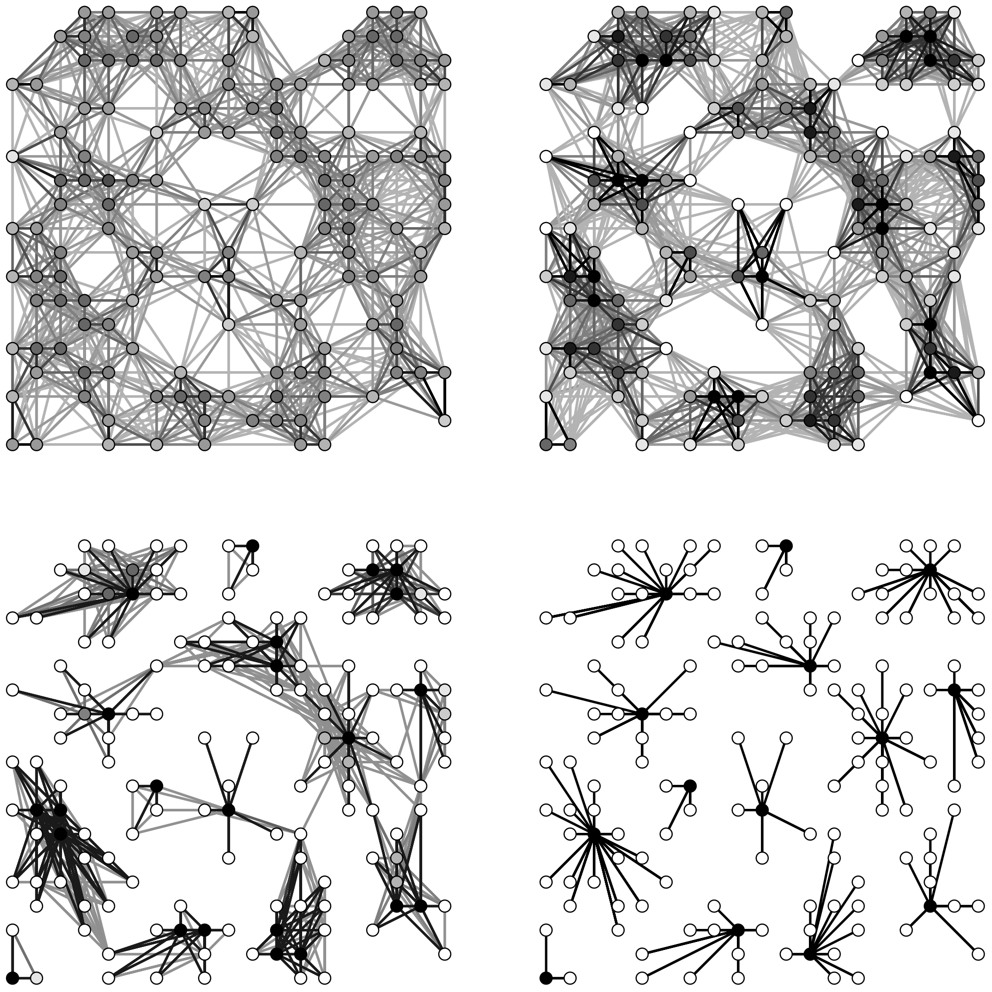


Figure 3: Successive stages of flow simulation by the MCL process.

distributed very inhomogeneously, especially if cluster structure is present and the diameter of the natural clusters is not too large. The heuristic behind the process supports the idea that it will not harm (for the purpose of clustering) if for each column only the largest entries are kept, if they account for a large fraction of the column mass. In order to maintain the stochastic interpretation of the process, it is then necessary to rescale a pruned column to have sum one again. These issues are discussed in some more detail in [6], though no attempt is made at a perturbation analysis of the proposed pruning schemes.

A particularly simple idea is to fix an integer k and prune the columns of each newly computed matrix product (after expansion) such that they have at most k nonzero entries again. However, it is quite costly to compute the k largest entries of a vector that may have k^2 nonzero entries. In order to overcome this problem, it is proposed to apply *automatic pruning*, that is, removal of all entries that are smaller than a bound which is computed as some function of the vector to be pruned. Automatic pruning is then followed by exact pruning (keeping at most k entries per column). Such a setup was created for the scaled experiments conducted in Section 12. The bounding function is defined in terms of a particular Schur convex function which also plays a role in a generic performance criterion for graphs defined in Section 10. The effect of pruning is studied in Section 12, using test graphs for which a priori good clusterings are known and using different pruning constants k.

Adding loops
For small graphs and graphs with bipartite characteristics such as rectangular grids, adding loops is a beneficial manoeuvre. The possible dependence of the transition probabilities on the parity of the simple path lengths in the graph is removed. More generally, adding loops of weight c to a graph has the effect of adding c to all the eigenvalues in its spectrum, and negative eigenvalues are known to correspond with oscillatory behaviour of the associated matrix. The effect of adding loops on the output clusterings of the *MCL* algorithm is that connectedness (with respect to the input graph) of the clusters in the output clustering is promoted, and that the granularity of the output clustering is increased. The latter is reflected in the fact that adding loops increases the number of endclasses of the associated *DAG* of a *dpsd* matrix (see Theorem 2).

2. Organization

In Sections 3–8 a series of examples is given illustrating various characteristics of the *MCL* algorithm. The phenomenon of attractor systems is briefly discussed in Section 3, and cluster overlap is the subject of Section 4. In Section 6 the *MCL* algorithm is applied to various small torus graphs to see whether it is able to recognize a particular characteristic of their structure. In Section 7 it is shown that the *MCL* algorithm is in general not suited for detecting cluster structure if the diameter of the natural clusters is large. The examples used are homogeneous neighbourhood graphs derived from two dimensional data. A common approach towards detection of clusters in neighbourhood graphs is by finding the borders that separate them. In Section 8 it is shown that early stages of the *MCL* process yield information that can be used to this end.

Sections 9–11 contain measures for judging the quality of clusterings. In Section 9 a criterion for simple graphs is derived by demanding that it yields the perfect score 1 if and only if the graph is a direct sum of complete graphs and the clustering is the corresponding canonical clustering. This criterion is first formulated globally in terms of the incidence matrix associated with the simple graph, and then in terms of an average of scores for all nodes in the graph. The latter formulation is used in Section 10 to derive a formula that measures how efficient a characteristic vector (corresponding with a cluster) captures the mass of a nonnegative vector (corresponding with an element or node contained within the cluster). The average score over all nodes in the graph yields the weighted performance criterion for clusterings of nonnegative graphs. The formula contains a scaling factor which is a Schur convex function on the set of nonnegative nonzero vectors. Several properties of this function are derived.

A formula for a metric distance between two partitions (non-overlapping clusterings) is given in Section 11. This distance is computed as the sum of two numbers, where each number represents the distance of one of the two partitions to the intersection of the two partitions. If one of the two numbers is very small compared to the other, this implies that the corresponding partition is nearly a subpartition of the other partition. The metric is compared to the metric known as the equivalence mismatch coefficient.

In Sections 12 and 13 the *MCL* algorithm is tested against randomly generated simple graphs for which it is a priori known that they posses natural cluster structure. The graphs which are used have small diameter and so do (the subgraphs induced by) their natural clusters. Section 12 describes how test graphs are generated and contains a small-scale example with a graph on 160 nodes. Section 13 gives an account of two experiments. In the first experiment the *MCL* algorithm was applied to three graphs on 10000 nodes with identical cluster structure but with different density characteristics. Each graph was input to several *MCL* runs. In each run a different pruning constant k was applied while holding other *MCL* parameters fixed. The results indicate that pruning works well, which is explained by the small diameter of the natural clusters. In the second experiment eight graphs were generated (again on 10000 nodes) having the same density characteristics but different granularity characteristics. They were clustered using the exact same *MCL* parametrization for each graph. The resulting clusterings are generally quite good, indicating that the *MCL* algorithm can handle graphs with large natural clusters, and that it depends mainly on the diameter and the density characteristics of the natural clusters which parametrizations give the best result. The experiments indicate that fine-tuning the pruning regime considerably helps the performance of the *MCL* algorithm.

MCL legend

In order to describe the results of *MCL*–runs on various test-graphs for various parametrizations, the legend for *MCL* parametrizations is introduced. The experiments allowed input rows $e_{(i)}$, $r_{(i)}$ which have very simple structure. The row $e_{(i)}$ simply consists of twos only. The row $r_{(i)}$ is constant on a tail of infinite length, and may assume another constant on a prefix of length l, where l can be specified as well. This amounts to three parameters related to the input rows specifying *MCL* process parameters. The fourth parameter indicates whether loops are added to an input graph G. If this parameter assumes a value $c \in I\!R_{\geq 0}$, the program takes the graph $G + cI$ as actual input. The parameter labels, their meaning, and default setting are found in Table 1. The length l of the initial prefix is indicated by '`l`', the constant value assumed by $r_{(i)}$ on the initial prefix by '`r`', the constant value on the infinite postfix by '`R`', and the loop weight by '`a`' (stemming from auto-nearness). The main use of introducing a default setting is that in many examples the simplest possible parametrization is chosen, where the initial prefix length is equal to zero. The default setting corresponds with an *MCL* process for which $e_{(i)} \overset{c}{=} 2$ and $r_{(i)} \overset{c}{=} 2$ and no loops added[6].

Parameter	Meaning	Default setting
`a`	Loop weight	0
`r`	Initial inflation constant	2
`l`	Initial prefix length	0
`R`	Main inflation constant	2

Table 1: *MCL* implementation legend.

3. Attractors

In practice, for input graphs constructed from real applications for the purpose of clustering (and thus with no strong symmetries present), the equivalence classes $E_1, \ldots, E_d$ (see Definition 3) tend to be singleton sets. In all situations observed so far where this is not the case, the elements in an equivalence class of cardinality greater than one are the orbit of a graph automorphism.

Consider the graph G_1 in Figure 6 on page 9. An *MCL* run with parametrization $\texttt{a} = 1$ results in 4 clusters, each of which is a triangle in the graph, and where each node is an attractor. Printing attractors in boldface, this is the clustering $\{\{\mathbf{1}, \mathbf{2}, \mathbf{3}\}, \{\mathbf{4}, \mathbf{5}, \mathbf{6}\}, \{\mathbf{7}, \mathbf{8}, \mathbf{9}\}, \{\mathbf{10}, \mathbf{11}, \mathbf{12}\}\}$. The cluster $\{\mathbf{1}, \mathbf{2}, \mathbf{3}\}$, and likewise the other clusters, is the orbit of either of its elements under rotation of G_1 around the symmetry axis orthogonal to the plane spanned by $1, 2$, and 3.

For a less symmetric example, consider the graph G_2 in Figure 5. Clustering G_2 with parametrization $\texttt{a} = 1$ results in the clustering $\{4, 8, \mathbf{9}, \mathbf{11}, 12\}$, $\{1, 6, 7, 10\}$, $\{2, 3, 5\}$. The attractor system $\{9, 11\}$ corresponds with the graph automorphism interchanging 9 and 11 and keeping all other nodes fixed.

Generally, attractors are located in local centra of density, which is best illustrated by large graphs with clear islands of cohesion. If a graph fits the picture, so to speak, of a 'gradient of density', the attractors are found in the thickest parts of the graph. This effect can be seen in Figure 3, and it is to some extent also illustrated by the graph shown in Figure 7 on page 11, with several clusterings resulting from differently parametrized

[6]The notation $e_{(i)} \overset{c}{=} 2$ is shorthand for $e_i = 2, i \in I\!N$.

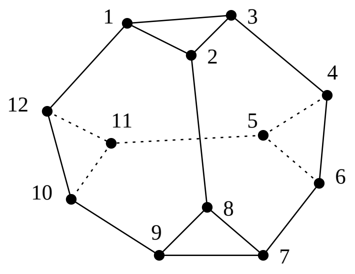


Figure 4: Graph G_1.

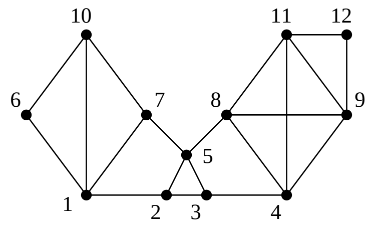


Figure 5: Graph G_2.

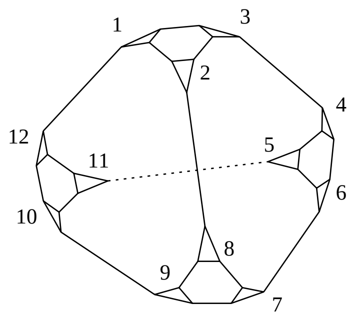


Figure 6: Graph G_3.

MCL processes. For this graph, it interferes however with another phenomenon that occurs when a graph possesses borders. In that case, the return probabilities of nodes which lie just before those borders, profit immediately and maximally after one expansion step from the 'dead end' characteristic of the border. The border of the graph in Figure 7 is the outline of the grid. This explains why all of the attractors in Figure 7 are nodes lying at distance one from the outline of the grid.

4. Overlap

The phenomenon of overlap in the setting of undirected graphs has only been observed so far for input graphs with specific symmetry properties. For these cases, if the *MCL* algorithm produces two clusters C_1, C_2 with nonempty intersection, there exists an automorphism which transforms C_1 into C_2 while leaving the intersection invariant. Existence of such an automorphism means that the overlapping part forms a subset of the graph from which the graph looks the same in different directions. If those different directions correspond also with different islands of cohesion, it is rather nice if the overlapping part is not arbitrarily divided among the resulting clusters. An example of this phenomenon can be found in Figure 7. Overlap occurs at several levels of granularity, and it always corresponds with a symmetry of the graph. For undirected graphs, the amount of possible overlap tends to be proportional to the amount of symmetry present. Any amount of overlap can be constructed by taking appropriate directed input graphs.

Small perturbations in the input graph generally do not affect the output clustering produced by the *MCL* algorithm. Overlap is an exception to this, as discussed in [6]. If the symmetry corresponding with the overlap is perturbed, the overlap disappears. Moreover, the phenomenon of overlap is intrinsically unstable. For small-scale graphs such as in Figure 7 the *MCL* process generally converges so fast that idempotency is reached before instability starts to play a role, and the matrix computations can be done without pruning. This is certainly not the case for large graphs.

5. The effect of inflation on cluster granularity

There is a clear correlation between the inflation parameter and the granularity of the resulting output. The higher the parameter r, the more the inflation operator Γ_r demotes flow along long path distances in the input graph. This is illustrated in Figure 7, where the result of six *MCL* runs for the same halter-shaped grid are given. The inflation parameter is varied from 1.4 to 2.5, while all other parameters are kept the same (i.e. $\texttt{a} = 1$ $\texttt{E} = 2$). Note that the corresponding overlapping clusterings are strongly related to each other. The set of all clusterings excluding the one corresponding with inflation parameter $\texttt{R} = 1.4$ is a set of nested overlapping clusterings. This is very satisfactory, as one expects clusters at different levels of granularity to be related to each other. The clusterings at the first three levels $\texttt{R} = x$, $x \in \{1.4, 1.5, 1.7\}$, have good visual appeal. It holds for all clusterings that the sizes of the respective clusters are evenly distributed, except perhaps for the clustering with parameter R $= 2.0$.

The second example in which the inflation parameter is varied while other parameters are kept the same concerns the graph G_3 in Figure 4. It is derived from the graph G_1 in Figure 6 by replacing each of the 12 nodes in G_1 by a triangle. Note that G_3 is a simple graph: the lengths of the edges in the picture do not correspond with edge weights. Now G_3 clearly allows two extreme clusterings $\mathcal{P}_1 = \{\text{singletons}(V)\}$ and $\mathcal{P}_4 = \{V\}$, a clustering $\mathcal{P}_2$ in which each of the newly formed triangles forms a cluster by itself, and a clustering $\mathcal{P}_3$ with 4 clusters in which each cluster consists of the 9 nodes corresponding with 3 newly formed triangles. Clustering with parameters a $= 1$ E $= 2$ R $= x$, where x varies, yields the following. Choosing $x \in [1.0, 1.2]$ results in the top extreme clustering $\mathcal{P}_4$, choosing $x \in [1.3, 1.4]$ in the clustering $\mathcal{P}_3$, choosing $x \in [1.4, 3.0]$ in the clustering $\mathcal{P}_2$, and choosing $x \in [3.1, \infty]$ results in the bottom extreme clustering $\mathcal{P}_1$. The range of x for which the clustering $\mathcal{P}_4$ results is small. This has to do with the fact that the clustering $\mathcal{P}_4$ is rather coarse. The dependencies associated with $\mathcal{P}_4$ correspond with longer distances in the graph G_3 than the dependencies associated with $\mathcal{P}_3$. If the inflation parameter increases, the latter dependencies (in the form of random walks) soon profit much more from the inflation step than the former dependencies. By letting expansion continue a while before starting inflation, this can be remedied. Table 2 shows several parameter settings and the resulting clusterings.

6. Flow on torus graphs

The following examples are rectangular torus-graphs. A k-dimensional rectangular torus graph generalizes a ring graph in k dimensions. It is most conveniently defined as a sum of ring graphs, defined on the Cartesian product of the respective node sets.

Definition 4 *Let $(G_i = (V_i, w_i)), i = 1, \ldots, n$ be an n-tuple of simple graphs. The* **sum graph** *S of $G_1, \ldots, G_n$*

Parametrization			
l	r	R	Clustering
0	–	1.0 – 1.2	$\mathcal{P}_4$
0	–	1.3 – 1.4	$\mathcal{P}_3$
0	–	1.5 – 3.0	$\mathcal{P}_2$
0	–	3.0 – ∞	$\mathcal{P}_1$
1	1	1.0 – 1.3	$\mathcal{P}_4$
1	1	1.4 – 1.7	$\mathcal{P}_3$
1	1	1.8 – 5.3	$\mathcal{P}_2$
1	1	5.4 – ∞	$\mathcal{P}_1$
2	1	1.0 – 1.4	$\mathcal{P}_4$
2	1	1.5 – 2.4	$\mathcal{P}_3$
2	1	2.5 – 6.8	$\mathcal{P}_2$
2	1	6.9 – ∞	$\mathcal{P}_1$

a $= 1$ set everywhere

Table 2: *MCL* runs for the graph G_3 in Figure 4. The clusterings $\mathcal{P}_1, \ldots, \mathcal{P}_4$ are defined in the text above.

x	l	r	R
5	2	1.2	2.1 – 3.2
	2	1.0	2.1 – 4.0
	2	0.8	2.1 – 5.3
6	2	1.2	2.1 – 2.4
	2	1.0	2.1 – 2.8
	2	0.8	2.1 – 3.3
7	3	1.2	2.1 – 2.7
	3	1.0	2.3 – 3.9
	3	0.8	2.9 – 6.4
8	3	1.2	2.1 – 2.3
	3	1.0	2.3 – 2.9
	3	0.8	2.9 – 4.3
9	3	1.2	2.1
	3	1.0	2.3 – 2.5
	3	0.8	2.9 – 3.3

a $= 1$ set everywhere

Table 3: Parametrizations for which the *MCL* algorithm finds 10 clusters of size x each for the input graph $TORUS(10, x)$, $x = 5, \ldots, 9$.

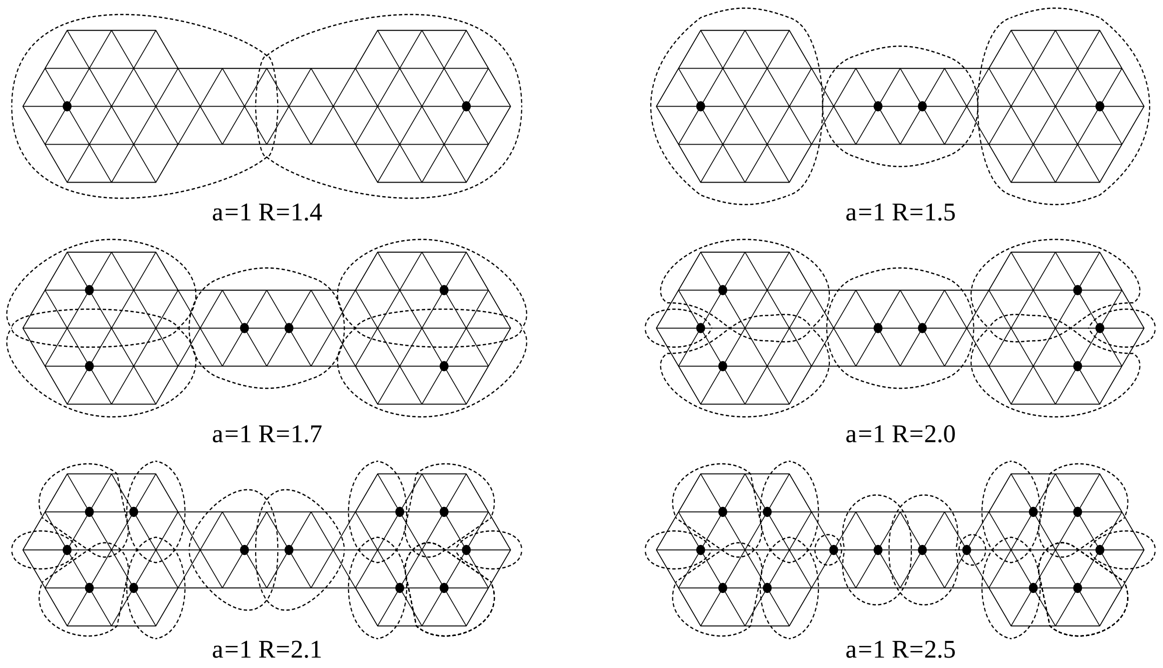


Figure 7: Different clustering of a grid for varying inflation values. Dotted nodes are attractors.

is defined on the Cartesian product $V_1 \times \cdots \times V_n$. Two vertices $(x_1, \ldots, x_n)$ and $(y_1, \ldots, y_n)$ are connected in S if exactly one of the pairs (x_i, y_i) is connected in G_i, and $x_i = y_i$ for the remaining $n-1$ pairs.

Definition 5 *The* 1*-dimensional* **torus graph** *or* **ring graph** *of cardinality t is the simple graph defined on the integers modulo t: $0, \ldots, t-1$, where there is an edge between i and j iff $i \equiv j+1 \pmod{t}$ or $j \equiv i+1 \pmod{t}$.*

A graph is called a k-dimensional torus graph if it is the sum graph of k ring graphs. It can be identified with a k-tuple $(t_1, \ldots, t_k)$, where t_i is the cardinality of the node set of the i^{th} ring graph. The torus graph corresponding with this k-tuple is denoted $TORUS(t_1, \ldots, t_k)$. □

Here I will use only 2- and 3-dimensional simple torus graphs. A 2-dimensional torus graph $TORUS(k, l)$ can be thought of as a rectangular grid of width k and depth l, where nodes lying opposite on parallel borders are connected. In [6] it appeared that periodic MCL limits exist which have the same automorphism group as ring graphs. A two dimensional torus graph $G = TORUS(k, l)$ where $k = l$ has the same homogeneity properties as ring graphs. It is interesting to see what happens if $k > l$. Consider a node pair (u_1, u_2) lying on a ring of length l in G at a (shortest path) distance $t \leq l$ from each other, and a node pair (v_1, v_2) in G, also lying at distance t from each other, but not lying on such a ring. The transition probability associated with going in l steps from u_1 to u_2 is larger than the transition probability associated with going in l steps from v_1 to v_2, because u_1 can reach u_2 in two ways along the ring on which they both lie, while this is not true for v_1 and v_2. Is it possible to find an MCL process in which this effect is boosted such that a clustering of G in k clusters of size l each results? This is indeed the case, and it requires the usage of input rows $r_{(i)}$ which are not constant everywhere. If l is very close to k, it is furthermore beneficial to use an initial inflation parameter which is close to or smaller than 1. Without this, the return probability of each node grows too large before paths of length l start to have influence, which is after $\lceil \log_2(l) \rceil$ expansion steps (assuming $e_{(i)} \overset{c}{=} 2$). Table 3 shows parameter settings for which the MCL algorithm output divides the graphs $TORUS(10, x)$ in 10 clusters of cardinality x each, $x = 5, \ldots, 9$. These are of course not the only parametrizations achieving this, but among the parametrizations found they lead to fast convergence of the *MCL* process.

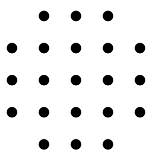
The last torus example is the 3-dimensional torus graph $TORUS(3,4,5)$. A priori it is to be expected that the non-extreme clusterings which the *MCL* algorithm can possibly produce are the clustering $\mathcal{P}_2$ corresponding with 20 subgraphs isomorphic to $TORUS(3)$ and the clustering $\mathcal{P}_3$ corresponding with 5 subgraphs isomorphic to $TORUS(3,4)$. Denote the bottom and top extreme clusterings by $\mathcal{P}_1 = \{\text{singletons}(V)\}$ and $\mathcal{P}_4 = \{V\}$ respectively. The table below gives four parameter ranges yielding the four clusterings $\mathcal{P}_i$.

Parametrization			
`l`	`r`	`R`	Clustering
2	1.2	1.0 − 2.3	$\mathcal{P}_4$
2	1.2	2.4 − 3.3	$\mathcal{P}_3$
2	1.2	3.4 − 6.4	$\mathcal{P}_2$
2	1.2	6.5 − ∞	$\mathcal{P}_1$

The torus examples illustrate the strong separating power of the *MCL* process. This is mainly interesting for a better understanding of the process, and probably not of much help in using the algorithm. The rewarding aspect of the torus examples is that abstract reasoning about the process applied to extreme cases is confirmed by experiments. The clusterings shown for the torus graphs, the tetraeder-shaped graphs in Figure 4, and the grid in Figure 7 illustrate that the *MCL* algorithm 'recognizes' structure even if the node degrees in the input graph are homogeneously distributed and the connectivity of the graph is high. The inflation parameter clearly is the main factor influencing the granularity of the output clusterings. The output clustering changes at specific values of the inflation parameter constant, and stays the same for the intervals in between. By prolonging expansion, coarser clusterings can be found.

The further experiments described below will exhibit a characteristic of the *MCL* algorithm that may be considered a weakness. It concerns the formation of clusters in graphs consisting of weakly connected grids, where certain clusters connect parts of different grids. The phenomenon is rather surprising at first, but it can be be understood in abstract terms. It is indicative for the fact that there are severe problems involved in applying graph cluster methods to neighbourhood graphs derived from vector data.

7. The *MCL* algorithm applied to grid–like graphs



In the upper left of Figure 8 a rectangular assembly of points in the plane is shown. A graph is defined on the black spots using the neighbourhood relation depicted to the right of this text (i.e. nodes correspond with black spots and there is an edge between two nodes if they are at most $\sqrt{5}$ units away). The weight of an edge is inversely proportional to the distance between the coordinates of its incident nodes. This is not essential for what follows, nor is the particular neighbourhood structure used. In Figure 8 three clusterings are depicted, corresponding with the simple parametrizations `R` $= 1.9$, `R` $= 2.3$, and `R` $= 2.7$. The *MCL* process applied to grid-like graphs such as these has the property that columns (equivalently, probability distributions of a node) begin to convergence towards a homogeneous state in the corners and borders first. While this happens, the converging parts of the distributions begin to assume the characteristics of a border themselves, as flow from the border region towards the centre is demoted. This explains the neat division of the graph into blocked patterns.

If the inflation parameter is chosen sufficiently low, the graph will be clustered into a single cluster. This requires considerable time, as the diameter of the graph is ten. Diagonally opposite corners build up distinctly

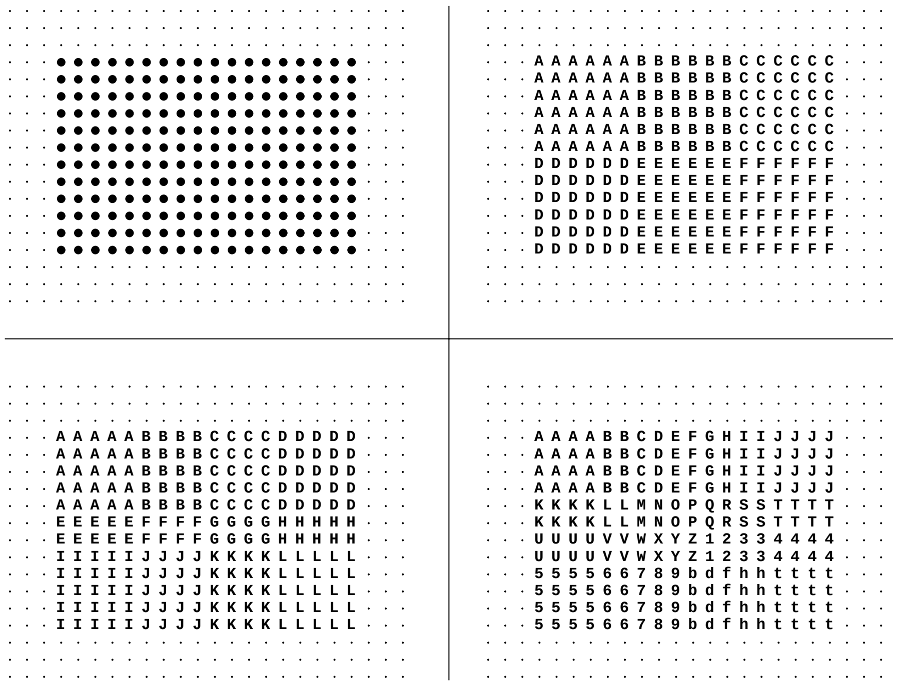


Figure 8: Three clusterings of the 18×12-node graph in the upper left for respective inflation values $\mathtt{R} = 1.9$, $\mathtt{R} = 2.3$, and $\mathtt{R} = 2.7$. The initial edges are defined by the neighbourhood structure depicted on page 12.

different probability distributions, and it requires several expansion steps at low inflation parameter to let them equalize. Though it is possible (using the parametrization $\mathtt{R} = 1.3$) in this simple setting, matters become more complicated if the graph is made part of a larger constellation. Four such constellations are depicted in Figure 9, where the one in the upper left is the same as before. The graphs on the grids are derived using the same neighbourhood structure as before, and Figure 9 shows the result of applying the *MCL* algorithm with parametrization $\mathtt{R} = 1.3$. The upper right clustering is remarkable in that the two small satellite subgraphs form clusters together with regions of the large rectangle. This is explained by the effect that the small subgraphs 'inflate' the probability distributions of the nodes of the rectangle lying on the opposing border (i.e. cause them to be less homogeneous). Random walks departing from these border nodes crossing over to the satellite subgraphs have a relatively small probability initially, but due to the low inflation parameter, and the fact that a satellite subgraph has a great absorbing quality, these probabilities win out easily in the end. This is further illustrated by Figure 10, in which all four constellations are again clustered for parametrization $\mathtt{R} = 1.5$. The crossing characteristics of the upper right constellation are now much less dramatic, but still present.

The clustering of the lower left constellation in Figure 9 is remarkable in that the natural bipartition of the large rectangle is rather along the south/north axis than the along the east/west axis. If the rectangle is clustered without satellite systems, then this is the only bipartition which can possibly result, which is explained by the fact that flow is sooner bound along the north/south axis than it is along the east/west axis. This can be compared to the clustering of a (k, l) torus graph, $k < l$, for which an *MCL* process will never result in a clustering into k rings of size l.

In the lower right constellation of Figure 9 it is noteworthy that the corner nodes are all attracted to the

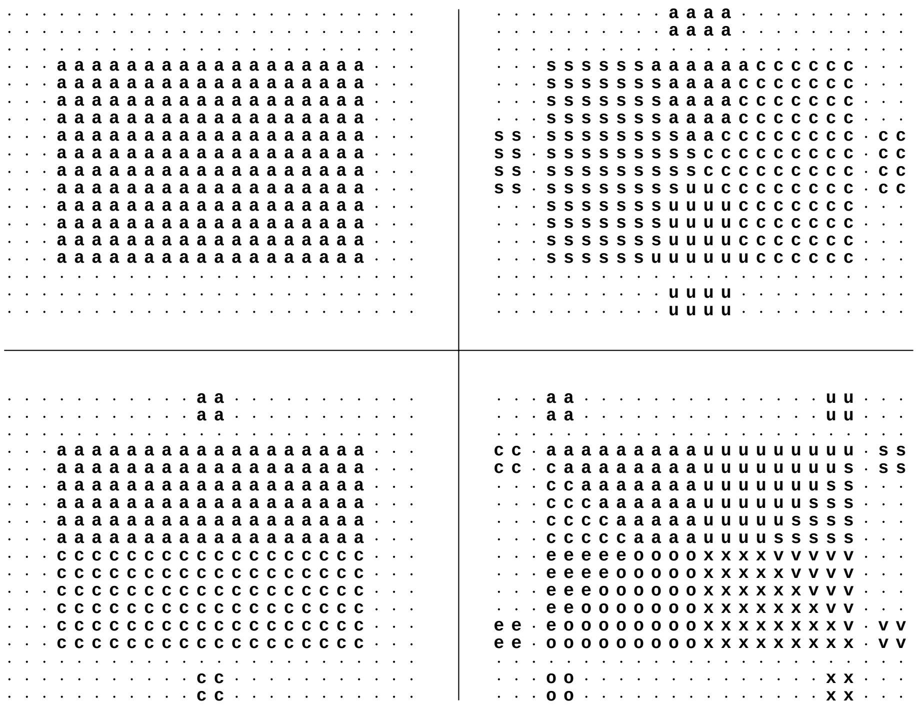


Figure 9: Graph from Figure 8 with different satellite constellations clustered with constant inflation equal to 1.3. The initial edges are defined by the neighbourhood structure depicted on page 12. The upper right clustering exhibits clear border crossing caused by inhomogeneity differentiation. The lower left clustering induces an unnatural bipartition of the large rectangle.

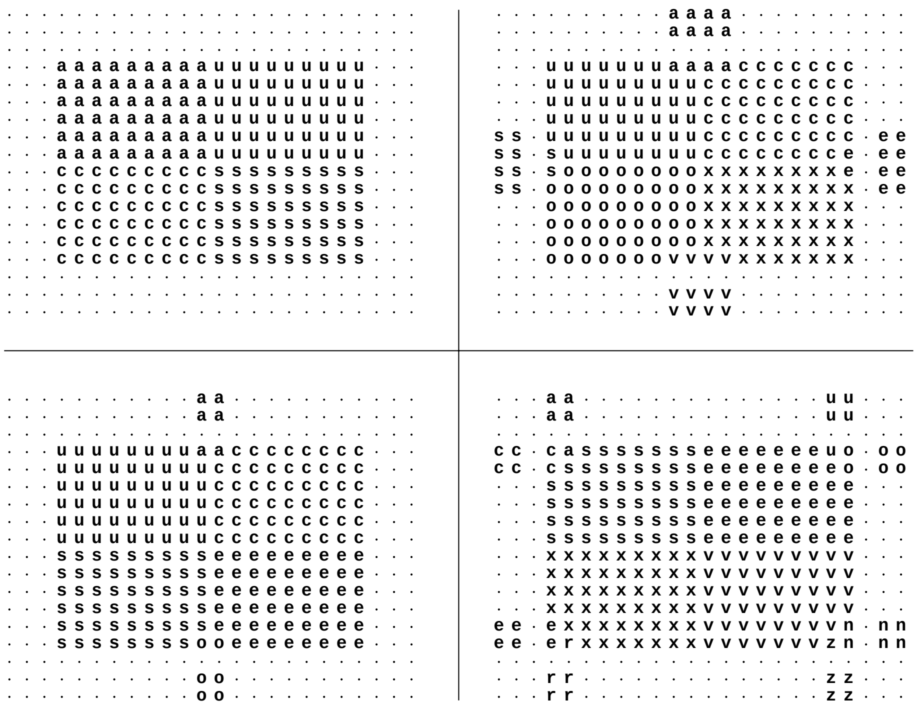


Figure 10: 18×12 Graph with different satellite constellations clustered with constant inflation equal to 1.5. The clusterings exhibit slight border crossing caused by inhomogeneity differentiation.

clustering corresponding with the neighbouring south or north satellite rather than the east or west satellite. This is probably caused by the fact that the border between the ‘s’ and ‘v’ clusters is closer by than the border between the ‘a’ and ‘u’ clusters, so that the distribution of the corner node transition probabilities is steeper along the south/north axis than it is along the east/west axis. The situation is different for the constellation in Figure 10, because the convergence of the cluster structure corresponding with the satellite systems took place much sooner than the convergence of the four large clusters in the rectangle.

The behaviour of the *MCL* process for the grid-like graphs in this section has two reasons. The fact that the natural clusters have sizes differing by orders of magnitudes is an important factor. However, examples exhibiting the same border-crossing behaviour can be constructed with sets of grids of the same size, simply by tiling them such that corners are aligned with borders. The most significant factor is the prolonged expansion at low inflation parameter required in order to equalize the probability distributions of opposing corners and opposing borders. The main characteristics of the subgraph corresponding with the large rectangle are that it is rather large (216 nodes) and has relatively large diameter. The process of equalizing distributions via expansion at low inflation values is costly in terms of space due to the large number of elements, and it is costly in terms of time due to the large diameter. The time requirement causes the process to be sensitive to perturbations in the input graph. This was demonstrated by adding small extra grids; similar phenomena occur if for example some nodes are given greater initial return probabilities than other nodes.

The significance of these observations is that one must be careful in applying the *MCL* algorithm to neighbourhood graphs derived from vector data, especially if it is known or unknown whether the diameter of the natural clusters is large. If it is known that this quantity is not too large, then the *MCL* algorithm will work well, as illustrated by the geometric graph example shown in Section 1. The principal cause for the behaviour of the *MCL* algorithm — large diameter and dimension of clusters — will affect any graph clustering algorithm that is based on the the computation of long distance dependencies, be it via random walks, paths, or shortest paths (several of these are discussed in [6]). Such type of algorithm will in each case be costly and prone to be sensitive to local fluctuations in density of the vectors inducing the neighbourhood graph.

The detection of clusters in a grid-like setting may be better served by a procedure such as border-detection. It is interesting to try and devise such a procedure using flow formulation and the graph cluster paradigm. The following section describes a small experiment in this direction.

8. Towards border detection using flow simulation

Clustering in the setting of (graphs derived from) grids has its limitations, as argued in the previous section. It was seen that clusters which correspond with large regions are difficult to detect using the graph clustering paradigm. However, the early stages of flow simulation yield information that can be used to detect the presence of borders between regions that have for example different shades (grey-levels) or colour. This is first illustrated for a simple example in which the mutual attraction between nodes depends on the associated Euclidean distance only. The graph shown on the left of Figure 11 is defined on four neighbouring rectangles, each of size 9×6, using the neighbourhood relationship on page 12. The borders of the four rectangles are thus weakly connected. The *MCL* process was applied to this graph with standard parameters, i.e. inflation and expansion both equal to two. The graph shown on the right of the same figure corresponds with the sixth iterand T of the process. The grey level of a node i is inversely proportional to the ratio T_{ii}/c_i, where c_i denotes the mass centre[7] of order 2 of the i^{th} column of T. Nodes i at the borders of the four rectangles thus have low value T_{ii}/c_i, and nodes lying in the centre have a high value. This is explained by the fact that convergence begins first at the borders, with border nodes becoming attracted to nodes which are one step away from the border. The nodes in the centre initially develop rather homogeneous and stable probability distributions.

[7]The mass centre function is defined on page 20. It can be viewed as a suitable notion of average for a stochastic vector. See also the remarks on page 32.

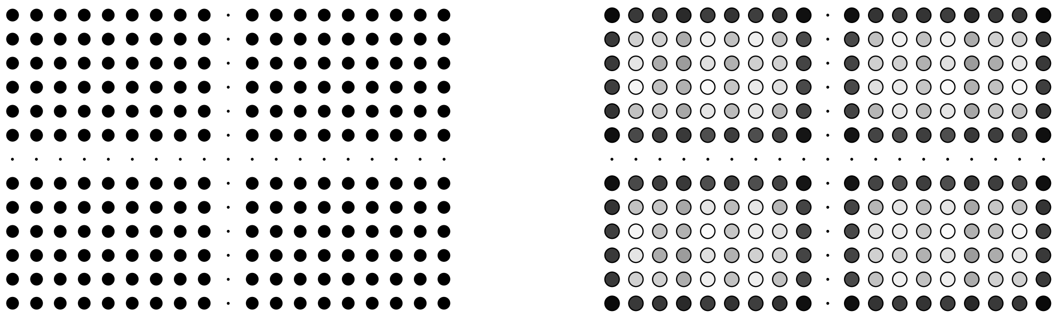


Figure 11: Using flow to find borders. The picture on the right hand side corresponds with the sixth iterand of an *MCL* process applied to the graph on the left using the neighbourhood structure on page 12. The grey level of the nodes on the right hand side is inversely proportional to the extent to which the nodes attract flow.

Next consider an image in the form of a bitmap with different grey-levels. An example is given in Figure 12. This image is a bitmap of dimension 284 × 380, where each pixel may assume 256 grey-levels. A graph was created from this bitmap with a node for each pixel. Horizontally, vertically, and diagonally neighbouring nodes (pixels) where connected via an edge with weight inversely proportional to the difference in grey level between the pixels. Loops were added such that the return probability of a node equalled the mass centre of order 2 of the stochastic vector associated with this node.

A variant of the *MCL* process was applied to this graph which incorporated aggressive pruning. That is, after each matrix multiplication, all nodes were allowed to have at most nine neighbours. The natural choice for this is to pick the nine neighbours with greatest associated probability. After removal of all but the nine largest neighbours of a node, the corresponding pruned column is rescaled to have weight one again. If a pixel is situated in a homogeneous region, then for early stages of the process the neighbours with largest associated transition probability will be just the set of its initial neighbours (including itself), since there is no reason for the symmetry to be broken. Moreover, the return probability will be the largest value in the column, since the symmetry leaves no room for any other direction of attraction. On the other hand, if a pixel is situated near a border or edge in the image, then the distribution of the associated probability vector will be asymmetric with respect to the initial neighbourhood constellation. This will cause the emergence of a direction of attraction, just as in the example in Figure 11. Figure 13 shows the result of interpreting the third iterand of the resulting process using the same principle as in Figure 11 and using a threshold for the indicator value T_{ii}/c_i.

The resulting image (Figure 13) indeed shows that the indicator value causes homogeneous regions to become blank and causes clear borders in the image to reappear as such. This is a tentative result, as there is information present in the processed image that hampers further contour detection (i.e. a true symbolic representation of borders), and there is also information lacking that one would like to be present (i.e. the arcs in the original image do not fully reappear in the processed image). However, it must be kept in mind that the chosen approach was extremely simple and naive. This use of the *MCL* process may well serve as an intermediate processing step in more sophisticated approaches. The value of the *MCL* process in this application is that it offers a generic modus via which neighbouring and super-neighbouring pixels may influence each other.

9. Performance criteria for simple graphs

Consider a simple graph and a clustering of this graph. If the rows and columns of the incidence matrix of the graph are permuted such that nodes in the same cluster correspond with consecutive columns (and rows), then the clustering is pictorially represented by a diagonal of blocks in the permuted matrix.

It is desirable that the clustering covers as many edges as possible, and does so efficiently. This means that

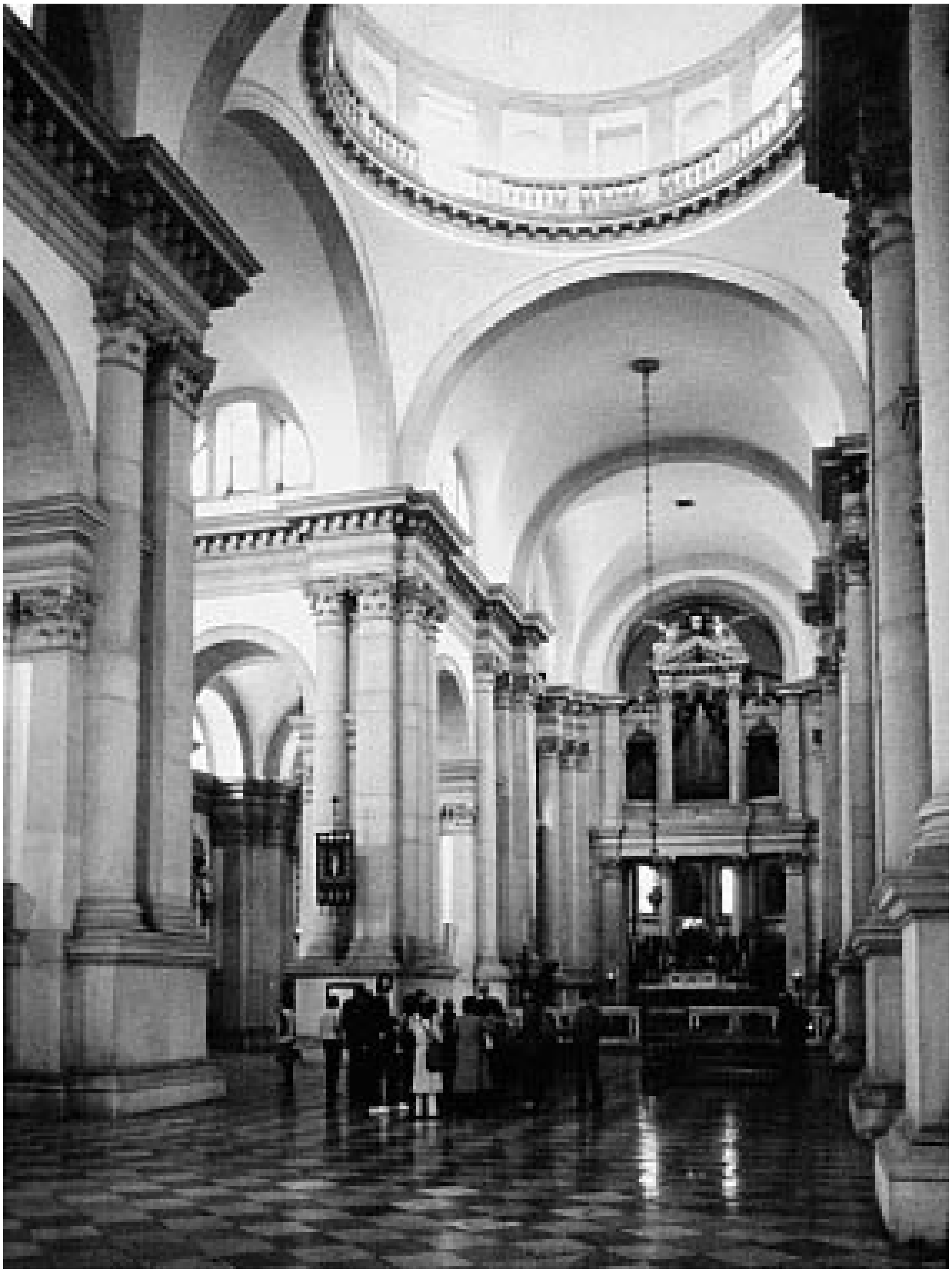


Figure 12: San Giorgio Maggiore in Venice.

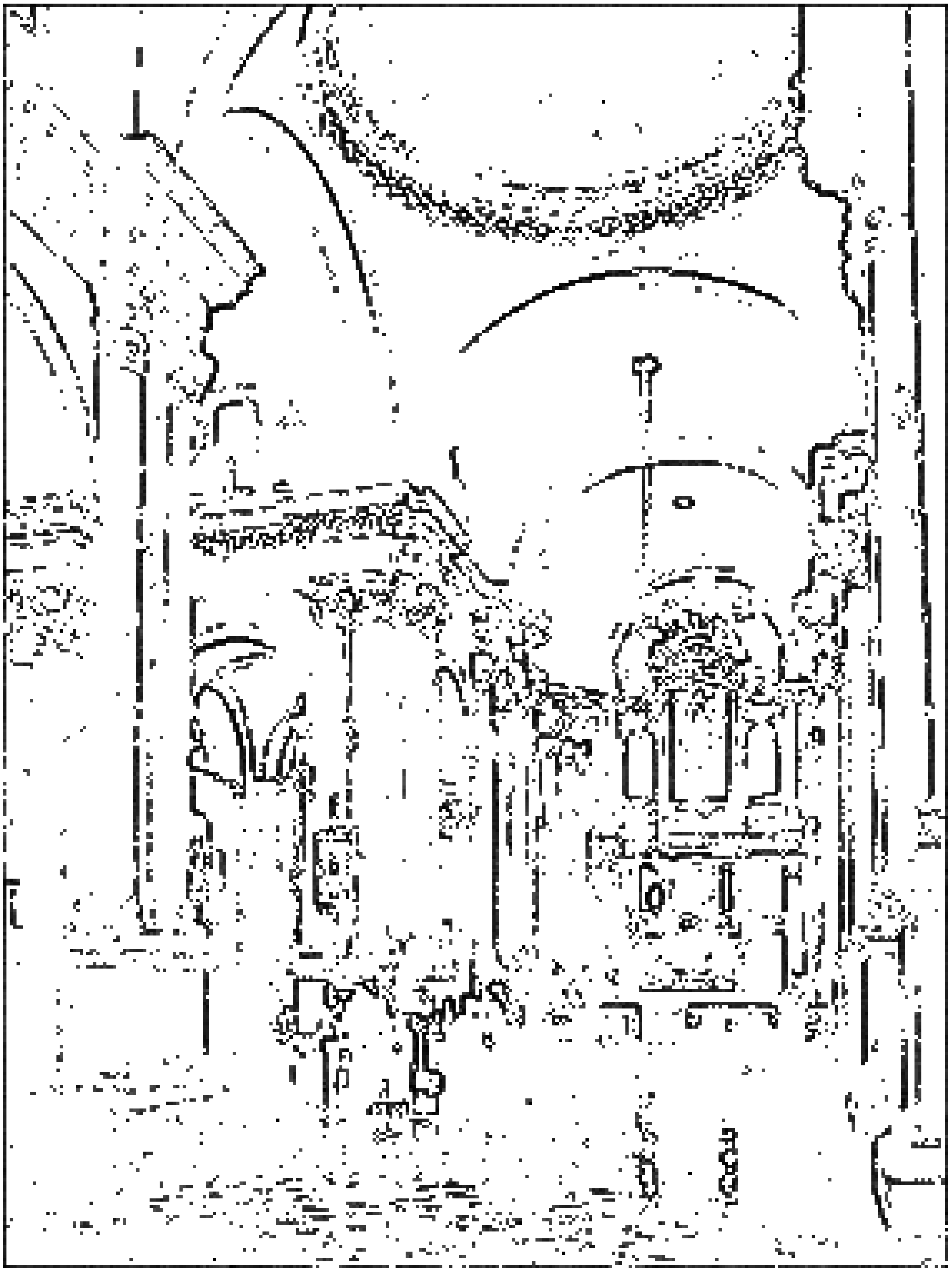


Figure 13: Result of a bordering process based on the MCL process.

the blocks should cover as many edges or 'ones' as possible, while there should be few zeroes within the blocks and few edges outside. It is easy to cover all of the edges by taking the top clustering (implying a single block consisting of the whole matrix), but then all zeroes are covered as well. This suggests that clusterings should be punished for each edge that is not covered by the clustering, and for each edge that is suggested by the clustering but not present in the graph, i.e. a within–block zero. Definition 6 gives a simple formula expressing this idea.

Definition 6 *Let G be a simple graph on n nodes with or without loops and let $\mathcal{P}$ be a partition of G. The* naive performance criterion *for G and $\mathcal{P}$ is defined to be*

$$\mathrm{Perf}(G,\mathcal{P}) = 1 - \frac{\#^1_{\mathrm{out}}(G,\mathcal{P}) + \#^0_{\mathrm{in}}(G,\mathcal{P})}{n(n-1)} \qquad \textit{(naive)} \tag{9.1}$$

where $\#^1_{\mathrm{out}}(G,\mathcal{P})$ denotes the number of edges not covered by the clustering (i.e. an edge (i,j) in G for which i and j are in different clusters of $\mathcal{P}$), and where $\#^0_{\mathrm{in}}(G,\mathcal{P})$ denotes the number of edges suggested by $\mathcal{P}$ absent in G (i.e. all pairs $i \neq j$ for which i and j are in the same cluster of $\mathcal{P}$ and (i,j) is not an edge in G). □

Note that by disregarding loops in the definition (i.e. only considering pairs $i \neq j$ for $\#^0_{\mathrm{in}}(G,\mathcal{P})$) it is achieved that clusterings are not punished for putting a node i in the same cluster as itself, if a loop from i to i is absent. This ensures that the best clustering (scoring the maximum performance 1) for the empty graph on n nodes (with no links between different nodes) is the bottom clustering consisting of n singletons. Note that this graph can be viewed as a direct sum of n times the complete graph K_1. In general, direct sums of complete graphs are the only graphs for which a clustering yielding performance 1 exists, and this is exactly what intuition demands. The scaling factor $n(n-1)$ guarantees that the performance criterion lies between 0 and 1 for all simple graphs. The performances of the top and bottom clusterings are related to each other by the fact that they add up to 1, as stated in the following basic property.

Property. Let G be a simple graph on n nodes with or without loops. Denote the top and bottom clusterings respectively by Top and Bottom. The following identity holds:

$$\mathrm{Perf}(G,\mathrm{Top}) + \mathrm{Perf}(G,\mathrm{Bottom}) = 1 \tag{9.2}$$

The proof is nearly trivial and is omitted here. The criterion in Definition 9.1 is useful only for clusterings of graphs which are not too sparse. For large sparse graphs the numerator in (9.1) usually pales in comparison to the denominator. This implies that for such graphs the naive performance criterion is close to one and almost constant for different clusterings as long as they are not too absurd. A slight modification of the definition remedies this drawback. To this end, the quality of a clustering is measured per node rather than in one stroke. First, consider what it takes to formulate definition 6 in terms of coverage of the respective nodes.

Definition 7 *Let G be a simple graph on n nodes with or without loops, let M be its incidence matrix, let $\mathcal{P}$ be a partition of G, and let v be a node in G. Denote the cluster in $\mathcal{P}$ containing v by P_v. The* naive coverage measure *of P_v for v is defined to be*

$$\mathrm{Cov}(G,P_v,v) = 1 - \frac{\#^1_{\mathrm{out}}(G,P_v,v) + \#^0_{\mathrm{in}}(G,P_v,v)}{n-1} \qquad \textit{(naive)} \tag{9.3}$$

where $\#^1_{\mathrm{out}}(G,P_v,v)$ denotes the number of edges going out from v not covered by P_v (i.e. a nonzero entry M_{iv} for which $i \notin P_v$) and where $\#^0_{\mathrm{in}}(G,P_v,v)$ denotes the number of edges suggested by P_v for v absent in G (i.e. all labels $i \neq v$ for which $i \in P_v$ and M_{iv} is zero). □

Averaging the coverage of a node according to Definition 7 over all nodes in a graph, yields the naive performance criterion from Definition 6. The coverage according to Definition 7 has the advantage that it can be easily modified such that its discriminating power applies to clusterings of sparse graphs as well. This is done by making the denominator smaller, whilest maintaining the property that the whole expression lies between 0 and 1.

Definition 8 *Let G be a simple graph on n nodes with or without loops, let M be its incidence matrix, let $\mathcal{P}$ be a partition of G, and let v be a node in G. Denote the set of neighbours of v (excluding v itself) in G by S_v, and denote the cluster in $\mathcal{P}$ containing v by P_v. The* simple coverage measure *of P_v for v is defined to be*

$$\mathrm{Cov}(G, P_v, v) = 1 - \frac{\#^1_{\mathrm{out}}(G, P_v, v) + \#^0_{\mathrm{in}}(G, P_v, v)}{|S_v \cup P_v|} \qquad \textit{(scaled)} \tag{9.4}$$

where $\#^1_{\mathrm{out}}(G, P_v, v)$ and and $\#^0_{\mathrm{in}}(G, P_v, v)$ are as in the previous definition. □

It is easy to see that the quantity defined by 9.4 lies between 0 and 1, and is equal to 1 only if the sets S_v and P_v are the same. It should be noted that this definition of coverage no longer yields the property that the respective performances of top and bottom clustering add to 1, which is a small sacrifice for its increased expressive power.

10. Performance criteria for weighted graphs

Definition 7 provides the inspiration for a measure of coverage telling how well a certain characteristic vector (representing a cluster) captures the mass of a nonnegative vector. The measure (given in Definition 10) is independent of scaling. An important distinction from the preceding definitions is that the loop (c.q. return probability) is no longer treated as a special case. The measure uses the notion of *mass centre* defined below, of which several properties are derived first.

Definition 9 *Let π be a probability vector of dimension n, let r be a real number greater than zero. The* **mass centre** *of order r of π is defined as*

$$\mathrm{ctr}_r \pi \;=\; \left(\sum_{i=1}^{n} {\pi_i}^r \right)^{\frac{1}{r-1}} \tag{10.1}$$

If the subscript is omitted, it is understood that the mass centre is of order 2. □

Lemma 1 *The mass centre of order r is an increasing function in r for all stochastic vectors π. It is monotone increasing for all non-homogeneous vectors π and constant for all homogeneous vectors π.*

PROOF. The derivative of $\mathrm{ctr}_r(\pi)$ equals $\mathrm{ctr}_r(\pi) f_\pi(r) g_\pi(r)$ where

$$f_\pi(r) = -\big(\sum_i {\pi_i}^r\big) \ln \big(\sum_i {\pi_i}^r\big) + (r-1) \sum_i {\pi_i}^r \ln \pi_i$$

$$g_\pi(r) = 1/\big(\sum_i {\pi_i}^r\big)\,(r-1)^2$$

The sign of $d/dr\ \mathrm{ctr}_r(\pi)$ depends on $f_\pi(r)$ only. Jensen's inequality states that

$$\sum_i w_i x_i \ln x_i - \big(\sum_i w_i x_i\big) \ln \big(\sum_i w_i x_i\big) \geq 0$$

where $\sum_i w_i = 1$ and the x_i are all positive. This inequality is strict unless all the x_i are equal. A proof is given in [1], page 17. Alternatively, this inequality follows easily from letting ϵ approach zero in the inequality

$$\sum_i w_i {x_i}^{1+\epsilon} - \big(\sum_i w_i x_i\big)^{1+\epsilon} \geq 0$$

The latter inequality follows from the convexity of the mapping $x \to x^\gamma$, $\gamma \geq 1$. Assume without loss of generality that all the π_i are positive (simply by pruning all zero entries of π and rescaling its dimension). Substituting $w_i = \pi_i$ and $x_i = {\pi_i}^{r-1}$ yields

$$\sum_i \pi_i {\pi_i}^{r-1} \ln {\pi_i}^{r-1} - \big(\sum_i \pi_i {\pi_i}^{r-1}\big) \ln \big(\sum_i \pi_i {\pi_i}^{r-1}\big) \geq 0$$

which is equivalent to saying that $f_\pi(r) \geq 0$, with equality iff π is homogeneous. This proves the lemma. □
The behaviour of $\mathrm{ctr}_r(\pi)$ for the limit cases is easily described.

Lemma 2 *Let π be a probability vector of dimension n, and suppose that it has k positive entries and $n-k$ entries equal to zero. By convention, define 0^0 to be equal to one.*

$$\lim_{r \downarrow 0} \mathrm{ctr}_r(\pi) = 1/k \tag{10.2}$$

$$\lim_{r \to 1} \mathrm{ctr}_r(\pi) = \prod_i {\pi_i}^{\pi_i} \tag{10.3}$$

$$\lim_{r \to \infty} \mathrm{ctr}_r(\pi) = \max_i \pi_i \tag{10.4}$$

PROOF. The first and last identities are elementary. The second follows from the derivation given below.

$$\begin{aligned}
\Big[\sum_i {\pi_i}^{1+\epsilon}\Big]^{1/\epsilon} &= \Big[\sum_i \pi_i e^{\epsilon \ln \pi_i}\Big]^{1/\epsilon} \\
&= \Big[\sum_i \pi_i \big(1 + \epsilon \ln \pi_i + \mathcal{O}(\epsilon^2)\big)\Big]^{1/\epsilon} \\
&= \big(1 + \epsilon \sum_i \pi_i \ln \pi_i + \mathcal{O}(\epsilon^2)\big)^{1/\epsilon} \\
&\to e^{\sum \pi_i \ln \pi_i} \qquad (\epsilon \to 0) \\
&= \prod_i {\pi_i}^{\pi_i}
\end{aligned}$$

□

Lemma 3 *Extend the definition of ctr_r to ${I\!R_{\geq 0}}^n \backslash \{0^n\}$. Let u and v be nonnegative vectors of the same dimension n having the same weight, that is, $\sum u_i = \sum v_i$. Let r be greater than zero and not equal to one. If u is majorized by v then $\mathrm{ctr}_r(u) \leq \mathrm{ctr}_r(v)$. This inequality is strict if u is not a permutation of v.*

PROOF. When the implication $u \prec v \implies \mathrm{ctr}_r(u) \leq \mathrm{ctr}_r(v)$ holds for all $u, v \in {I\!R_{\geq 0}}^n$, this is known as the property of *Schur convexity* of the function ctr_r on the set ${I\!R_{\geq 0}}^n$. A sufficient condition for Schur convexity of a continuously differentiable function ϕ defined on I^n, where I is an open interval in $I\!R$, is that ϕ is symmetric and that the expression

$$(x_i - x_j)\Big[\frac{\partial \phi(x)}{\partial x_i} - \frac{\partial \phi(x)}{\partial x_j}\Big]$$

is nonnegative for all $x \in I^n$ ([15], page 57). Setting ϕ to ctr_r expands the above to

$$\tfrac{r}{r-1}(x_i - x_j)({x_i}^{r-1} - {x_j}^{r-1})\big(\sum {x_i}^r\big)^{\frac{1}{r-1}-1}$$

Indeed, this expression is nonnegative for all $x \in I^n$, where I is chosen as the interval $(0, \infty)$. A limiting argument now establishes the first statement in the lemma. A sufficient condition for strictness of the majorization is that whenever (for ϕ symmetric)

$$\frac{\partial \phi(X)}{\partial x_i} = \frac{\partial \phi(X)}{\partial x_j} \tag{10.5}$$

for given $X \in I^n$, one must have

$$\phi_{(i,i)}(X) + \phi_{(j,j)}(X) - \phi_{(i,j)}(X) - \phi_{(j,i)}(X) > 0 \tag{10.6}$$

where $\phi_{(i,j)}(x)$ denotes the partial derivative $\partial\phi(x)/\partial x_i \partial x_j$. This condition is given in [15], pages 56–57. Equation (10.5) is satisfied only if $x_i = x_j$. Using this equality and the identities

$$\begin{aligned}
\frac{\partial \mathrm{ctr}_r(x)}{\partial x_i \partial x_j} &= \tfrac{r^2}{r-1}(\tfrac{1}{r-1}-1){x_i}^{r-1}{x_j}^{r-1}\Big[\sum {x_i}^r\Big]^{\frac{1}{r-1}-2} \\
\frac{\partial \mathrm{ctr}_r(x)}{\partial x_i \partial x_i} &= \tfrac{r^2}{r-1}(\tfrac{1}{r-1}-1){x_i}^{r-1}{x_i}^{r-1}\Big[\sum {x_i}^r\Big]^{\frac{1}{r-1}-2} + r{x_i}^{r-2}\Big[\sum {x_i}^r\Big]^{\frac{1}{r-1}-1}
\end{aligned}$$

validates Inequality (10.6). This proves the second statement for all $u, v \in {I\!R_{>0}}^n$, and its general form (for $u, v \in {I\!R_{\geq 0}}^n$) follows again from a limiting argument. □

The Schur convexity of ctr_r implies that ctr_r is a measure for the deviation from homogeneity of a nonnegative vector. The inverse of the mass centre of order r $(r \geq 2)$ applied to stochastic vectors has an interpretation as the number of nonzero entries in a weighted sense. This statement is justified by Definition 10 and Theorem 3.

Definition 10 *Let u be a nonnegative vector, and let π be the scalar multiple of u such that π is stochastic. Let P be a subset of $\{1,\dots,n\}$, let P^c denote its complement in this same set. For $r \geq 2$ the* weighted coverage measure of order *r for P and u is defined to be*

$$\mathrm{Cov}_r(P,u) = 1 - \frac{|P| - \dfrac{1}{\mathrm{ctr}_r(\pi)}\left(\sum_{i\in P}\pi_i - \sum_{i\in P^c}\pi_i\right)}{n} \qquad \textit{(weighted)} \tag{10.7}$$

If the subscript is omitted, it is understood that the measure is of order 2. □

The interpretation of the measure is as follows. The quantity $1/\mathrm{ctr}_r(\pi)$ can be viewed as the size of the 'ideal' clustering for the vectors u and π. This is certainly true if u is homogeneous. If the actual clustering equals the ideal clustering of u (picking out precisely all positive elements), then the numerator in the fraction cancels, and the measure yields the perfect score one. If the vector u is not homogeneous, then it is impossible for any cluster to yield this perfect score. This makes sense if e.g. the vectors $a = (100,0,0)$, $b = (98,1,1)$, $c = (58,21,21)$, and $d = (50,25,25)$ are considered. The cluster (represented by a characteristic vector) $(1,0,0)$ should (and does) have a coverage measure equal to one for a, because it perfectly captures all of the mass of a. It does not perfectly capture all of the mass of b, but the clustering $(1,1,1)$ for b is inefficient in the sense that the cluster uses two positions accounting for two percent of the mass, and one position accounting for ninety-eight percent of the mass. The performance coefficients (of order 2) of the respective clusterings $(1,1,1)$ and $(1,0,0)$ for the vectors a, b, c, and d are respectively $(0.333, 1.000)$, $(0.347, 0.999)$, $(0.785, 0.792)$, and $(0.889, 0.667)$. The 'max' coefficients of order $r = \infty$ are respectively $(0.333, 1.000)$, $(0.340, 0.993)$, $(0.575, 0.759)$, and $(0.667, 0.667)$. The coefficient of order 2 tends to be more rewarding if all mass is covered; the max coefficient tends to be more rewarding if relatively small elements are not covered.

The weighted coverage measure $\mathrm{Cov}_r(P,u)$ has the property that it lies between 0 and 1 for all u and P iff $r \geq 2$.

Theorem 3 *Let π and P be as in Definition 10. Let $\mathrm{Cov}_r(P,\pi)$ denote the weighted coverage measure $(r \geq 2)$. Then*

$$0 \leq \mathrm{Cov}_r(P,\pi) \leq 1 \tag{10.8}$$

with equality in the second inequality if and only if the set of nonzero positions of π coincides with P and the nonzero entries are homogeneously distributed.

For $r < 2$ there exist vectors π and clusterings P such that $\mathrm{Cov}_r(P, \pi) > 1$.

PROOF. For the first part of the theorem only one inequality needs to be proven, since replacement of P with P^c interchanges the inequalities. It is easy to see that $\mathrm{Cov}_r(P, \pi) + \mathrm{Cov}_r(P^c, \pi) = 1$, so that the coverage measure $\mathrm{Cov}_r(P, \pi)$ is equal to zero iff the nonzero entries are homogeneously distributed and P covers all the zero entries of π.

Using the identity $\sum_{i \in P} \pi_i = 1 - \sum_{i \in P^c} \pi_i$ leaves the inequality $2 \sum_{i \in P} \pi_i \leq \mathrm{ctr}_r(\pi)|P| + 1$ to be proven. This inequality is proven for the special case that $r = 2$; the case $r > 2$ then follows from the monotonicity of $\mathrm{ctr}_r(\pi)$ in r.

The inequality $2x \leq x^2 + 1$ (valid for $x \in [0, 1]$) and the inequality $2ab \leq a^2 + b^2$ (valid for all real a and b) validate the following derivation.

$$\begin{aligned}
2\sum_{i \in P} \pi_i &\leq \left(\sum_{i \in P} \pi_i\right)^2 + 1 \\
&= \sum_{i \in P} {\pi_i}^2 + \left(\sum_{i,j \in P, i<j} 2\pi_i\pi_j\right) + 1 \\
&\leq \sum_{i \in P} {\pi_i}^2 + \left(\sum_{i,j \in P, i<j} {\pi_i}^2 + {\pi_j}^2\right) + 1 \\
&= \mathrm{ctr}(\pi) + \big(|P| - 1\big)\mathrm{ctr}(\pi) + 1
\end{aligned}$$

The second part of the theorem follows from an explicit construction. Let ϵ be an arbitrarily small (fixed) positive number, and set $r = 2 - \epsilon$. Let a be a small positive number, let π be the two-dimensional vector $(1 - a, a)$, and let P be the cluster $\{1\}$. It is shown that for a small enough the inequality $2 \sum_{i \in P} \pi_i \leq \mathrm{ctr}_r(\pi)|P| + 1$ is invalidated. First rewrite this inequality in terms of the chosen parameters as

$$1 - 2a \leq \left[(1-a)^{2-\epsilon} + a^{2-\epsilon}\right]^{\frac{1}{1-\epsilon}}$$

Estimate the right-hand side (which equals $\mathrm{ctr}_r(\pi)$) as follows.

$$\begin{aligned}
\left[(1-a)^{2-\epsilon} + a^{2-\epsilon}\right]^{\frac{1}{1-\epsilon}} &= \left[1 - \big(2a - \epsilon a + \mathcal{O}(a^2)\big) + a^{2-\epsilon}\right]^{1+\epsilon+\mathcal{O}(\epsilon^2)} \\
&= \left[1 - \big(2a - \epsilon a + \mathcal{O}(a^{2-\epsilon})\big)\right]^{1+\epsilon+\mathcal{O}(\epsilon^2)} \\
&= 1 - \big(2a - \epsilon a + \mathcal{O}(a^{2-\epsilon})\big)\big(1 + \epsilon + \mathcal{O}(\epsilon^2)\big) + \mathcal{O}(a^2) \\
&= 1 - 2a + \epsilon a - 2\epsilon a + \mathcal{O}(\epsilon^2 a) + \mathcal{O}(a^{2-\epsilon}) \\
&= 1 - 2a - \epsilon a + \mathcal{O}(\epsilon^2 a) + \mathcal{O}(a^{2-\epsilon})
\end{aligned}$$

It follows that the inequality $2 \sum_{i \in P} \pi_i \leq \mathrm{ctr}_r(\pi)|P| + 1$ will fail to be true for the chosen parameters if a is chosen sufficiently small (much smaller than ϵ). Setting $\epsilon = 0.1$ and $a = 0.01$ yields that $\mathrm{ctr}_r(\pi)$ is approximately equal to 0.9792, whereas $1 - 2a$ equals 0.98. □

Combining the modifications from Definitions 8 and 10 yields the following performance criterion for clusterings of weighted graphs.

Definition 11 *Let $r \geq 2$ be real, let u be a nonnegative vector of dimension n, and let π be the scalar multiple of u such that π is stochastic. Denote the set of indices i with u_i nonzero by S. Let P be a subset of $\{1, \ldots, n\}$, let P^c denote its complement in this same set. The* weighted coverage measure of order r *for P and u is defined as*

$$\mathrm{Cov}_r(P, u) = 1 - \frac{|P| - \dfrac{1}{\mathrm{ctr}_r(\pi)}\left(\sum_{i \in P} \pi_i - \sum_{i \in P^c} \pi_i\right)}{|P \cup S|} \qquad \textit{(weighted, scaled)} \tag{10.9}$$

If the subscript is omitted, it is understood that the measure is of order 2. Next, let G be a graph with associated matrix M, let $\mathcal{P}$ be a partition of $\{1, \ldots, n\}$, and let P_u be the set in $\mathcal{P}$ containing u, $u \in \{1, \ldots, n\}$. The weighted performance measure $\mathrm{Perf}_r(G, \mathcal{P})$ is obtained by averaging the summed coverage measures $\mathrm{Cov}_r(P_u, u)$ for all columns u of M. □

Both coverage and performance lie between 0 and 1. This is easily seen; if v is the vector u with those zero entries pruned which are not in S (leaving precisely the entries corresponding with $P \cup S$) then the coverage of u according to Definition 11 is the coverage of v according to Definition 10.

11. A distance on the space of partitions

The following distance defined on the space of partitions of a given set is used in judging the continuity properties of clusterings generated by the *MCL* algorithm at different levels of granularity, and for measuring the distance between *MCL* clusterings and clusterings of randomly generated test graphs with a priori known density characteristics.

Definition 12 *Let S be the set $\{1, \ldots, n\}$. Let $\mathcal{A}$ and $\mathcal{B}$ be arbitrary partitions of S. The projection number $p_{\mathcal{A}}(\mathcal{B})$ of $\mathcal{A}$ onto $\mathcal{B}$ is defined as*

$$p_{\mathcal{A}}(\mathcal{B}) = \sum_{a \in \mathcal{A}} \max_{b \in \mathcal{B}} |a \cap b| \tag{11.1}$$

The distance d between $\mathcal{A}$ and $\mathcal{B}$ is defined as

$$d(\mathcal{A}, \mathcal{B}) = 2n - p_{\mathcal{A}}(\mathcal{B}) - p_{\mathcal{B}}(\mathcal{A}) \tag{11.2}$$

It will customarily be written as the pair of nonnegative integers $(n - p_{\mathcal{A}}(\mathcal{B}), n - p_{\mathcal{B}}(\mathcal{A}))$, which is equal to the pair $(d(\mathcal{A}, \mathcal{A} \cap \mathcal{B}), d(\mathcal{B}, \mathcal{A} \cap \mathcal{B}))$ (see below).

□

Example. Let $n = 12$, let A and B be the respective partitions $\{\{1,2,3,4\}, \{5,6,7\}, \{8,9,10,11,12\}\}$ and $\{\{2,4,6,8,10\}, \{3,9,12\}, \{1,5,7\}, \{11\}\}$, which have meet $\{\{1\}, \{2,4\}, \{3\}, \{5,7\}, \{6\}, \{8,10\}, \{9,12\}, \{11\}\}$. Then $p_{\mathcal{A}}(\mathcal{B})$ equals $2+2+2$ and $p_{\mathcal{B}}(\mathcal{A})$ equals $2+2+2+1$, the distance $d(\mathcal{A}, \mathcal{B})$ equals $24 - 6 - 7 = 11$ and is presented as the pair $(6, 5)$.

It is useful to present the distance as a pair, because if either of the pair elements is zero, this implies that the corresponding partition is a subpartition of the other. This is easy to verify, and expressed in the following identities which are valid for all $\mathcal{A}$ and $\mathcal{B}$.

$$\begin{aligned} p_{\mathcal{A}}(\mathcal{A} \cap \mathcal{B}) &= p_{\mathcal{A}}(\mathcal{B}) \\ p_{\mathcal{A} \cap \mathcal{B}}(\mathcal{A}) &= n \\ d(\mathcal{A}, \mathcal{B}) &= d(\mathcal{A}, \mathcal{A} \cap \mathcal{B}) + d(\mathcal{B}, \mathcal{A} \cap \mathcal{B}) \end{aligned}$$

More generally, a small projection number $p_{\mathcal{A}}(\mathcal{B})$ implies that $\mathcal{A}$ is close to being a subpartition of $\mathcal{B}$. This is intuitively clear, and it is formally expressed in the following theorem, by identifying $d(\mathcal{A},\mathcal{B})$ with the weight of a shortest path in a suitable graph.

Theorem 4 *The distance defined in Definition 12 satisfies the axioms for a metric.*

PROOF. Clearly it is only the triangle inequality that is of concern. The proof follows by showing that the distance corresponds to the shortest distance between $\mathcal{A}$ and $\mathcal{B}$ in a particular undirected weighted graph constructed on the set of all partitions of the set $\{1,\ldots,n\}$. In this graph two partitions are connected via an edge iff one can be constructed from the other by joining two of its sets (equivalently the other is constructed from the first by splitting a set into two). The weight of the edge equals the size of the smallest of the two sets. So the claim is that $d(\mathcal{A},\mathcal{B})$ is the length of a shortest path (in terms of total weight) between $\mathcal{A}$ and $\mathcal{B}$. Denote a split of a set UV into two parts U and V by $(UV)\searrow(U|V)$, denote a join of two parts U and V by $(U|V)\nearrow(UV)$. Denote a path between A and B by a sequence of splits and joins. Now it is easy to verify that $d(\mathcal{A},\mathcal{B})$ is the cost of a path consisting of successive down arrows $()\searrow()$ starting from $\mathcal{A}$ all the way down to the meet $\mathcal{A}\cap\mathcal{B}$, followed by a sequence of up arrows $()\nearrow()$ up to $\mathcal{B}$, and that there is no similar down/up path (with only one reversal in the orientation of the arrows) of lower weight. The crux is now that any other path through the graph can be converted to a one-reversal down/up path without gaining weight. To this end, it is only necessary to convert an up/down arrow pair to a down/up arrow sequence without gaining weight. Repeated application of this manoeuvre yields the desired result. Two cases must be distinguished. First consider the sequence $(TU|VW)\nearrow(TUVW)\searrow(TV|UW)$, with associated cost

$$\min(|T|+|U|,|V|+|W|)+\min(|T|+|V|,|U|+|W|)$$

It can be converted into the sequence

$$(TU|VW)\searrow(T|U|VW)\searrow(T|U|V|W)\nearrow(TV|U|W)\nearrow(TV|UW)$$

with associated cost

$$\min(|T|,|U|)+\min(|V|,|W|)+\min(|T|,|V|)+\min(|U|,|W|)$$

The following inequalities yield that the latter cost does not exceed the former.

$$\begin{aligned}\min(|T|,|U|)+\min(|V|,|W|) &\le \min(|T|+|V|,|U|+|W|)\\ \min(|T|,|V|)+\min(|U|,|W|) &\le \min(|T|+|U|,|V|+|W|)\end{aligned}$$

Second, consider the sequence $(U|V|XY)\nearrow(UV|XY)\searrow(UV|X|Y)$ with associated cost $\min(|U|,|V|)+\min(|X|,|Y|)$. It can be converted to the sequence $(U|V|XY)\searrow(U|V|X|Y)\nearrow(UV|X|Y)$, which has identical cost as the former sequence. The theorem follows. □

A well-known distance on the space of partitions ([18], page 241) is the equivalence mismatch coefficient emc defined below.

$$\operatorname{emc}(\mathcal{A},\mathcal{B})=\sum_{a\in\mathcal{A}}|a|^2+\sum_{b\in\mathcal{B}}|b|^2-2\sum_{\substack{a\in\mathcal{A}\\ b\in\mathcal{B}}}|a\cap b|^2 \tag{11.3}$$

This coefficient is usually scaled with the factor $1/n^2$ in order to restrict its range to the interval $[0,1]$. The distance is precisely the Hamming distance for binary vectors[8] if the set of all pairs $(i,j)\in S^2$ is enumerated

[8]The Hamming distance between two 0/1 vectors of the same length is the number of positions in which the vectors differ.

and a partition is represented via a characteristic vector defined on this enumeration[9]. The definition of emc shows that its interpretation must be formulated in terms of edge numbers of complete graphs. The distance d can be interpreted as the number of node moves needed to convert one partition into the other. The following example shows that the distances behave very dissimilar for two extreme cases, where the behaviour of d appears to be preferable to that of emc.

Suppose that the number of nodes n is a square, say $n = m^2$. Let Top denote the partition $\{V\}$, let Bottom denote the partition $\{\text{singletons}(V)\}$. Denote the partition $\big\{\{1, 2, \ldots, m\}, \{m+1, m+2, \ldots, 2m\}, \ldots, \{m^2 - m+1, m^2-m+2, \ldots, m^2\}\big\}$ by Left, denote the partition $\big\{\{1, m+1, \ldots, m^2-m+1\}, \{2, m+2, \ldots, m^2-m+2\}, \ldots, \{m, 2m, \ldots, m^2\}\big\}$ by Right. Now $d(\text{Top}, \text{Bottom})$ equals $n-1$ and $d(\text{Left}, \text{Right})$ equals $2m(m-1) = 2\sqrt{n}(\sqrt{n}-1)$. For emc one has $\text{emc}(\text{Top}, \text{Bottom})$ equalling $n(n-1)$ and $\text{emc}(\text{Left}, \text{Right})$ equalling $2mm^2 - 2n = 2n(\sqrt{n}-1)$. It is noteworthy that emc judges Left and Right to be closer to each other than Top and Bottom, whereas the former two are nested partitions and the latter two are maximally conflicting. The cause for this is clearly that emc measures volumes (i.e. edge numbers) of complete graphs induced by the partitions, so that the sizes of partition elements play an important role.

12. Randomly generating test graphs

For the purpose of testing the *MCL* algorithm, simple graphs are generated via a simple modification of the generic random graph model, in which each edge is realized with some fixed probability p. In the generic random graph model, the parameters are the dimension of the graph n and the probability p.

Definition 13 *A* **random cluster/graph generator** $\mathfrak{G}$ *(Fraktur G) is a tuple* $(n, p, q, \mathfrak{O})$*, where* n *is a positive integer,* p *and* q *are probabilities satisfying* $1 \geq p \geq q \geq 0$*, and* $\mathfrak{O}$ *(Fraktur O) is a generator producing partitions of the set* $\{1, \ldots, n\}$*.*

The generator $\mathfrak{G}$ *generates a* **cluster test graph** *by obtaining a partition* $\mathcal{P}$ *from* $\mathfrak{O}$*, and subsequently realizing an edge* (k, l) *with respectively probability* p *if* k *and* l *are in the same partition element of* $\mathcal{P}$*, and probability* q *if* k *and* l *are in different partition elements.* □

The partition generator has not been further specified in this definition, as it is convenient to be able to plug in different types of generators, which differ with respect to the granularity and homogeneity (i.e. variation in the subset sizes) of the partitions generated.

The symmetric nature of the way in which edges are locally realized within partition elements implies that the corresponding subgraphs are unlikely to have long chains. A simple way of seeing this is by envisioning the corresponding incidence submatrix. For any permutation of this matrix, the nonzero entries are expected to be homogeneously distributed, whereas a long chain corresponds with a (part of the) submatrix in which all elements are close to the diagonal. The theory of randomly generated graphs confirms that the expected diameter of the connected components is small [3, 4]. The upshot is that the generated cluster structure is spherical rather than chain-like.

The discussion in this section will mostly follow heuristic arguments. Whereas the simple setup in Definition 13 should allow mathematical reasoning by probabilistic arguments about the generated graphs, this is entirely beyond the scope of this thesis. The mathematics behind random graph-theory is rather involved, and constitutes a large branch of research in itself (for results on the diameter of random graphs, consult [3, 4, 19] and references therein). I do claim however that the random cluster/graph generator is a canonical model

[9] A node pair (i, j) for which i and j are in the same partition element induces a '1' in this characteristic vector, and a node pair (i, j) for which i and j are in different elements induces a '0'.

for generating cluster test graphs. Consider a graph G generated by a $(n, p, q, \mathfrak{O})$ generator for a partition $\mathcal{P}$ from $\mathfrak{O}$. The graph can be viewed as a random graph generated with parameters (n, q), after which extra edges are added *solely in the subgraphs corresponding with elements of* $\mathcal{P}$. Assuming that p is much larger than q, it follows that $\mathcal{P}$ must approximately be the best clustering of G; it is simply extremely unlikely that any other block diagonalization achieves the same high density p of nonzero entries within the blocks, and the low density q of nonzero entries outside the blocks.

The tests with the *MCL* algorithm were carried out with the randomized parametrized generator defined below.

Definition 14 *A* **grid partition generator** $\mathfrak{P}$ *(Fraktur P) is a pair* (n, g)*, where n and g are positive integers with* $g \leq n$*. Let r be the remainder of n modulo g, and let k be the integer part of* n/g*, so that* $n = kg + r$*.*

The generator $\mathfrak{P}$ *generates a partition of* $\{1, \dots, n\}$ *by generating k random permutations of the interval* $\{1, \dots, g\}$*, and taking as partition sizes the lengths of the cycles. The partition sizes derived from the* i^{th} *permutation,* $i = 1, \dots, k$*, are mapped onto a consecutive set of elements in the set* $\{(i-1)g + 1, \dots, ig\}$*. The last r entries are partitioned similarly by a random permutation of the set* $\{1, \dots, r\}$*.* □

Note. Throughout this section, the word 'partition' will be used to refer to the partition used in generating a test graph. The word 'clustering' is used to refer to a clustering generated by the *MCL* algorithm. A cluster is an element of the clustering, i.e. a set of nodes or node indices. Its counterpart in a partition is called a partition element. The words 'graph' and 'matrix' will be used almost interchangingly. In particular, the diagonal block structure of a (permuted according to cluster structure) matrix is in one–one correspondence with the clustering of the underlying graph of the matrix.

The incidence matrix in Figure 14 was generated with parameters $n = 160$, $g = 60$, $p = 0.25$, and $q = 0.03$. The underlying graph of this particular matrix has diameter four. The generated partition sizes (after rearrangement) equalled 1^5, 2, 2, 3, 16, 20, 35, 36, and 41. A random permutation of this matrix is shown in Figure 15, to illustrate the serious challenge of finding block structure in matrices (i.e. clustering the underlying graph).

A singular property of the random cluster/graph generator model is that small partition elements do *not* result in clear clusters in the generated graph, and thus introduce the phenomenon of noise in the test graphs. This is illustrated by the matrix in Figure 14. The partition elements of size 1, 2, and 3 correspond with nodes of lowest index, inducing the leftmost columns and uppermost rows. Had clusters of size up to approximately ten been present, the same would apply to them. Apart from modifying the partition generator, this phenomenon can be remedied by extending the random cluster/graph generator model by introducing functions $f(p, q, n, x)$ and $g(p, q, n, x, y)$ such that an edge is realized with probability $f(p, q, n, x)$ if its incident nodes are in the same partition element of size x, and with probability $g(p, q, n, x, y)$ if its incident nodes are in partition elements of respective sizes x and y. This is in fact part of the generator implementation in use at CWI. However, all experiments discussed in this section were conducted without using this refinement. Introducing more parameters stand in the way of standardizing, repeating, comparing, and benchmarking of experiments. Moreover, the presence of noise (nodes and edges not really fitting in any larger scale cluster structure) is actually interesting in its own right, as it will surely pop up in real-life applications.

As a first elaborate example, several *MCL* runs were carried out for the matrix in Figure 14, with pruning constant set to 50. The results are depicted in Table 4. The partition $\mathcal{P}$ used in constructing this matrix has performance criterion (according to Definition 11) equal to 0.198 for the matrix itself and 0.607 for the square of the matrix. The partition consists of elements with respective sizes 1^5, 2, 2, 3, 16, 20, 35, 36, and 41. The number of clusters found is given in the column under $|\mathcal{C}|$. The distance between two different clusterings is

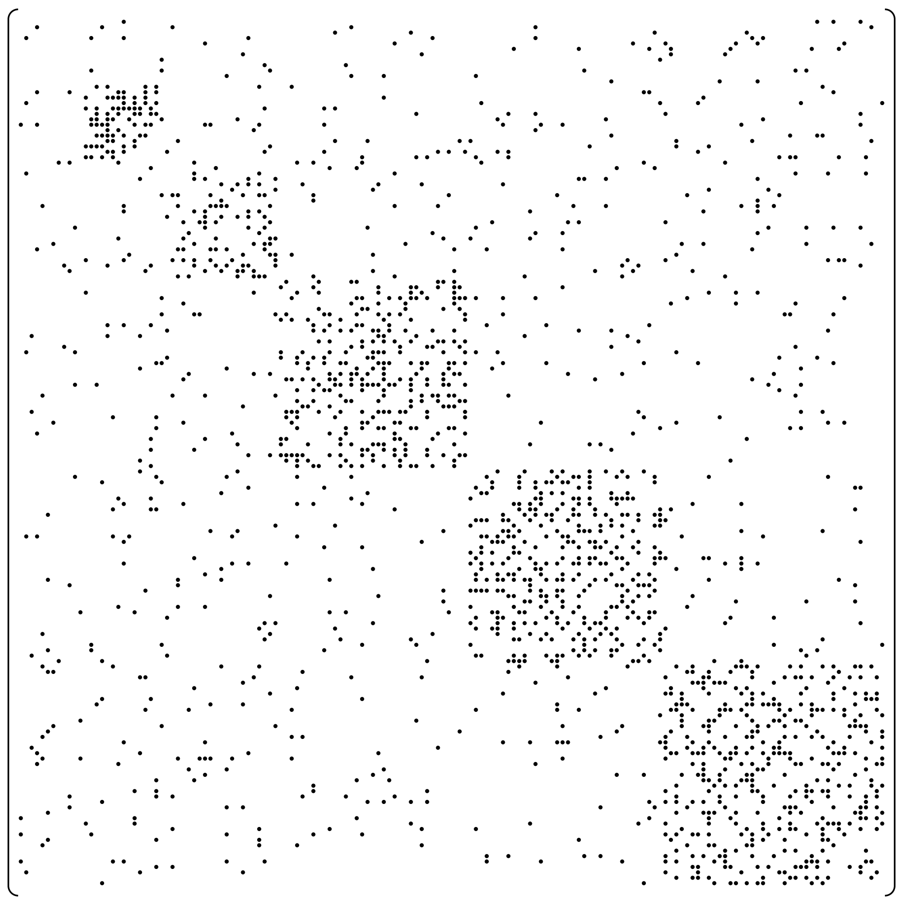


Figure 14: Randomly generated cluster test matrix. Dots indicate nonzero entries, which are all equal to one. The indices were aligned according to the generating partition.

measured by the pair-valued function d defined in Section 11. The density of the number of nonzero entries (as a fraction of the cluster area, which is the sum of the squares of the cluster sizes) is given in the column labelled p_i. The corresponding density of nonzero entries *not* covered (as a fraction of the remaining area) is given in the column labelled q_i.

The cluster sizes of the 'best' clustering in row 12 are respectively 1^3, 2^2, 3^2, 4, 6, 14, 15, 33, 34, and 41. The four largest clusters roughly correspond with the four largest partition elements — for the clusters of size 15, 33, 34, and 41 the size of the symmetric difference with a best matching partition element of $\mathcal{P}$ is respectively 3, 4, 2, and 4. The distance pair of this clustering with $\mathcal{P}$ is $(17, 23)$. The data in the figure shows that in general either the clustering or the partition is close to their intersection. The same holds for pairs of clustering at consecutive levels of granularity (corresponding with a small increase of the inflation power coefficient). This shows that the *MCL* algorithm has good properties with respect to continuity. The clustering/partition distances in the upper part of the table indicate that the maximum performance value is attained for the clustering which is approximately closest to the partition. This can be taken as evidence that the performance criterion is a good assistant in measuring the quality of a clustering. The drop in performance from 0.583 to 5.81 at inflation level 1.6 in the lower part of the table is a little peculiar, but the fact that the

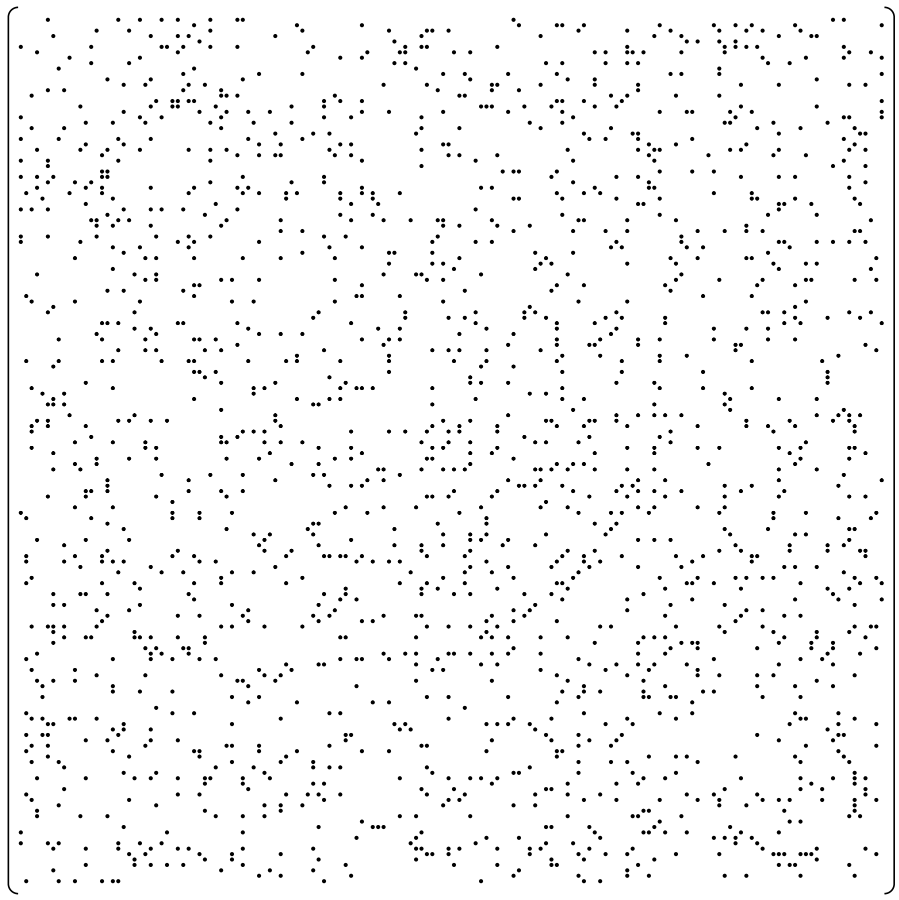


Figure 15: Random permutation of the matrix in Figure 14. A random clustering will have low performance coefficient.

corresponding clusterings are rather close can easily account for such a small fluctuation. Moreover, it is seen in both parts of the table that all clusterings which are slightly better than the partition with respect to the matrix, are slightly worse than the same partition with respect to the square of the matrix.

Finally, it is interesting to visualize the effect that longer distances have on the resulting clustering. To this end, a champion clustering with performance coefficient equal to 0.214 (resulting from loop weight 3 and inflation 1.7) is used to align the *square* of the matrix in Figure 14. The result is depicted in Figure 16. One sees very clearly off-diagonal bands in the matrix which correspond with (two-step) edges connecting different subgraphs corresponding with different diagonal blocks. These two step connections have 'helped' each of the diagonal blocks to become a cluster in the *MCL* process. Now it is natural to wonder whether this champion clustering cannot be further improved by joining the diagonal blocks connected by the bands. The answer is no, and the reason is that this *does* improve the clustering with respect to the performance criterion applied to the *square of the graph* — which is a mere 0.567 — but it does not improve the clustering with respect to the performance criterion applied to the graph itself.

i	a	R	pf$(G, \mathcal{C}_i)$	pf$(G^2, \mathcal{C}_i)$	p_i	q_i	$\lvert\mathcal{C}_i\rvert$	$d(\mathcal{C}_i, \mathcal{P})$	$d(\mathcal{C}_i, \mathcal{C}_{i-1})$
1	1	1.3	0.111	0.442	0.100	0.026	2	(84, 4)	–
2	1	1.4	0.149	0.546	0.150	0.029	3	(48, 12)	(0,45)
3	1	1.5	0.149	0.546	0.150	0.029	3	(48, 12)	(0,0)
4	1	1.6	0.193	0.598	0.219	0.032	8	(19, 15)	(9,38)
5	1	1.7	0.201	0.600	0.240	0.034	11	(18, 20)	(6,12)
6	1	1.8	0.186	0.581	0.256	0.042	24	(18, 39)	(3,23)
7	1	1.9	0.165	0.526	0.365	0.055	48	(17, 83)	(8,53)
								$d(\mathcal{C}_i, \mathcal{C}_{i-7})$	$d(\mathcal{C}, \mathcal{C}_{i-1})$
8	2	1.3	0.111	0.442	0.100	0.026	2	(0,0)	–
9	2	1.4	0.148	0.547	0.151	0.029	3	(2,2)	(1,47)
10	2	1.5	0.181	0.583	0.189	0.031	6	(7,25)	(6,25)
11	2	1.6	0.183	0.581	0.193	0.032	6	(8,21)	(7,7)
12	2	1.7	0.205	0.601	0.255	0.035	14	(5,9)	(6,28)
13	2	1.8	0.204	0.586	0.281	0.041	24	(13,14)	(5,22)
14	2	1.9	0.181	0.543	0.374	0.052	49	(24,17)	(6,43)

Table 4: Various *MCL* runs for the matrix in Figure 14 — pf is an abbreviation of Perf(ormance).

13. Scaled experiments

In this section two experiments are described in which several graphs with 10000 nodes are clustered. In the first experiment three graphs are generated using the same underlying partition $\mathcal{P}$, but with different probabilities for the realization of edges within partition elements. In the second experiment graphs are generated with the same probabilities but with underlying partitions which have different granularity. These graphs are clustered using the same *MCL* parametrization for each. The partition $\mathcal{P}$ which was used for generating the first three test graphs has grid parameter g equal to 500 and has partition element sizes

1^{20}, 2^{10}, 3^7, 4^6, 5^5, 6^5, 7, 8^2, 9^3, 10^3, 11, 12^2, 13, 14^3, 16^3, 20, 21^2 25, 27, 31, 32, 36, 38, 40^2, 41^2, 45, 54, 58, 63, 66, 70, 74^2, 78, 81, 85, 89, 92^2, 97, 108, 111, 116, 121^2, 125^2, 129, 131, 137, 169^2, 183, 226, 255, 284, 298, 314, 343, 350, 352, 374, 392, 401, 422, 428, 444, 484, 499, 500.

The probability p was chosen equal to 0.1 and the probability q was respectively chosen as 0.002, 0.004, and 0.006. Three graphs labelled H_1, H_2, and H_3, were generated this way. A node of H_1 in the partition element of size 500 has on average 50 neighbours on the inside (with respect to this element) and 19 neighbours on the outside. The higher values of q for H_2 and H_3 imply that the cluster structure is more concealed in these graphs. Optimal clustering parameters were sought for each graph using pruning constant $k = 200$. In doing so three parameters were varied, namely the loop weight a, the initial inflation constant r, and the main inflation constant R. The length of the initial prefix was in each case chosen equal to 2. Good clusterings were sought for each of the three graphs, judged by performance coefficient only. Parameters were varied according to their observed effect on the performance coefficient; a few different trials sufficed to find good parametrizations. The *MCL* algorithm was again applied holding the corresponding parameters fixed, except for the pruning constant k which was successively decreased with a decrement of 25. The invariant parts of this setup are summarized in Table 5 on page 34.

Table 6 on page 34 shows that the cluster structure found by the *MCL* algorithm matches the generating partition $\mathcal{P}$ rather well. Note that the size of the 22^{nd} largest partition element equals 131. The nodes in this partition element have approximately 13 neighbours within the same element (for all three of H_1, H_2,

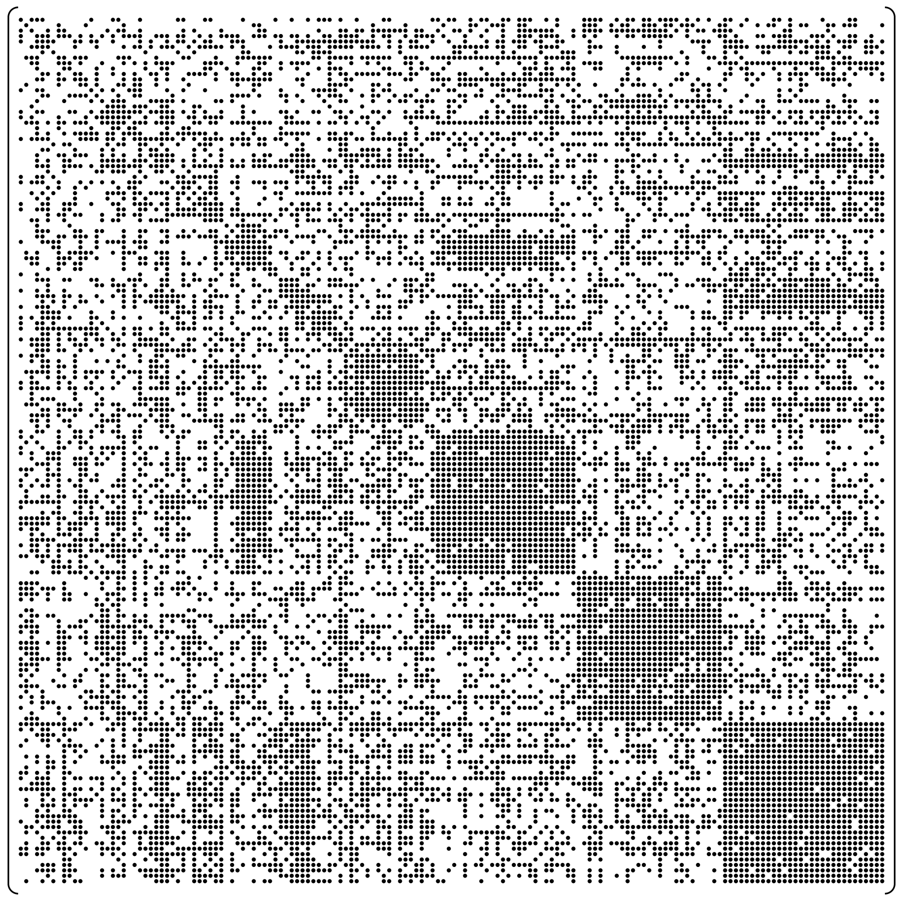


Figure 16: Square of matrix in Figure 14 aligned according to clustering found by the MCL algorithm. Dots denote nonzero entries, which are positive integers. The presence of off-diagonal vertical and horizontal bands is explained by the nature of the MCL process.

and H_3), and respectively approximately 20, 40, and 60 neighbours elsewhere (for H_1, H_2, and H_3). For all but two parametrizations this partition element is clearly recognized as a cluster. It is furthermore noteworthy that the clusterings are not very much affected by the value assumed by the pruning constant k, as long as it does not become critically low.

In the second experiment eight graphs were generated on 10000 nodes with probabilities $p = 0.1$ and $q = 0.004$, that is, the same probabilities used for H_2. The underlying partition was generated for each graph separately, where g varied from 300 to 1000. It was ensured (by repeated trials) that each partition generated with grid size g, $g = 300, 400, \ldots, 1000$ had several partition elements of size close to g. Each graph was clustered using the same set of parameters which worked best for H_2, and the pruning constant was chosen equal to 150. The results are depicted in Table 7 on page 35. They clearly show that the same parametrization works for graphs with distinctly different granularity characteristics with respect to their natural cluster structure. The MCL algorithm does not perform very well for the graph of lowest cluster granularity (corresponding with grid size g=300), which is explained by the fact that the corresponding generating partition has many clusters of small cardinality — approximately half of the nodes belong to a partition element of size 110 or less.

Pruning
A pruning scheme was used in which threshold pruning is applied first followed by exact pruning (see also page 5 and [6]). Thresholds of the form $\mathrm{ctr}(c)(1-t[\max_i(c_i)-\mathrm{ctr}(c)])$ (where c is the stochastic column vector to be pruned) seem to work best in practice, where t is chosen in the range $[1.0, 7.0]$. Considering vectors with some large and some small entries supplies the rationale for choosing $\mathrm{ctr}(c)$ as the 'reference value' for the threshold. For example, the vector $1/100(30,30,30,4,3,2,1)$ has ctr value equal to 0.273. Then it is natural to introduce $\max_i(c_i) - \mathrm{ctr}(c)$ as a correcting factor, because if this is factor is large, then the distribution of the entries of c will be smooth rather than peaked. The vector $1/100(25,20,20,15,10,5,5)$ has 0.18 as ctr value, and the amount $0.25-0.18$ seems a good 'base' correcting factor in order to prevent pruning from being too severe. Threshold pruning was *always* applied for each newly computed column vector, at every stage of the process. No attempt to readjustment was made in case thresholding caused the number of nonzero vector entries to drop below the pruning constant k or if thresholding left a number of nonzero entries much larger than k. However, early stages of the process need thresholding more severely than the middle stages of the process, as the newly computed column vectors during expansion tend to have a much larger number of nonzero entries during early stages. Thresholds that are too harsh during the middle stages have in general an adverse effect on the quality of the resulting clustering. For this reason thresholds of the form $a[\mathrm{ctr}(c)]^b$ appear to be disadvantageous, as they are difficult to tune. Using the threshold $\mathrm{ctr}(c)(1-t[\max_i(c_i)-\mathrm{ctr}(c)])$ and slowly increasing the parameter t during the process overcomes this problem. This approach is still a bit crude if t is increased regardless of the density characteristics of the iterands (such was the setup in conducting these experiments). It seems quite worthwhile to search for smart pruning schemes which are cheap to compute and effective during the various stages of the *MCL* process.

Clusterings associated with intermediate iterands
The implementation that was used in conducting these experiments computed a (possibly overlapping) clustering for each *MCL* iterand M after every expansion step. This was done by defining a directed graph G on the column indices of M by creating an arc $q \rightarrow p$ iff $M_{pq} \geq M_{qq}$. The overlapping clustering was computed as the set of all weakly connected components of G. In an ideal world (where tractability is not an issue and computers use real numbers instead of approximate values), the matrix M would have been guaranteed to be *dpsd* (since all input matrices are symmetric), and the overlapping clustering could have been computed as the set of all endclasses of the *DAG* associated with M (according to Theorem 2) joined with the nodes reaching them.

For the experiments in this section it took the *MCL* algorithm between 12 and 20 iterations (each iteration consisting of one expansion and one inflation step) to reach a matrix iterand for which the computed clustering was identical to the clustering associated with the eventual limit (which was typically reached 5 iterations later). However, the initial stages exhibited much more dramatic decreases in the number of clusters than later stages — the number of clusters was always decreasing, never increasing. The table below supplies the number of clusters and the amount of overlap associated with each iterand of the run for H_2 using pruning constant 150. The numbers are quite typical for the behaviour of the *MCL* algorithm during the scaled experiments.

Expansion step	#clusters	#nodes in overlap
1	10000	0
2	10000	0
3	7438	139
4	3430	1970
5	2455	166
6	1418	92
7	778	102
8	420	119
9	240	45
10	152	41
11	121	28
12	113	18
13	106	8
14	105	3
15	103	1
16	102	0
17	101	0
18	101	0

One of the many things that has not yet been investigated (theoretically nor empirically) is the relationship (distance) between the clusterings associated with different iterands, and the depths (i.e. the length of a longest path) of the $DAGs$ that occur. For the type of graph experimented with here I expect that the depth of the associated DAG will in general be small, on average at most 2. It is furthermore interesting to investigate how the MCL algorithm performs for graphs which have lower connectivity and natural clusters of somewhat larger diameter, but without the strong homogeneity properties of meshes and grids. Random geometric graphs such as in Figure 2 on page 5 seem suitable candidates, and I expect that for this type of graphs $DAGs$ may arise that have larger depth.

Graph label	p	q	Perf$(H_i, \mathcal{P})$	Cluster parameters			
H_1	0.1	0.002	0.0876	a=3.0	r=1.25	l=2	R=1.3
H_2	0.1	0.004	0.0805	a=3.0	r=1.20	l=2	R=1.3
H_3	0.1	0.006	0.0752	a=4.0	r=1.20	l=2	R=1.3

Table 5: Each graph H_i was generated on the same partition $\mathcal{P}$ (on 10000 nodes, grid size 500) with parameters p and q as indicated. Cluster parameters (see Table 1 on page 8) were chosen after a few trials on each graph. Table 6 gives the result of clustering each H_i with these parameters for varying pruning constants k.

k	pf$(C_i(k))$	$d(C_i(k), \mathcal{P})$	Best matches[⋆] between $\mathcal{P}$ and $C_i(k)$
			H_1
200	0.0882	(582,738)	0 0 0 0 0 1 0 0 1 0 0 0 0 2 0 1 1 3 4 2 8 7
175	0.0885	(584,780)	0 0 0 0 0 1 0 0 1 0 0 0 1 1 0 1 2 3 4 2 8 5
150	0.0893	(608,881)	1 0 0 0 0 1 0 0 0 0 0 0 1 1 0 1 5 5 4 2 7 5
125	0.0892	(613,938)	1 0 0 1 0 1 0 0 1 0 0 0 1 0 0 1 4 5 2 2 7 6
100	0.0891	(602,927)	1 2 0 0 0 1 0 0 3 0 0 0 0 0 0 1 2 3 6 2 6 9
75	0.0889	(633,1068)	6 8 0 0 0 1 0 0 3 0 0 0 1 0 0 0 1 3 4 3 6 5
			H_2
200	0.0800	(748,1154)	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 1 3 9 7
175	0.0801	(730,1043)	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 1 3 8 4
150	0.0796	(890, 722)	0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 2 1 0 9 4
125	0.0794	(764,864)	0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 1 2 1 0 12 9
100	0.0790	(760,964)	1 0 0 0 0 1 9 1 1 0 0 0 0 0 1 0 2 1 3 4 66 8
75	0.0749	(973,993)	5 0 22 8 4 6 67 5 14 7 17 7 3 51 4 2 18 31 21 40 52 26
			H_3
200	0.0731	(1014,1817)	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 4 4 24 8 17 16
175	0.0731	(1004,1791)	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 3 24 7 22 21
150	0.0730	(995,1786)	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 14 6 17 12
125	0.0723	(1045,1734)	2 3 1 1 3 1 2 5 0 2 1 0 9 1 1 1 11 2 14 8 41 18
100	0.0662	(1537,2233)	6 13 2 63 37 27 16 32 17 32 24 43 102 30 62 26 57 30 94 102 . . .
75	0.0444	(3328,6455)	256 320 161 354 236 . . . (more three digit numbers) . . .

Table 6: $C_i(k)$ denotes the clustering of graph H_i resulting from an *MCL* process with parameters as in Table 5 and pruning constant equal to k. In the second column, pf$(C_i(k))$ denotes the performance Perf$(H_i, C_i(k))$.

[⋆]The last column gives the sizes of the symmetric difference of the j^{th} largest element of $\mathcal{P}$ ($j = 1, \ldots, 22$), with that cluster of $C_i(k)$ which is the best match for the partition element.

g	pf$(\mathcal{P}_g)$	pf(C_g)	$d(C_g, \mathcal{P}_g)$	Best matches* between $\mathcal{P}_g$ and C_g
300	0.0681	0.0650	(2321,3206)	3 1 1 0 0 2 11 4 6 8 8 12 29 10 11 22 18 19 15 27
400	0.0755	0.0733	(1198,1574)	2 0 0 4 0 0 0 10 17 1 1 3 0 2 1 3 6 0 0 9
500	0.0810	0.0810	(740,870)	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
600	0.0829	0.0826	(576,632)	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
700	0.0843	0.0836	(575,671)	0 1 1 0 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 1
800	0.0838	0.0834	(499,635)	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 3 0
900	0.0863	0.0856	(368,407)	1 1 0 0 1 1 1 0 0 0 0 0 0 0 1 0 1 14 4 5
1000	0.0890	0.0888	(192,225)	1 2 2 2 1 1 1 2 1 2 0 0 1 0 1 0 0 0 1 2

Table 7: For each value of g a partition $\mathcal{P}_g$ was generated using g as grid size with n=10000. A graph H_g was generated using $\mathcal{P}_g$ and probabilities $p = 0.1$ and $q = 0.004$. Each graph H_g was clustered using the same MCL parameters a=3, r=1.2, l=2, R=1.3, k=150. In the second and third column, pf$(\mathcal{P}_g)$ and pf(C_g) respectively denote the performance coefficients Perf$(H_g, \mathcal{P}_g)$ and Perf(H_g, C_g).

*The last column gives the sizes of the symmetric difference of the j^{th} largest element of $\mathcal{P}_g$ ($j = 1, \ldots, 20$), with that cluster of C_g which is the best match for the partition element.

References

1. Edwin F. Beckenbach and Richard Bellman. *Inequalities.* Springer–Verlag, 1961.
2. Abraham Berman and Robert J. Plemmons. *Nonnegative Matrices In The Mathematical Sciences.* Number 9 in Classics in Applied Mathematics. SIAM, 1994. Corrected and extended republication of the 1979 book.
3. Béla Bollobás. The diameter of random graphs. *Transactions of the American Mathematical Society*, 267(1):41–52, September 1981.
4. Béla Bollobás. *The Evolution of Sparse Graphs*, pages 35–57. Academic Press, 1984. Proceedings of the Cambridge combinatorial conference in honour of Paul Erdős.
5. Stijn van Dongen. A new cluster algorithm for graphs. Technical Report INS–R9814, National Research Institute for Mathematics and Computer Science in the Netherlands, Amsterdam, December 1998.
6. Stijn van Dongen. A cluster algorithm for graphs. Technical report, National Research Institute for Mathematics and Computer Science in the Netherlands, Amsterdam, 2000. Revised version of [5]. To appear.
7. Stijn van Dongen. *Graph Clustering by Flow Simulation.* PhD thesis, University of Utrecht, May 2000.
8. Stijn van Dongen. A stochastic uncoupling process for graphs. Technical report, National Research Institute for Mathematics and Computer Science in the Netherlands, Amsterdam, 2000. To appear.
9. Brian S. Everitt. *Cluster Analysis.* Hodder & Stoughton, third edition, 1993.
10. Julie Falkner, Franz Rendl, and Henry Wolkowicz. A computational study of graph partitioning. *Mathematical Programming*, 66:211–239, 1994.
11. Michiel Hazewinkel. Tree–tree matrices and other combinatorial problems from taxonomy. Technical Report AM–R9507, National Research Institute for Mathematics and Computer Science in the Netherlands, Amsterdam, April 1995.
12. Roger A. Horn and Charles R. Johnson. *Matrix Analysis.* Cambridge University Press, 1990.
13. Nicholas Jardine and Robin Sibson. *Mathematical Taxonomy.* Wiley Series In Probabilistic And Mathematical Statistics. John Wiley & Sons, 1971.
14. L. Lovász. Random walks on graphs: A survey. In Miklos et al. [16], pages 353–397.
15. Albert W. Marshall and Ingram Olkin. *Inequalities: Theory of Majorization and Its Applications.* Number 143 in Mathematics In Science And Engineering. Academic Press, 1979.
16. D. Miklos et al., editors. *Combinatorics, Paul Erdős is eighty*, volume II. Janos Bolyai Mathematical Society, 1996.
17. Henryk Minc. *Nonnegative Matrices.* Wiley Interscience Series In Discrete Mathematics And Optimization. John Wiley & Sons, 1988.
18. Boris Mirkin. *Mathematical Classification and Clustering.* Kluwer Academic Publishers, 1996.
19. Thomas K. Philips, Donald F. Towsley, and Jack K. Wolf. On the diameter of a class of random graphs. *IEEE Transactions on Information Theory*, 36(2):285–288, March 1990.
20. E. Seneta. *Non–negative matrices and Markov chains.* Springer, second edition, 1981.
21. Oved Shisha, editor. *Inequalities.* Academic Press, 1965. Wright–Patterson Air Force Base, Ohio, August 19–27.