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 correspondence:

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

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:

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

where $\mu[a \dots b]$ 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.

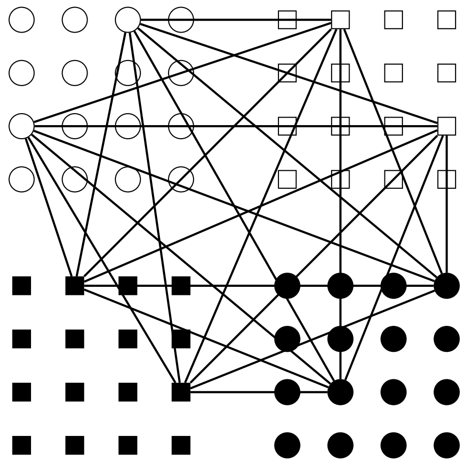

Figure 1. Illustration of the storage process of a message μ. In this setting the number of clusters is $c = 4$, each containing $\ell = 16$ units. The parameter a is 2.

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 co-occurrence 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 $\tilde{\mu}$, $\text{select}(\alpha, \tilde{\mu})$ is equal to the α-th greatest element of μ (counting repetitions): as an example $\text{select}(3, [4, 2, 1, 2, 0, 2]) = 2$. Then the selection of activated units is performed as follows:

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

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 describe some static properties of the networks as the density of edges.

III. Static properties of multipartite clique networks

We study the properties of the network once some messages have been stored in it. We are interested in how much edges have been added to the network We also compute the rate of the network in the sense of information theory (ratio between information stored in the network and the information used to store the network)

In this document, messages stored in the network will be considered independent and identically distributed according to the uniform distribution. This randomization allows us to compute the theoretical density of the network. Density is a crucial parameter of the network that can be used to predict retrieval rates for different problems. Moreover we compute asymptotic efficiency thanks to the formula for density.

A. Density

The density d of the network is the proportion of edges present in the network : $d = \frac{\#\{(i,i')\} \, | W_{ii'}=1\}}{c(c-1)\ell^2}$.

1) Theoretical density: We first compute the theoretical density of the network, that is to mean the probability for an edge (i, i') to be present in the network (i.e. $W_{ii'} = 1$).

The probability for i (and i') to be active in its cluster for one message μ is a/ℓ providing messages are uniformly distributed. It follows that the probability for the edge (i, i') of not being added in the network after μ is being stored is $1 - (a/l)^2$. Since messages are independent, the density d, which is the probability for an edge to be in the network, can be expressed as :

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

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

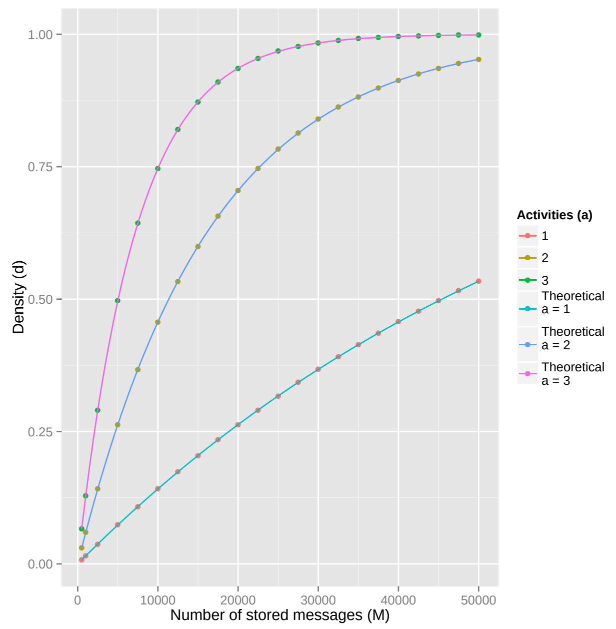

Figure 2. Theoretical and empirical densities for networks of 5 clusters, 256 units per cluster

2) Simulations: Theoretical and empirical densities match very closely, see Figure 2. The higher a is, the quicker density approaches 1 : multipartite cliques saturate quicker, which was expected since they take much more edges ($a^2\binom{c}{2}$).

B. Rate of the network

The rate R of the network measures how much information is stored in the network compared to what it takes to store the network itself. Note it doesn't take into account how effective retrieving messages is (if we store too much messages in the network, the proportion of retrieved messages approaches 0 but the rate is high)

The matrix W used to store the network is binary, symmetric, with null blocks of size ℓ^2 on the diagonal. It can therefore be stored in memory as $\frac{c(c-1)\ell^2}{2}$ bits.

Since messages are not stored in an ordered fashion, the entropy $H(\mathcal{M})$ of the set of messages is $\log_2(\frac{(\binom{\ell}{a}^c)^M}{M!}) =_{M\to+\infty} M(c\log_2\binom{\ell}{a} - \log_2(M) + \frac{1}{\log 2}) + O(\log_2(M))$ (thanks to Stirling's formula).

R can be written as $\frac{2H(\mathcal{M})}{c(c-1)\ell^2}$.

A very good approximation is thereby

$$R \approx \frac{2M\left(c\log_2\binom{\ell}{a} - \log_2(M) + \frac{1}{\log 2}\right)}{c(c-1)\ell^2} \quad (2)$$

The efficiency of the network measures how much information can be retrieved compared to how much could be at the maximum. The efficiency is called η. $\eta = P_{retrieve}Rh$.

C. 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.

Asymptotic efficiency of multipartite clique networks is also $\log 2$.

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

We now describe the distinct problems of messages retrieval that we study the performance of.

IV. Problems statements

We are interested in distinct problems of messages retrieval. We detail what those problems are. Since those problems vary in hardness, we propose distinct measures for efficiencies. Efficiency measures the ratio of information effectively retrieved times the hardness of the problem. It can be expressed as the product of the rate of the network R (how much information is stored, c.f. (2)), the retrieval rate $P_{retrieve}$ (how much is effectively retrieved) and some measure of hardness of the problem h (how hard the problem is).

$$\eta_M = P_{retrieve}R\,h \quad (3)$$

However what we are really interested in is the maximum as we want to exploit the network at its best. The maximum efficiency attainable with a given retrieving rule, which we will call efficiency, is thereby

$$\eta = \max_{M\geqslant 1}\eta_M \quad (4)$$

To account for hardness of problems, we consider an "optimal" associative memory. In this document, we are interested in retrieving messages with a few iterations at most. However an "optimal" memory would store $\mathcal{M}$ with $H(\mathcal{M})$ bits and recover messages by maximum likelihood (with ambiguities causing errors in our study, to simplify the problem). For example, such a memory would scan over all messages to find all messages corresponding to a partial probe. We detail the performance of such a memory for each problem, which gives use a measure for the hardness of each problem.

A. Erasures

1) Model: A message m can be recovered from a partial probe $\tilde{m}$ where some symbols, say c_e, of indexes $\{j_1,\dots,j_{c_e}\}$, have been replaced by the blank symbol $\top$, with $\top \notin \mathcal{A}$ (recall $\mathcal{A}$ is the alphabet we are working on). We have $\tilde{m} \in (\mathcal{A}\cup\{\top\})^c$.

We can extend the mapping f to $(\mathcal{A}\cup\{\top\})^c$, by associating to the symbol $\top$ an "erased cluster" where none of the units are activated. If we consider the binary representation μ of m, erased symbols corresponds to erased clusters. Formally the partial probe $\tilde{\mu}$ is such that for all $j\in[c]$ and $k\in[\ell]$,

$$\tilde{\mu}[(j-1)\ell+k] = \begin{cases} 0 & \text{if } j\in\{j_1,\dots,j_{c_e}\} \\ \mu[(j-1)\ell+k] & \text{otherwise} \end{cases}$$

2) Hardness of the problem: We compute the probability for a message m and a partial probe $\tilde{m}$ to cause confusion.

Let $j\notin\{j_1,\dots,j_{c_e}\}$ be the index of a non-erased symbol in $\tilde{m}$. The probability for another message m' in $\mathcal{M}$ to satisfy $m'_j = \tilde{m}_j$ is $1/|\mathcal{A}| = \binom{l}{a}^{-1}$. As symbols of a message are independent (since messages are uniformly distributed), the probability that for all $j \notin \{j_1, \dots, j_{c_e}\}$, $m'_j = \tilde{m}_j$ is $\binom{\ell}{a}^{-(c-c_e)}$. Since messages are independent the probability for all other messages in $\mathcal{M}$ not to be equal to $\tilde{m}$ on non-erased symbols is $\left(1 - \binom{\ell}{a}^{-(c-c_e)}\right)^{M-1}$. We define $h_{erasures}$ the hardness of the problem as the inverse of that last probability :

$$h_{erasures} = \left(1 - \binom{\ell}{a}^{-(c-c_e)}\right)^{-(M-1)} \tag{5}$$

Notice increasing the size of the alphabet makes the problem easier as confusion is less likely.

An erasure is tantamount to lack of information, but associative memories can also retrieve messages from corrupt probes.

B. Random replacements

1) Model: We propose one model of message corruption. As all symbols in $\mathcal{A}$ all contains exactly a ones, symbols which do not belong to $\mathcal{A}$ can be easily detected. We therefore restrict ourselves to random replacements by symbols in $\mathcal{A}$.

A message m can be recovered from a corrupt probe $\tilde{m}$ where some symbols, say c_e, of indexes $\{j_1, \dots, j_{c_e}\}$ have been replaced by random elements of $\mathcal{A}$. We have $\tilde{m} \in \mathcal{A}^c$.

Concerning the binary representation, we now have a corrupt probe $\tilde{\mu} = f(\tilde{m})$.

2) Hardness of the problem: If a message m' is equal to $\tilde{m}$ over k symbols with $k > c - c_e$, the likelihood principle leads to the conclusion $\tilde{m}$ is more likely to be a probe for m' than for m. This causes errors in the retrieving process. As ambiguities also causes errors in our model, there should not be a message $m' \neq m$ in $\mathcal{M}$ that is closer or as close as m of $\tilde{m}$ for the Haaming distance (number of different indexes where symbols differ).

The probability for an index j and a message m' that $m'_j = \tilde{m}_j$ is $|\mathcal{A}|^{-1} = \binom{\ell}{a}^{-1}$. The number of symbols over which m' and $\tilde{m}$ are equal follows a binomial law of Bernoulli parameter $\binom{\ell}{a}^{-1}$, therefore the probability that m' is at a Hamming distance of at least $c_e + 1$ of $\tilde{m}$ is $\sum_{k=c_e+1}^{c} \binom{c}{k} \left(1 - \binom{\ell}{a}^{-1}\right)^k \binom{\ell}{a}^{-(c-k)}$. As messages are independent we deduce :

$$h_{corrupt} = \left[\sum_{k=c_e+1}^{c} \binom{c}{k} \left(1 - \binom{\ell}{a}^{-1}\right)^k \binom{\ell}{a}^{-(c-k)}\right]^{-(M-1)} \tag{6}$$

C. Resilience

1) Model: The problem of resilience takes into account the possibility for the network of sustaining damages. Despite those, the network can still work as an associative memory, albeit with decreased performance. In our model, the network W first sustains random binary noises of magnitude ψ. The new matrix is called $\tilde{W}$ and satisfies for all i and i' in $[c \cdot \ell]$,

$$W_{ii'} = \begin{cases} 0 = W_{ii'} & \text{if } i \text{ and } i' \text{ are in the same cluster} \\ 1 - W_{ii'} & \text{with probability } \psi \\ W_{ii'} & \text{with probability } 1 - \psi \end{cases}$$

2) Hardness of the problem: The resilience problem is composed of two problems : transmitting information (the matrix W) through a binary symmetric channel and retrieving messages with the output of the channel $\tilde{W}$. We therefore express $h_{resilience}$ as the product of the two respective hardnesses. The capacity $\mathcal{C}(\psi)$ of the binary symmetric channel (with noise ψ) measures how much information can be transmitted through this channel with vanishing probability of error. We have $C(\psi) = 1 - H(\psi) = 1 + \psi \log_2(\psi) + (1-\psi)\log_2(1-\psi)$. We choose its inverse to qualify the hardness of transmitting through this canal. We have already computed the hardness of the "erasures" problem (c.f. (5)). Therefore :

$$h_{resilience} = h_{erasures} \times (1 - H(\psi))^{-1} \tag{7}$$

, where $H(\psi) = -(\psi \log_2(\psi) - (1-\psi)\log_2(1-\psi)$.

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

V. Retrieval performance

Thanks to the theoretical formula for density (1), we compute analytical 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 $a = 1$. Those efficiencies are computed using the empirical retrieval rate and formulas of section IV.

A. Retrieving partially erased messages

1) Analytical result: We 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.

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

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

vertices of one erased cluster, excepting the correct ones, to achieve the maximum is $\left(1 - d^{a(c-c_e)}\right)^{\ell - a}$.

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 $\left(1 - d^{a(c-c_e)}\right)^{c_e(\ell - a)}$.

Whence error rate in retrieving messages is :

$$P_{err} = 1 - P_{retrieve} = 1 - \left(1 - d^{a(c-c_e)}\right)^{c_e(\ell - a)} \quad (8)$$

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

For one iteration, as long as the parameter α is between 1 and a, there is no effect on retrieving (in this setting selecting amounts to take the maximum as all units of the original message achieve the same score). However for 4 iterations, the choice of $\alpha = a$ allows a shift of almost 5000 messages stored in addition to those stored for $\alpha = 1$. The variant rule can thus be considered as a significant improvement over the classic one for multipartite clique networks (recall that $\alpha = 1$ amounts to the classic retrieving rule).

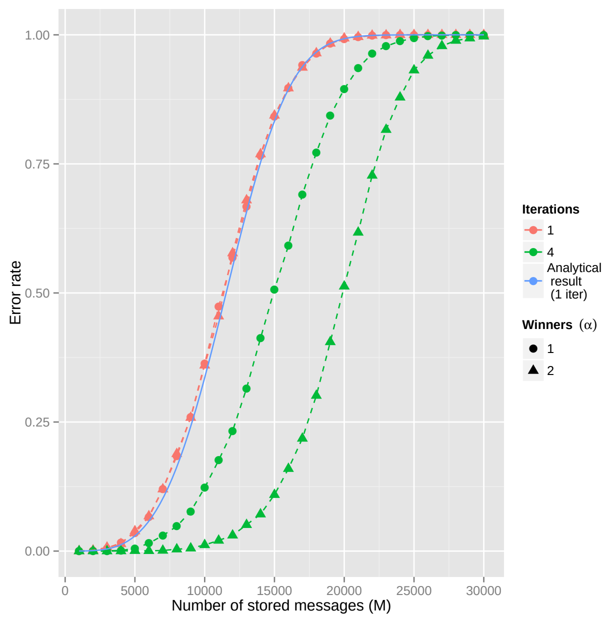

Figure 3. Evolution of error rate for increasing number of stored messages, 2 activities per cluster, 4 clusters, 512 units per cluster, two erasures, each point is the mean of 10 networks with 1000 sampled messages, analytical result (for one iteration) in continuous black

Thanks to the efficiency defined in the previous section, we can easily compare distinct choices of parameters a (we now take $\alpha = a$ 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.

We now examine the case of retrieving messages from corrupt probes.

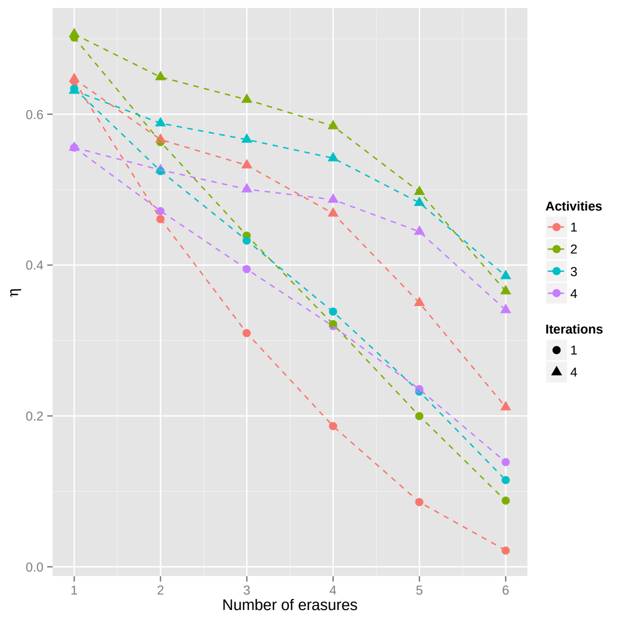

Figure 4. Comparison between different number of activities per cluster : 8 clusters, 256 units per cluster, 1000 sampled messages, pools of 5

B. *Retrieving corrupted messages*

1) Analytical result: We provide a formula for error rate in case of retrieving messages from a corrupt probe.

Knowing the density of the network, which follows formula (1), we can compute distributions for scores achieved by different type of units in different clusters.

We can divide units in 5 groups with 3 characteristics. Subscript g will indicate a unit being part of the message (it should be activated for the message to be retrieved), subscript b the contrary (the unit should not be activated). Subscript a means the unit is already activated, subscript d that it is deactivated at the beginning. Subscript c corresponds to a correct cluster, e to an erroneous cluster. For example s_{agc} designates the score achieved by an already *a*ctivated unit (a) which should stay activated (g) in an uncorrupted cluster (c). Note that there is some redundancy.

We consider a to be small in regard to ℓ which allows us to ignore the possibility for a random replacement cluster to share common activated units with the original.

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. Being connected to a given activated unit in a corrupted cluster occurs with probability d, since errors are identically distributed. Moreover 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 the law for its score s_{agc} is :

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

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

Activated units benefits of the memory effect γ : $P(s_{abe} = \gamma + x) = \binom{a(c-1)}{x} d^x (1-d)^{a(c-1)-x}$

Finally $P(s_{dbe} = x) = P(s_{dbc} = x) = \binom{a(c-1)}{x} d^x (1-d)^{a(c-1)-x}$

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. $P_e(x) = \sum_{k=1}^{a} \binom{a}{k} P(s_{dge} = x)^k P(s_{dge} > x)^{a-k}$)

$$P_{retrieve} = \left[\sum_{x=1}^{a(c-1)+\gamma} P_{ac}(x) P(s_{dbc} < x)^{\ell - a} \right]^{c-c_e} \times \left[\sum_{x=1}^{a(c-1)+\gamma} P_e(x) P(s_{dbe} < x)^{\ell-2a} P(s_{abe} < x)^a \right]^{c_e} \quad (9)$$

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

We can of course compute the error rate : $P_{err} = 1 - P_{retrieve}$

2) Simulations: Figure 5 compares the two choices $\alpha = 1$ and $\alpha = a$. The choice of $\alpha = a$ is a considerable improvement over $\alpha = 1$ for retrieving messages from corrupt probe.

The analytical error rate is close to the empirical one, albeit slightly optimistic.

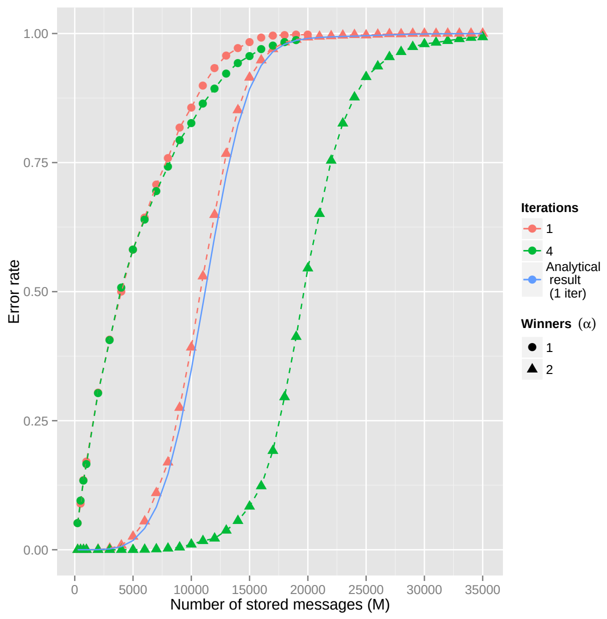

Figure 5. 2 activities per cluster, 4 clusters, 512 units per cluster, one erroneous cluster, each point is the mean of 10 networks with 1000 sampled messages, analytical result in continuous black

We now study the influence of the parameter a on retrieving messages from partial probes after the network has sustained damages.

C. Resilience

1) Analytical result: As information on a message is shared across edges, networks with higher values of a are more resilient to damages (as information on one message 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 take $\gamma = \infty$ (or big enough) for one iteration as it prevents activated neurons to be deactivated. Another good choice is $\gamma = a$ in practice.

With probability $1 - \psi$, a correct vertex is still connected to a given activated unit in an intact cluster. Thus $P(s_g = x) = \binom{a(c_k-1)}{x} (1-\psi)^x \psi^{a(c-c_e-1)-x}$

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 $P(s_b = x) = \binom{a(c-c_e)}{x} P_+^x (1-P_+)^{a(c-c_e)-x}$.

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 $P_g(x) = \sum_{k=1}^{a} \binom{a}{k} P(s_g = x)^k P(s_g > x)^{a-k}$

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

$$P_{retrieve} = \left(\sum_{x=1}^{a(c-c_e)} P_g(x) P(s_b < x)^{\ell-a} \right)^{c_e} \quad (10)$$

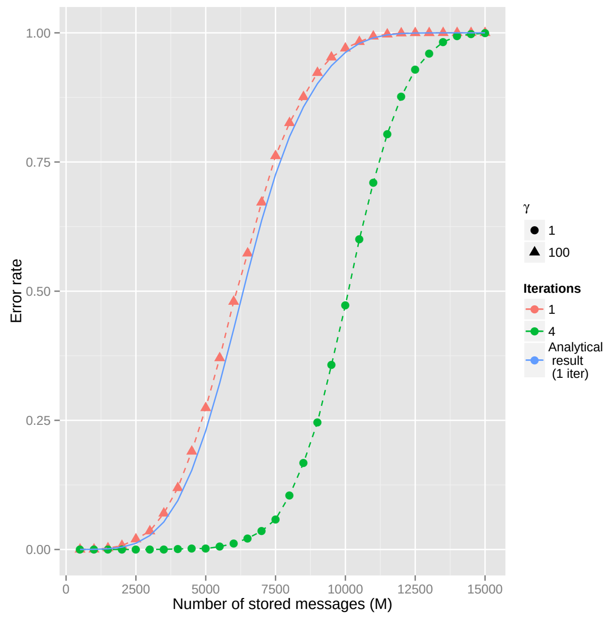

Figure 6. $\psi = 0.02$, 2 activities per cluster, 8 clusters, 256 units per cluster, 4 erasures, each point is the mean of 10 networks with 1000 sampled messages, analytical result in continuous black

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

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

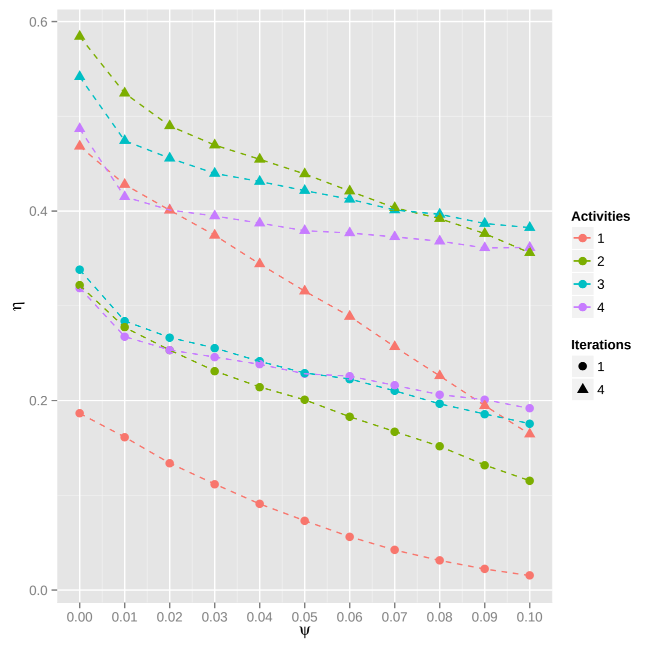

Figure 7. Comparison of evolutions of efficiency with increasing damages between different number of activities per cluster : 8 clusters, 256 units per cluster

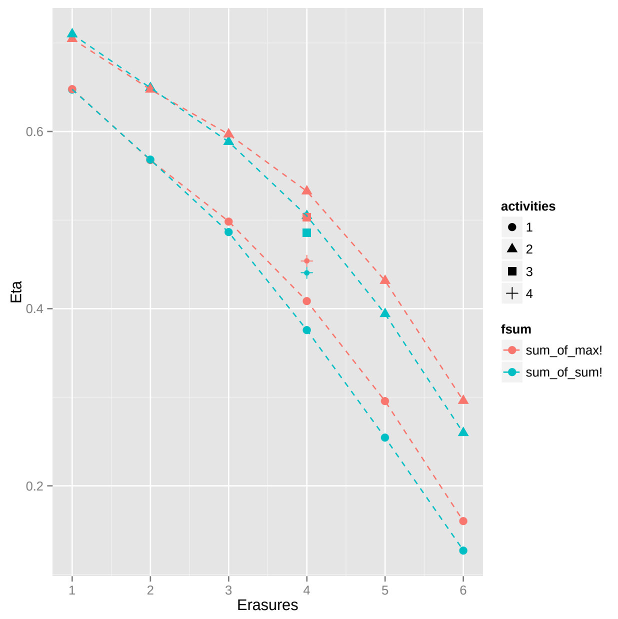

Figure 8. Comparison for different retrieval rules, 8 clusters, 256 units per cluster, 4 erasures, each point is the mean of 5 networks with 1000 sampled messages

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.

VI. Conclusion

Annex

Proof for asymtptotic efficiency

The entropy of the messages set can be approximated as $H(\mathcal{M}) \approx Mc\log_2\binom{\ell}{a}$. From $d = 1 - \left(1-(a/\ell)^2\right)^M$, we can deduce $M \sim \frac{\log_2(1-d)}{\log_2(1-(\frac{a}{\ell})^2)} \sim -\frac{\ell^2\log_2(1-d)\log 2}{a^2}$. We set $\log_2\binom{\ell}{a} = kc$ (otherwise density approaches 0 or 1, and efficiency 0). It follows that $\eta \sim \frac{Mc\log_2\binom{\ell}{a}}{\ell^2\binom{c}{2}} \sim \frac{kc^2\log_2(1-d)\log 2}{a^2\binom{c}{2}}$.

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

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 $P_{\exists e} \leqslant \binom{\ell}{a}^c d^{a^2\binom{c}{2}} = 2^{c\log_2\binom{l}{a}+a^2\binom{c}{2}\log_2(d)} \approx 2^{c^2(k+\frac{a^2}{2}\log_2(d))}$.

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

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

Adapting other rules

SUM-OF-MAX can be adapted to multipartie clique networks and still delivers improvements

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.