Storing messages with multipartite neural cliques

Nissim Zerbib[1], Vincent Gripon[2], and ?[3]

[1]*Département d'Informatique, École normale supérieure, Paris, France*
[2]*Département d'Électronique, Télécom Bretagne, Brest, France*

***Abstract*—We extend recently introduced associative memories based on clustered cliques to multipartite cliques. We propose a variant of the classic retrieving rule. We study its performance relatively to the former one for retrieving partially erased or corrupted messages. We provide both analytical and simulation results showing improvements in both networks efficiencies and resilience to damages. We compute asymptotic capacities of these networks.**

***Index Terms*—associative memory, error correcting code, cliques, multipartite cliques, neural networks**

I. Introduction

Associative memories

Recently, a new type of associative memories was proposed by Gripon and Berrou [3].

GBNN can also be used to retrieve messages from erroneous versions of them. We derive a formula for error rate in case of errors on messages.

Fault tolerance of those networks was extensively studied in [5]. We extend this to multipartite cliques and show significant improvements in resilience to faults.

II. Networks of neural cliques

An associative memory is able to store messages then retrieve them given a partial or corrupt probe.

We first describe how messages can be stored in a recently introduced neural network based associative memory []. Secondly, we detail how messages can be retrieved by these networks.

A. Storing messages

Consider a set $\mathcal{M}$ of messages of fixed length c over a finite alphabet $\mathcal{A}$.

Messages in $\mathcal{M}$ can be stored in a clustered neural network made of c clusters containing ℓ units each. Units can take binary values : 0 (deactivated) or 1 (activated). A message is represented by a set of activated units in the network.

One can choose the association between a message and its corresponding set of active units. In previous works [2], [4], each coordinate of a message m is associated with a cluster. Inside a cluster, each unit is associated with a symbol in $\mathcal{A}$. As a consequence, to a message m corresponds one active unit in each cluster of the network.

In this document, we are interested in assessing the performance of such systems when several units are associated with a given symbol for each cluster. To simplify this generalisation and allow for optimized retrieval strategies, we fix this number of activated units a in each cluster.

Note that there are $\binom{\ell}{a}$ possible choices for a activations among ℓ units, giving the following correspondance:

$$\binom{\ell}{a} = |\mathcal{A}| \ .$$

Without loss of generality, we therefore consider that $\mathcal{A}$ is composed of the list of all binary vectors containing exactly a ones that we index from 1 to $\binom{\ell}{a}$: $\mathcal{A} = \{\mathcal{A}_1, \dots, \mathcal{A}_{\binom{\ell}{a}}\}$.

Previous works [2], [4] consider the case $a = 1$.

To provide convenient notations, we propose to linearise messages using the following function mapping f:

$$f : \left\{ \begin{array}{rcl} \mathcal{A}^c & \rightarrow & \{0,1\}^{c \cdot \ell} \\ m & \mapsto & \mu | \forall j, \mu[(j-1)\ell + 1 \dots j \times \ell] = m_j \end{array} \right. ,$$

where $\mu[a..b]$ is the subarray of μ from coordinate a to coordinate b.

By abusing notations, and since function f is injective, we shall make no distinction between m and $f(m) = \mu$.

To store a message m in the network, we use the edges between units. More precisely, to store a message μ all edges between activated units are added to the network, with the exception of units in a same cluster. In other words, storing a message consists in printing a multipartite clique into the network. This process is depicted in Figure 1. Notice that edges already present in the network remain unchanged throughout the storing process.

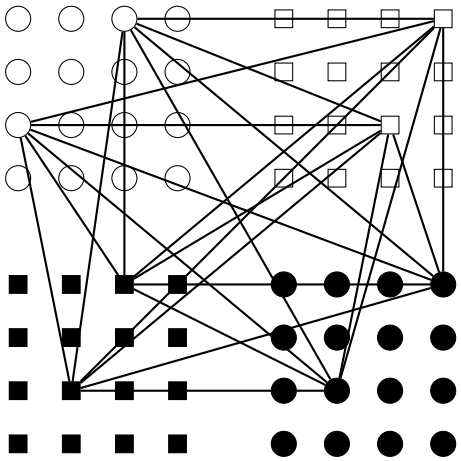

Figure 1. Illustration of the storage process of a message μ. In this setting the number of clusters is $c = 4$, each containing $\ell = 16$ units. The parameter a is 2.

Note that this representation is lossy, as the number of possible sets of messages is much larger than the number of possible networks.

Formally, we can describe the network using the adjacency matrix W of the underlying graph. By construction this matrix is symmetric and binary. For a message μ, note that $\mu\mu^\intercal$ is the cooccurence matrix associated with μ. Before defining W, we first introduce the following matrix:

$$\overline{W} = \max_{\mu \in \mathcal{M}} \mu\mu^\intercal \ ,$$

where $\max$ is a coefficient-to-coefficient max function. W is obtained from $\overline{W}$ by removing all edges connecting units in a same cluster. We obtain a binary symmetric matrix W with null blocks on the diagonal. The reason why we remove intracluster connections will be discussed in the following section.

As $\max$ is associative and commutative, storing can be done online and is independent of the order in which messages are stored.

B. Retrieving messages

Once a set of messages has been stored in some matrix W, we are interested in retrieving a stored message given a partial or corrupt probe $\tilde{\mu}$. To perform this operation, we use an iterative algorithm derived from the one proposed in [3].

The algorithm relies on two steps. First it computes *scores* for each unit in the network and then select which units should be activated based on their scores. For a review on different strategies for both steps in the classical case where $a = 1$, consider [1]. To adapt this algorithm in the case $a \geqslant 2$, we only modify the second step for choosing which units to be activated.

There are several strategies for computing scores of the units, but our simulations suggest that their impact on performance is not significative. We therefore use what is called the SUM-OF-SUM rule in which scores are computed as follows:

$$s = \gamma\tilde{\mu} + W\tilde{\mu} \ .$$

In other words, the score of a unit is the number of activated units it is connected to, plus a memory effect with parameter γ. This memory effect can have significative impact on performance and its optimal value depends on the problem considered. This is the main reason why intracluster edges are forbidden.

Once scores are computed, we are interested in choosing which units to be activated. In order to account for the fact several units should be activated in each cluster, we propose to activate the α units achieving the maximum score in each cluster. Note that in case of equality of scores, we select all tied units resulting in possibly more than α units activated in each cluster. Intuitively, the best possible choice for α is a, but it implies to have some prior knowledge about the value of a. If one does not know the value of a, choosing $\alpha = 1$ should lead to reasonable performance.

Formally, we introduce a function select such as for an array $\tilde{\mu}$, select$(\alpha, \tilde{\mu})$ is equal to the α-th greatest element of μ (counting repetitions): as an example select$(3, [4, 2, 1, 2, 0, 2]) = 2$. Then the selection of activated units is performed as follows:

$$\tilde{\mu}'[(j-1)l + k] = \begin{cases} 1 & \text{if } s[(j-1)\ell + k] \geqslant \text{select}(\alpha, s[(j-1)\ell + 1 \dots j\ell]) \\ 0 & \text{otherwise} \end{cases} .$$

Thus we use an Heavyside function with a parameter depending on each cluster. $\tilde{\mu}'$ is the output of the algorithm after one iteration. The process can be repeated providing $\tilde{\mu}'$ as the new input to the network. Several stopping criterion can be used [1]. In the remaining of this work, we will consider the number of iterations to be fixed. Simulations show that increasing the number of iterations can result in significatively better performance.

The network is considered to succeed if the final output of the algorithm matches the original message of which $\tilde{\mu}$ is a partial or corrupted version of.

We now study the performance of the new retrieving rule on distinct problems of retrieval : first partially erased messages, then erroneous messages.

III. Retrieval performance

Messages will be considered as independent and identically distributed according to the uniform distribution. We compute theoretical error rates after one iteration of the retrieving rule. We compare those results to empirical error rates obtained through simulations[1]. We also compare efficiencies of distinct networks for distinct values of the parameter a (weight of the constant weight code) in the sense of information theory and show improvements over the case where $a = 1$.

A. Retrieving partially erased messages

One can see a partial version of a message x as the result of erasures of symbols of x. Some symbols of x are replaced by the blank character $\intercal$. Erasures of a symbol in a message corresponds to an "erased" cluster in the network. An erased cluster is a cluster of units which are all inactive, that is to mean set to 0.

1) Analytical result: We first compute the theoretical density of the network, that is to mean the probability for a random edge (i, i') to be present in the network (i.e. $W_{ii'} = 1$). We then express the theoretical error rate after one iteration of the retrieving rule in function of the density. All this computations (on error rates) are done assuming existences of edges between random units are independent, which is false but is a convenient approximation that gives results matching very closely results from simulations.

As the probability for i (and i') to be active in its cluster for one message x is $\frac{a}{l}$ providing messages are uniformly distributed, it follows that the probability for the edge (i, i') of not being added in the network for x is $1 - (a/l)^2$. Since messages are independent, the density d, which is the

[1]Simulations are done by taking means on pool of five networks and messages sets, and each point is drawn for a different network and messages set.

probability for an edge to be in the network, can be expressed as :

$$d = 1 - \left(1 - \left(\frac{a}{l}\right)^2\right)^M \quad (1)$$

, recalling that M is the number of messages in $\mathcal{M}$.

Suppose we are trying to retrieve a message where c_e clusters have been erased. Thanks to the memory effect γ, activated units in a non-erased cluster stay activated and are the only ones being so in their cluster (as the others can't achieve a higher score thanks to not being already activated). For units in erased clusters, the maximum attainable score is $a(c - c_e)$. Units in erased clusters corresponding to the original message achieve this score. Since scores follow a binomial law, the probability for a random unit to achieve the maximum score in such a cluster, that is to mean $a(c - c_e)$ is $d^{a(c-c_e)}$.

Since unit activations are independent (as messages are themselves independent), the probability for none of the vertices of one erased cluster, excepting the correct ones, to achieve the maximum is $\left(1 - d^{a(c-c_e)}\right)^{l-a}$.

Scores in clusters being also independent, the probability for none of the vertices in any erased cluster, excepting the correct ones, to achieve the maximum in their respective clusters is $\left(1 - d^{a(c-c_e)}\right)^{c_e(l-a)}$.

Whence error rate in retrieving messages is :

$$P_{err} = 1 - \left(1 - d^{a(c-c_e)}\right)^{c_e(l-a)} \quad (2)$$

In order to compare distinct networks using distinct alphabets (due to variations of the weight parameter a), we define the efficiency of the network relatively to the hardness of the problem (how much it is efficient compared to a perfect associative memory). A perfect associative memory would use exactly the same number of bits than the original messages and retrieve messages by maximum likelihood (ambiguities causing errors). Efficiency is the amount of information retrieved divided by the hardness of the problem.

The network is a binary symmetric matrix with null diagonal and can therefore be stored in memory as $\frac{c(c-1)l^2}{2}$ bits. Since messages are not stored in an ordered fashion, the entropy of the set of messages is $\log_2(\frac{(\binom{l}{a}^c)^M}{M!}) =_{M\to+\infty} M(c\log_2\binom{l}{a} - \log_2(M) + \frac{1}{\log 2} + o(1))$ (thanks to Stirling's formula). The ratio of information effectively stored in the network is $P_{retrieve} \times \frac{2M\left(c\log_2\binom{l}{a} - \log_2(M) + \frac{1}{\log 2}\right)}{c(c-1)l^2}$.

Ambiguities would occur in a perfect associative memory if at least two messages were identical over non-erased letters. The probability it doesn't happen is $(1 - \binom{l}{a}^{c-c_e})^{M-1}$.

$\eta_M = P_{retrieve}\frac{2M\left(c\log_2\binom{l}{a} - \log_2(M) + \frac{1}{\log 2}\right)}{c(c-1)l^2} \times (1 - (\frac{1}{\binom{l}{a}})^{c-c_e})^{-(M-1)}$

The maximum ratio of information that can be stored in the network is therefore $\eta = \max_x \eta_x$, which we will define as the efficiency of the network.

2) Simulations: Theoretical and empirical densities match very closely, see Figure 2.

Simulations agree with the analytical results for error rate, albeit we can observe a small shift, see Figure 3. The theoretical error rate is a bit optimistic which is due to the connections not being independent.

For one iteration, as long as the parameter w is between 1 and a, there is no effect on retrieving. However for 4 iterations, the choice of $w = a$ allows a shift of about 5000 messages stored on top of those stored for $w = 1$. The variant rule can thus be considered a significant improvement over the classic one for multipartite clique networks (recall that $w = 1$ amounts to the classic retrieving rule).

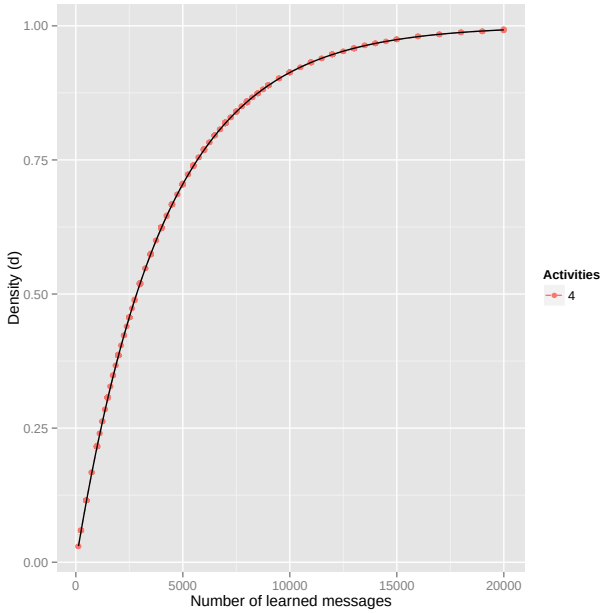

Figure 2. Theoretical and empirical densities for a network of 8 clusters, 256 units per cluster and 4 activated units per cluster

Thanks to the efficiency defined in the previous section, we can easily compare distinct choices of parameters a (we now take $w = a$ since it is the most efficient choice in practice) over distinct number of erased clusters. Figure 4 shows improvements in efficiencies over parameter $a = 1$. The more difficult the problem is, the more redundancies introduced by multipartite cliques are useful.

Messages can suffer many types of alteration. Multipartite clique neural networks can also correct errors.

B. Retrieving corrupted messages

1) Analytical result: We provide a formula for error rate in case of messages with corrupted clusters.

Messages can be corrupted, our models for errors is : given c_e the number of erroneous clusters, we draw at random c_e clusters and in each of these draw at random, uniformly, a new letter in the alphabet that will replace the old one (it could be the same).

Knowing the density of the network, which follows formula (1), we can compute the laws of probabilities for scores achieved by different type of units in different clusters. We can divide units in 5 groups. Subscript g will indicate a unit

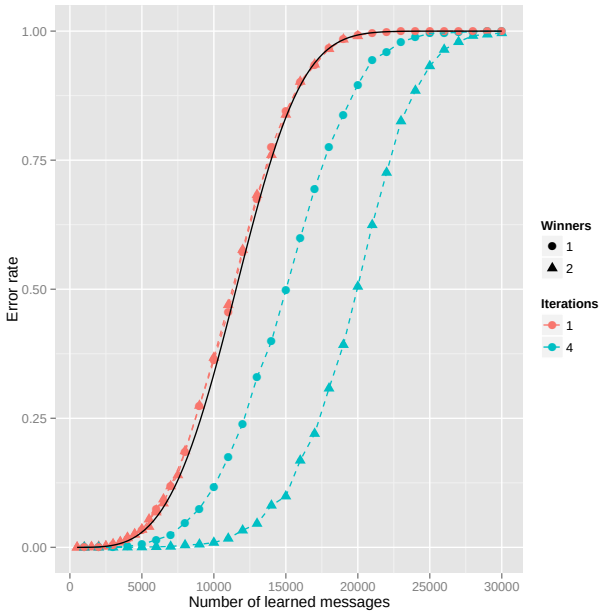

Figure 3. Evolution of error rate for increasing number of stored messages, 2 activities per cluster, 4 clusters, 512 units per cluster, two erasures, each point is the mean of 5 networks with 1000 sampled messages, analytical result (for one iteration) in continuous black

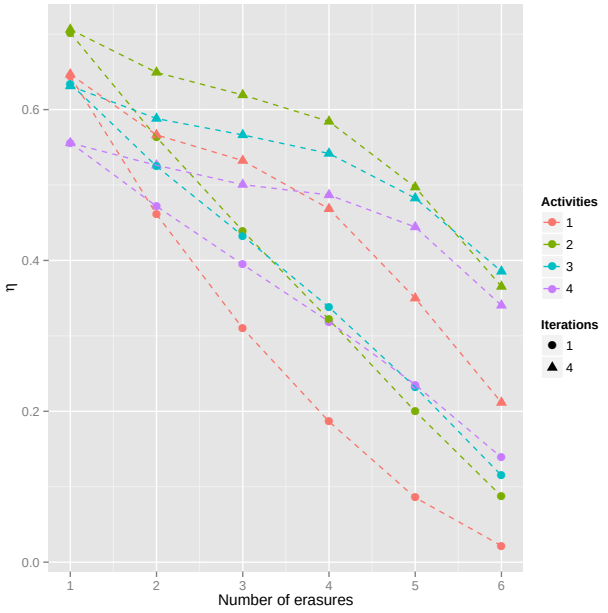

Figure 4. Comparison between different number of activities per cluster : 8 clusters, 256 units per cluster, 1000 sampled messages, pools of 5

being part of the message (it should be activated), subscript b the contrary (the unit should not be activated). Subscript a means the unit is already activated. Subscript c corresponds to a correct cluster, e to an erroneous cluster. Correct units in uncorrupted clusters achieve at least a score of $a(c-c_e-1)+\gamma$. Being connected to a given activated unit in a corrupted cluster occurs with probability d, since errors are identically distributed. Moreover corrects units see their scores shift, as do all activated units respectively thanks to being part of the original message and the memory effect γ. We can express those probability laws as : For example an **a**ctivated **g**ood unit in a **c**orrect cluster starts with a base score of $a(c-c_e-1)+\gamma$. The probability it has to be connected to a random activated unit in an erroneous cluster is d. Thereby its law is :

$P(n_{agc} = a(c-c_e-1)+\gamma+x) = \binom{ac_e}{x} d^x (1-d)^{ace-x}$

Correct units in corrupted clusters achieve at least a score of $a(c-c_e)$. $P(n_{ge} = a(c-c_e)+x) = \binom{a(c_e-1)}{x} d^x (1-d)^{a(c_e-1)-x}$

Activated units benefits of the memory effect γ :

$P(n_{abe} = \gamma + x) = \binom{a(c-1)}{x} d^x (1-d)^{a(c-1)-x}$

Finally $P(n_{be} = x) = P(n_{bc} = x) = \binom{a(c-1)}{x} d^x (1-d)^{a(c-1)-x}$

A message is retrieved after one iteration, if and only if correct units achieve strictly the best scores.

The probability for all good units of a correct cluster (resp. of an erroneous cluster) to achieve a score equal or greater than x, with at least one having a score of x is $P_{ac}(x) = \sum_{k=1}^{a} \binom{a}{k} P(n_{agc} = x)^k P(n_{agc} > x)^{a-k}$ (resp. $P_e(x) = \sum_{k=1}^{a} \binom{a}{k} P(n_{ge} = x)^k P(n_{ge} > x)^{a-k}$)

$$P_{retrieve} = \left[\sum_{x=1}^{a(c-1)+\gamma} P_{ac}(x) P(n_{bc} < x)^{l-a} \right]^{c-c_e} \times \left[\sum_{x=1}^{a(c-1)+\gamma} P_e(x) P(n_{be} < x)^{l-2a} P(n_{abe} < x)^a \right]^{c_e} \quad (3)$$

with the approximation that there is few activated units that should be kept active in an erronenous cluster.

$P_{err} = 1 - P_{retrieve}$

2) Simulations: See Figure 5 compares the two rules. "a-winners take all" better than "winner takes all" with errors instead of erasures, even for one iteration.

$$\eta_m = P_{retrieve} \frac{2m\left(c\log_2\binom{l}{a} - \log_2(m) + \frac{1}{\log 2}\right)}{c(c-1)l^2} \times \left[\sum_{k=c_e+1}^{c} \left(1 - \frac{1}{\binom{l}{a}}\right)^k \left(\frac{1}{\binom{l}{a}}\right)^{c-k} \right]^{-(m-1)}$$

IV. Resilience

A. Analytical result

As information on a message is shared across edges with multiple activated units per cluster, such networks are more resilient to damages (for one message information is carried by $a^2\binom{c}{2}$ edges. v_c will indicate a correct unit that should be activated, v the other type of units.

Our deviation model is the same as [5][2], that is random binary noise on edges. We generalise results of [5] to multi-partite cliques.

We assume $\gamma = \infty$ for one iteration as it prevents activated neurons to be deactivated. With probability $1-\psi$, one correct vertex is still connected to a given activated unit in an intact cluster. Thus $P(n_{v_c} = x) = \binom{a(c_k-1)}{x}(1-\psi)^x \psi^{a(c-c_e-1)-x}$

[2]Compared to this article, we do not draw at random a letter in case of ambiguity, which hurts performance a bit.

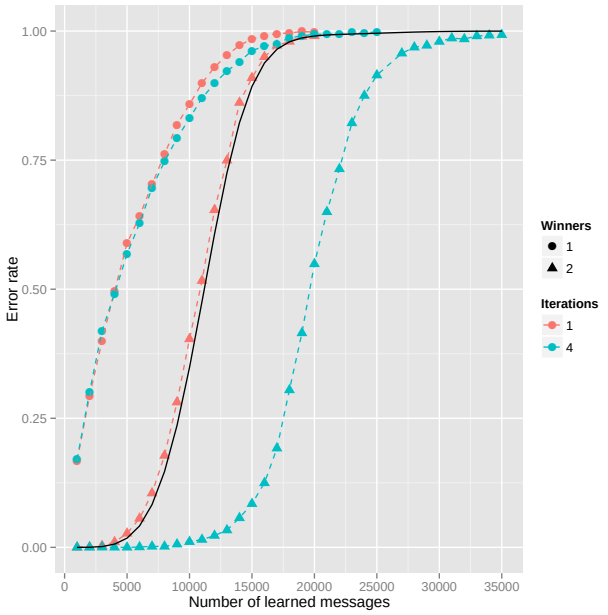

Figure 5. 2 activities per cluster, 4 clusters, 512 units per cluster, one erroneous cluster, each point is the mean of 5 networks with 1000 sampled messages, analytical result in continuous black

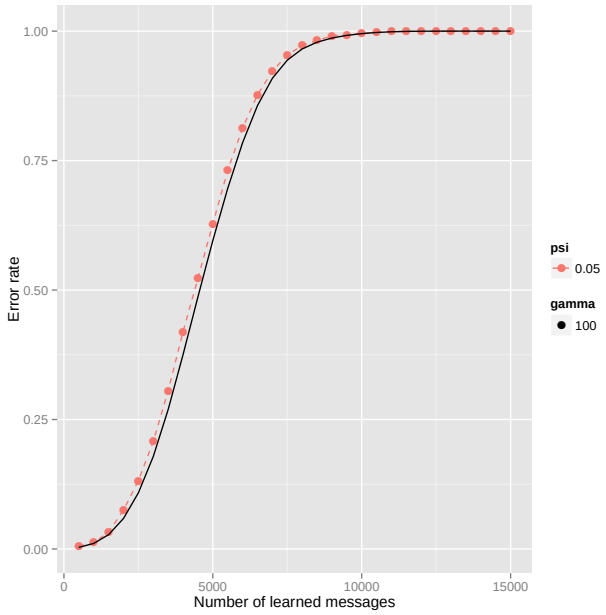

Figure 6. $\psi = 0.05$, 2 activities per cluster, 8 clusters, 256 units per cluster, 4 erasures, each point is the mean of 10 networks with 1000 sampled messages, analytical result in continuous black

The probability for an edge to be present in the network after damages is : $P_+ = \psi(1-d) + (1-\psi)d$. So for other units $P(n_v = x) = \binom{a(c-c_e)}{x} P_+^x (1-P_+)^{a(c-c_e)-x}$.

The probability for all correct units to achieve a score equal or greater than x, with at least one achieving a score of x is $P_c(x) = \sum_{k=1}^{a} \binom{a}{k} P(n_{v_c} = x)^k P(n_{v_c} > x)^{a-k}$

It follows that $P(\text{no other vertex activated in one cluster}) = \sum_{x=1}^{a(c-c_e)} P_c(x) P(n_v < x)^{l-a}$.

$$P_{retrieve} = \left(\sum_{x=1}^{a(c-c_e)} P_c(x) P(n_v < x)^{l-a} \right)^{c_e} \tag{4}$$

B. Simulations

With random noise on edges, hardness of the problem must take into account the capacity of the binary symmetric channel (the closer ψ is of 0.5, the harder the problem is). Our new is efficiency is written as : $\eta_{m,\psi} = P_{retrieve} m \frac{2\left(c\log_2\binom{l}{a} - \log_2(m) + \frac{1}{\log 2}\right)}{c(c-1)l^2} \times \left(1 - \binom{l}{a}^{c-c_e}\right)^{-(m-1)} \times \frac{1}{1+\psi\log_2(\psi)+(1-\psi)\log_2(1-\psi)}$

$\eta_\psi = max_x\, \eta_{x,\psi}$

We compute values represented on figure 7 by scanning on the number of messages, so as to find the maximum efficiency.

After a transition, efficiencies seem to follow a linear decreasing low according to ψ.

Absolute values of slopes diminish for increasing number of activities per cluster. We can conclude usage of multipartite cliques improve the resilience of the network.

Figure 7 seems to indicate a linear relationship (after a transition for multiple activities) between $\eta_\psi - \eta_0$ and ψ.

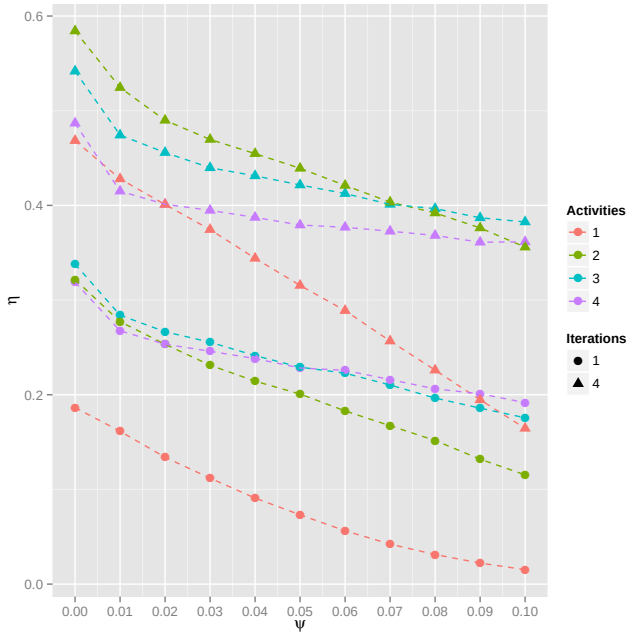

Figure 7. Comparison of evolutions of efficiency with increasing damages between different number of activities per cluster : 8 clusters, 256 units per cluster

V. Maximum theoretical efficiency

It was shown in [6] that maximal asymptotic capacity of Willshaw networks is $\log 2$. We adapt this result to multipartite clique clustered networks.

The entropy of the messages set can be approximated as $H(\mathcal{M}) \approx mc\log_2\binom{l}{a}$. From $d = 1 - \left(1 - \left(\frac{a}{l}\right)^2\right)^m$, we

can deduce $m \sim \frac{\log_2(1-d)}{\log_2(1-(\frac{a}{l})^2)} \sim -\frac{l^2 \log_2(1-d)\log 2}{a^2}$. We set $\log_2 \binom{l}{a} = kc$ (otherwise density approaches 0 or 1, and efficiency 0). It follows that $\eta \sim \frac{mc \log_2 \binom{l}{a}}{l^2 \binom{c}{2}} \sim \frac{kc^2 \log_2(1-d)\log 2}{a^2 \binom{c}{2}}$.

$\eta \sim -2ka^{-2} \log 2 \times \log_2(1-d)$.

We can bound the probability $P_{\exists e}$ for a message being "wrongly present" in the network, that is to mean adding it will not add any edge to the network albeit it is not in $\mathcal{M}$. By union-bound $P_{\exists e} \leqslant \binom{l}{a}^c d^{a^2\binom{c}{2}} = 2^{c \log_2 \binom{l}{a} + a^2 \binom{c}{2} \log_2(d)} \approx 2^{c^2(k + \frac{a^2}{2}\log_2(d))}$.

Fixing $k^* = -\frac{a^2}{2}\log_2(d)$ gives us $P_{\exists e}$ and $\eta \sim \log_2 d \log_2(1-d) \log 2$ and therefore

$$\eta_{max} = \log 2$$

Multipartite cliques don't improve theoretical asymptotic efficiency (for not accepting messages that weren't previously seen by the network).

VI. Conclusion

References

[1] Ala Aboudib, Vincent Gripon, and Xiaoran Jiang. A study of retrieval algorithms of sparse messages in networks of neural cliques. In *Proceedings of Cognitive 2014*, May 2014. To appear.

[2] Vincent Gripon and Claude Berrou. A simple and efficient way to store many messages using neural cliques. In *Proceedings of IEEE Symposium on Computational Intelligence, Cognitive Algorithms, Mind, and Brain*, pages 54–58, Paris, France, April 2011.

[3] Vincent Gripon and Claude Berrou. Sparse neural networks with large learning diversity. *IEEE Transactions on Neural Networks*, 22(7):1087–1096, July 2011.

[4] Vincent Gripon and Claude Berrou. Nearly-optimal associative memories based on distributed constant weight codes. In *Proceedings of Information Theory and Applications Workshop*, pages 269–273, San Diego, CA, USA, February 2012.

[5] François Leduc-Primeau, Vincent Gripon, Michael Rabbat, and Warren Gross. Cluster-based associative memories built from unreliable storage. In *ICASSP*, May 2014. To appear.

[6] Günther Palm. On associative memories. 1980.