Storing Sequences in Binary Neural Networks with High Efficiency

Xiaoran Jiang

N° d'ordre : 2014telb0303

Sous le sceau de l'Université européenne de Bretagne

Télécom Bretagne

En accréditation conjointe avec l'Ecole Doctorale Sicma

Storing sequences in binary neural networks with high efficiency

---

Thèse de Doctorat

Mention : STIC (Sciences et technologies de l'information et de la communication)

Présentée par **Xiaoran Jiang**

Département : Electronique

Laboratoire : Lab-STICC Pôle : CACS

Directeur de thèse : Claude Berrou

Soutenue le 8 janvier 2014

Jury :

M. Olivier Michel, Professeur, Grenoble INP (Rapporteur)
M. Michael Rabbat, Professeur, McGill University (Rapporteur)
M. Olivier Faugeras, Directeur de Recherche, INRIA (Examinateur)
M. Pierre De Loor, Professeur, ENIB (Co-directeur de thèse)
M. Claude Berrou, Professeur, Télécom Bretagne (Co-directeur de thèse)
M. Olivier Rosec, Ingénieur, VOXYGEN (Invité)

Résumé

Nous proposons et développons un modèle de mémoires associatives permettant d'apprendre et de récupérer par un mécanisme d'anticipation, des séquences de longueur quelconque, scalaires ou vectorielles, dans des réseaux de neurones binaires. Ce modèle permet notamment de lever certains verrous bien connus dans la littérature tels que l'intolérance aux erreurs, l'oubli catastrophique lors de nouvelles acquisitions et le problème de l'interférence dans le stockage des séquences complexes, etc. La quantité d'information totale pouvant être stockée par ce modèle suit une loi quadratique vis-à-vis du nombre de neurones du réseau. Et l'efficacité d'apprentissage peut atteindre environ 30% en pratique.

La séquentialité est omniprésente dans la cognition humaine et dans le raisonnement. L'apprentissage de séquences temporelles dans les réseaux de neurones est un sujet de recherche important depuis de nombreuses années. Différentes approches ont été conduites. Certaines sont basées sur la dynamique des neurones impulsionnels [Bar03] [BSP11], d'autres s'intéressent aux modèles théoriques de neurones binaires, similaires au modèle très simple de McCulloch et Pitts [MP43], avec des entrées et des sorties binaires. Plusieurs modèles théoriques de l'apprentissage sont basés sur l'hypothèse très généralement admise de la corrélation Hebbienne, c'est-à-dire des changements dans les poids synaptiques liés à la corrélation des activations pré- et post-synaptiques. Parmi eux, les réseaux de Hopfield [Hop82], connus principalement pour leur capacité à coder et décoder des motifs statiques, peuvent être adaptés pour stocker des séquences temporelles d'une manière incrémentale. Par exemple, dans [MHB05], le modèle de Hopfield est étendu pour coder une série temporelle de motifs. Les séquences sont stockées dans une série de réseaux de Hopfield liés les uns aux autres à travers une matrice de coefficients de pondération. Cependant, le problème avec les réseaux de Hopfield en général est bien connu : seulement 0,15 N motifs peuvent être mémorisés avant que le taux d'erreur dans le rappel ne devienne trop important, N étant le nombre de neurones dans le réseau.

Cette lacune est due au phénomène connu sous le nom de "interférence catas-

trophique" [MC89] ou "oubli catastrophique" [Fre99]. En effet, dans de nombreux réseaux du type Hopfield, puisque l'information est exclusivement stockée dans les poids des connexions, l'information nouvellement acquise va modifier les poids des connexions anciennement établies. Par conséquent, il n'y a aucune garantie que l'aptitude du réseau à retrouver les messages sera conservée après un nouvel apprentissage. Un problème similaire existe également dans les modèles [HB00] [SHT09] basés sur des machines de Boltzmann [AH85], dans lequel les poids des connexions sont continus.

Contrairement aux modèles ci-dessus, des modèles de type Willshaw [WBLH69] [SP99] considèrent des connexions binaires au lieu de connexions pondérées. Le problème de l'interférence catastrophique est alors résolu : un nouveau motif stocké pourra chevaucher un autre motif plus ancien, mais un motif particulier en chevauche peu d'autres. Dans les réseaux de Willshaw, il est possible de stocker les séquences les plus élémentaires (celles ne contenant que deux motifs, un motif de déclenchement et un motif cible) dans un graphe biparti via des connexions hétéro-associatives. Un algorithme de décodage en une seule étape permet la récupération du motif cible à partir du motif de déclenchement.

Le travail de cette thèse fait suite à l'étude des réseaux de cliques neurales [GB11b] précédemment menée au sein de l'équipe d'accueil. Ces réseaux formés par des connexions binaires exploitent au plus haut degré les concepts de redondance et de parcimonie pour offrir une diversité d'apprentissage suivant une loi quadratique. Dans le Chapitre 2, nous rappelons les principes de ces réseaux. Les noeuds du réseau, appelés fanaux, sont répartis dans différents clusters, i.e. des parties disjointes du réseau. Pour le codage d'un élément d'information, nous combinons le code économe local, très facile à décoder, avec le code à cliques global, fort de sa distance de Hamming minimale. Ainsi, apprendre un message est équivalent à inscrire une clique, un sommet dans chaque cluster. D'une manière correspondante, le décodage est constitué de deux étapes: une étape de passage de message global qui exploite le redondance de la clique, suivie d'une étape locale de sélection du gagnant, matérialisée par le principe du Winner-take-all (WTA). La diversité d'apprentissage est proportionnelle au carré

du nombre l de fanaux par cluster et à la densité du réseau d : $M \approx l^2 d$. De plus, ce modèle répond à un certain degré de vraisemblance biologique, concernant par exemple les notions de codage parcimonieux, de WTA et de l'organisation du cerveau en colonnes. En effet, un fanal dans ce modèle peut être assimilé à une microcolonne (ou minicolonne), unité de traitement d'information la plus homogène dans le cerveau. De même, en suivant cette logique, nous associons un cluster à une colonne, et le réseau lui-même à une macrocolonne, plusieurs macrocolonnes constituant une aire fonctionnelle du cerveau.

Cependant, ce modèle affiche certaines limitations. Pour citer quelques uns : 1) Il ne sait mémoriser que des messages d'ordre fixe. 2) La totalité du réseau doit être impliquée dans chaque élément d'information acquis. 3) Il ne sait apprendre que des messages statiques. 4) Les synapses bidirectionnelles ne sont pas plausibles biologiquement. Les deux premiers verrous ont été traités dans [Ali13] [ABGJ13] pour une adaptation de ce réseau à des messages parcimonieux d'ordre variable c. Cette adaptation nous amène à une diversité encore meilleure, proportionnelle au carré du nombre total de fanaux n : $M_{sparse} \approx \frac{n^2 d}{c(c-1)}$.

Néanmoins, malgré cette performance théorique prometteuse, [ABGJ13] applique l'algorithme de décodage WTA Global (GWTA) qui ne profite pas pleinement de l'avantage du décodage itératif. En effet, GWTA est une règle peu tolérante qui ne sélectionne que les fanaux ayant le score maximal global. Un phénomène d'oscillation a été observé : au cours des itérations, le réseau oscille entre deux états, l'un avec très peu de fanaux activés et l'autre avec beaucoup de fanaux activés dont le nombre est supérieur à l'ordre de la clique recherchée. Afin de résoudre ce problème, nous proposons dans le Chapitre 2 deux nouvelles règles de décodage : Winners-take-all Global (GWsTA) et Losers-kicked-out (LsKO), toutes les deux utilisant le principe du "décodage soft". A chaque itération, GWsTA sélectionne les Σ premier fanaux dans un pile où les fanaux sont empilés dans un ordre décroissant selon leurs scores. Quant à LsKO, il élimine les fanaux ayant le score minimal à chaque itération. Par conséquent, l'ensemble de fanaux activés est un ensemble fini décroissant, donc la convergence de l'algorithme est garantie. En outre, l'algorithme complet de LsKO doit contenir deux

phases itératives de LsKO local et une seule itération de GWTA ou GWsTA entre les deux précédentes. A l'aide de simulations, nous observons que l'aptitude de GWsTA et LsKO à restaurer des messages distordus est très largement supérieure à celle de GWTA dans une structure parcimonieuse. Les courbes de performance de ces deux algorithmes sont très proches de celle du décodage à maximum de vraisemblance, ce dernier ayant pourtant une complexité exponentielle.

D'un point de vue informationnel, le mémoire cérébrale est robuste et durable, et donc nécessairement codée avec de la redondance. De nombreux travaux récents tels que [BS09] [LMZ+12] ont suggéré d'étudier la redondance de la connectivité des réseaux cérébraux complexes à l'aide de la théorie des graphes. Cependant, une redondance excessive fait diminuer la capacité de la mémoire et peut limiter la quantité d'information portée par le système neural. Dans le Chapitre 3, nous proposons quelques méthodes pour diminuer la redondance d'une clique en supprimant certaines connexions de manière intelligente. Nous constatons que la structure de chaîne de cliques est l'un des bons choix en raison de sa connectivité homogène. Par la suite, nous introduisons l'unidirectionnalité des connexions pour renforcer la plausibilité biologique du modèle. En effet, la propagation des stimuli via des synapses cérébrales est toujours orientée. Ceci nous conduit à proposer une structure appelée "chaîne de tournois" bouclée. Un tournoi est simplement obtenu en remplaçant chaque connexion bidirectionnelle par une connexion unidirectionnelle. Nous étudions par la suite la capacité d'une chaîne de tournois en tant que mémoire associative. Pour un taux d'erreur P_e donné, il est possible d'optimiser l'ordre r de chaque petit tournoi afin de maximiser l'efficacité informationnelle : $r_{\text{opt}} = \ln\left(\frac{l}{P_e}\right)$.

Le Chapitre 4 étudie l'aptitude de la chaîne de tournois bouclée à apprendre des séquences de symboles. Chaque sommet dans ce graphe de chaîne de tournois bouclée est matérialisé par un cluster. Lors de l'apprentissage, un symbole, matérialisé par un fanal dans un cluster, est connecté aux r symboles suivants localisés dans les r clusters qui suivent. La ressource neurale peut être réutilisée le cas échéant à partir du deuxième passage de la séquence dans un même cluster. Ceci permet au réseau d'apprendre des séquences dont la longueur est quelconque, indépendante du nombre

de clusters, et uniquement limitée par la ressource totale disponible du réseau. Grâce à la redondance de la chaîne de tournois, (en effet, r peut être vu comme un paramètre de chevauchement temporel.) le processus de restitution, aidé par un mécanisme d'anticipation, est tolérant aux erreurs transitoires. En pratique, pour un réseau de taille biologiquement plausible, i.e. l de l'ordre de 100, l'efficacité peut atteindre environ 30%.

Dans le Chapitre 5, nous proposons un cadre général pour le stockage de séquences de vecteurs ou de motifs parcimonieux. Un motif peut être matérialisé par un ensemble de fanaux, au plus un fanal par cluster. Deux structures de codage développées à partir d'une chaîne de tournois respectivement bouclée ou ouverte, sont proposées. Si la première s'appuie sur un ensemble de sous-réseaux disjoints, la seconde utilise la totalité du réseau. Pour la chaîne ouverte, la difficulté supplémentaire lors du décodage réside dans le manque d'information sur la localisation de chaque motif. Non seulement une règle globale de sélection du gagnant telle que GWTA ou GWsTA est absolument nécessaire, mais nous devons aussi imposer une restriction supplémentaire sur les activités des clusters lors de la construction de séquences. Plus précisément, si un cluster est activé par un motif, il lui sera interdit d'être activé à nouveau pendant les r prochain instants. Nous constatons aussi que le taux d'erreurs de récupération ainsi que la résistance aux erreurs sont étroitement liés au produit de l'ordre du motif c qui représente la diversité spatiale et du paramètre r qui représente la diversité temporelle.

Une structure séquentielle interdit par sa nature les processus de décodage itératif. En effet, d'une manière générale, une séquence est codée par une chaîne de connexions dans laquelle le motif A est connecté au motif B, ce dernier se connectant alors au motif C, etc. Le problème de ce mécanisme de chaînage est simple à constater : une erreur dans le motif A activerait potentiellement des fanaux qui n'appartiennent pas au motif B, tandis qu'une partie de B pourrait ne pas être activée. B est alors transformé en une version corrompue B'. Par conséquent, la récupération du motif C sera encore plus difficile et plus vulnérable. Pour résoudre ce problème de l'avalanche des erreurs, nous proposons dans la suite du Chapitre 5 une structure à double couche

combinant des tournois et des cliques. La séquence est stockée dans une couche hétéro-associative de chaînes de tournois et en parallèle, les motifs qui composent la séquence sont stockés dans une autre couche, cette fois-ci auto-associative, sous la forme de cliques. La restitution s'effectue alors par des allers et retours entre ces deux couches. Une erreur survenant dans la couche des tournois pourra être éventuellement corrigée par ce décodage itératif, en l'occurrence grâce aux algorithmes GWsTA ou LsKO, dans la couche des cliques.

Nous traitons aussi dans le Chapitre 5 le problème des interférences pour des séquences complexes. Cette situation est d'autant plus susceptible de se produire que les séquences considérées sont construites sur la base d'un dictionnaire de faible nombre d'entrées. L'idée principale pour lever ce verrou est de transformer des séquences complexes en des séquences simples. Pour ce faire, nous proposons de fournir plusieurs versions possibles pour différentes instances d'un même motif. Plus précisément, quatre solutions sont proposées : 1) Segmenter le réseau en plusieurs sous-réseaux identiques. Cela permet à un motif d'avoir autant de représentations possibles que le nombre de sous-réseaux. 2) Établir un réseau supplémentaire spécialement dédié à une signature aléatoire. Différentes instances d'un même motif sont associées avec des signatures aléatoires, qui sont bien distinctes. 3) Élargir la taille des clusters de sorte que le nombre de fanaux susceptibles de représenter un motif, soit augmenté. A chaque instance de ce motif, un motif aléatoire est généré. 4) Construire des représentations différentes en sous-échantillonnant aléatoirement le motif d'origine. Contrairement aux trois autres, cette quatrième solution ne consomme pas de ressource supplémentaire. Il est aussi possible de combiner cette méthode de sous-échantillonnage avec la structure à double couche pour construire un réseau hiérarchique. Le taux d'échantillonnage optimal est sujet à un compromis entre de nombreux facteurs, lesquels sont validés par les simulations.

Nous concluons nos travaux dans le Chapitre 6. La principale contribution est de proposer et d'étudier un modèle de mémoires associatives permettant d'apprendre et de récupérer par un mécanisme d'anticipation, des séquences de longueur quelconque, dans des réseaux de neurones binaires. Les considérations de plausibilité biologique

sont présentes tout au long de ces travaux.

Les modèle que nous avons mis en œuvre dans notre thèse ouvre de nombreuses perspectives. Ainsi, d'une manière générale, la notion du temps est incorporée dans un message temporel de deux façons: l'ordre temporel et la durée. Seul le premier aspect a été pris en compte dans notre travail. Il serait donc intéressant d'introduire la notion de durée dans les mémoires associatives, ce qui aurait une utilité pour des applications telles que le traitement de séquences de phonèmes en langage naturel, la robotique, etc.

Notre modèle possède encore quelques degrés de liberté vis-à-vis de plusieurs paramètres du réseau, en particulier le seuil σ dont nous pourrions faire usage dans l'avenir. Nous pourrions ainsi imaginer la mise en œuvre d'un seuil non uniformément distribué sur l'ensemble du réseau. Les clusters avec des seuils plus élevés seraient donc pénalisés par rapport à ceux qui ont les seuils les plus faibles. De même, le seuil pourrait aussi varier dans le temps et serait alors capable de guider la séquence vers des endroits appropriés du réseau. Ceci serait aussi utile pour réaliser des associations ou discriminations entre les séquences, car le seuil pourrait être codé de telle manière à autoriser des commutations entre les séquences. N'est-il pas possible d'imaginer que l'intelligence reposerait, au moins partiellement, sur la façon dont ce genre de commutation ou d'association est effectué dans le cerveau ?

Enfin, la structure hiérarchique à deux niveaux pourrait être étendue à une structure multicouche telle que [SH07]. En particulier, nous pourrions imaginer un réseau hiérarchisé où chaque niveau passe une partie de l'information sous-échantillonnée au niveau hiérarchique immédiatement supérieur, lequel agrègerait ensuite ces informations sous forme de cliques ou de chaînes de tournois d'un plus haut degré d'abstraction.

Abstract

Sequential structure imposed by the forward linear progression of time is omnipresent in all cognitive behaviors. This thesis proposes a novel model to store sequences of any length, scalar or vectorial, in binary neural networks. Particularly, the model that we introduce allows resolving some well known problems in sequential learning, such as error intolerance, catastrophic forgetting and the interference issue while storing complex sequences, etc. The quantity of the total sequential information that the network is able to store grows quadratically with the number of nodes. And the efficiency - the ratio between the capacity and the total amount of information consumed by the storage device - can reach around 30%.

This work could be considered as an extension of the non oriented clique-based neural networks previously proposed and studied within our team. Those networks composed of binary neurons and binary connections utilize the concept of graph redundancy and sparsity in order to acquire a quadratic learning diversity.

To obtain the ability to store sequences, connections are provided with orientation to form a tournament-based neural network. This is more natural biologically speaking, since communication between neurons is unidirectional, from axons to synapses. Any component of the network, a cluster or a node, can be revisited several times within a sequence or by multiple sequences. This allows the network to store sequences of any length, independent of the number of clusters and only limited by the total available resource of the network.

Moreover, in order to allow error correction and provide robustness to the network, both spatial assembly redundancy and sequential redundancy, with or without anticipation, may be combined to offer a large amount of redundancy in the activation of a node. Subsequently, a double layered structure is introduced with the purpose of accurate retrieval. The lower layer of tournament-based hetero-associative network stores sequential oriented associations between patterns. An upper auto-associative layer of mirror nodes is superposed to emphasize the co-occurrence of the elements belonging to the same pattern, in the form of a clique. This model is then extended

to a hierarchical structure, which helps resolve the interference issue while storing complex sequences.

This thesis also contributes in proposing and assessing new decoding rules with respect to sparse messages in order to fully benefit from the theoretical quadratic law of the learning diversity. Besides the performance aspect, the biological plausibility is also a constant concern during this thesis work.

Acknowledgments

First of all, I would like to express my gratitudes to my thesis advisor Claude Berrou for his continuous help and support. It was really an honour having the opportunity to work with him during these three years.

I thank all my colleagues in the Neucod project with whom I interacted, Ala Aboudib, Behrooz Kamary Aliabadi, Benoît Larras, Deok-Hee Kim-Dufor, Dominique Pastor, Ehsan Sdegh Gooya and Philippe Tigréat, etc. I thank especially Vincent Gripon and Olivier Dufor who have given me many suggestions and helps during the long of my thesis.

Many thanks to the reviewers - Olivier Michel and Michael Rabbat -, members of the jury - Olivier Faugeras, Pierre De Loor, Claude Berrou and Olivier Rosec - and invited guests of my thesis defense for showing interest in our work.

Finally, I would like to thank my parents and my wife for their support and inspiration throughout my life.

Contents

Chapter 1

Introduction

1.1 Biological neural networks

Before representing the nervous system as a whole, which includes the brain, spinal cord, and peripheral ganglia, let us start with the core components of the nervous system, the neurons. A neuron is a special type of cell that processes and transmits some kind of information through electrochemical signals. A neuron is composed of a cell body or *soma* that contains all the structural and metabolic machinery necessary to the life of the neuron, and the specialized branches, i.e. one axon and many dendrites. Dendrites are usually thick projections with many branches receiving electrochemical stimulation from other neurons at contacts called synapses. Being located after the synapses, with respect to the direction of signal transmission, dendrites are said to be postsynaptic. The axon is another type of projection that conducts electrical impulses away from the neuron's cell body. The axon is said to be presynaptic. Impulses coming from neurons will either favour or inhibit the excitation of other ones. If the sum of excitatory and inhibitory signals surpasses some threshold, the neurons receiving impulses are likely to fire.

Neurons can connect to each other via synapses to form neural networks. The human brain has a huge number of synapses. Within the liter and a half of the human brain, there are approximately 20 billion neocortical neurons, with an average of 7,000 synaptic connections each. The cerebral cortex has about 0.15 quadrillion

synapses, that is, about a trillion synapses per cubic centimeter of cortex [Dra05]. Despite the gigantic number of components in the brain, it is estimated that each neuron is able to contact any other neuron with no more than six interneuronal connections –"six degrees of separation", i.e. the small-world network phenomenon [SvS12] [BS09] [BB06].

Currently, it is generally assumed that neural networks are structured in a hierarchical manner. At the lowest level of hierarchy, about $80-120$ neurons form a cortical microcolumn (also called the minicolumn), which is a vertical cylinder through the cortical layers of the brain. The diameter of a microcolumn is about $28-40\mu$m. There are about 2×10^8 microcolumns in the human neocortex [JL07]. Neurons within a microcolumn encode similar features. From the point of view of informational organization, the microcolumn is likely the most basic and consistent information processing unit in the brain, instead of a single neuron [Jon00] [JL07] [CBP+05]. There are about 50 to 100 cortical microcolumns to form a macrocolumn (also called hypercolumn). Then, several macrocolumns are grouped together to constitute the functional areas of the brain.

1.1.1 Functions of a neuron

Despite its complex electro-chemical nature, the operations that a neuron performs locally can be summarized as a simple integration and a comparison to some threshold. When a given target neuron receives impulses (also called spikes) from multiple neurons, those impulses can be spatially integrated if they arrive closely enough in time before the influence of each has decayed. If a target neuron receives an impulse from a single axon and if it occurs repeatedly at short intervals, these impulses will be integrated temporally.

A very important rule found in the literature on biological neural networks is the *winner-take-all* (WTA) rule [LIKB99], [CGL92] . This corresponds to a selection of the neuron (or the small set of neurons) that achieves maximum activity in a given region of the neural network. Those that have been selected remain active, whereas others return to a silent mode. The WTA mechanism is necessary for the brain to

be at the same time energy-efficient and well-functioning, which prevents too many neurons from activating at the same time. For example, it is believed that some diseases such as epilepsy are caused by abnormal, excessive neuronal activity in the brain [FvEBB+05].

All these notions: minicolumns, signal integration, thresholding, and WTA will be very important in the sequel of this document. Obviously, we will not attempt to investigate deeply into electro-chemical stimuli exchanged between neurons, the whole mechanism still being partially unknown nowadays. What we are interested in and what we consider here is its macroscopic behavior. All these operations, i.e. summation, thresholding and WTA, even though represented here in an extremely simplified version, are curiously of digital nature. These last two operations (thresholding and WTA) remind us more directly of digital information and more specifically binary information (i.e., bits of information that take either the value 0 or 1). Indeed, no matter the continuous activity levels a neuron can represent, any neuron with an activity lower than some threshold (in the case of WTA, that is the maximum activity in a certain region) would be inhibited. This observation has led some authors to an audacious speculation [BGD+13] : is memory in the brain finally digital?

1.2 Artificial neural networks

Artificial neural networks (ANN) are justified as abstractions of the architecture of biological neural systems. A mathematical neural network can be generally represented as a graph, composed of a number of vertices, which are computing units assumed to mimic neurons at the informational level, and connections which represent synapses. In the connectionist approach [FB82] [HB90], information stored in the network is borne by a group of computational units, that is, related to a pattern of activity.

1.2.1 McCulloch-Pitts neuron

Here, we focus on the McCulloch-Pitts neuron [MP43], which is an abstract neuron model which combines neurophysiology and mathematical logic, using the all-or-none

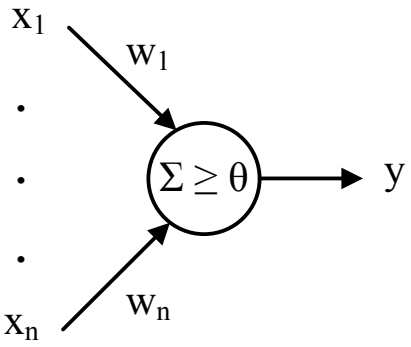

Figure 1-1: A McCulloch-Pitts neuron operating on a discrete-time scale. Each input x_i is associated with a weight w_i, and the neuron activity is conditioned by a threshold value θ. The output function is then a Heaviside step function shifted by θ units with the sum of weighted input values as the abscissa.

property of neuron firing to model the neuron as a binary discrete-time element. The basic idea is to divide time into units comparable to refractory periods so that, in each time period, at most one impulse can be generated in the axon of a given neuron. Thus, the McCulloch-Pitts neuron operates on a discrete time scale where, biologically speaking, the time unit is on the order of a millisecond. Denoting by $x(t)$ one synaptic signal at time t, we have $x(t) = 1$ or 0 according to the presence of an active input or not. Each connection, or synapse, from the output of a neuron to the input of another one is weighted. The presence of a weight $w_i > 0$ means that the connection is excitatory, and $w_i < 0$ means that the connection is inhibitory. Each neuron is associated with a *threshold* value θ. A neuron will be fired at time $t + 1$ if and only if the sum of its weighted input values at time t is at least equal to θ. Formally, if at time t, the input values are $x_i(t)$, the weights of corresponding connections are w_i, and the output at time $t + 1$ is $y(t + 1)$, then we have

$$y(t+1) = 1 \Leftrightarrow \sum_i w_i x_i(t) \geq \theta. \tag{1.1}$$

Even though many artificial neural network models are based on the McCulloch-Pitts neuron, today we know there are a number of fundamental differences between computations in such artificial neural networks and those in real biological neurons.

As a matter of fact, the output of a biological neuron consists of sharp potential impulses or spikes, which last for $1-2$ ms. And unlike in artificial neural networks functioning in a synchronous mode, there is no central clock in a biological neural circuit. The *Spiking neuron* model is supposed to be biologically more realistic, the current state of the art of which can be found in [GMM01].

1.2.2 Association in neural networks

The operation of *association* involves the linkage of two or more pieces of information. An associative memory takes some "keys" as inputs and returns some "associated pattern" as the output. For example, given a misspelled word, we may wish to recall the correct spelling. Given the name of a football team, we may wish to recall the score of its last match. The first example involves *auto-associative* memory, which is able to retrieve a piece of information upon presentation of only partial or degraded data from this same piece of information. On the other hand, the second example involves the use of *hetero-associative* memory, able to recall an associated piece of information from one category exploiting information from another category. The previously state-of-the-art Hopfield neural networks (HNN) [Hop82] have been designed to act as an auto-associative memory, since they are able to retrieve information given only some partial clues.

There are two major architectures implementing neural associative memories. In $non-recurrent$ networks, there are no loops in the associated graph. In other words, starting from a neuron, there is no path to return back to it. The input pattern is fixed steady in the network and the output pattern will not interfere with the input because of this absence of loops. On the other hand, in *recurrent* networks, the input pattern is only used to give the initial state of the network. The subsequent retrieval process is then that of a looped non-linear dynamical system, and the output pattern is read out until a certain stopping condition is satisfied, for example, when an equilibrium state is reached.

The *data representation* of input and output patterns is of critical importance in two aspects: on the one side, the representation should correspond to information

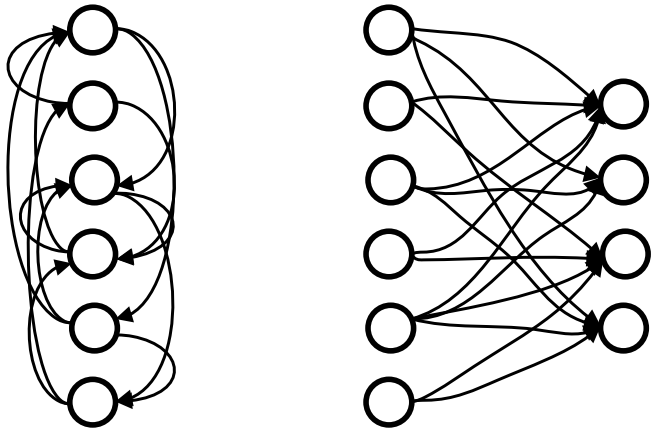

Figure 1-2: Two different approaches of associations in neural networks. The figure on the left represents the auto-associative approach: a set of units is recurrently self-connected. The figure on the right represents a feedforward network with the hetero-associative approach: the input and output patterns are represented by separated sets of units, and the input pattern triggers the output without feedback.

about the real world; on the other side, this representation must guarantee the network's retrieval performance. Two opposite approaches are worth mentioning. On one hand, a knowledge element can be materialized by a highly localized element, called the *grandmother cell* [Bar72] [Gro02] representation. In this approach, a single neuron would respond to a specific object. On the other hand, the principle of *distributed representation* [RHM86] states that a specific stimulus is encoded by its unique pattern of activity over a group of neurons. The problem with the grandmother cell theory is the explosion of material if one would represent a very large number of knowledge elements. For example, in a face recognition task, one would need thousands of neurons for each face, since any given face should be recognized in any angle, e.g. profile, 3/4 view, full frontal, from above, etc., and with various facial expressions, e.g. smile, grimace, cries, etc. In this document, for both biological and performance-wise reasons, we adopt a sparse distributed representation, in which each concept or object is represented by a small number of active units. However, the degree of distributed encoding and of sparsity is still a matter for discussion.

1.2.3 Sequences in neural networks

Forward linear progression of time is a fundamental property of human intelligence. For example, to make a cake, we should follow operations in a given order: we begin with measuring the flour, sugar and baking powder, then we beat the egg yolk with the sugar, then mix these ingredients by adding the butter and finally the flour, egg white and baking powder. This order should be respected, otherwise the taste will not be as expected. Even when it is not necessary to remember the exact order of actions, the brain would probably exploit temporal information in order to better reconstruct past events. Let us recall what we have done yesterday. Probably, we remember what kind of food we ate for breakfast, that we went to classes, the conversation that we had with friends during the coffee time after the lunch, and finally how we spent our evening, studying hard for today's examination, or watching a football match with our family. Obviously, it is not necessary to recall all these activities in the exact order, but when recalling them, we exploit without doubt the temporal indices to obtain an easier organization.

For all the above reasons, learning temporal sequences in neural networks has been an important research topic for many years. Different approaches have been carried on. Some are based on the dynamics of spiking neurons [Bar03] [BSP11], others interest in theoretical models of binary neurons, similar to the simple McCulloch-Pitts model with binary inputs and outputs.

Many theoretical models of learning are based on correlation Hebbian assumptions, that is, changes in synaptic efficacy are related to correlations of pre- and post-synaptic firing. Among them, Hopfield networks [Hop82], known predominantly for their ability to encode static patterns, could be adapted to store temporal sequences in an incremental manner. For example, in [MHB05], the Hopfield model is extended to encode a time series of patterns. The sequences are stored in a series of Hopfield networks linked to one another through the matrix of weights. However, the problem with Hopfield networks in general is well known: only "about $0.15N$ states can be simultaneously remembered before error in recall is severe", if N is the number

of units in the network. This is due to "catastrophic interference" (CI) [MC89] or "catastrophic forgetting" (CF) [Fre99]. Indeed, in many Hopfield-like networks, the storage of new information completely disrupts or even eliminates that previously learnt by the network. Since the learning process relies on changing the connection weights, there is no guarantee that the ability of the network to recall messages will remain still when learning new ones. A similar issue exists also in the models [HB00] [SHT09] based on Boltzmann machines [AH85], in which the connection weights are continuous.

In contrast to the above models, Willshaw type models [WBLH69] [SP99] consider binary connections instead of weighted ones. CI is well resolved: a newly stored pattern could overlap an older one, but the latter will not be eliminated. It is possible to store the most elementary sequences (those containing only two patterns, a cue pattern and a target pattern) in a bipartite graph. A sparse pattern a represented in the memory x is associated to another sparse pattern b in the memory y, via hetero-associative connections. A one-step retrieval algorithm enables the retrieval of b provided a.

A recently proposed neural network based on cliques and sparsity properties [GB11b] offers a novel vision of the Hebbian rule: instead of strengthening the weight of a frequently active connection, the weight of a connection belonging to several cliques remains equal to 1. A lost connection would be retrieved or repaired by the co-activation of other parts of the cliques it belongs to. This model demonstrates large storage diversity (the number of storable messages with a relatively high reliable recovering) and capacity (the amount of storable binary information), as well as strong robustness with respect to erasures and errors. However, the non-oriented connections are not relevant biologically. It is then of high interest to replace them with oriented ones. Orientation of connections would naturally offer the network with the ability to exhibit sequential behavior. One may expect the novel structure to inherit good proprieties of clique-based networks to solve some well-known issues such as CI in sequence learning.

1.3 Thesis structure

The structure of this document is depicted in the following list.

- **Chapter 2** recalls the principle of clique-based neural networks to store and retrieve maximum-length messages as well as their adaptation to sparse messages. New decoding rules are introduced in order to fully benefit from the theoretical quadratic law of the learning diversity with respect to sparse messages. Their performance is assessed, in particular, retrieval capability, complexity, robustness, etc. Some extensions of the use of associative memory to this model are also discussed.

- **Chapter 3** studies some ways to smartly reduce redundancy in the structure of cliques in order to maximize information efficiency. Different strategies are discussed. Performance is assessed both theoretically and by simulation using this model as an associative memory given only partial information. Parameters are optimized.

- **Chapter 4** introduces unidirectionality into the structure discussed in Chapter 3 for the concern of plausibility. This modification enables the model to store sequential messages, the length of which is variable and only limited by the available neural resources. Some rules of storage and retrieval are proposed. Parameters are optimized with respect to the efficiency.

- **Chapter 5** proposes a general framework to store vectorial sequences. Single-layered structures with looped or unlooped chains of sparse distributed patterns are proposed and evaluated. Subsequently, a double layered structure combining a hetero-associative network with the auto-associative clique-based network is introduced with the purpose of accurate retrieval. This model is then extended to a hierarchical structure, which helps resolve the interference issue with complex sequences. Different strategies in our model to get around this interference issue are also discussed.

Chapter 2

Network of neural cliques

2.1 Structure

Let us recall the key points of the GB networks, a type of clique-based neural networks introduced in [GB11b]. Such a network is composed of n binary nodes organized in χ clusters. For the reason of simplicity, we consider clusters of same size $l = n/\chi$ each, but this is not a necessity. Though any alphabet with cardinality l could be considered in the representation of information stored by the network, we focus on binary messages in order to allow classic computations or estimations of storage properties. Therefore, l is taken to be a power of 2: $l = 2^\kappa$. We denote by n_{ij} the j^{th} node in the i^{th} cluster, the value of which $v(n_{ij})$ is respectively one or zero, activated or not. This node can be also simply noted in the form of the pair (i, j). The binary edge weight between nodes n_{ij} and $n_{i'j'}$ is noted as $w_{(ij)(i'j')}$. In our vocabulary, the nodes are called "fanals" [1], since during the storage and recovery process, in normal conditions, only one of them in the same cluster can be activated at the same time.

[1] Definition of the word "fanal" according to Oxford Dictionaries Online: "a lighthouse or beacon for guiding ships." It is from the Italian word "finale". A fanal is often the only source light we can see during the navigation.

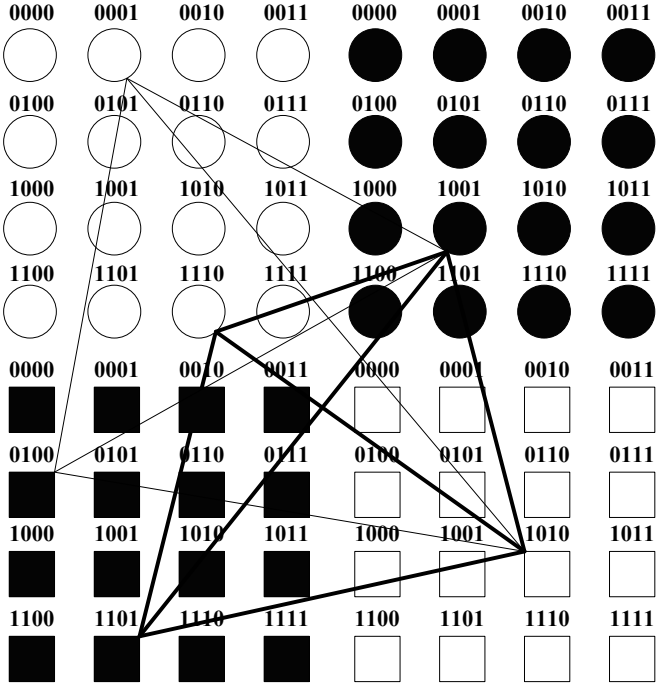

Figure 2-1: Storage process illustration for non-oriented clique-based networks. The pattern to store (with thick edges) connects fanals from four clusters composed of 16 fanals each (filled circles, filled rectangles, rectangles and circles).

2.1.1 Storage

Let $\mathfrak{M}_k$ be the set of binary messages of fixed length k bits to be stored by the network. For each message $m \in \mathfrak{M}_k$, we split it into χ sub-messages of length $\kappa = \frac{k}{\chi}$: $m = m^1 m^2 ... m^\chi$. Each sub-message m^i is mapped to a unique fanal in the cluster i. Let us denote this mapping in the cluster i by $f_i : \{-1;1\}^\kappa \to \{n_{ij} \mid j \in [|1;l|]\}$. As a consequence, storing a binary message of k bits is equivalent to connecting the corresponding pattern of χ fanals. This pattern is then represented by a clique, that is, a set of fanals such that each one is connected to the others. It has been proven in [GB11a] that cliques are codewords of a good error correcting code. The binary edge weight $w_{(ij)(i'j')}$ represents whether a connection exists or not. As a result, the weight of an existing connection remains unchanged even if the same pair of fanals appears in two or more messages.

An example is represented in Fig. 2-1, in which there are $\chi = 4$ clusters (filled

circles, filled rectangles, rectangles and circles) of $l = 16$ fanals each. Two binary messages of 16 bits have been stored. For instance, the message $m = 1110100111011010$ is split into 4 sub-messages, $m^1 = 1110, m^2 = 1001, m^3 = 1101, m^4 = 1010$. Each sub-message is then mapped to a unique fanal in the corresponding cluster ($f_1(m^1) = n_{1,15}, f_2(m^2) = n_{2,10}, f_3(m^3) = n_{3,14}$ and $f_4(m^4) = n_{4,11}$), and then these fanals are fully interconnected to build a clique. The connection weight is not incremental: though these two messages share a pair of sub-messages: $m^2 = 1001$ and $m^4 = 1010$, the weight of the corresponding edge remains one.

Formally, if we denote $W(m_i)$ the connection set of the corresponding clique after storing the message m_i, the connection set of the associated graph after storing $(m_1, m_2, ..., m_N)$ can therefore be defined by the union:

$$W(m_1, m_2, ..., m_N) = \bigcup_{i=1}^{N} W(m_i). \tag{2.1}$$

2.1.2 Network density

We define the density of the network as the ratio between the number of established connections via the storage process and that of all possible ones. In order to estimate the network density, some important hypotheses should be assumed.

Let us suppose that the stored messages are randomly generated. They are independently and uniformly distributed (i.i.d.) messages chosen from the possible ones. In the network described above, a possible message is composed of χ fanals, each of them being randomly and independently chosen in its corresponding cluster. The number of possible messages is thus l^χ. Furthermore, we assume that the connections are independent in the graph in order to apply the binomial distribution law. Strictly speaking, this hypothesis is not entirely true, since connections of a clique that belong to the same message are obviously dependent on each other. One connection of a clique can be deduced by the existence of others. However, this approximation can be interpreted intuitively by the fact that with a large number of stored messages, two connections chosen at random in this graph are unlikely to have been added

simultaneously.

The connections can then be seen as independent random variables with Bernoulli distribution of parameter d. The value of d for a connection can be obtained by binomial arguments. Indeed, the probability of the existence of a connection between two vertices (i,j) and (i',j') in the graph, with $i \neq i'$, is by construction the probability of having at least a stored message containing (i,j) and (i',j'):

$$\begin{aligned} d &= p\left(\exists m \in \mathfrak{M}, m^i = (i,j) \wedge m^{i'} = (i',j')\right) \\ &= 1 - p\left(\forall m \in \mathfrak{M}, m^i \neq (i,j) \vee m^{i'} \neq (i',j')\right) \\ &= 1 - p\left(m^i \neq (i,j) \vee m^{i'} \neq (i',j')\right)^M \\ &= 1 - \left\{1 - p\left(m^i = (i,j) \wedge m^{i'} = (i',j')\right)\right\}^M \\ &= 1 - \left\{1 - p\left(m^i = (i,j)\right) p\left(m^{i'} = (i',j')\right)\right\}^M \\ &= 1 - \left(1 - \frac{1}{l^2}\right)^M \quad . \end{aligned}$$

For the subsequent development in this document, if not mentioned in particular, we assume that the messages stored in the network are i.i.d. and connections are independent variables with Bernoulli distribution of parameter d. Some strategies for storing non-uniformly distributed messages in clique-based networks have been studied in [BGSH13]. Even though storing non-uniformly distributed messages is not the main interest of this document, these strategies can easily be adapted to the networks described in Chapter 3, 4 and 5.

2.1.3 Retrieval

At the retrieval stage, the network is provided with a message $\hat{m}$, possibly a degraded version of one of the stored messages m. Several forms of degradations are envisaged. We call *erasure* the situation when a cluster is not provided with information, in other words none of the fanals in this cluster are activated. An *error* is when the fanal activated in a cluster does not correspond to the right one. Based on the set

of existing connections, an appropriate fault tolerant decoding algorithm should be able to give an estimate $\tilde{m}$, which is the nearest message to m in terms of Hamming distance.

The decoding procedure consists of two steps: a *message passing* step at the global scale; and a *selection of the winner* step at the local scale. The former exchanges stimuli (that is, some of the sub-messages m^i) within the whole network via established connections, and the contributions are added at each fanal. The latter makes a local decision within a given cluster following the winner-take-all principle. The fanals that have the largest weight are selected, provided that this weight is superior to a predefined threshold value σ. All other fanals are reset to zero. Note that several fanals may be allowed to be activated at the same time in the same cluster, resulting in a soft local decoding. This decoding procedure is possibly iterative, as the winners after the local selection step become the new stimulus for the global message passing. Formally, the equation of global message passing is expressed as:

$$\forall i,j,v(n_{ij}) \leftarrow \sum_{i'=1}^{\chi} \max_{1\leq j'\leq l} \left[w_{(i,j)(i',j')} v(n_{i'j'}) \right] + \gamma v(n_{ij}) \tag{2.2}$$

where γ is the parameter of a memory effect which assures that a stored message will be recognized by the network. However, the value of γ should not be too high to allow error correction. Equation (2.2) is an enhanced version of the decoding algorithm compared to that in [GB11b]. In fact, the algorithm described in [GB11b] suffers from a minor drawback: a cluster with ambiguities - that is, with several fanals activated - has potentially a more important impact on the decoding in the other clusters than that of an unambiguous cluster. Equation (2.2) eliminates this accumulated contribution due to ambiguities in the same cluster [GB12]. Applied to associative memory, this gain is much more noticeable when a large proportion of erasures or errors is present at the input of the decoder.

The previous algorithm is then followed by a local selection of the winner in each cluster i, which is expressed as:

$$\forall j \in [|1, l|], v(n_{ij}) \leftarrow \begin{cases} 1 & \text{if } v(n_{ij}) = v_i^{\max} \text{ and } v_i^{\max} \geq \sigma \\ 0 & \text{otherwise} \end{cases} \tag{2.3}$$

where $v_i^{\max}$ is the largest score in the cluster i after the message passing step. After these operations, all fanals have value 1 or 0. This is quite comparable to the neurons of the very simple McCulloch-Pitts model [MP43] with binary inputs and outputs, in which the only functionality of a neuron is the summation of inputs and the comparison with some firing threshold.

A more generalized framework of decoding algorithms will be studied in Section 2.2.

2.1.4 Measures of performance

In this document, there are mainly three measures of performance. The first one is the *diversity*, that is to say, the number of units of information (for example, messages) a device can store in the case that a certain retrieval error rate is guaranteed. The second one is the *capacity*, the total amount of information a device can store, in bits. The last one is the *efficiency*, the ratio between the capacity and the total amount of information consumed by the storage device. As far as cognitive issues are concerned, the diversity of a network seems to be a more important parameter than the capacity. Indeed, as said in [Gri11], it is better to be able to store a lot of short messages than to store a small amount of long messages as it is the ability to confront many pieces of information which is the very foundation of human thinking.

2.1.5 Sparsity

This network organization represents three levels of sparsity. The first level is the sparsity of messages. Compared to Hopfield networks, this network stores "short" messages, the length of which is significantly smaller than the size of the network (that is, the number of neurons). As long as a message is represented by a spatial pattern in the graph, this is simply the consequence of the second level of sparsity,

imposed by the local winner-take-all rule: only one fanal per cluster is eligible to carry on information for a message at a precise time. This could be slightly extended to some more tolerant rules, but the number of firing fanals should remain very small before the total number of fanals. This finds a direct correspondence to precisely what biologists call sparse coding in the brain [FÖ3] [PB04] [OF04]. They present it as the fact that only a few neurons are firing at a specific time. Finally, the third level of sparsity is given by the network connections, which could be represented by a sparse connection matrix.

All the above could be enhanced by a supplementary level of sparsity, which is exterior to the network. In fact, in terms of knowledge perception, the brain could be considered as a selective filter. Among the very rich information received by the brain at one moment, only a small part of it will be considered relevant and possibly exploited later.

It is biologically relevant to evoke sparsity at several levels of our network organization. Actually, sparsity has a direct analogy to energy minimization which is often a good explanation of the life strategies.

2.1.6 Biological and information considerations

More and more researchers have adopted the idea that memories are stored in the brain using a *distributed representation* across numerous cortical and subcortical regions [EC01] [Fus09], which means that each memory is stored across thousands of synapses and neurons. Also each neuron or synapse is involved in thousands of memories.

The cortical minicolumn (also called microcolumn) is likely the most basic and consistent information processing unit in the brain [Jon00] [JL07] [CBP+05]. The neurons within a minicolumn encode similar features, as said in [CBP+05]: "current data on the microcolumn indicate that the neurons within the microcolumn receive common inputs, have common outputs, are interconnected, and may well constitute a fundamental computational unit of the cerebral cortex". As a consequence, a node in our network, the unit being able to send and receive information towards and from

the rest of the network, may correspond to a minicolumn, each comprising around 80 neurons in reality. To make it clear, we deliberately call a node as a "fanal" instead of a "neuron".

Following this logic, we propose to liken clusters in the clique-based networks to the cortical column, generally composed of 50 to 100 minicolumns. The columns are then grouped into macrocolumns, in the analogical way that clusters compose the network. A set of clique-based networks that cooperate with each other would likely correspond to the so-called functional areas of the brain.

The general concept of the brain as an information processing machine underlines the importance of information theory in cognition problems. Many research efforts [Mil56] [Bad94] [Cow00] point out that the *chunk* of information, an abstract but nevertheless measurable unit of information, is relevant to several human cognitive features, such as the span of immediate memory, rather than the absolute amount of information, that is, the *bit*. In the clique-based network, the division of information into clusters is in fact a way for *chunking*. A sparse message may be composed of a number of chunks, which is much smaller than the number of fanals in the whole network. The subsampling of chunks could be furthermore applied to resonate with the experienced limitations on the number of information items that we are able to receive, process and remember. And a hierarchical structure could be envisaged to form the clusters of clusters, as well as the chunks of chunks.

2.1.7 Advantages and drawbacks

This network model based on neural cliques has several main advantages:

1. Quadratic performance: the quantity of information that this model is able to store and retrieve outperforms dramatically the previous state-of-the-art Hopfield neural networks (HNN), or other comparable laws obtained with networks based on weighted connections. In particular, given a density d, the number M

of i.i.d. stored messages in such a clique-based network is expressed as:

$$M = \frac{\log(1-d)}{\log\left(1-\frac{1}{l^2}\right)} \approx l^2 d. \tag{2.4}$$

This approximation is close to the exact value for low values of d. This simple result shows the performance follows a quadratic law of the number of fanals per clusters, while that of HNN follows a sub linear law. For instance, [Gri11] shows that for the same amount of used memory of 1.8×10^6 bits, the clique-based network with $\chi = 8$ and $l = 256$ is 250 times superior to HNN in terms of diversity, 20 times superior in terms of capacity, and 20 times superior in terms of efficiency. This performance enhancement is achieved by exploiting the sparsity of the network at several levels.

2. Biological plausibility: This model echoes to some biological reality, such as sparse coding, winner-take-all rule and the columnar organization of the brain, etc. This has been explained more in details in Section 2.1.6.

The main interest of this model is that one does not need to find a trade-off between biological plausibility and retrieval performance. Both aspects can be plainly satisfied without any conflict. However, this model still has limitations:

1. The entire network is involved in the storing and retrieving of every piece of information. This does not correspond directly to the "sparse coding" vision of mental information [FÖ3] [PB04] [OF04]. It is contradictory to the biological reality that in the neocortex, the information received will only address a restricted part of the material, rather than the entire neocortex.

2. By conception, the network is only able to store messages with fixed length. This is a consequence of the first point.

3. The network is able to store only static spatial messages, while temporality and sequentiality is omnipresent in human cognitive behavior.

4. Symmetrical connection matrix, in other words, bidirectional synapse is not likely biologically plausible.

The first two points have been addressed in [Ali13] [ABGJ13] and will be re-explained in Section 2.2 with newly introduced decoding algorithms, whereas the last two points will be addressed in Chapter 4 and 5.

2.2 Adaptation to sparse structure

As already pointed out previously, parsimony is a prominent characteristic of the brain organization and functioning. The amount of data which is continually conveyed by the nervous system, the visual cortex for instance, is gigantic. And yet, what is retained by the brain for a possible later exploitation requires much less information than the original physical stimuli. These stimuli undertake several filtering operations, from the sensory levels (e.g. points, lines, shapes, colors) to the most cognitive ones (names, concepts, etc.), each level adapting the elements of knowledge to its own mode of representation and storage [Ben09].

Kamary Aliabadi [Ali13] describes how to adapt our network structure to sparse message storage and retrieval, where only a small part of the network is addressed at a time for information acquisition. An illustration is given in Fig. 2-2. Three sparse messages are stored in a network composed of 16 clusters of 8 fanals each. Messages are not necessarily of same order, two of them are of order 4 (on black and on blue), and the other one is of order 5 (on red). The local activation restriction is still respected: only one fanal at most can be active in a cluster. And once again, the weight of a connection shared by several cliques remains 1.

This further level of sparsity makes it possible to increase furthermore learning diversity. As studied in [ABGJ13], for a given density d, the number of i.i.d. sparse messages of order c stored in a network of χ clusters of l fanals each is expressed as:

$$M_{sparse} = \frac{\log(1-d)}{\log\left(1-\frac{c(c-1)}{\chi(\chi-1)l^2}\right)} \approx \frac{n^2 d}{c(c-1)}. \tag{2.5}$$

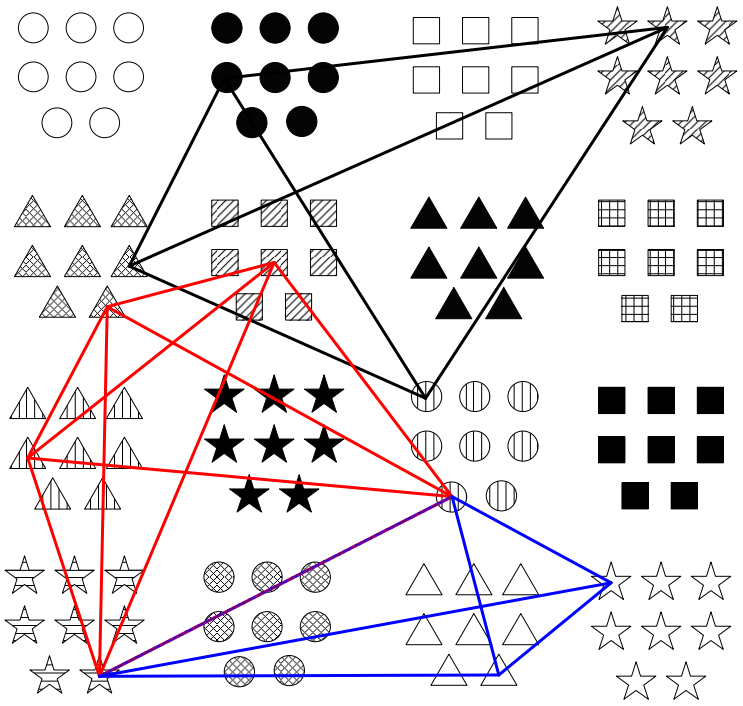

Figure 2-2: Storage of sparse messages in a network composed of 16 clusters of 8 fanals each. Cliques corresponding to three sparse messages: {(2,4),(4,2),(5,6),(11,1)}, {(5,8),(6,5),(9,4),(11,7),(13,8)}, {(11,7),(13,8),(15,8),(16,1)} are represented, two of them are of order 4, and the other one is of order 5. The clique on blue and that on red share the same edge.

Equation (2.5) shows that for a particular value of the clique order, the number of messages that the network is able to store becomes proportional to the square of the total number of fanals n instead of being proportional to the square of the number of fanals per cluster l (cf. (2.4)).

2.2.1 Generalized framework of decoding algorithms

Despite of promising theoretical performance, [ABGJ13] applies a Global WTA (GWTA) retrieval algorithm that does not fully take advantage of the sparse organization of the data structure from a practical point of view. In fact, GWTA is a simple adaptation of the local WTA algorithm used in the classical GB networks [GB11b] to the new sparse structure: instead of selecting the fanals with the local maximum score in each cluster, it selects those with the global maximum score through the whole network.

In order to benefit from algorithm's iterative feature, one should set a supplementary limitation and carry out a "guided decoding", i.e. the indices of the clusters addressed by a "to be retrieved" message should be known beforehand. In the opposite case, for a "blind decoding" where this information is not available for the decoder, the improvement of iterative algorithms is not observed in our simulations.

Here in this Section, we propose a generalized framework of decoding algorithms, which can be applied to the message retrieval process in both the classical GB networks and the sparse networks. In addition, this will be also applicable in sequence retrieval in tournament-based networks, which will be introduced later in this document. The retrieval performance of these newly proposed algorithms is very close to some algorithms based on brute-force search or exhaustive search principle, while still maintaining a certain degree of biological plausibility and relatively low complexity.

In general, the decoding process can be summarized as the following algorithm (algorithm 1). A similar algorithm has been depicted in [AGJ13].

begin
  Insert an input message $\hat{m}$;
  Initialize $it = 0$;
  Initialize $\widetilde{m} = \emptyset$;
  Suppose IT the predefined number of applied iterations. $IT = +\infty$ if this number is not specified;
  Suppose the algorithm stopping condition: $\hat{m} = \widetilde{m}$ or $it = IT$;
  while *the stopping condition is not satisfied* **do**
    $\hat{m} \longleftarrow \widetilde{m}$;
    Apply a dynamic rule to $\hat{m}$ and message passing;
    Apply an activation rule. $\widetilde{m}$ is updated;
    $it = it + 1$;
  end
  Return $\widetilde{m}$;
end

Algorithm 1: Generalized decoding process for recover the stored message m from an input degraded message $\hat{m}$.

2.2.2 Dynamic rules

The dynamic rule defines the initial dynamic level of each fanal, and the way how they contribute to other clusters during message passing.

Sum-of-Sum rule (SoS)

By this Sum-of-Sum rule, the dynamic level of each fanal contained in $\hat{m}$ is 1, all the rest being 0. These fanals then contribute equally to other clusters, disregarding the fact that some of them belong to the same cluster or not. This rule is used for the classical retrieval algorithms in GB networks in [Gri11] [GB11b].

$$\forall i,j,v(n_{ij}) \leftarrow \sum_{i'=1}^{\chi}\sum_{j'=1}^{l}\left[w_{(i,j)(i',j')}v(n_{i'j'})\right] + \gamma v(n_{ij}). \tag{2.6}$$

Normalization rule (NORM)

By this normalization rule, the dynamic level is normalized by the number of active fanals in each cluster. That is, if a cluster contains q active fanals, then the dynamic of each fanal viewed from outside of this cluster becomes $1/q$. These fanals then contribute equally to other clusters, disregarding the fact that some of them belong to the same cluster or not.

$$\forall i,j,v(n_{ij}) \leftarrow \sum_{i'=1}^{\chi}\frac{1}{|m_{i'}|}\sum_{j'=1}^{l}\left[w_{(i,j)(i',j')}v(n_{i'j'})\right] + \gamma v(n_{ij}). \tag{2.7}$$

Sum-of-Max rule (SoM)

By this Sum-of-Max rule, the contribution from a given cluster is limited by the activation of only one fanal. It has been shown in [GB12] that in classical GB networks, the SoM rule guarantees that the target message is always contained in the retrieved active pattern: $m \subseteq \tilde{m}$, although the equality is not always assured.

$$\forall i,j,v(n_{ij}) \leftarrow \sum_{i'=1}^{\chi}\max_{1\leq j'\leq l}\left[w_{(i,j)(i',j')}v(n_{i'j'})\right] + \gamma v(n_{ij}). \tag{2.8}$$

With the SoS rule, a cluster with ambiguities - that is, with several fanals activated - has potentially a more important impact on the decoding in the other clusters than that of an unambiguous cluster. NORM and SoM are proposed with the purpose of

eliminating this accumulated contribution due to ambiguities in the same cluster. Let us take an intuitive example, where we gather concepts of the same type into the same cluster, e.g. colors such as red, blue or black in a cluster, and the categories of the object such as car, mobile phone or computer in another cluster, etc. The network has stored the object "a red car", but somehow we have forgotten the exact color of this car. So we activate the fanal "car", and also all the fanals that correspond to a possible color. A retrieval algorithm applying the SoS rule will have more chance to give a wrong answer such as "a red computer" than the SoM or NORM rules. However, if we increase the size of this clique, for example, adding some other clusters for storing the brand of car, the year of production and the type of engines, etc, the fanal "car" will be much better protected. In this case, the SoS rule would perform almost as well as SoM or NORM.

2.2.3 Activation rules

An activation rule defines how the fanals are selected with respect to their scores after the message passing step. This section proposes some modifications to the previous works.

Global winner-takes-all

Unlike in the classical GB network presented in Section 2.1, when trying to recover a sparse message, the location of the target clusters may be unknown. As a consequence, one has to process a global selection of winners instead of a local one. Indeed, the local WTA automatically leaves residues in every clusters.

The algorithm of Global winner-take-all (GWTA) is simply an adaptation of the local WTA rule to the sparse organization of the network. Instead of selecting the fanals with the maximal score within each cluster, we select those having the global maximum.

Global winners-take-all

The algorithm of Global winners-take-all (GWsTA) is based on the initial parameter Σ. Ideally, this parameter should be fixed to c, the size of the message to recover. In the case where the messages are of variable sizes from $c_{\min}$ to $c_{\max}$, and one does not have a priori knowledge of the target message size c, the optimal choice is the smallest size of the stored messages: $\Sigma = c_{\min}$.

During the decoding, instead of only activating the fanals with the maximal score, the GWsTA algorithm takes the Σ fanals with the maximal or near maximal scores. If several fanals have the same score as the Σ^{th} does, these fanals are activated as well. All the rest are extinguished. Note that the GWTS rule is a particular case of GWsTA, where $\Sigma = 1$.

A simple version of this algorithm is proposed in Algorithm 2. This version uses a "max heap", that is, a data structure in which we add elements and extract those having the highest score or scores.

begin
  Suppose $\mathcal{T}$ a "max heap";
  for $1 \leq i \leq n$ **do**
    add (i, V_i) to $\mathcal{T}$;
  end
  Suppose $\mathcal{S}$ an empty set;
  for $1 \leq k \leq s$ **do**
    extract (i, V_i) the first element (with the highest score) in $\mathcal{T}$;
    add (i, V_i) in $\mathcal{S}$;
  end
  while *The score of the first element of* $\mathcal{T}$ *is the same as that of the last element of* $\mathcal{S}$ **do**
    extract (i, V_i) the first element (with the highest score) in $\mathcal{T}$;
    add (i, V_i) in $\mathcal{S}$;
  end
  Return $\mathcal{S}$;
end

Algorithm 2: A simple implementation of the "Global winners-take-all" algorithm.

A comparison between GWTS and GWsTA decoding process is illustrated by the example in Fig. 2-3. All the 7 vertices, A, B, C, D, E, F and G belong to distinct clusters. The target pattern is the clique of order 4, A-B-C-D, while the provided

pattern is composed of A, B and E, a corrupted version with erasure and insertion at the same time. Besides this clique, some connections have been established by other stored cliques (not represented in Fig. 2-3).

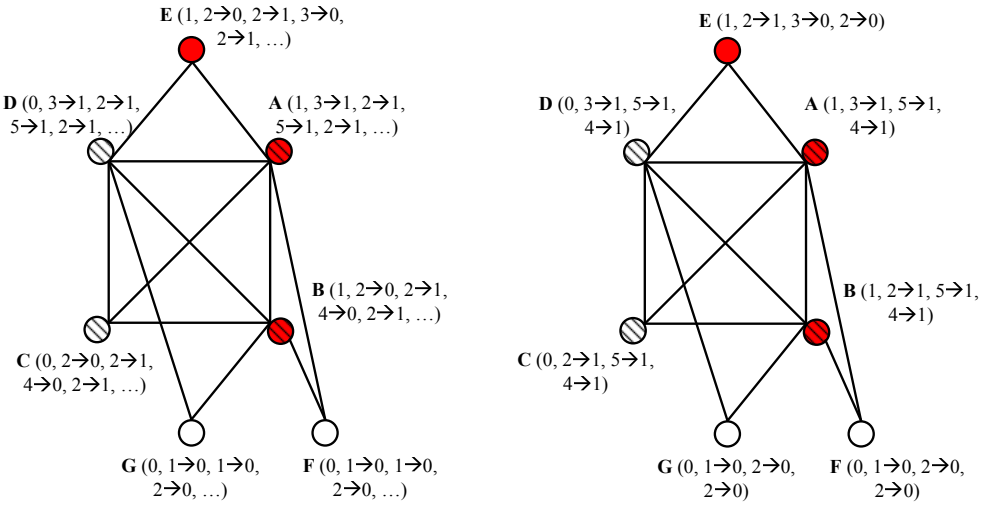

Figure 2-3: An example of different decoding scenarios of Global winner-takes-all and Global winners-take-all.

So, at the beginning of the retrieving process, A, B and E have initial values 1 whereas the others have value 0. Since all vertices belong to distinct clusters, the three dynamic rules, i.e. SoS, NORM and SoM are equivalent: the unitary signals are sent through the graph with memory effect $\gamma = 1$. With GWTA, after the first iteration, only A and D, which have the maximum of scores 3, will be activated. When the second iteration is carried out, the vertices A, B, C, D and E will be activated, since these 5 vertices are all connected to A and D ($\gamma = 1$ is equivalent to an auto-connection). One would expect the third iteration to eliminate the vertex E (score of 3), but since GWTA only takes the vertices with the maximal score, once again only A and D (score of 5) will remain active, whereas B and C (score of 4) are reset to 0. Thereby, we return to the same configuration after the first iteration. If further iterations are carried out, the retrieving process will be oscillating between these two configurations: (A-D) and (A-B-C-D-E), and will repeat the same cycle. This is the reason why we have not observed any iteration gain when executing "blind decoding" for sparse messages in [Ali13].

On the other hand, instead of just considering the maximal score, the GWsTA algorithm takes at least $\Sigma = 4$ vertices with the maximal or near maximal scores after the first iteration. In this case, A = D = 3 and B = C = E = 2 are set to 1, while F = G = 1 are set to 0. The second iteration enables to recover the clique: A = B = C = D = 5. The next iteration will converge to the same clique: A = B = C = D = 4, and that ends up the process.

Losers-kicked-out

The Losers-kicked-out (LsKO) algorithm was initially designed in order to find a maximal clique (a clique that cannot be enlarged) that contains a given set of vertices. As it is well known, the problem of listing all maximal cliques is NP-complete: it may require exponential time as there exist graphs with exponentially many maximal cliques. LsKO is not for the purpose of listing all maximal cliques, but to give an answer that is as more probably as possible to be a maximal clique and maintains a polynomial complexity as well as a certain degree of biological plausibility.

As a matter of fact, LsKO could be regarded as the opposite algorithm of WTA to some extent. The message passing is locally performed within the pattern previously activated. It calculates the scores of these activated fanals, and then eliminates those with the minimum of scores: the "losers". This process is iterative. In this way, the activated fanals at each iteration form a decreasing set, and thus the algorithm necessarily converges. One of the simplest manners to limit the set of previously activated fanals is to take a very large memory effect (e.g. larger than the number of fanals in the network, $\gamma > n$).

If the input pattern contains only "insertions", in other words, it contains the entire target message, but also some excess part of information, the LsKO algorithm alone would probably be sufficient to find the target message. In any other cases, e.g. when the input is erroneous or partially erased, etc., this algorithm should cooperate with GWTA or GWsTA to perform the retrieval task. In the sequel, we present a general framework that can be applied to all kinds of inputs.

At Phase 1, some iterations of "local loser-kicked-out" are applied, where "local"

here is in the sense that it focuses only on the activated pattern at the end of the previous iteration. This first decoding phase ends when all the remaining fanals have the same score. If the original message has undergone errors or has insertions, then at the end of this phase, we would obtain a message only with some erasure. If the initial message has just undergone some erasure, this first phase is unnecessary and convergence is immediate. Then, the algorithm enters in Phase 2 to apply a single iteration of GWTA with a memory effect $\gamma = 1$. This time, message passing is executed through the whole network. It therefore retains fanals only with the global maximum of scores. Note that GWsTA is also possible. After Phase 2, the entire target message is hopefully contained in the active pattern. Therefore, the decoder enters in Phase 3, where the exactly same iterations of "local loser-kicked-out" as in Phase 1 are carried out. The decoding is completed until the convergence is detected.

Formally, this complete algorithm can be expressed by the following equations. Let us denote by $\tilde{m}$ the estimated version of the message m given $\hat{m}$, the corrupted message provided to the network. At first, the fanal values are updated according to the input pattern:

$$\tilde{m} \leftarrow \hat{m} \tag{2.9}$$

$$v(n_{ij}) \leftarrow \begin{cases} 1, \forall n_{ij} \in \tilde{m} \\ 0, \text{ otherwise.} \end{cases} \tag{2.10}$$

LsKO is then applied locally on the set of activated fanals:
For $n_{ij} \in \tilde{m}, \forall n_{i'j'} \in \tilde{m}$,

$$v(n_{ij}) \leftarrow \sum_{i'} \max_{j'} \left[\omega_{(i,j)(i',j')} v(n_{i'j'}) + \gamma v(n_{i'j'})\right] \tag{2.11}$$

$$(v^{loc}_{\max}, v^{loc}_{\min}) \leftarrow \left(\max_{n_{ij} \in \tilde{m}} v(n_{ij}), \min_{n_{ij} \in \tilde{m}} v(n_{ij})\right). \tag{2.12}$$

This process can be iterative. If the stopping condition is not satisfied, that is, all the remaining fanals do not have the same score, more iterations of LsKO should be

applied. If $v_{\max}^{loc} \neq v_{\min}^{loc}$,

$$\tilde{m} \leftarrow \{n_{ij} \mid n_{ij} \in \tilde{m}, v(n_{ij}) > v_{min}^{loc}\} \tag{2.13}$$

$$\text{restart from (2.10).} \tag{2.14}$$

Otherwise, the first phase of LsKO ends here. We redo the value updating step (2.10) and then we apply a single iteration of GWTA with a memory effect $\gamma = 1$:

$\forall i \in [|1; \chi|], \forall j \in [|1; l|]$,

$$v(n_{ij}) \leftarrow \sum_{i'} \max_{j'} \left[\omega_{(i,j)(i',j')} v(n_{i'j'}) + \gamma v(n_{i'j'})\right] \tag{2.15}$$

$$v_{\max}^{glob} \leftarrow \max_{i,j} v(n_{ij}) \tag{2.16}$$

$$\tilde{m} \leftarrow \{n_{ij} \mid v(n_{ij}) = v_{\max}^{glob}\}. \tag{2.17}$$

Finally, a phase of "local loser-re-kicked-out" is carried out. The estimate is not read out until the convergence is detected.

$$\text{Repeat (2.10)} \rightarrow \text{(2.19) until } v_{\max}^{loc} = v_{\min}^{loc}. \tag{2.18}$$

$$\text{Return } \tilde{m}. \tag{2.19}$$

Fig. 2-4 depicts the LsKO retrieval scenario with the same example as in Fig. 2-3. A corrupted pattern composed of A, B and E is injected into the network. So, at the beginning of the retrieving process, A, B and E have initial values 1 whereas the others have value 0. At the first phase of LsKO decoding, only the vertex A ($v_A = 3$) is conserved, since B and E both obtain a score of 2, and are thus eliminated as "losers". This first phase, which is supposed to eliminate the insertions or errors in the input corrupted pattern, ends at the first iteration since there are only one active vertex, and necessarily has an unique score. The algorithm then continues to search all the direct neighbors of the active pattern (in this case, the vertex A), which is equivalent to a single GWTA step. After this step, A, B, C, D, E and F are

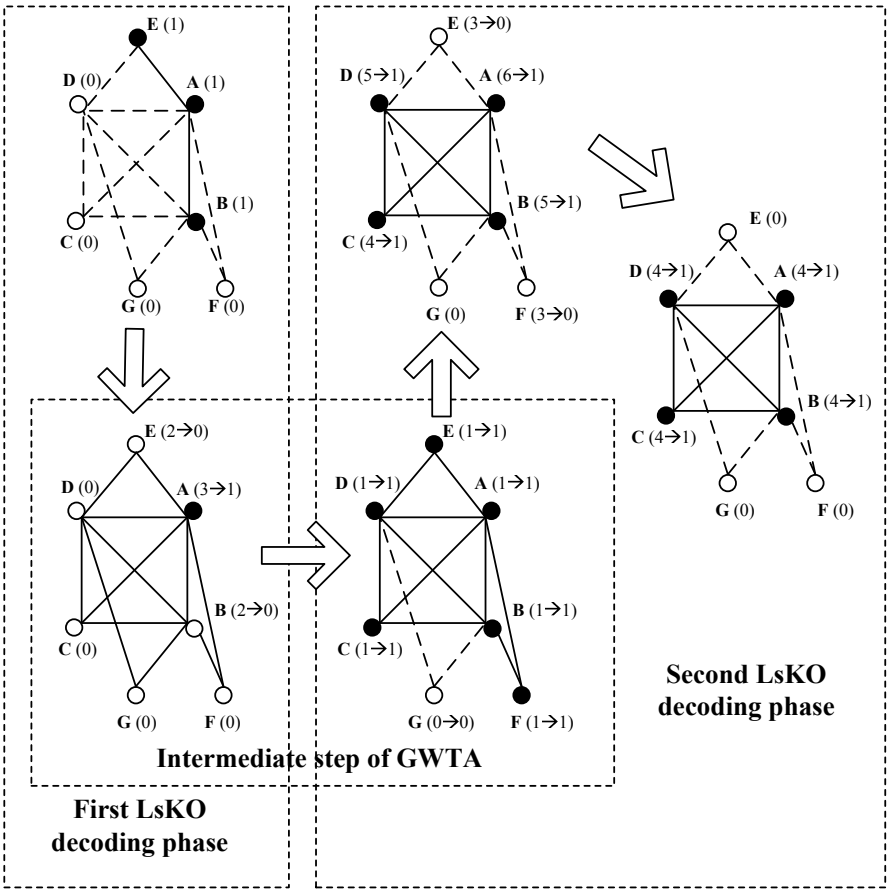

Figure 2-4: An example of Losers-kicked-out decoding process

activated. The second phase of LskO is then carried on among these 6 vertices. The first iteration eliminates the vertices E and F, and the second iteration verifies the convergence: A, B, C and D all have a score of 4. The clique is correctly retrieved.

Performance comparison

Fig. 2-5 compares the performance between GWTA, GWsTA and LsKO in terms of message retrieval error rate (MRER, defined as the probability of errors when retrieving a single message) and average number of iterations in a classical GB network composed of $\chi = 8$ clusters of $l = 256$ fanals. The stored messages are of size $c = 8$, in other words, all the clusters are involved. In this case, it is possible to apply the classic algorithm introduced in [GB11b], where the dynamic rule is SoS and the activation rule is Local WTA. Two cases are studied: 1) 4 clusters are not

provided with any information; 2) information provided in 2 clusters is erroneous. In the case of erasure, the curves of Maximum Likelihood (ML) decoding is also provided as a reference. In fact, the ML algorithm we use here is an adaptation of backtracking algorithm [Eit99] in the clique-based network. This algorithm finds at first all the fanals directly connected to the totality of the partially erased message. This preliminary step limits considerably the range of search, which makes it less complex than a typical brute-force search. Among the fanals found after the first step, the algorithm chooses randomly one fanal at a time to verify if it forms a new clique together with the previous one or not. This process continues until the size of a clique reaches c. An exhaustive search could be carried out to find all possible cliques of size c that contain the partially erased version. If there are several, one of them is chosen randomly at the end of the algorithm. ML is optimal in terms of retrieval performance, but obviously has an exponential complexity, and thus is not biologically plausible.

Fig. 2-5 shows that in the case of erasure, GWTA, GWsTA and LsKO have all similar, if not the same, retrieval error rate, which slightly outperforms the classic algorithm, and is very closed to the optimal value of ML. LsKO needs approximately one more iteration than the others before convergence. With a relatively high density, GWsTA and LsKO converge immediately to a response, which is nevertheless very likely a wrong one. In the case of erroneous inputs, LsKO outperforms the other algorithms. Moreover, all these three newly introduced algorithms need fewer iterations than the classic one, except for LsKO with a low density.

Fig. 2-6 compares their performance in retrieving sparse messages ($c < \chi$) in a network composed of $\chi = 100$ clusters of $l = 64$ fanals. The input messages are heavily corrupted: the top-left figure corresponds to messages of size 24, among which 12 fanals that forms the original message, and 12 others as insertions; the top-right figure corresponds to messages of size 50, among which 25 fanals are provided erroneously; finally on the bottom corresponds to original messages of size 50, 40 of them are erased. In all these three cases, the simulation results confirm that GWTA is not adapted to the sparse organization of the network. For instance, GWTA can

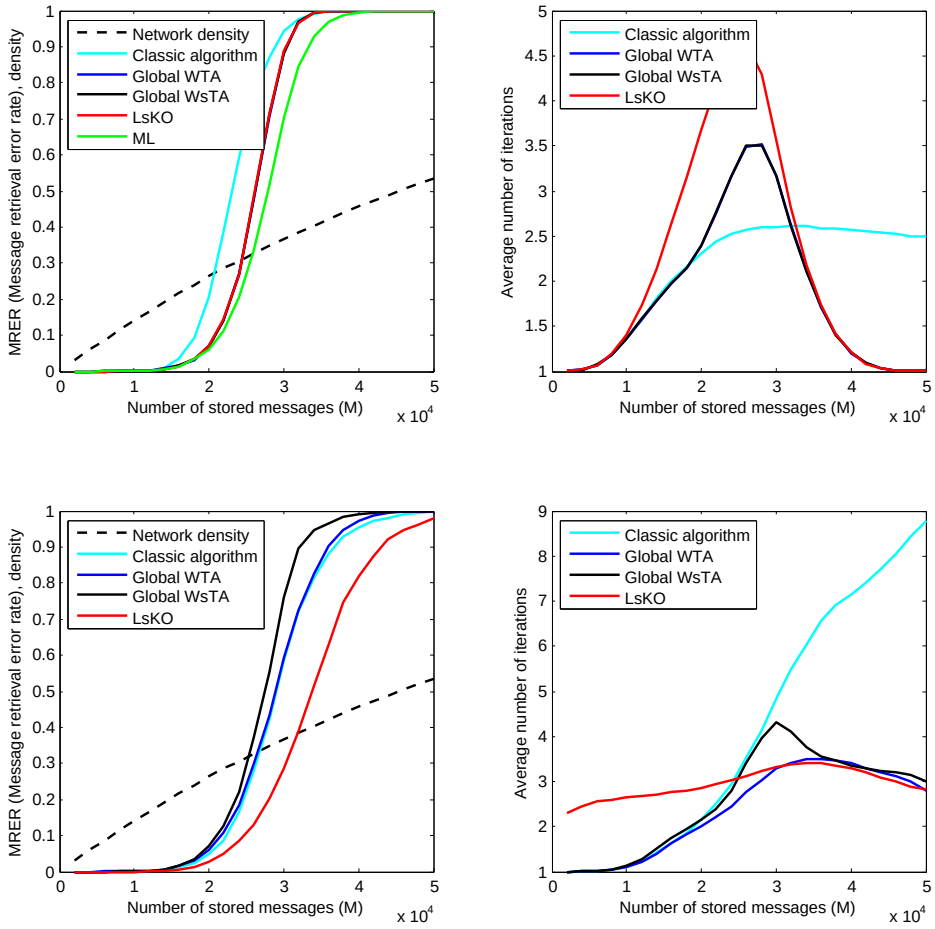

Figure 2-5: Performance comparison between GWTA, GWsTA and LsKO in terms of retrieval error rate and average number of iterations. The network is composed of $\chi = 8$ clusters of $l = 256$ fanals. Stored messages are of size $c = 8$ (all the cluster are involved). Above: 4 clusters are not provided with any information; below: the information provided in 2 clusters is erroneous.

hardly retrieve messages with heavily erroneous inputs (cf. Fig. 2-6 on the top-right). In the case of insertions and erroneous inputs, LsKO largely outperforms GWsTA in terms of retrieval error rate. In the case of erasure, GWsTA can be slightly better than LsKO by carefully choosing the parameter γ ($\gamma = 1$). In terms of average number of iterations, on the part that we interest in, i.e. corresponding to a low error rate, LsKO needs more iterations to converge than GWsTA with $\gamma = 1$.

Fig. 2-7 assesses their performance in retrieving messages of variable size. The simulated network is composed of $\chi = 100$ clusters of $l = 64$ fanals. The stored

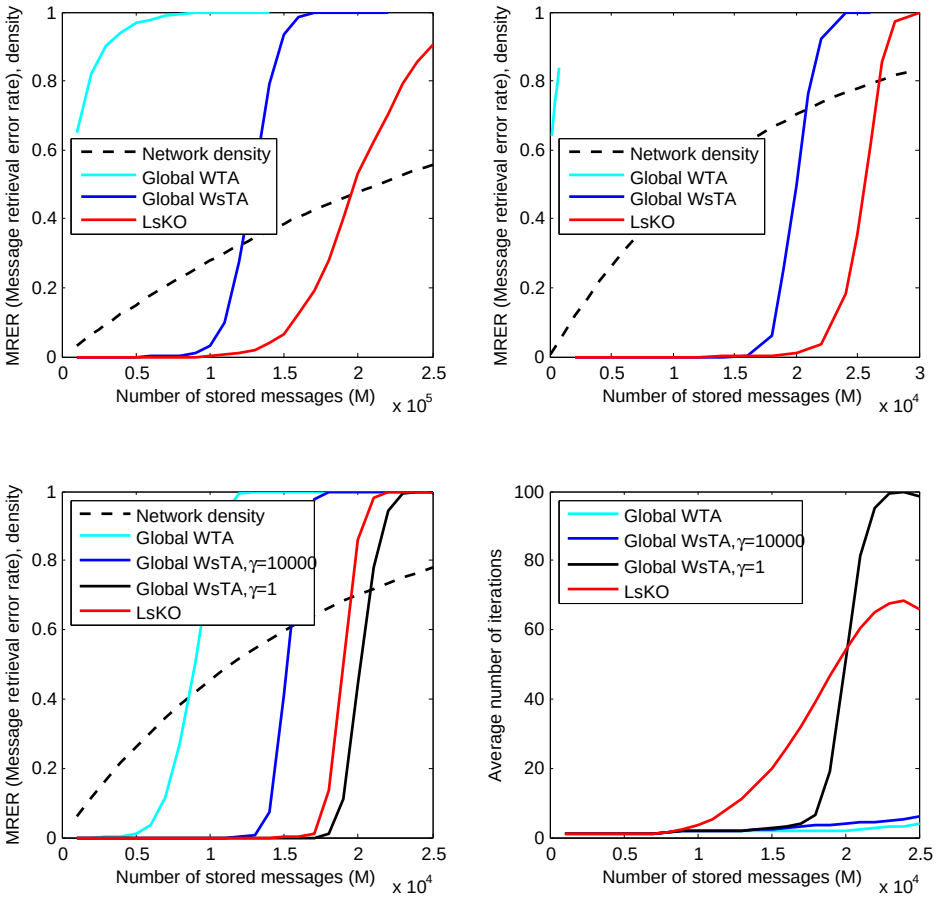

Figure 2-6: Performance comparison between GWTA, GWsTA and LsKO in terms of retrieval error rate and average number of iterations. The network is composed of $\chi = 100$ clusters of $l = 64$ fanals. The stored messages are sparse ($c < \chi$) Top-left: $c = 12$ and there 12 insertions; Top-right: $c = 50$ and the information provided in 25 clusters among them is erroneous; Bottom: $c = 50$ and 40 clusters among them are not provided with any information.

messages are of variable size c chosen uniformly from 20 to 30. For GWsTA, Σ is optimally fixed to value 20. One fourth of the clusters involved in the original message are not provided with any information. Theoretically speaking, LsKO is independent of the size of messages, so is better adapted than GWsTA, where the choice of $\Sigma = 20$ will not be optimal for retrieving messages of size 30 for example. The simulation results, however, show that GWsTA is flexible enough with respect to variable sized messages. LsKO has the lowest retrieval error rate, which is once again very close to the complex ML algorithm. But the average number of iterations also increases

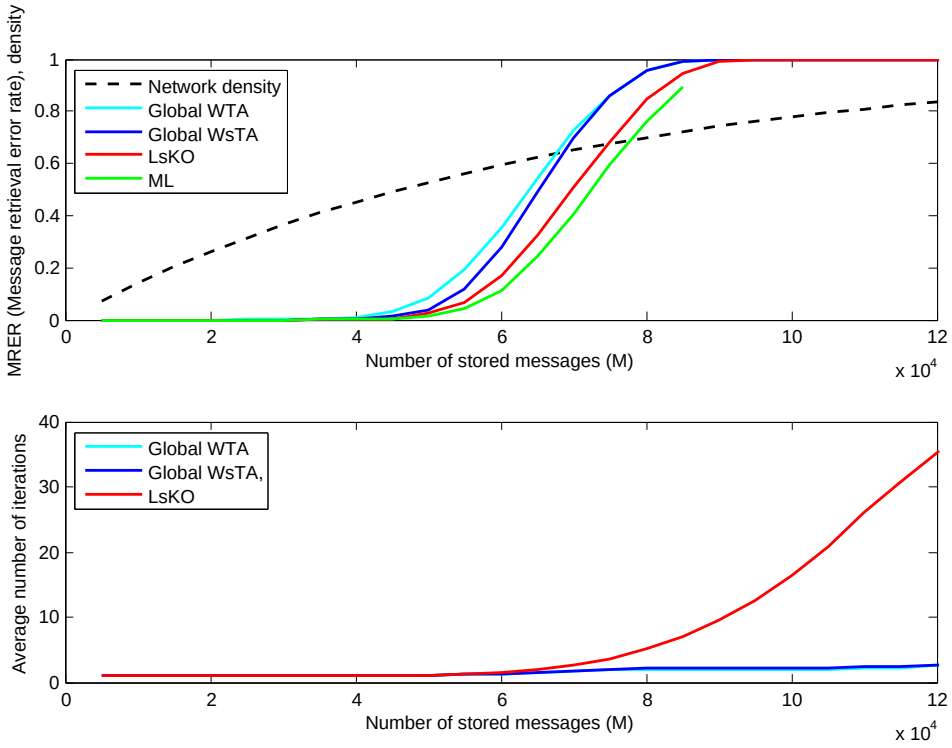

Figure 2-7: Performance comparison between GWTA, GWsTA and LsKO in terms of retrieval error rate and average number of iterations. The network is composed of $\chi = 100$ clusters of $l = 64$ fanals. The stored messages are of variable size c chosen uniformly from 20 to 30. For GWsTA, $\Sigma = 20$. One fourth of the clusters are not provided with information.

considerably for LsKO.

Note that although LsKO generally needs more iterations, each iteration involves only a small number of fanals, and this number decreases at each iteration. For instance, let us consider an input pattern composed of b fanals, one fanal per cluster. The number of additions an iteration of LsKO operates is borne by $b(b-1)$. This is the similar case for GWsTA with an extremely large memory effect, for example, $\gamma = 10000$. But with $\gamma = 1$, scores are calculated for every fanal in the whole network. In the previously mentioned case, this number is borne by $b.n$, n being the number of fanals in the whole network. The ratio is $\frac{b(b-1)}{bn} \approx \frac{b}{n} \ll 1$ in a sparse organization. Thus a GWsTA iteration is generally much more expensive in terms of number of executed operations than that of LsKO. And also, the convergence is guaranteed

with LsKO, and this convergence is independent of the nature of input messages, whereas GWsTA and GWTA guarantees the convergence only in the case of erasure.

However, the LsKO algorithm is originally designed for the purpose of finding a maximal clique in a graph, so it is seriously dependent on the structure of the clique. As a consequence, the main drawback of LsKO is the lack of flexibility with respect to damaged networks where connections are partially erased. GWsTA largely outperforms other algorithms in terms of robustness. For example, with 5% of connections erased, GWsTA allows retrieving 75000 messages of size 12 in a network of $\chi = 100$ clusters of $l = 64$ fanals with MRER inferior to 10%. And when this proportion increases to 10%, GWsTA still allows retrieving 45000 messages. When the connections are perfect, this number is approximately 130000.

Table 2.1 summarizes the comparison of these algorithms.

Table 2.1: Comparison between different activation rules

	Erasures	Errors/insertions	Variable size	Convergence	Simplicity	Robustness
GWTA	OK	Bad	Good	Erasures	Simple	Bad
GWsTA	Very Good	Good	Good	Erasures	Ok	Good
LsKO	Good	Very good	Very good	All	A bit complex	Bad
ML	Ideal	-	-	All	Very complex	-

2.3 Extensions

2.3.1 Blurred messages

Existing associative memory architectures are not adapted to blurred inputs, of which none of the data are known precisely. These inputs may arise in some applications such as data mining: retrieving information from approximate criteria, e.g. find a user who has logged in the last three minutes and visited one of the web pages in a given category. In this subsection, we demonstrate how to modify the introduced associative memories in order to deal with the problem of the recognition of blurred

messages. This modification does not affect the capacity of the model, it becomes just as well suited to blurred inputs as to those partially erased [GJ13].

We call the *neighborhood relation* a binary relation F on the vertices in the same cluster, which is symmetric and reflexive, i.e. :

$$\forall v, v', (v, v') \in F \Rightarrow (v', v) \in F \wedge \forall v, (v, v) \in F. \tag{2.20}$$

Equation (2.20) means that if the vertex v is the neighbor of v', then v' is also necessarily neighbor of v. And a vertex is neighbor of itself.

And we denote by $F_v = \{v' | (v, v') \in F\}$ the set of neighbors of the vertex v. We have

$$v' \in F_v \Leftrightarrow v \in F_{v'}. \tag{2.21}$$

For the sake of simplicity, we consider that F is b-regular: $v \in V, \#(F_v) = b$. We call blurring a message m in $\mathfrak{M}$ the operation to replace all the symbols v in m by a symbol v' that is uniformly and randomly chosen among the symbols in F_v. The problem to solve is the following:

Given a blurred version m' of a message m in $\mathfrak{M}$, retrieve m.

If we denote by $m' = \{v'_1, v'_2, ..., v'_c\}$ and $m = \{v_1, v_2, ..., v_c\}$, by definition, we have $\forall i, v'_i \in F_{v_i}$.

From m', an input version is constructed: $\tilde{m} = \{v | \exists i \in [|1, c|], v \in F_{v'_i}\}$, which includes all the neighbors of the blurred vertices. From (2.21), we get that $\forall i, v \in F_{v'_i}$. Thus, we have $m \subseteq \tilde{m}$: the target message is necessarily a part of this input version.

The retrieval algorithms introduced in Section 2.2.1 are just as well suited to blurred inputs as to those with erasure, errors or insertions. As a matter of fact, the blurred message has been transformed into a version simply with insertions, $b - 1$ insertions per segment. All of these b vertices are possible candidates activated at an equal dynamic level at the beginning. The objective of the retrieval algorithm is to reduce the window of possible candidates until only one remains in each cluster i. The SoM dynamic rule is well suited in this case, since it guarantees for each iteration

t, $\tilde{m}^{t+1} \subseteq \tilde{m}^t$ and $m \subseteq \tilde{m}^{t+1}$. In other words, the set of active nodes decreases in the sense of set inclusion, with iterations. It is therefore a decrease in well-founded sets, necessarily convergent. A trivial upper bound for the number of iterations is thus $c(b-1)$. The simulations, however, seem to indicate that a much smaller number of iterations is sufficient. The fixed point of the algorithm which defines the state of convergence does not necessarily correspond to a message because several vertices can remain active in the same cluster. In other terms, a retrieval failure occurs when a vertex not belonging to m reaches and maintains a maximum score during iterations. However, as already explained, the target message is always contained in this fixed point.

2.3.2 Binary errors

The network is also able to retrieve messages with binary errors instead of symbol errors. We suppose that there is at most one binary error in each sub-message. For retrieving the original message, we can follow the same approach as in Section 2.3.1. For example, if the sub-message provided to the decoder is 00100001, the decoder will consider a neighborhood window F_v of 9 elements: F_v ={10100001, 01100001, 00000001, 00100001, 00110001, 00101001, 00100101, 00100011, 00100000} containing the sub-message itself and its distance-1 neighbors in the sense of Hamming distance. The following retrieval algorithm is exactly the same as with blurred messages.

The error probability is limited by the probability of having another clique among the initially activated pattern. Supposing there is an interfering clique with i vertices that differs from the target clique. i can be varied between 1 and χ. Besides the connections that it shares with the target clique, the number of supplementary connections is $(\chi - i) \times i + \frac{i \times (i-1)}{2}$. And there could be $\binom{\chi}{i} \times (\log_2 l)^i$ such cliques. So, the probability of not having an interfering cliques with i vertices that differs from the target clique is:

$$\left(1 - d^{(\chi-i)\times i + \frac{i\times(i-1)}{2}}\right)^{\binom{\chi}{i}\times(\log_2 l)^i}. \tag{2.22}$$

This should be verified for all i between 1 and χ. Thus, the theoretical lower bound

P_e^b of error probability P_e can be expressed as (2.23):

$$P_e \geq P_e^b = 1 - \prod_{i=1}^{c} \left(1 - d^{(\chi-i)\times i + \frac{i\times(i-1)}{2}}\right)^{\binom{\chi}{i}\times(\log_2 l)^i} \tag{2.23}$$

with

$$d = 1 - \left(1 - \frac{1}{l^2}\right)^M. \tag{2.24}$$

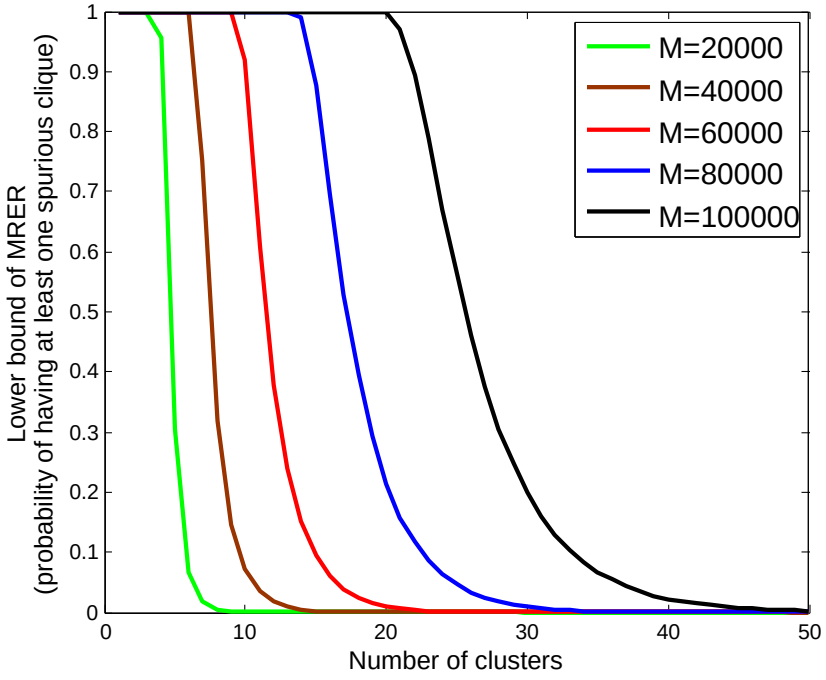

Figure 2-8: Theoretical lower bound of MRER (probability of having an interfering clique) in function of χ. $l = 256$. $M = 20000, 40000, 60000, 80000$ and 100000. At most one binary error in each sub-message. Soft decoding is applied.

From the appearance of (2.23), it is not obvious that the lower bound of error probability is a monotonically decreasing function of χ, since the three χ in this expression have opposite effects. Fig. 2-8 verifies however this monotonicity. The above analysis is also completely valid for blurred messages. It is easier to retrieve a clique of higher order from approximate inputs than that of a lower order, despite the fact that the number insertions is also linearly increased with respect to the clique order.

2.3.3 Fractal network

Consider a network composed of 4 clusters, containing l fanals each. This network stores messages of length $4 \times \log_2 l$, which are materialized by cliques of 4 vertices. We suppose that the first two bits of each sub-message m_i of length $\kappa = \log_2 l$ serve to encode their physical emplacement: 00 for top left, 01 for top right, 10 for bottom left and 11 for bottom right. So m_i can be regarded as the concatenation of two sub-sub-messages: $m_i = m_i^1||m_i^2$. m_i^1 containing 2 bits offers localization information, and m_i^2 containing $(\kappa - 2)$ bits offers content information, that is, to which fanal corresponds it in this particular location. We call it "fractal network" because this process can be repeated. For example, m_i^2 can be once more recomposed in m_i^{21} and m_i^{22}. m_i^{21} is again of 2 bits refining the localization of the corresponding fanal in the previously indicated part of the network. The precision is more and more increased as long as this fractal decomposition process continues.

This fractal approach, which has been introduced in [Gri11], makes it possible to retrieve from messages with partially erased parts. Fig. 2-9 illustrates a network composed of 4 clusters of 64 fanals each. The message $m = m_1||m_2||m_3||m_4 =$ 00001011||01011001||10110001||11100011 is stored in the network in the form of a clique. The goal of the decoder is to retrieve the original message from an altered one: $\hat{m} =$ 00__1011||0101____||1011__01||111000__, each sub-message being partially erased. In the cluster on the top left, which corresponds to $\hat{m}_1$, we have perfect content information: $\hat{m}_1^2 = 1011$, but localization information $\hat{m}_1^1$ is lost. So the decoder activates the 4 fanals that corresponds to 1011 on the top left, the top right, the bottom left and the bottom right. In a similar manner, $\hat{m}_2$ contains perfect localization information, but this time the content information is lost. The decoder activates then the whole top right part of this cluster, and knows that the correct answer is contained within this sub-cluster. A similar process is repeated for the third and the fourth clusters, where one more level of precision is added. The following retrieval process could be identical to those described in this chapter.

One can also add a new decoding rule: in each cluster one single sub-cluster can

only be activated. This can be applied to deeper hierarchy, for instance, in each sub-cluster one single sub-sub-cluster can only be activated, etc. This rule echoes to neurobiology literature where one can read that the activity of a neocortex column is likely to prevent the activity of its neighbor ones.

This construction does not affect the input message length, but reduces significantly the size of clusters that are involved in the retrieval process and thus as well as the complexity. In Fig. 2-9 for example, the sizes of clusters are limited to 64, 16, 16 and 4. And we know that the complexity is proportional to the square of cluster size.

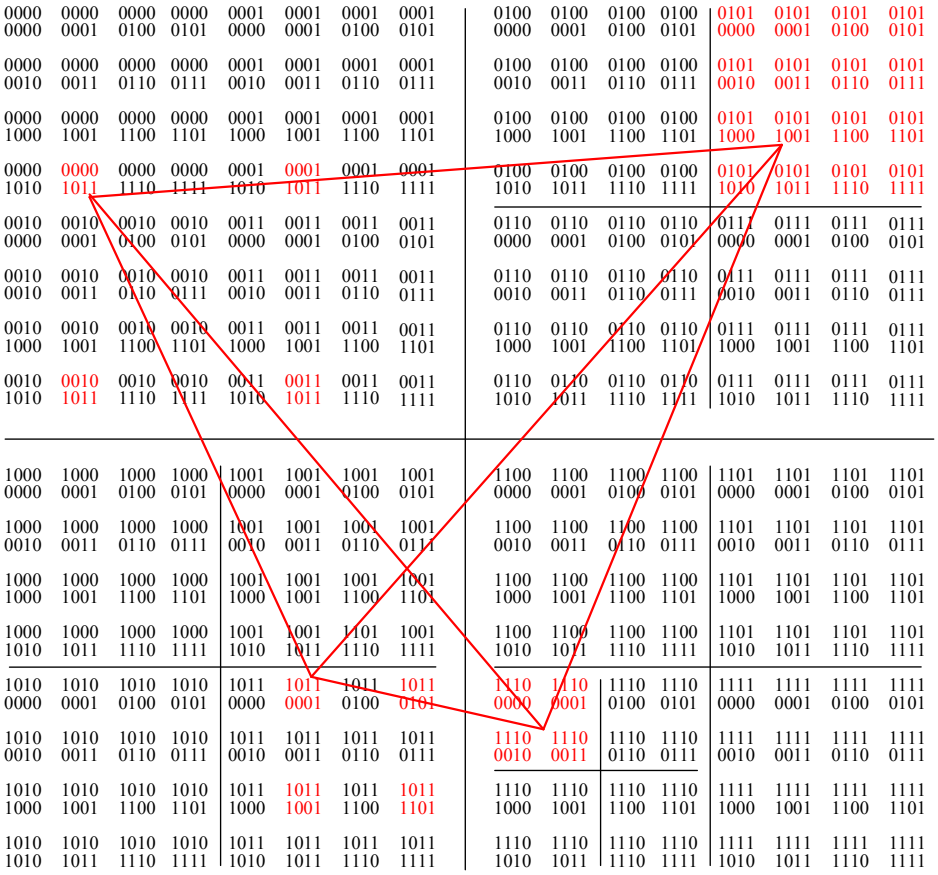

Figure 2-9: Example of fractal addressing for the input message 00 _ _ 1011 || 0101 _ _ _ _ || 1011 _ _ 01 || 111000 _ _ in a network composed of 4 sub-networks, containing 64 fanals each. The first two bits of the sub-messages serve to encode their physical emplacement: 00 for top left, 01 for top right, 10 for bottom left and 11 for bottom right. The stored message is represented by a clique.

Chapter 3

Chains of cliques and chains of tournaments

3.1 Redundancy reduction in a clique

The retrieval process described in Chapter 2 makes use of high redundancy of cliques. As a matter of fact, in terms of reachability from any node to another, a clique composed of four nodes for example can be uniquely defined by three connections if well chosen, and the other three serve as redundant information (see Fig. 3-1).

From the information point of view, the cerebral memory is robust and durable, and therefore it must be encoded redundantly. The concept of informational redundancy was originally defined by Shannon in the context of communication theory [SW49]. Recently numerous studies have proposed to investigate connectivity redundancy of the complex brain networks with the introduction of basic principles of graph theory, such as [BS09] [LMZ+12]. However, excessive redundancy is what wastes channel capacity and can limit the amount of information carried by a neural system. And as said in [FP03]: "in neuroscience, redundancy implies inefficiency". Information maximization leads naturally to redundancy reduction.

One can consider weakening redundancy of a clique by removing some of the connections. Fig. 3-2 illustrates three ways to achieve this with the example of having stored the word "learning" by a clustered network. On the top left, the original clique

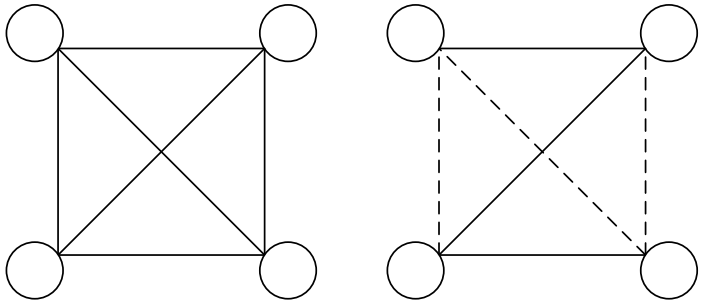

Figure 3-1: Redundancy of the clique. A clique can be specified by smaller number of connections than the complete graph (on the left). For example, in terms of reachability from any node to another, a clique composed of four nodes can be uniquely defined by three connections if well chosen (on the right), and the other three serve as redundant information.

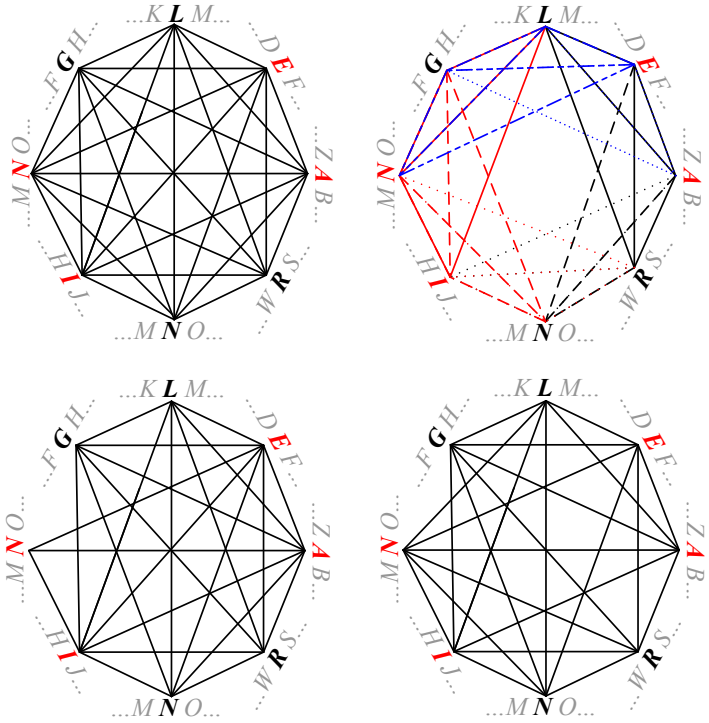

Figure 3-2: Redundancy reduction in clique-base neural networks. Example of having learnt the word "learning". Top left: clique of 8 vertices (maximum redundancy). Top right: chain of 8 cliques of 4 vertices each. Bottom left: 4 connections are randomly removed; an extreme case where all the connections removed are concentrated on only one vertex. Bottom right: 4 connections are randomly removed such that each vertex has 6 connections.

with the maximum redundancy is represented. Four connections are then removed in three manners:

1. Instead of embedding a single clique of 8 vertices, 8 small cliques of 4 vertices are constructed, the two adjacent cliques sharing three vertices (top right);

2. Connections are randomly removed. On the bottom left is an extreme example where all the connections removed are linked to the same vertex;

3. Connections are semi-randomly removed with the limitation that each vertex possesses an identical number of connections (bottom right).

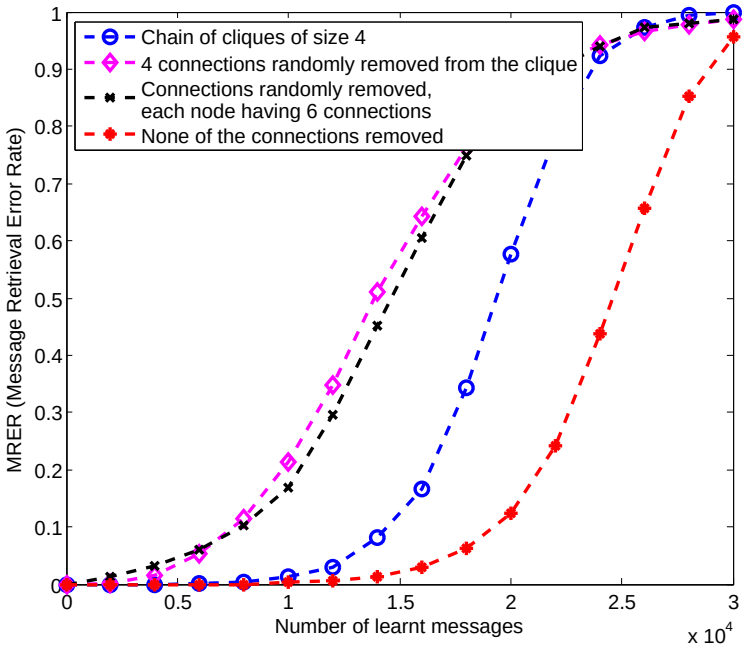

Figure 3-3: Several possibility to reduce the redundancy in a clique. Comparison of the retrieval error rate when retrieving a partially erased message in function of the number of stored messages. The network is composed of 8 clusters of 256 fanals. The number of iteration is 4. The information in half of the clusters is lost.

Fig. 3-3 evaluates the performance of these 3 schemes in terms of the MRER. The performance of the scheme with the maximum redundancy is given as a reference. Randomly generated binary messages are stored by the networks composed of 8 clusters of 256 fanals each. Each scheme (except that with the maximum redundancy)

results in the same number of connections per message and the same rate of redundancy on average. The networks are then provided with partially erased messages where the information in half of the clusters is not known. The number of iterations to recover the entire messages is fixed to 4 for all the schemes. The simulation results depicted in Fig. 3-3 suggest that homogeneous connectivity of the chain of cliques offers the best retrieval performance. Indeed, if one refers to the extreme example represented on the bottom left of Fig. 3-2, the disparity of connectivity makes it impossible to recover the entire word, given a partially erased input "L-*-*-R-N-*-*-G".

3.2 Towards unidirectionality: chain of tournaments

In a non-oriented network, the connection matrix is symmetric: $\forall (i,j),(i',j') \in [|1,\chi|] \times [|1,l|], w_{(ij)(i'j')} = w_{(i'j')(ij)}$. This is however not biologically relevant. Indeed, in the neural system, a neuron is composed of a cell body and two types of branched projections: the axon and the dendrites. The dendrites of a neuron conduct the electrochemical stimuli received from the axon of the other neural cells to the cell body, which then propagates the stimuli to the axon in certain conditions. This type of propagation of stimuli can only be realized in this unique direction. Considering oriented networks not only provides more biological credibility, but it is also more general, since in a graph, one non-oriented edge is simply two oriented arrows.

Unidirectionality seems, at first glance, to be a handicap to obtain good performance: it requires twice as much material resources (one has to use two bits in order to specify a connection between two nodes in an oriented graph, instead of only one bit in non-oriented one) for a lower redundancy. However, if the clique maximizes the number of connections between the set of fanals corresponding to a stored message, so as well as the redundancy, it requires an important number of connections which is proportional to the square of the number of corresponding nodes.

If one replaces non-oriented connections by oriented ones in a chain of cliques presented in Section 3.1, one obtains a novel structure that we name "chain of tournaments". In graph theory, a tournament is a directed graph obtained by assigning

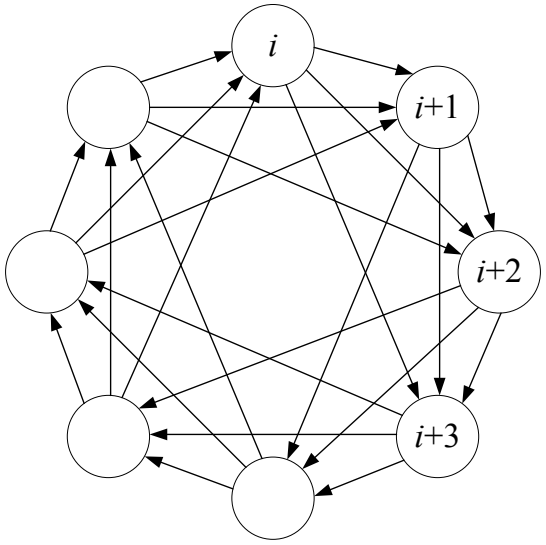

Figure 3-4: Structure of the chain of tournaments with 8 clusters and incident degree $r = 3$.

a direction to each edge in an undirected complete graph (clique). A tournament offers half of the redundancy of a clique, since the number of connections is divided by two (one can consider an edge in non oriented graphs as two arrows in opposite direction). An example of "chain of tournaments" is illustrated in Fig. 3-4. Clusters are represented by circles, and an arrow represents not a single connection between two fanals, but a set of possible connections between two clusters. One can consider such an arrow as a vectorial connection. The connections are authorized between the cluster i and j, only if $|j - i| \leq r$. r is the incident degree, which is the number of incoming vectorial connections of any cluster. In a non-oriented network embedding cliques, we have $r = \chi - 1$. We call this a "chain of tournaments" of parameter r. For example, Fig. 3-4 illustrates a chain of tournaments composed of 8 clusters of parameter $r = 3$.

The storage procedure here no longer inserts a complete clique, but a chain of χ tournaments. Each tournament contains $r + 1$ vertices, and shares r vertices respectively with the previous and following tournaments. Formally, after storing a set of messages $\mathfrak{M}_k$, this storage process can be then summarized as:

$\forall (i,i') \in [|1;\chi|]^2, \forall (j,j') \in [|1;l|]^2$,

$$w_{(i,j)(i',j')} = \begin{cases} \text{if } 1 \leq \delta(i',i) \leq r \\ 1, \quad \text{and } \exists m \in \mathfrak{M}_k, \begin{cases} f_i(m^i) = n_{ij} \\ f_{i'}(m^{i'}) = n_{i'j'} \end{cases} \\ 0, \text{ otherwise} \end{cases} \tag{3.1}$$

where $\delta(i',i) = (i'-i) \bmod \chi$.

The decoding procedure remains exactly the same as in non-oriented networks, except that one may limit the summation range to reduce complexity. Equation (2.2) is adapted as

$$\forall i,j, v(n_{ij}) \leftarrow \sum_{1 \leq \delta(i') \leq r} \max_{1 \leq j' \leq l} \left[w_{(i,j)(i',j')} v(n_{i'j'}) \right] + \gamma v(n_{ij}). \tag{3.2}$$

3.3 Associative memory with chains of tournaments

As for non-oriented clique-based networks, let us evaluate the ability of the chain of tournaments to retrieve the correct data given only part of it or in presence of errors. For networks of the same size, it is expected that the fully connected non-oriented networks outperform the oriented ones, which possess fewer incident degrees, and thus provides less information as the input to the global decoder. As a consequence, the proposed oriented network should conserve a comparable incident degree so as to obtain similar performance, and it therefore leads to a larger size of the network and a larger number of requested fanals.

Let us consider such a chain of tournaments composed of χ clusters of size l with incident degree r ($r < \chi$). Note that the number of possible connections of such a network is $\chi r l^2$, compared to $\chi(\chi-1)l^2$ (with orientation) for clique-based networks. The expected density d after learning M i.i.d. messages is then described by (3.3), which is independent of r:

$$d = 1 - \left(1 - \frac{1}{l^2}\right)^M. \tag{3.3}$$

Given the same number of stored messages and same number of fanals per cluster, the density estimation remains the same as in a non-oriented clique-based network. As a matter of fact, compared to a clique-based network, as long as the connections are more sparse in a chain of tournaments, the number of all possible connections is also reduced by the same proportion.

3.3.1 One erased cluster

It is possible to estimate the error probability after a single iteration when only one cluster is not provided with information. From now on, $\sigma = 0$ and $\gamma = 1$. The probability of correctly retrieving the erased fanal is given by (3.4):

$$P_{retrieve} = (1 - d^r)^{l-1}. \tag{3.4}$$

Considering the non zero memory effect actually used, the probability of conserving the correct fanals in other clusters is 1. The error probability when retrieving the entire message is then:

$$P_e = 1 - (1 - d^r)^{l-1}. \tag{3.5}$$

Given (3.3), this error probability can be described as:

$$P_e = 1 - \left(1 - \left[1 - \left(1 - \frac{1}{l^2}\right)^M\right]^r\right)^{l-1}. \tag{3.6}$$

Note that this probability is independent of the number of clusters χ. For instance, when only one cluster is not provided with information, the chain of tournaments composed of 8 clusters of 256 fanals each with incident degree equal to 3 offers exactly the same performance as the clique-based network composed of 4 clusters of 256 fanals each.

Referring to [GB11b], for a clique-based network composed of χ clusters of a reasonable size $l \gg 1$, if $M \ll l^2$, the retrieval error probability when one cluster is

erased is close to

$$P_e \approx l \left(\frac{M}{l^2}\right)^{\chi-1}. \tag{3.7}$$

The diversity $M^{\mathrm{c}}_{\max}$ for a given P_e is then

$$M^{\mathrm{c}}_{\max} \approx l^2 \left(\frac{P_e}{l}\right)^{\frac{1}{\chi-1}}. \tag{3.8}$$

And the efficiency of the network with complete cliques can be written as

$$\begin{aligned} \eta^{\mathrm{c}} = \frac{C}{Q} &\approx \frac{M^{\mathrm{c}}_{\max}\chi \log_2 l}{\chi(\chi-1)l^2} \\ &= \frac{\left(\frac{P_e}{l}\right)^{\frac{1}{\chi-1}} \log_2 l}{(\chi-1)}. \end{aligned} \tag{3.9}$$

In a similar way, we deduce from (3.7) the diversity $M^{\mathrm{ct}}_{\max}$ in the chain of tournaments

$$M^{\mathrm{ct}}_{\max} \approx l^2 \left(\frac{P_e}{l}\right)^{\frac{1}{r}} \tag{3.10}$$

and the efficiency

$$\eta^{\mathrm{ct}} \approx \frac{\left(\frac{P_e}{l}\right)^{\frac{1}{r}} \log_2 l}{r}. \tag{3.11}$$

Thus, the ratio of the efficiencies is

$$\frac{\eta^{\mathrm{ct}}}{\eta^{\mathrm{c}}} = \frac{(\chi-1)\left(\frac{P_e}{l}\right)^{\frac{1}{r}}}{r\left(\frac{P_e}{l}\right)^{\frac{1}{\chi-1}}}. \tag{3.12}$$

For a network of given neural resource (χ and l fixed), taking the derivative of (3.12) with respect to r gives

$$r_{\mathrm{opt}} = \ln\left(\frac{l}{P_e}\right). \tag{3.13}$$

One can conclude that

1. if $\chi \leq r_{\mathrm{opt}}$, the network with complete cliques offers the maximum efficiency possible. In other words, $\frac{\eta^{\mathrm{ct}}}{\eta^{\mathrm{c}}} < 1$ for any r;

2. if $\chi > r_{\mathrm{opt}}$, there exists an optimal r for the network based on tournaments that

leads to $\frac{\eta^{\text{ct}}}{\eta^{\text{c}}} \geq 1$.

3.3.2 Two erased clusters

In contrast to clique-based networks where we were able to give a general formula if the number of clusters without provided information χ_e is larger than one, it is generally more difficult to give such a formula for a chain of tournaments in which $r < \chi - 1$. For the reason of simplicity, from now on a cluster without input information is called a "dark" cluster. Here, since the connectivity of a cluster is only concentrated in a neighborhood of $(2r+1)$ clusters, dark clusters located in a dependent neighborhood will not have the same effect as those separated far away.

Let us consider the simplest case where the number of dark clusters is $\chi_e = 2$. If these two dark clusters are separated far away such that there are no connections between them (cluster i and $i+4$ in Fig. 3-4, for example), the probability of correctly selecting the erased fanal in each dark cluster follows (3.6). Nevertheless, if the two dark clusters are separated within r positions (cluster i and $i+2$ in Fig. 3-4, for example), the downstream dark cluster is then provided with less information, in this case $(r-1)\kappa$ bits, the probability of selecting the erased fanal in this cluster is then written as:

$$P_{retrieve} = (1 - d^{r-1})^{l-1}. \tag{3.14}$$

We only consider networks where the connections are merely unidirectional which implies $\chi > (2r+1)$. The mean probability of correctly retrieving the entire messages is then:

$$\begin{aligned} P_e = \frac{2r}{\chi - 1} \times \{1 - (1 - d^r)^{l-1} \times (1 - d^{r-1})^{l-1}\} \\ + \frac{\chi - 2r - 1}{\chi - 1} \times \{1 - (1 - d^r)^{2(l-1)}\}. \end{aligned} \tag{3.15}$$

Fig. 3-5 compares retrieval performance after one iteration between the clique-based network and chains of tournaments with the same incident degree and the same number of dark clusters. The clique-based network is composed of 4 clusters of 512 fanals, while the chains of tournaments are composed of 8, 10 or 12 clusters of 512 fanals. Both theoretical and experimental results are represented. Generally,

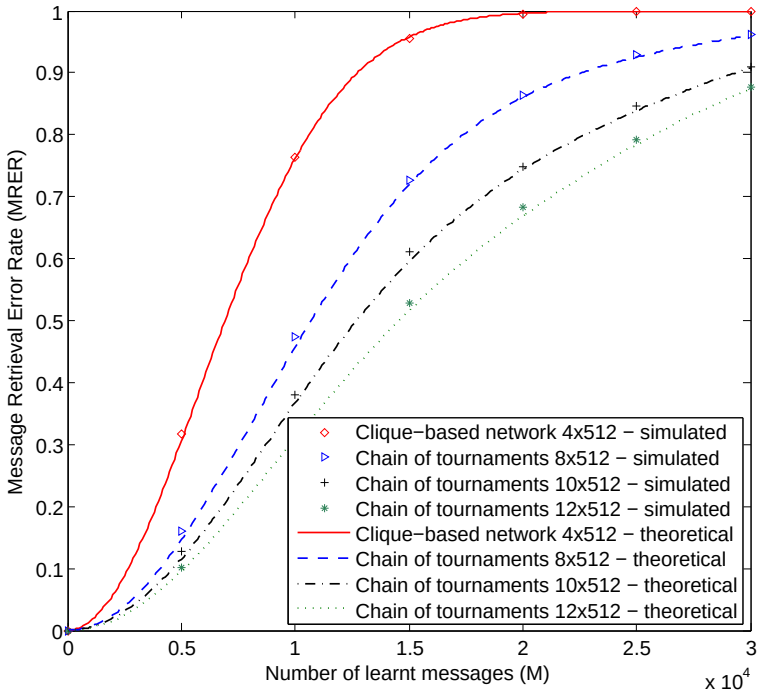

Figure 3-5: Evolution of the error rate when retrieving a stored message after 1 iteration in function of the number of stored messages. The clique-based network is composed of 4 clusters of 512 fanals, while the chains of tournaments are composed of 8, 10 or 12 clusters of 512 fanals. The incident degree is 3. There are 2 dark clusters in all considered networks. Both the theoretical curve for a single iteration and the simulation results are drawn.

for the same number of dark clusters $\chi_e > 1$, the chain of tournaments offers better performance than the clique-based one, since the lost information in the downstream dark cluster will not impact negatively on the decision in the upstream dark cluster. The larger the number of clusters is, the lower the error rate. Indeed, with a larger number of clusters, the dark clusters are more likely to locate far enough from each other, so that the lost information in one dark cluster will not impact the decision in the other. Note that the hypothesis that we made in Section 2.1.2, that is, the independence of connections is supported by simulations. The experimental results almost coincide with the theoretical curves.

However, the simulation condition in Fig. 3-5 favors the chain of tournaments. With $\chi_e = 2$, half of the clusters in the clique-based network are dark clusters, while this proportion is only $\frac{1}{6}$ for the chain of tournaments of 12 clusters, for example.

It is thus interesting to build an oriented network which offers nearly equivalent performance compared to a given non-oriented configuration with the same proportion of dark clusters.

Fig. 3-6 fixes this proportion to $\frac{1}{4}$ both for the clique-based network and chains of tournaments. It is shown that the clique-based network ($\chi = 4$, $l = 512$) possesses similar MRER after 1 iteration compared to the chain of tournaments ($\chi = 4$, $l = 512$, $r = 4$), which consumes twice as many as neural resources. Note that a slight difference of the incident degree can lead to a significant gap in terms of MRER. For instance, with $M = 20000$ messages stored, MRER after 1 iteration falls from 86.4% for $r = 3$ to 2.3% for $r = 5$. The gain of iterations (red curves correspond to 4 iterations) is then evaluated for the two similar schemes. After only one iteration, the retrieval procedure is already optimal for the corresponding clique-based network, and therefore the gain of iterations is not observed. On the contrary, for the chain of tournaments, the decision on the dark clusters after one iteration is made only based on the information provided by their direct upstream neighbors. More iterations are useful to propagate existing correct information so as to achieve a more accurate estimate.

A similar conclusion is drawn in Fig. 3-7 simulating the retrieval performance in the presence of $\frac{1}{4}$ of errors. Once again the gain of iterations in the chain of tournaments is more important than that in the clique-based network. The presence of incorrect information is a more demanding scenario than that of erasures. As a consequence one iteration is no longer sufficient for either of these two schemes.

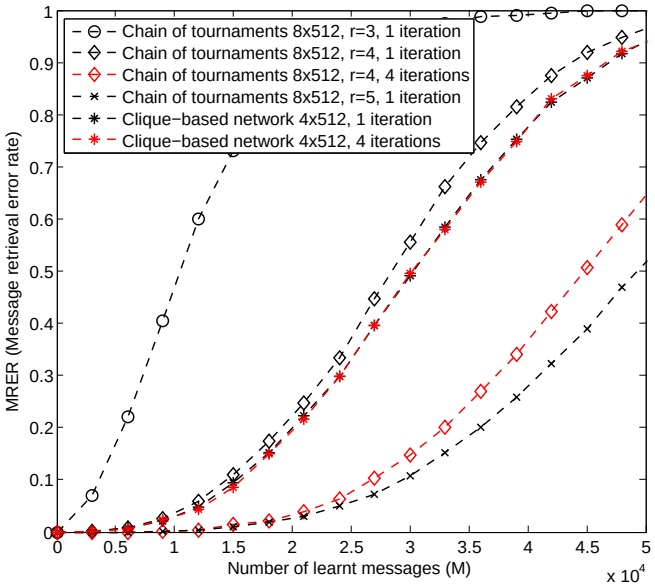

Figure 3-6: Evolution of the error rate when retrieving a message in function of the number of stored messages. The clique-based network is composed of 4 clusters of 512 fanals, while the chains of tournaments are composed of 8 clusters of 512 fanals. The incident degree is 3, 4 or 5. Proportion of the dark clusters is $\frac{1}{4}$. Black curves correspond to 1 iteration while red ones correspond to 4 iterations.

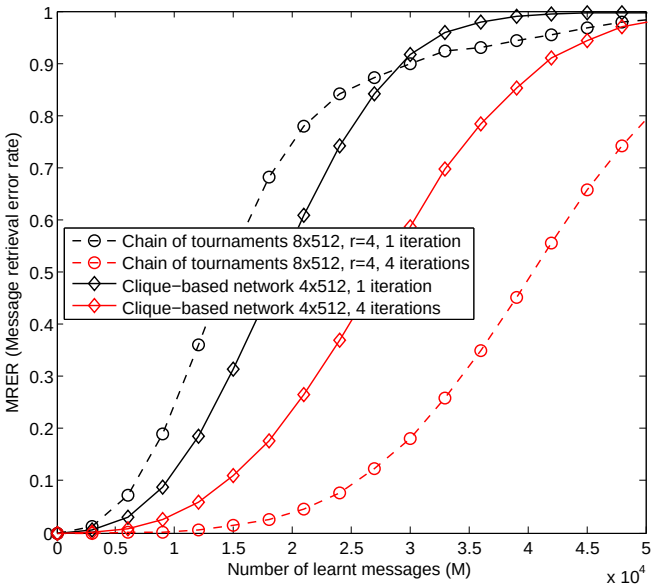

Figure 3-7: Evolution of the error rate when retrieving a message in function of the number of stored messages. The clique-based network is composed of 4 clusters of 512 fanals, while the chain of tournaments is composed of 8 clusters of 512 fanals. The incident degree is 4. Proportion of clusters provided with incorrect information is $\frac{1}{4}$. The simulation results after 1 and 4 iterations are drawn.

Chapter 4

Storage of sequences in tournament-based neural networks

Unidirectionality enables the network to exhibit sequential or temporal behavior, which opens a novel horizon compared to non-oriented networks. In fact, sequential structure imposed by the forward linear progression of time is omnipresent in all cognitive behaviors. Cognitive sequential learning is rather unidirectional (irreversible) than bidirectional (reversible). Indeed, while learning a song, the succession of the lyrics as well as the melody rather follows the forward progression of time, and the attempt to sing a song in reversed order reveals to be too difficult for human cognitive capacities. In this Chapter, we will explain how to store and decode sequential messages with a variable length in a chain of tournaments, whereas the length of messages stored by clique-based networks must accord with the number of clusters [JGB12].

4.1 Definition of the sequence

Before further studies, let us define a temporal sequence following the terminology introduced by Wang and Arbib [WA90]. A temporal sequence $\mathbf{S}$ of length L is defined as

$$p_1 - p_2 - ... - p_L.$$

Each $p_i (i = 1, 2, ..., L)$ is called a component of $\mathbf{S}$, which can be just a symbol, or a spatial pattern composed of several symbols in parallel. The length of a sequence is the number of components in the sequence. In this chapter, we are only interested in the sequence of symbols.

Generally, a temporal sequence can contain repetitions of a same subsequence belonging to different contexts. The degree of a current component p_i is the number of the previous components required to find p_i again. The *degree* or the *complexity* of a sequence is the maximum degree of its components. A sequence of degree one is called a *simple* sequence, otherwise it is a *complex* sequence. For example, in the sequence $\mathbf{S}_1$: C-A-B-C-A-D-C-D, the degree of the second "D" is 2. $\mathbf{S}_1$ is a complex sequence of degree 3. And the sequence $\mathbf{S}_2$: A-B-C-D-E-F-G is a simple sequence, thus of degree 1.

4.2 Storing

The principle of storing a sequence is illustrated in Fig. 4-1. Clusters are represented by circles, and an arrow represents not a single connection between two fanals, but a set of possible connections between two clusters. Each symbol p_i is mapped to a particular fanal in a corresponding cluster. During the storage process, the directed connections are established successively between the current cluster and its r predecessors. For instance, the fanal corresponding to p_4 in the cluster 4 receives synapses from those corresponding to p_1, p_2 and p_3, respectively in cluster 1, 2 and 3; that of p_5 receives synapses from those of p_2, p_3 and p_4, etc. The loop structure enables the reuse of the resources. A cluster may be solicited at several times if L, the length of the sequence, surpasses the number of clusters χ. In the example illustrated in Fig. 4-1, cluster 1 is solicited three times by symbols p_1, p_9 and p_{17}. In this manner, the network is able to store sequences of any length, not limited by the number of clusters. In particular, the encoded sequences can for example correspond to voluminous multimedia streams. In fact, the sequences are not mandatorily binary, and the number of fanals l in each cluster can correspond to the number of symbols in a

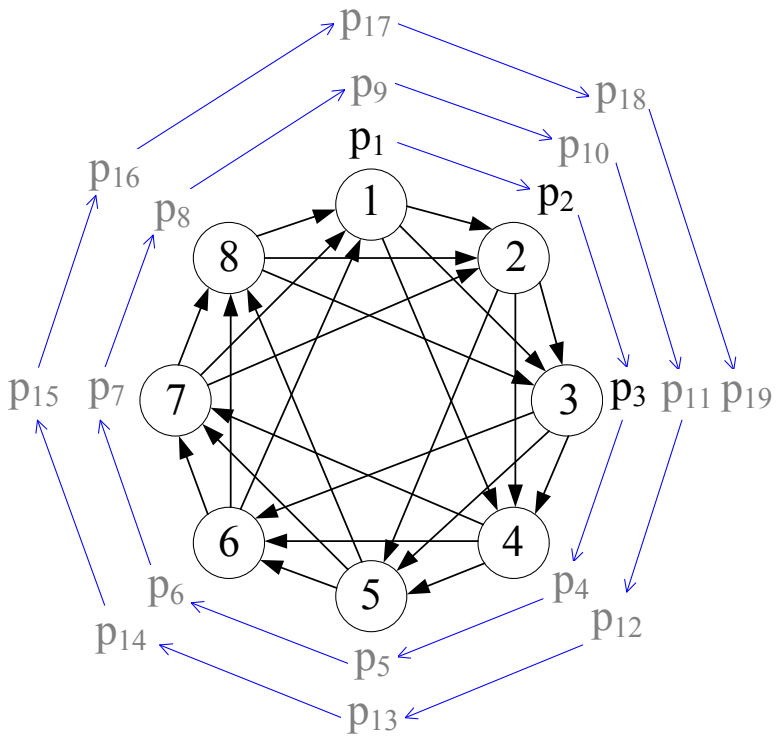

Figure 4-1: Use of chain of tournaments for successive storing and decoding arbitrary long sequential messages.

predefined finite alphabet.

We assume that the storage of a sequence always starts at Cluster 1. After storing a set denoted $\mathfrak{S}$ of S sequences of length L, the network (chain of tournaments of parameter r) is defined formally by:
$\forall (i,i') \in [|1;\chi|]^2, \forall (j,j') \in [|1;l|]^2$,

$$w_{(i,j)(i',j')} = \begin{cases} 1, & \text{if } 1 \leq \delta(i') \leq r \\ & \text{and } \exists \mathbf{S} \in \mathfrak{S}, \exists \pi \leq \dfrac{L}{\chi}, \\ & \begin{cases} f_i(p_{i+(\pi-1)\chi}) = n_{ij} \\ f_{i'}(p_{i'+(\pi-1)\chi}) = n_{i'j'} \end{cases} \\ 0, & \text{otherwise.} \end{cases} \tag{4.1}$$

The mapping function f_i maps the symbol $p_{i+(\pi-1)\chi}$ to a corresponding fanal in the

cluster i on the π_{th} passage of this sequence.

In this way, the corresponding set of fanals is related to each other only if they are consecutive within r positions in the considered sequence.

We suppose the stored sequences are randomly, uniformly and independently generated among all the possible ones. Once again, we presume the independence of connections in order to apply the binomial law. As already explained, this approximation, verified by simulations, could be intuitively explained by the fact that with a large number of long sequences stored in the network, two connections taken randomly from the graph are unlikely to have been added at the same time.

The density of the network after learning S sequences can be then expressed as

$$\begin{aligned} d^{seq} &= p\left(\exists \mathbf{S} \in \mathfrak{S}, \exists \pi \in \left[\left|1, \frac{L}{\chi}\right|\right], p_{i+(\pi-1)\chi} = n_{ij} \wedge p_{i'+(\pi-1)\chi} = n_{i'j'}\right) \\ &= 1 - p\left(\forall \mathbf{S} \in \mathfrak{S}, \forall \pi \in \left[\left|1, \frac{L}{\chi}\right|\right], p_{i+(\pi-1)\chi} \neq n_{ij} \vee p_{i'+(\pi-1)\chi} \neq n_{i'j'}\right) \\ &= 1 - p\left(p_{i+(\pi-1)\chi} \neq n_{ij} \vee p_{i'+(\pi-1)\chi} \neq n_{i'j'}\right)^{\frac{SL}{\chi}} \\ &= 1 - \left\{1 - p(p_{i+(\pi-1)\chi} = n_{ij}) p(p_{i'+(\pi-1)\chi} = n_{i'j'})\right\}^{\frac{SL}{\chi}} \\ &= 1 - \left(1 - \frac{1}{l^2}\right)^{\frac{SL}{\chi}}. \end{aligned} \tag{4.2}$$

4.3 Error tolerant decoding

To start the decoding process, the network should be provided with any subsequence of r consecutive symbols, in particular the first r symbols if one would like to retrieve the sequence from the beginning, but this is not necessary. The 3 starting symbols are illustrated in black in Fig. 4-1 (The symbols to be deduced are in gray). Note that if the supplied subsequence is in the middle of the sequence, the decoder should be provided with supplementary information where the corresponding clusters' emplacements are. Actually this is way the biological brain behaves, since if one gets stuck in the middle of a melody when he is playing on the piano, it seems easier to recall it if he restarts from the beginning of the tune.

The decoding process is progressive because at each step, the decision to make in

the subsequent cluster is based on contributions from the r previous ones. Formally, the decoding can be expressed as follows:

$$\text{for } r+1 \leq \lambda \leq L : \begin{cases} i \leftarrow \lambda \bmod \chi + 1 \\ \forall i,j, v(n_{ij}) \leftarrow \\ \quad \sum_{1 \leq \delta(i') \leq r} \max_{1 \leq j' \leq l} \left[w_{(i,j)(i',j')} v(n_{i'j'}) \right] \\ \quad\quad + \gamma v(n_{ij}) \\ v_i^{\max} \leftarrow \max_j \left[v(n_{ij}) \right] \\ \forall j, v(n_{ij}) \leftarrow \begin{cases} 1 \text{ if } v(n_{ij}) = v_i^{\max} \\ 0 \text{ otherwise.} \end{cases} \end{cases} \tag{4.3}$$

An important comment concerns the anticipation deduced from this decoding principle. Indeed, in Fig. 4-1, while the network decodes in cluster 4, given the dynamics in cluster 1, 2 and 3, activity is already emerging in cluster 5, receiving synapses from cluster 2 and 3; and cluster 6, receiving synapses from cluster 3. These anticipated information will give a rough estimate of the network dynamics in the following scale of time. This echoes some biological observations such as [LLZ+09], according to which the very existence of anticipatory imagery suggests that retrieving stored sequences of any kind involves predictive readout of upcoming information before the actual sensorimotor event.

It is also important to note that not only the decisions will never come back to previous symbols, but also the iteration gain is definitely lost, since only one iteration at each step is applicable in this process. In Fig. 4-1, the decoder is provided with 3 starting symbols, and it requires 16 successive 1-iteration steps to decode the entire sequence.

The clique-based networks are robust to errors. This property is hopefully inherited by chains of tournaments to be error tolerant in sequential decoding. Indeed,

(4.3) guarantees the activation of the target fanal in any case, and thus will always contribute at the next step. Let us define symbol retrieval error rate (SBRER) as the fraction of the symbols that are not correctly retrieved within any test sequence. One should distinguish two different types of errors: the ones generated because there have been errors within the r previous positions, and those generated by the network density being too high. Note that the errors occuring because the network density is too high will occur even if perfect information is provided, and we call these the *innate* symbol retrieval error rate. For an error-intolerant system, practically all errors are of the former type. One can roughly estimate the latter as the error rate at a single decoding step when perfect information is provided:

$$1 - (1 - d^r)^{l-1} \,. \tag{4.4}$$

This should be compared to the real SBRER to evaluate the degree of error tolerance.

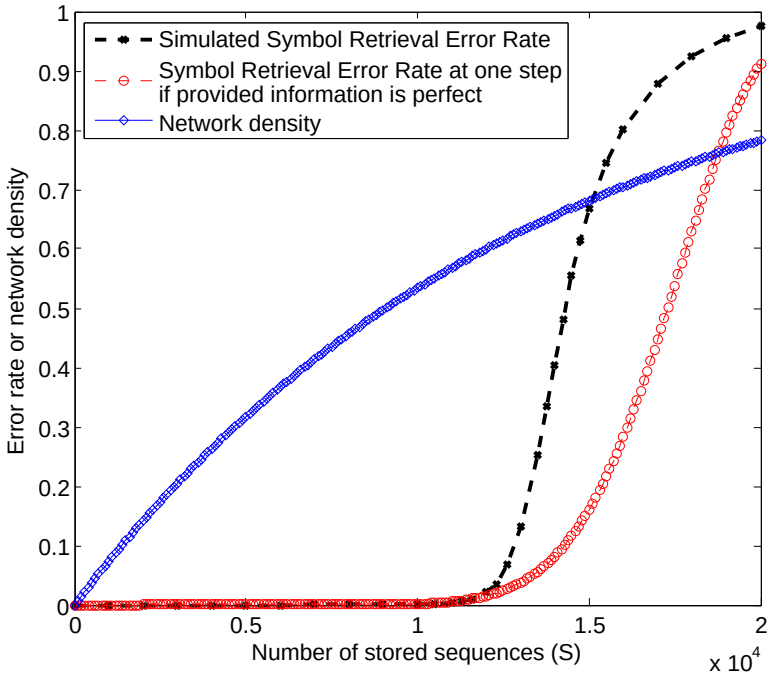

Figure 4-2: Performance of the error tolerant decoding in a chain of tournaments with $\chi = 20$, $l = 256$ and $r = 19$. The length of each sequence is $L = 100$. Three curves are illustrated: simulated symbol retrieval error rate, symbol retrieval error rate for one decoding step when provided with perfect information, and the network density.

The simulation is done in Fig. 4-2 for the chain of tournaments with parameters $\chi = 20$, $l = 256$ and $r = 19$. The length of each sequence is $L = 100$. We observe that the occurrence of errors in the previous positions does make the task more difficult for the decoder, but the performance is far from catastrophic. For example, for an innate SBRER= 20%, the network is able to store 15000 sequences of length 100. In practice, this same error rate corresponds to 13000 sequences. Roughly half of symbol retrieval errors are innate.

4.4 Sequence retrieval error rate

As discussed before, giving a theoretical estimation of SBRER is not trivial in sequential decoding, since the occurrence of one retrieval error at a given step will provide the next r steps with erroneous information, and thus potentially leading to a cumulated number of retrieval errors. We are interested therefore in the sequence retrieval error rate (SQRER), which is defined as the ratio between the number of incorrectly retrieved sequences over the total number of test sequences. Test sequences are among those having been stored by the network. The retrieval of any test sequence will be considered as a failure as soon as a single error at the symbol level appears.

Each decoding step is a small segmented problem to identify the target fanal by exploiting the stimuli of the r previous fanals and the established synapses. An error will occur when at least one fanal other than the target one in the current cluster is also connected to these r previous fanals. The probability of the existence of such a connection is d, the network density. Thus, the probability of making a right decision for a certain decoding step is

$$(1 - d^r)^{l-1} . \tag{4.5}$$

There are $(L - r)$ steps for decoding a whole sequence of length L. The error rate SQRER is finally estimated by the following formula:

$$P_e^{\text{seq}} = 1 - (1 - d^r)^{(l-1)(L-r)} . \tag{4.6}$$

One question is posed here: for a limited quantity of neural resource (n fanals), and a given set to store (S sequences of length L), how should the number of clusters χ be chosen in order to minimize SQRER? We fix $r = \chi - 1$, since obviously it is of interest to take r as large as possible to minimize the error rate, the density d being independent of r.

Some assumptions may be made in order to simplify the demonstration. By supposing $d \ll 1$, we deduce from (4.2) that

$$d \approx \frac{SL}{\chi l^2}. \tag{4.7}$$

For $r = \chi - 1$, we have

$$\begin{aligned} P_e^{\text{seq}} &= 1 - \left(1 - d^{\chi-1}\right)^{(l-1)(L-\chi+1)} \\ &\approx l(L-\chi)d^{\chi}, \text{ if } P_e^{\text{seq}} \ll 1,\ l \gg 1 \text{ and } \chi \gg 1 \\ &\approx l(L-\chi)\left(\frac{SL\chi}{n^2}\right)^{\chi}. \end{aligned} \tag{4.8}$$

By differentiating $\log(P_e^{\text{seq}})$ with respect to χ, we obtain

$$\begin{aligned} \frac{d\log(P_e^{\text{seq}})}{d\chi} &= \log\left(\frac{SL\chi}{n^2}\right) - \frac{1}{\chi} + 1 - \frac{1}{L-\chi} \\ &\approx \log\left(\frac{SL\chi}{n^2}\right) + 1, \text{ if } 1 \ll \chi \ll L. \end{aligned} \tag{4.9}$$

Finally, setting the derivative equal to zero gives

$$\chi_{opt} \approx \frac{n^2}{eSL}. \tag{4.10}$$

For example, for $n = 4096, S = 3000$ and $L = 100$, the optimal number of clusters is $\chi_{\text{opt}} \approx 20$, for a corresponding density $d \approx \frac{1}{e} \approx 0.37$. This is acceptable for a rough estimate, since with the same set of parameters, the exact formula of density (4.2) gives $d \approx 0.30$. Note that χ_{opt} is proportional to the square of the number of fanals and inversely proportional to the sequence length and the number of sequences.

4.5 Capacity and efficiency

Each sequence is composed of $L\log_2(l)$ bits. If the number of stored sequences is small compared to the total number of possible ones, what we consider in the sequel, it has been shown in [GRSG12] that the capacity, that is, the number of bits stored in the network approaches:

$$C^{\text{seq}} = SL\log_2(l). \tag{4.11}$$

Each cluster is connected via a vectorial arrow towards each of its r downstream neighbors. A vectorial arrow is potentially composed of l^2 arrows, each of which can be encoded by 1 bit. Thus, Q, the quantity of memory used by the network is:

$$Q = r\chi l^2. \tag{4.12}$$

With the same hypothesis as above, this leads to the expression of network efficiency:

$$\begin{aligned} \eta^{\text{seq}} &= \frac{C}{Q} = \frac{SL\log_2(l)}{r\chi l^2} \\ &\approx \frac{\log_2(n/\chi)}{e(\chi-1)}, \text{ for } r = \chi - 1. \end{aligned} \tag{4.13}$$

For the same configuration as in the previous subsection, we have $\eta^{\text{seq}} = 14.1\%$. This calculation is valid in the case of $r = \chi - 1$, however, a higher efficiency is usually observed for a lower degree of anticipation r. Table 4.1 describes theoretical values for several different configurations of the network. Some conclusions are drawn: the number of clusters is proportional to the diversity, but does not have any effect on the efficiency; a relatively large incident degree r, and thus a high level of redundancy, does not necessarily mean a high informational efficiency; the efficiency tends asymptotically to 69% [KPS10] when the number of fanals per cluster l tends to infinity, but the convergence is very slow. For a reasonably sized network with the concern for biological plausibility, the efficiency is around 20% to 30%. Indeed, as we have already stated, a macrocolumn contains about 50 to 100 microcolumns. If we let a fanal represent a microcolumn and let a cluster represent a macrocolumn, the

Table 4.1: Maximum number of sequences (diversity) S^{seq} that a chain of tournaments is able to store and retrieve with an error probability smaller than 0.01, for different values of c, l, r and L. The values of corresponding efficiency η^{seq} are also mentioned.

χ	l	r	L	S^{seq}	η^{seq}
8	512	3	16	1513	3.5%
50	128	10	100	2335	20.0%
50	128	20	100	5693	24.3%
50	128	49	100	11728	20.5%
30	512	23	100	57206	28.5%
30	512	29	100	70914	28.0%
100	2^{26}	40	200	$1.6.10^{15}$	45.1%

number of fanals per cluster should be also of the order of 100. Obviously, it is not reasonable to consider a cluster composed of 2^{26} fanals. The last line of Table. 4.1 is given only to illustrate the extremely slow convergence to the asymptotical bound of 69%.

Chapter 5

Storage of vectorial sequences in generalized chains of tournaments

The structure represented in Fig. 3-4 is a generalization of the clique-based networks. It becomes a clique by setting $r = \chi - 1$. It is still possible to generalize this topology further:

1. A chain of tournaments is not necessarily a closed loop;

2. A given component of the sequence at time τ, p_τ, is not necessarily a single symbol, but a set of parallel symbols that corresponds to a set of fanals in different clusters. The sequence then becomes vectorial. We call these sets of parallel symbols as "vectors" or "patterns".

In this chapter, we propose some solutions to storing vectorial sequences in generalized chains of tournaments.

5.1 Sparse random patterns

As already explained in Chapter 2, a *sparse message* or a *sparse pattern* is a message that contains few expressed elements. Back to our clique-based network models, a *pattern* is mapped to a set of c fanals, distributed in a equal number of clusters, with only one fanal per cluster. The pattern is called *sparse* if $c < \chi$. Here, we are not

interested in the way to store and recall separate sparse patterns, the different aspects of which were originally assessed in [Ali13] [ABGJ13] and were improved in Chapter 2, but in the association between these patterns to build up sequential behavior.

5.2 Storing vectorial sequences in looped chains of tournaments

In Chapter 4, we have considered a circular structure where the clusters get involved periodically and successively one after the other to store a sequence of symbols. This periodicity enables the network to store sequences of any length, the capacity being only limited by the total neural resources. This structure can be extended to store sequences of patterns. Let us consider a network composed of α sub-networks, each of which contains $\frac{\chi}{\alpha}$ clusters. During the storage process, a spatial pattern belonging to a sub-network is associated with the r ($r < \alpha$) patterns belonging to subsequent sub-networks through oriented complete bipartite graphs. For the reason of simplicity, we consider all the patterns having the same size c. At each step of storage, in a corresponding sub-network, we choose randomly c out of $\frac{\chi}{\alpha}$ clusters, and in these chosen clusters, we select randomly one fanal out of l to be part of a pattern. The probability of a fanal to be selected is thus $\frac{c}{n/\alpha}$. An arrow exists if and only if the upstream fanal and the downstream fanal are both selected, the probability of which is $\frac{c^2}{(\frac{n}{\alpha})^2}$. After storing S sequences of length L, an arrow does not exist if and only if this pair of fanals is not selected (the probability is $1 - \frac{c^2}{(\frac{n}{\alpha})^2}$), and this should be true for approximately $\frac{LS}{\alpha}$ passages on the same pair of sub-networks. Finally, the density of the network after storing S sequences of L patterns of c fanals each can be expressed as

$$d = 1 - \left(1 - \frac{c^2}{(\frac{n}{\alpha})^2}\right)^{\frac{LS}{\alpha}}. \tag{5.1}$$

Once more, the parameter of time overlapping r does not impact the expression of the network density, but only the number of all possible connections.

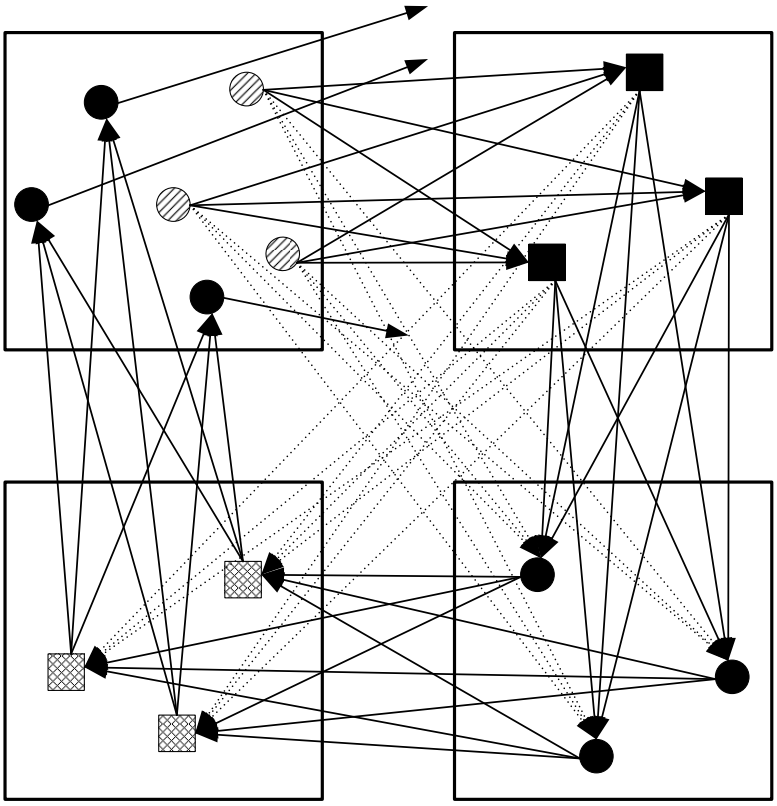

Figure 5-1: Storing vectorial sequences in a looped chain of tournaments. The network is divided into four distinct sub-networks composed of several clusters for each. A spatial pattern belonging to a sub-network is associated with the following two patterns belonging to other sub-networks through oriented complete bipartite graphs. The pattern p_τ is linked to $p_{\tau+1}$ by continued lines with a flash end, and to $p_{\tau+2}$ by dotted lines with a flash end. Neural resources (i.e. fanals) can be reused as long as the sequence continues being stored in this circular manner.

At each step of the retrieval process, the decision will be made within the current sub-network. There will be failure if one or more fanals, other than the c target ones, are connected to all of the rc fanals previously activated. The probability of correct retrieval in a sub-network at time τ is $(1-d^{rc})^{\frac{n}{\alpha}-c}$. And for the correct retrieval of the whole sequence, this should be verified for $L-r$ steps. SQRER is finally expressed as

$$P_e = 1 - \left[(1-d^{rc})^{\frac{n}{\alpha}-c}\right]^{L-r}. \tag{5.2}$$

The performance of this structure will be discussed in Section 5.3.2.

5.3 Storing vectorial sequences in unlooped chain of tournaments

As already explained, the ability to reuse neural resources is the key feature that enables the network to store sequences of any length. This idea is implemented in Chapter 4 and Chapter 5.2 both by a segmentation of the network and a periodical addressing of each disjoint segment. It is possible to push this approach even further. In fact, the reuse of neural resources need not be periodic. A cluster can be addressed at any time slice of the sequence. And the network could perform as a whole during the retrieval process.

This principle is illustrated in Fig. 5-2. In this example, the network is composed of $\chi = 100$ clusters, represented by squares in the grid. Each pattern generates

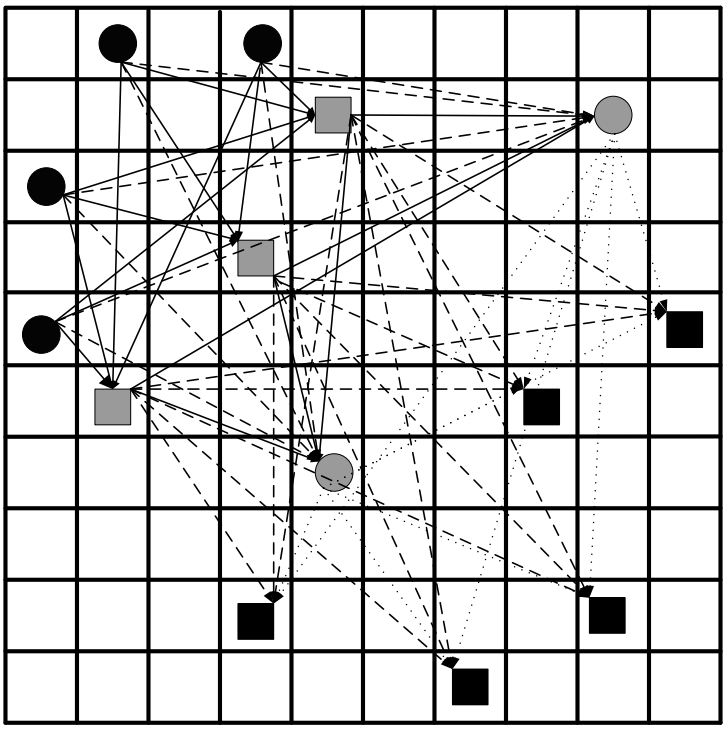

Figure 5-2: Learning vectorial sequences in a network composed of 100 clusters by the generalized chain of tournaments. Clusters are represented by squares in the grid. Four patterns with different sizes are represented: filled circles, grey rectangles, grey circles and filled rectangles. The incident degree is $r = 2$.

activities in a limited number of distributed clusters, very small compared to the total number of clusters. Four sparse patterns with different sizes are represented: filled circles (size 4), grey rectangles (size 3), grey circles (size 2) and filled rectangles (size 5). The elements of a single pattern are not interconnected. On the other hand, two patterns that are linked are associated through an oriented complete bipartite graph. The succession of patterns is then carried by a chain of tournaments with parameter r. In Fig. 5-2, we have $r = 2$, since the first pattern (filled circles) is connected (via solid arrows) to the second (grey rectangles) and to the third (grey circles, via dotted arrows), but not to the fourth (filled rectangles). This connectivity structure is similar to the sparse, distributed model of Rinkus [Rin04], although the latter only considers connections between two consecutive patterns, and the way of the organization in clusters is different.

5.3.1 Global decoding schemes

As usual, to retrieve the whole sequence, the network should be provided at least with a cue sequence of any r successive patterns (a cue sequence of less than r patterns makes the retrieval task not optimal but still possible), not necessarily at the beginning. The actual state of the network, denoted by $\widetilde{\Phi}_\tau$, is defined by the collection of the fanals that constitute r estimated successive patterns at the time τ:

$$\widetilde{\Phi}_\tau = \bigcup_{t=\tau}^{\tau+r} \tilde{p}_t \text{ , with } \tilde{p}_t = \{n_{ij}^t, n_{i'j'}^t, ...\}. \tag{5.3}$$

All the fanals in $\widetilde{\Phi}_\tau$ are activated:

$$\forall n_{ij} \in \widetilde{\Phi}_\tau, v(n_{ij}) = 1 \text{ , otherwise } v(n_{ij}) = 0. \tag{5.4}$$

A step of message passing is then done for all the fanals in the network:

$$\forall i \in [|1, \chi|], j \in [|1, l|], v(n_{ij}) \leftarrow \sum_{i'=1}^{\chi} \sum_{j'=1}^{l} w_{(i,j)(i',j')} v(n_{i'j'}). \tag{5.5}$$

Two differences compared to the previously presented message passing deserve being pointed out. Firstly, since the r successive patterns potentially use some same clusters, a cluster is authorized here to contribute more than 1 when calculating scores. SoS rule should be applied, instead of SoM or NORM rules. Secondly, the memory effect γ is omitted, since here we are in the case of hetero-association rather than that of auto-association.

Unlike in looped tournament-based networks (cf. Fig. 4-1 and Fig. 5-1), a priori, the placement of the target pattern is unknown. As a consequence, one has to process a *global* selection of winners rule instead of a local selection such as (2.3). Indeed, the local WTA leaves automatically residues in every clusters.

Several strategies are possible for a global selection. Three of them are presented here with the example where the set of 10 fanal scores is $\{5, 6, 1, 8, 7, 7, 8, 5, 0, 8\}$ after the message passing step.

1. Threshold Selection (TS): only the fanals with a score superior or equal to a predefined threshold are selected. For example, if $\sigma = 6$, the next estimated pattern is $\{2, 4, 5, 6, 7, 10\}$;

2. Global Winner-Takes-All: only the fanals with the maximal score are selected. The next estimated pattern is $\{4, 7, 10\}$;

3. Global Winners-Take-All: the Σ fanals with the maximal or near maximal scores are selected. If several fanals has the same score as the the Σ^{th} does, these fanals are selected as well. For example, if $\Sigma = 4$, the next estimated pattern is $\{4, 5, 6, 7, 10\}$. In this case, the number of selected fanals can be slightly greater than the targeted number Σ.

GWTA and GWsTA are exactly identical to those used for clique retrieval in Chapter 2. Note that LsKO is not suitable here, since finding a triggered pattern in sequence retrieving is no longer a problem of finding maximal cliques in a graph.

TS is simple and efficient when the size c of each pattern in sequences is identical and any of r successive patterns do not overlap. In this case, one can simply apply

a threshold $\sigma = rc$ to reach good retrieval performance. However, the performance of TS is very sensitive to the chosen threshold. In a general case, if we denote $\Phi_\tau = \bigcup_{t=\tau}^{\tau+r} p_t$, the optimal σ_τ is then the cardinality of the intersection $\Phi_\tau \cap \widetilde{\Phi}_\tau$, denoted by $|\Phi_\tau \cap \widetilde{\Phi}_\tau|$. This value represents in fact the quantity of available useful information to trigger the following pattern. This makes it not applicable to sequences with variable sized patterns if one does not know beforehand the size of previous patterns. In a similar manner, the choice of parameter Σ in GWsTA is dependent on the size of the target pattern, the optimal choice being $\Sigma = c$. Nevertheless, compared to TS, GWsTA is much more flexible concerning variable sized patterns. Setting the value of Σ equal to the smallest size $c_{\min}$ is generally good enough, if the size of patterns does not vary very much and $c_{\min}$ is not too small. On the other hand, GWTA is totally independent of the pattern size. The counter part is that a fanal obtaining erroneously the highest score will extinguish all the rest.

The network state is then updated by replacing the oldest pattern in Φ with the most recently decoded pattern. The process since (5.3) repeats itself for the following decoding step, and so on.

Fig. 5-3 compares in terms of PRER different decoding schemes: TS, GWTA and GWsTA. The simulated network is composed of 100 clusters of 64 fanals each. The length of the sequences to store is 100. Two cases are simulated: all patterns contain 20 fanals; or the pattern size varies uniformly from 10 to 20. The parameters: σ for TS and Σ for GWsTA are optimized. TS and GWsTA have almost equivalent performance when retrieving sequences of fixed sized patterns, and both outperform GWTA. With variable sized patterns, GWsTA with optimally chosen Σ outperforms GWTA once again, while TS has very poor performance in this case. In overall, GWsTA is generally the best decoding scheme in terms of PRER, while GWTA is more flexible, thus is a good tradeoff if the sizes of patterns vary considerably or are completely unknown.

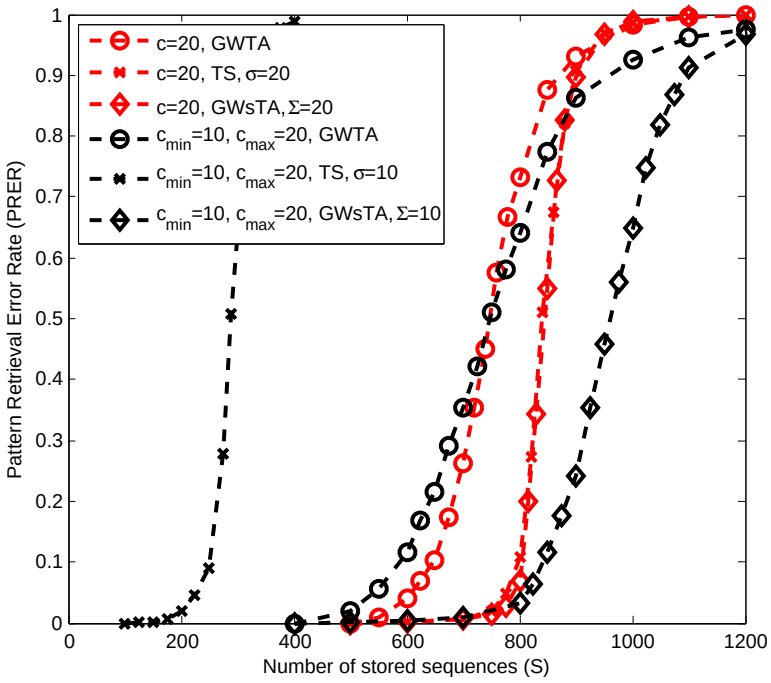

Figure 5-3: Comparison of PRER in a single layer structure by applying different decoding schemes: TS, GWTA and GWsTA. The network is composed of 100 clusters of 64 fanals. The length of each sequence to store is 100. Two cases are simulated: all patterns contain 20 fanals; or the pattern size varies uniformly from 10 to 20. The parameters: σ for TS and Σ for GWsTA are optimized.

5.3.2 Cluster activity restriction

The decoding in 5.3.1 could be problematic if a fanal is shared by two or more patterns within the range of r. In this case, the optimal threshold rc in TS is overestimated, which leads to inhibiting fanals supposed to be activated.

The number of sequences of big patterns that the network is able to store is more subject to the density, whereas that for small patterns is more subject to the incident degree. Sequences of small patterns can be better retrieved by increasing r, but the suitable range of r is very limited. The reason is illustrated by Fig. 5-4. Actually 3 patterns represented in the lattice are activated, each pattern comprising c fanals. The target pattern $p_{\tau+3}$ will be correctly recovered, since it receives contributions from the pattern p_τ, $p_{\tau+1}$ and $p_{\tau+2}$. But elements of $p_{\tau+2}$ will be potentially activated as well if some connections (represented by the dotted arrow) exist. The Hamming

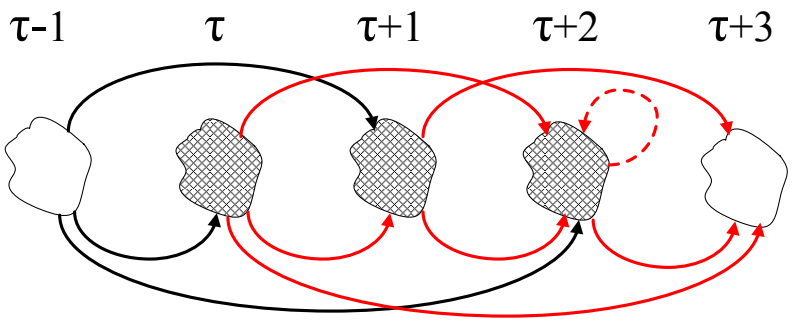

Figure 5-4: Illustration of the decoding issue with sequences of small patterns and $r > 1$.

distance between the set of elements that triggers $p_{\tau+2}$ and the one that triggers $p_{\tau+3}$ is at best c instead of rc as expected.

To avoid this interference issue, one trick is to set restriction on the cluster activity: the clusters involved in a pattern are not allowed to participate in the activities of the r following ones. In this case, it is possible to estimate SQRER. At each decoding step, we estimate $\tilde{p}_{\tau+r+1}$ from the r previous estimations $\{\tilde{p}_t, t \in [|\tau, \tau + r|]\}$. Supposing the previous estimations are correct, $\forall t \in [|\tau, \tau + r|], \tilde{p}_t = p_t$. Three groups of fanals should be distinguished for error probability estimation:

1. All the fanals in the clusters involved in $\{p_t, t \in [|\tau, \tau + r|]\}$, the number of which is rl. The probability of these fanals to be incorrectly stimulated is zero. In fact, cluster activity restriction forbids connections within the same cluster, that makes the scores of these fanals always inferior to rc.

2. The fanals involved in the following patterns $\{p_t, t \in [|\tau+r+2, \tau+2r+1|]\}$, the number of which is $(r-1)c$. These fanals receive deterministically stimuli from part of currently activated patterns. For example, the scores of the fanals in $p_{\tau+r+2}$ reach at least $(r-1)c$, since they are connected to $\{p_t, t \in [|\tau+1, \tau+r|]\}$. As a consequence, c spurious connections, the probability of each of them being d, are enough to lead to a wrong decision. The probability of these $(r-1)c$ fanals not to be incorrectly stimulated is thus $\prod_{i=1}^{r-1}(1-d^{ic})^c$.

3. All the other fanals except those in the target pattern $p_{\tau+r+1}$, the number of

which is $n - rl - rc$. The probability of these fanals not to be incorrectly stimulated is $(1 - d^{rc})^{n-rl-rc}$.

This estimation repeats $L - r$ times if r patterns at the beginning of the sequence is provided. Finally, SQRER is estimated as

$$P_e = 1 - \left[(1 - d^{rc})^{n-rl-rc} \prod_{i=1}^{r-1} \left(1 - d^{ic}\right)^c \right]^{L-r} \tag{5.6}$$

with

$$d \approx 1 - \left(1 - \frac{rc^2}{n^2}\right)^{SL}. \tag{5.7}$$

From Fig. 5-5, one could observe the optimal choice of r for different sizes c, leading to the lowest SQRER. Generally speaking, it is the product of r and c that determines the system retrieval performance. For example, in the network composed

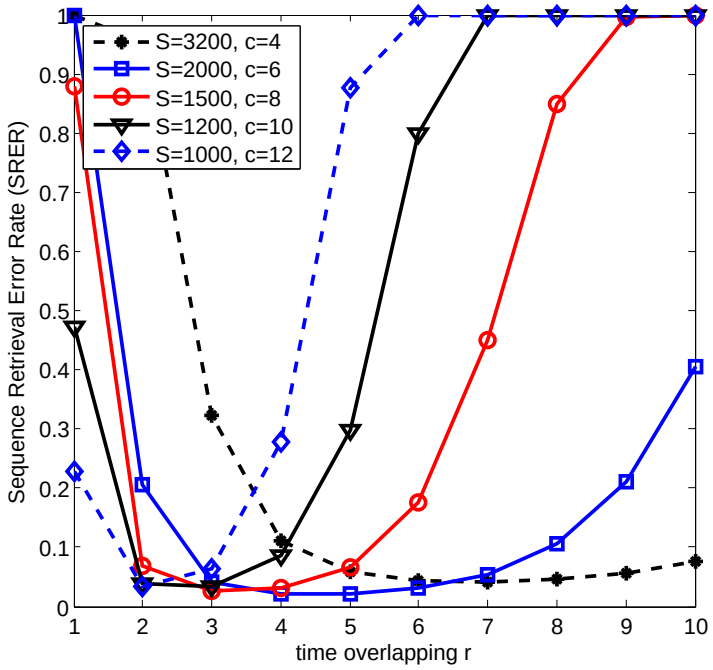

Figure 5-5: Theoritical SQRER in the single layered network in function of the time overlapping r. The network is composed of 100 clusters of 64 fanals. The length of each sequence to store is 100. The curves correspond to sequences of patterns of different sizes c: 4, 6, 8, 10 and 12. The number of sequences to store S is adjusted for each c in the way that the optimal choice of r leads to a SQRER near 1%.

of 100 clusters of 64 fanals storing sequences of length 100, the optimal value of this product is near 30. Indeed, the product rc corresponds to the quantity of available input information for the next pattern decoding. For sequences composed of large sized patterns, only few previous instant information is needed to decode the next pattern, while for small sized patterns more temporally distant information could be useful. However, an augmentation of either c or r will increase network density. A tradeoff should be made between the quantity of available input information and the density.

Fig. 5-6 shows that the unlooped structure with cluster activity restriction outperforms the looped chain of tournaments depicted in Section 5.2 in terms of retrieval error rate. The more is the number of subnetworks, the more the retrieval tasks will be difficult. As a matter of fact, even if the number of possible candidates for each decoding step decreases linearly with the number of subnetworks, the density increases correspondingly. However, the looped chain of tournaments is one of the solutions

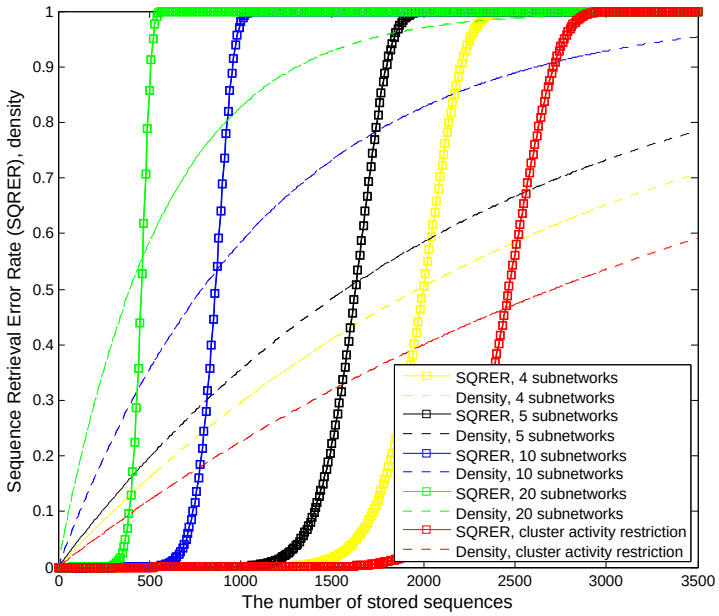

Figure 5-6: Comparison between looped and unlooped structure. The network is composed of 100 clusters of 64 fanals. The length of each sequence to store is 100. The size of each pattern is 6. The time overlapping is $r = 3$. For the looped structure, the curves are drawn for $\alpha = 4, 5, 10$ and 20. In the unlooped structure, the cluster activity restriction is applied.

to resolve the interference issue with complex sequences, which will be depicted in Section 5.5.2.

5.4 Double layered structure

5.4.1 Error avalanche issue

Up to the current subsection, the decoding operation in tournament-based networks can not benefit from iterative processing. Generally speaking in the neural network, an overall sequence is encoded by connecting the group of neurons that represents memory A to those that represents memory B, which then connects to memory C, etc. The problem of this simple chaining process is that an error in the memory A will potentially fire neurons that do not take part of memory B, while some part of memory B will not be fired. B is transformed to a corrupted version B'. Therefore, the retrieval of memory C will be even more challenging and more subjected to errors. We call this issue "avalanche of errors". We have discussed the similar issue in Section 4.3 with the decoding of sequence of symbols. Our solution was to guarantee a sufficient number of r in order to assure an error tolerant decoding, to some extent. One may want to introduce an iterative decoding as well into sequential retrieval that enables to correct such transitory errors.

Some studies [Kle86] [SK86] suggested theoretical solutions to the error avalanche problem, which are based on the error correction properties of auto-associative networks. Here, auto-association refers to the strengthening of synaptic connections between cells encoding the same memory. It was proposed that the error avalanche problem could be avoided if every hetero-associative step in the chaining process is followed by an auto-associative step, which potentially enables to transform a corrupted memory to the accurate version. But in all these models, synaptic connections are weighted, which is different from our model.

5.4.2 Structure

In a similar approach, we propose a double layered structure as depicted in Fig. 5-7. The lower layer is *tournament-based hetero-associative layer*, identical to the single layered network which stores the sequential associations between patterns. An upper layer of mirror fanals is superimposed to emphasize the co-occurrence of the elements belonging to the same pattern, in the form of a clique. Therefore, this second layer is called *clique-based auto-associative layer*, similar to the sparse clique-based networks.

Generally speaking, all that we have proposed in Chapter 2 for clique-based sparse networks are valid here. In particular, with the help of some iterative decoding algorithms such as GWsTA and LsKO, the clique-based auto-associative layer should be able to remove the insertions which are potentially activated because of the spurious connections in the tournament-based layer. For instance in Fig. 5-7, the upstream pattern represented by 4 circles triggers the downstream pattern represented by 4 squares, but also an insertion, the filled square. A structure with only a single layered network will in this case continue on decoding the following patterns, with the

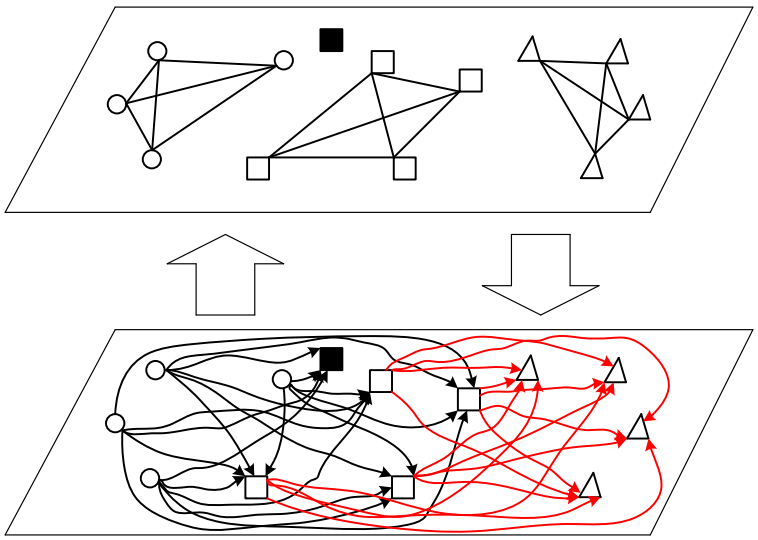

Figure 5-7: Double layered structure. Upper: clique-based auto-associative layer for sparse pattern learning and decoding; lower: tournament-based hetero-associative layer for sequential learning and decoding. The interaction between these two layers assures accurate retrieval.

risk of amplifying the error. On the contrary, the filled square does not take part of the clique in the clique-based layer, therefore will hopefully turn to be silent after several iterations. The accurate information is then retransmitted back to the tournament-based hetero-associative layer to pursue the sequential decoding.

Fig. 5-8 shows the performance gain of the double layered structure compared to the single layered structure. The simulated network is composed of 100 clusters of 64 fanals. The length of each sequence to store is 100. Two cases are simulated: all patterns contain 20 fanals; or the pattern size varies uniformly from 10 to 20. Four iterations of GWsTA with $\gamma = 1000$ are applied in clique-based layer , whereas one step of GWsTA is applied in tournament-based layer. Σ is optimized for both layers.

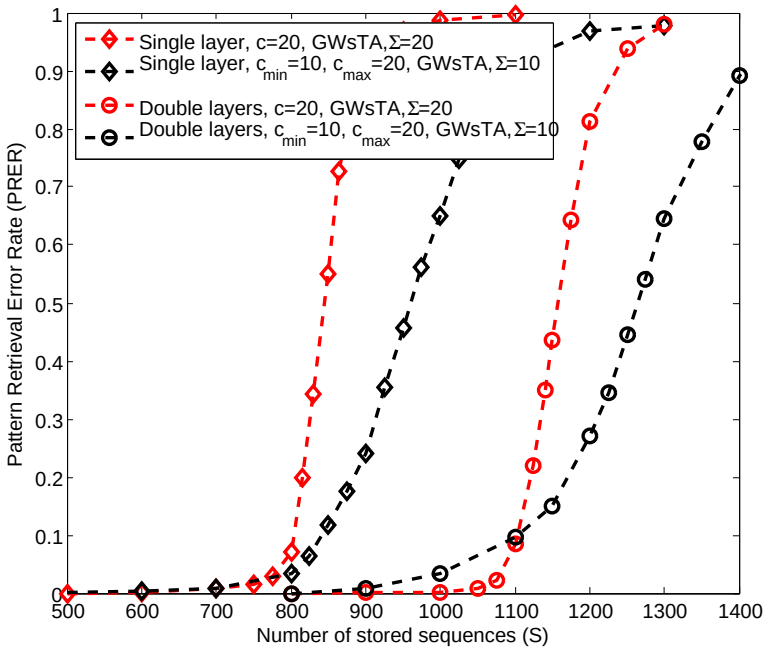

Figure 5-8: Comparison of the single layered structure and the double layered structure in terms of PRER. The network is composed of 100 clusters of 64 fanals. The length of each sequence to store is 100. Two cases are simulated: all patterns contain 20 fanals; or the pattern size varies uniformly from 10 to 20. Decoding scheme in clique-based layer: GWsTA with 4 iterations applied and $\gamma = 1000$. Decoding scheme in tournament-based layer: GWsTA. Σ is optimized for both layers. The number of required connections is doubled for the double layered structure.

5.4.3 Biological consideration

Recent studies [LTR05] have suggested that in the hippocampus the dentate and CA3 networks reciprocally cooperate for accurate recall of memory sequences. Auto-association occurs in CA3 to accentuate the simultaneity of elements in the same memory, whereas hetero-association occurs in the dentate to progress in the sequential decoding. Several anatomical and electrophysiological observations point out that dentate granule cells have axons called mossy fibers, which powerfully excite CA3 cells. And there is also a feedback information pathway from CA3 back to the dentate [Lis99].

The double layered structure with duplication of neurons may appear irrelevant in terms of biological plausibility. Albeit mirror neurons as well as mirror systems have been directly observed in primate and other species, and it is speculated that this system provides the physiological mechanism for the perception/action coupling, it is not obvious at all that such system exists also in cerebral memory mechanisms. To further approach to the biological reality, one may consider this structure as a single layer of unique neurons connected by relatively long and short synapses. The memory at one instant is represented by a pattern locally embedded in terms of a clique. The information is exchanged rapidly between active neurons via short auto-associative synapses until the neural pattern reaches a stable representation. Then this information is distributedly spread by long hetero-associative synapses over the network to emerge the following local pattern, etc. From an implementation point of view, these two types of connections can be distinguished in practice.

5.5 Interference issue with a limited dictionary

5.5.1 Interference issue

Until the current subsection, the cardinality of the dictionary we use is gigantic compared to the reality. For example, if we consider a dictionary of all the words with 4 letters, with the discussed network composed of 100 clusters of 64 fanals

each, the cardinality of the dictionary, i.e. the number of all the possible words is, $\binom{100}{4}64^4 \approx 6.6 \times 10^{13}$. To give some numbers in the real world: the Oxford English Dictionary contained approximately 301,100 main entries; and the Chinese language contains up to 90000 characters, while only 7000 of them are commonly used. An extreme example is given by DNA sequences, which are composed of only 4 nucleobases (G, A, T, C), yet rich in information.

In the network as we have introduced, the retrieval is much easier for a sequence of words that are uniformly chosen within a huge dictionary than within a very limited dictionary, let us say, a sequence of English letters for example. In fact, the larger the dictionary is, the more unlikely it is that a word or succession of words appears several times with a sequence, and there will be less interference present. We take the example as below to illustrate this issue:

A-B-C-D-B-E-C-B-...

Each letter in this sequence can be represented in the graph by a unique vertex, or a set of vertices, i.e. a spatial pattern. In the latter case, we assume that these spatial patterns are completely disjoint (two patterns do not share any vertices) in order to simplify the analysis. The figure on the top of Fig. 5-9 illustrates the case where this sequence is stored with $r = 1$, i.e. a letter is connected only to its direct neighbor during the storage process, and accordingly during the retrieval process the decision is made only given the letter that precedes. The letter A will prime B, since it is the only connection that starts from A. But at the next step, since the network makes its decision only based on the letter B, E will interfere with C. Given the network construction, these two are equiprobable. Either the network decides to take a random choice (it is equivalent to say, taking a random path on the high left in Fig. 5-9, (For instance, A-B-C-B-C-... is a possible obtained sequence), or the network conserves both possibilities and then two sequences will propagate in parallel in the network and will interfere with each other. (To see the obtained sequence, it is sufficient to check the top left of Fig. 5-9, and pick up the vertex with the maximum appearance at each column. In this case, the retrieved sequence will be A-B-CE-BCD-B-...)

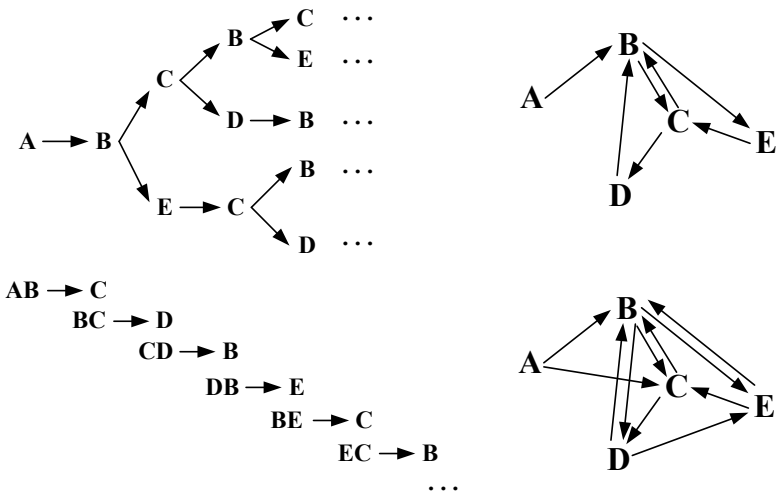

Figure 5-9: Interference issue for decoding the sequence A-B-C-D-B-E-C-B-... Top: situation where $r = 1$; Bottom: situation where $r = 2$; Left: decoding results; Right: network connections after the storage.

A simple and naive solution is to increase the time overlapping r. For example, in Fig. 5-9 a time overlapping $r = 2$ allows resolving this interference issue. With A and B activated, C ($v_C = 2$) is the only choice, the score of other vertices being: $v_A = 0, v_B = 1, v_D = 1$ and $v_E = 1$. At the next step, with B and C activated, D ($v_D = 2$) is the only choice, since $v_A = 0, v_B = 1, v_C = 1$ and $v_E = 1$, etc. In fact, the retrieval process will certainly fail if one sets a time overlapping r inferior to the degree of the sequence to store. This issue is due to the nature of sequence, but not due to our network structure.

However, the way that connections are established in our network makes the reverse proposition not true: a time overlapping r superior or equal to the degree of the sequence would not necessarily guarantee the retrieval success. For instance, the degree of the sequence A-B-C-B-D-C-E-C-D is 2, but the interference issue still exists even with $r = 2$ (cf. Fig. 5-10). The activation of A and B at the same time will not only stimulate C, but also B, since B is auto-connected to itself due to the subsequence B-C-B. Nevertheless, the subsequence A-B appears only once in the sequence. Another source of error is that the decoding algorithm does not make any difference between different pattern orders within a subsequence that triggers the

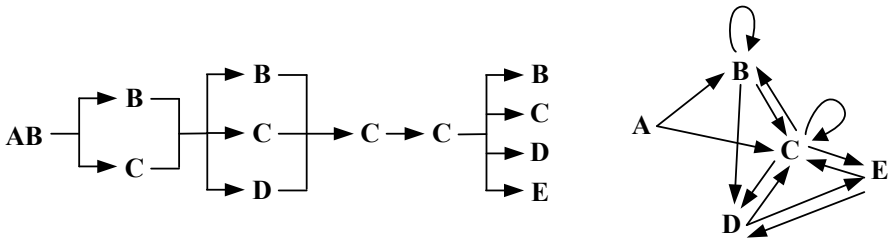

Figure 5-10: Interference issue for decoding the sequence A-B-C-B-D-C-E-C-D for $r = 2$. Left: decoding results; right: network connections after the storage.

next pattern. For instance, A-B-C-D-B-A-D will have decoding issue with $r = 2$, since the pattern A and B trigger C and also D.

This suggests that the choice of r that optimizes the decoding algorithm is highly dependent on the nature of sequences to store. In order to correctly configure the network, not only do we have to get the prior information on sequences even before the storage, but it is also required that all the sequences stored should possess the same optimal r.

5.5.2 Multiple solutions

Several solutions are proposed here in this subsection. The main idea is to provide multiple versions of the same pattern, which can differ accordingly to different contexts of this pattern in the sequence.

Segmentation of network

We segment the whole network into several sub-networks in the similar manner described by Fig. 5-1, and a pattern in a sub-network differs from the same pattern in another sub-network. The number of versions of a pattern is equal to the number of sub-networks. The structures in Chapter 4 and Section 5.2 follow this strategy. For instance, let us consider the storage of the sequence A-B-C-A-B-A-D in a network composed of 4 sub-networks with $r = 2$ (Fig. 5-1). The first occurrence of A-B (A in the first subnetwork, and B in the second subnetwork) is embedded differently compared to its second occurrence (A in the fourth sub-network, and B in the first

sub-network). In the same way, the pattern order within a sub-sequence is also fully taken into account: the activation of A-B and B-A represent now different neural activity in the network, and will trigger different patterns.

Association with signature

It is possible to associate a pattern with a random signature that differs at each occurrence of this pattern. The A in the first position of the sequence above will have a different signature from the A in the fourth position. The informational pattern, fixed and unique for each letter, combined with its randomly variable signature forms a larger spatial pattern, and then is connected to the following patterns (informational pattern plus signature as well). The informational pattern and its own signature are not necessarily interconnected. At the retrieval stage, the informational pattern is extracted from the large pattern simply by eliminating the signature part. To make it simple, the network can be constructed such that the sub-network storing real information is totally disjoint from that storing context information.

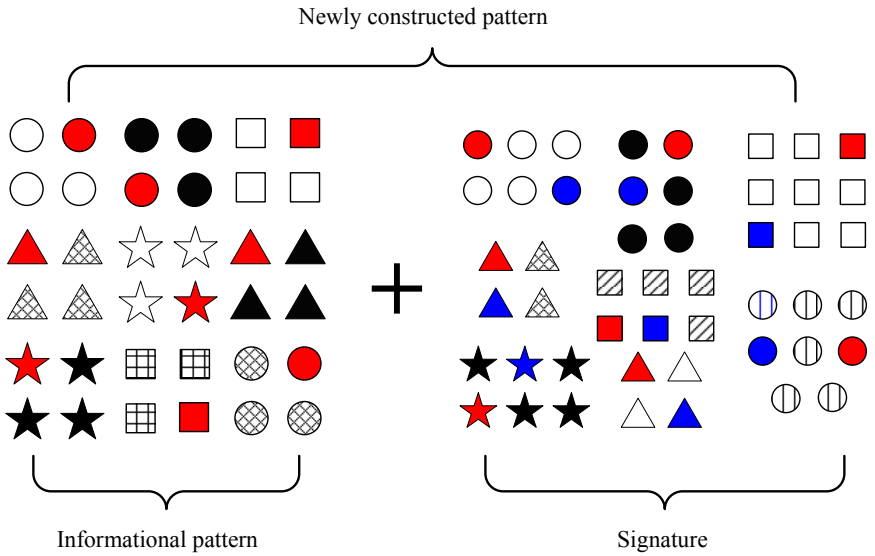

Figure 5-11: Solution 2 to the interference issue: associating a informational pattern with its signature. The signature can differ at each occurrence of this same informational pattern. Two possible versions of signature corresponding to the same informational pattern are represented: one in red and the other in blue.

Cluster enlargement

Another solution consists in increasing the cluster size by multiplying the number of fanals that represents the same symbol. Fig. 5-12 illustrates this principle, where the cluster size is multiplied by 4. The 4 fanals encircled are equivalent in terms of representation. At each occurrence of a pattern, one fanal among these 4 is randomly chosen to construct a new version of this pattern. On the left represents the original pattern A. On the right two versions A' and A" are represented respectively in red and in blue. The number of possible versions of a pattern is increasing exponentially with respect to the multiplication ratio β: c^{β}. It is possible that different versions overlap with each other.

It is important to note that multiplication of fanals can be carried out dynamically during the storage process. If the number of connections that involves a fanal surpasses a predefined threshold, a new fanal that has an equivalent representation will be created. This is especially interesting for storage of non-i.i.d patterns, which

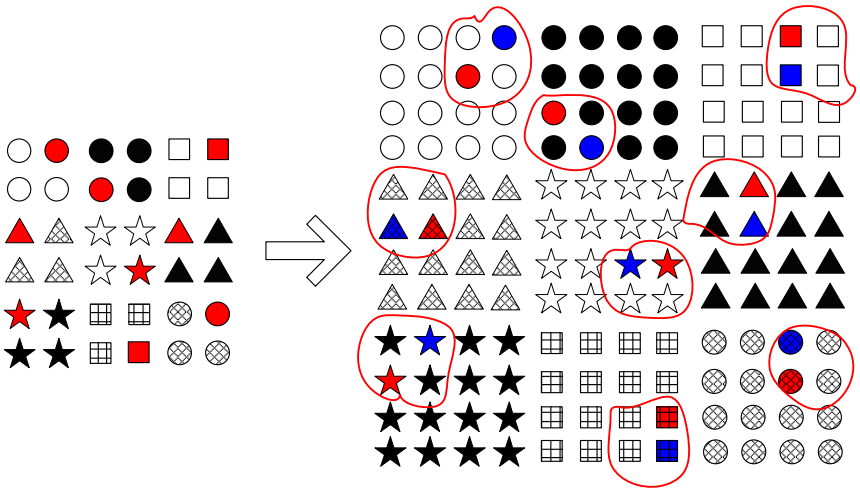

Figure 5-12: Solution 3 to the interference issue: increasing the cluster size by multiplying the number of fanals that represents the same symbol. The 4 fanals encircled are equivalent in terms of representation. At each occurrence of a pattern, one fanal among these 4 is randomly chosen to construct a new version of this pattern. On the left represents the original pattern A. On the right two versions A' and A" are represented respectively in red and in blue. It is possible that different versions of A overlap with each other.

will avoid from inefficiently creating fanals that are rarely sollicited in a sequence.

Subsampling

All the three solutions above require supplementary resources: Solution 1 requires multiplying the number of clusters, whereas Solution 3 requires increasing the cluster size. Solution 2 needs a supplementary part of the network dedicated to context information. The method of subsampling consists in choosing randomly f_s fanals among c that compose the original pattern to form a new subsampled version. As a consequence, the method of subsampling enables to provide multiple versions (the number is $\binom{n}{f_s}$) without consuming extra resource.

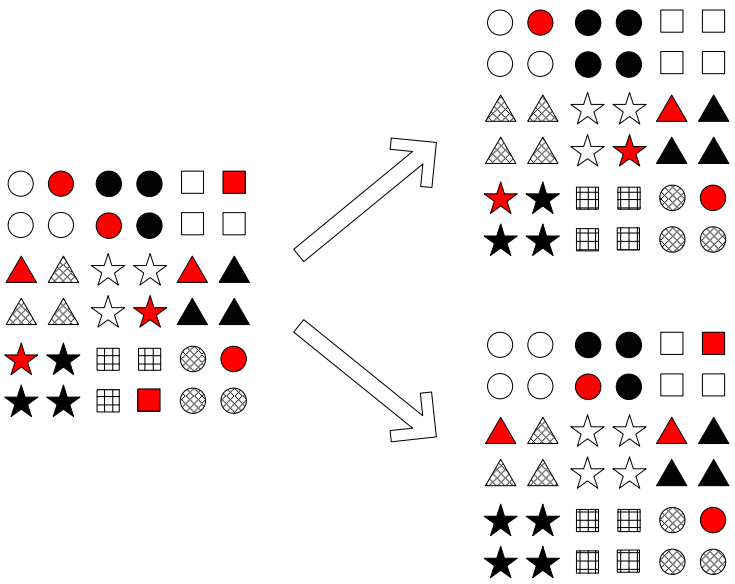

Figure 5-13: Solution 4 to the interference issue: subsampling the original pattern. The figure on the left represents the original pattern A. The figure on the right represents two subsampled versions A' and A". It is possible that different subsampled versions of A overlap with each other.

5.6 Hierarchical structure

The solution of subsampling may be combined with the previously proposed double layered structure. The complex sequence is not directly encoded by chains of tourna-

ments. As described in Section 5.5.2, a subsampling step makes it possible to decrease considerably the degree of the sequence. In order to maximize the performance, for a pattern that appears multiple times, their different subsampled versions are preferable to be less overlapped as possible. This sequence composed of subsampled patterns is then stored in the classical way on the tournament-based layer. On parallel, the complete patterns without subsampling are stored on the clique-based layer. During the retrieval process, a subsampled pattern is at first triggered by previous neural activities on the tournament-based layer. The decoder then hands over the upper layer's decision to the lower layer, where the complete pattern is retrieved from the subsampled version. This is equivalent to retrieve from a partially erased pattern, possibly with errors and insertions. This principle is illustrated in Fig. 5-14.

Unlike in the previously introduced double layered structure (cf. Fig. 5-7), the motivation of which is to resolve error avalanche issue, it is important to note that the feedback interaction from the clique-based layer to the tournament-based layer is no longer adapted. As a matter of fact, the complete clique in the lower layer contains fanals that are not encoded in the upper layer. Thus, if a feedback is carried out,

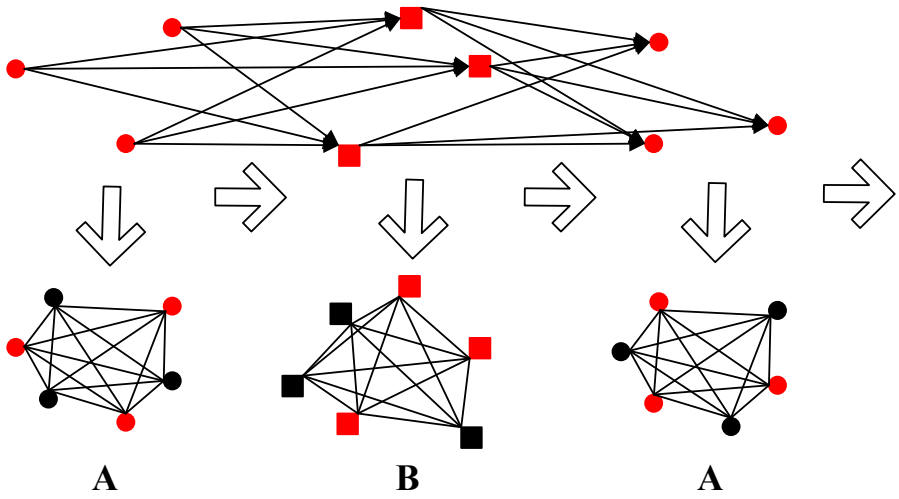

Figure 5-14: Two layered hierarchical structure. Bottom: clique-based autoassociative layer for sparse pattern storing and decoding; top: tournament-based heteroassociative layer for sequential storing and decoding. The patterns in the clique-based layer are subsampled versions of those in the tournament-based layer. Interaction between these two layers is top-down.

the upper layer decoder would find a corrupted pattern with insertions at the input. Moreover, this complete pattern contains by construction all its subsampled versions that have been encoded in the upper layer, as a consequence, all the patterns that are linked to them are likely to be triggered at the next decoding step.

5.6.1 Performance analysis

Fig. 5-15 compares retrieval performance in hierarchical structure with respect to different subsampling rates. The simulated network is composed of $\chi = 100$ clusters of $l = 64$ fanals each. Each sequences to store is of length $L = 100$ and the size of each pattern is $c = 20$. The subsampling rate is 25%, 40%, 50% and 65%, respectively. The time overlapping r is 1. The number of iterations executed in the clique-based layer is 4. It is shown that there exists an optimal subsampling rate in order to maximize the retrieval performance in terms of the diversity. In the above simulated case, the best choice of the subsampling rate is approximately 40%, i.e. 8 fanals out of 20 constitutes the pattern in the sequence layer. As a matter of fact, the subsampling rate can impact on the retrieval performance in the four following aspects:

1. The higher the subsampling rate is, bigger the pattern size, thus much the quantity of information available to trigger the following patterns, and finally better the retrieval performance.

2. The subsampling rate will impact on the diversity of each word. In particular, a subsampling rate of 50% corresponds to the maximum number of versions of a single word, that is $\binom{c}{c/2}$.

3. The lower the subsampling rate is, lower the network density will be in the tournament-based layer, thus better the retrieval performance.

4. The higher the subsampling rate is, more information will be handed over to the clique-based layer, thus better the retrieval performance.

Overall, the optimal subsampling rate should be set such that a tradeoff between all these four aspects are found.

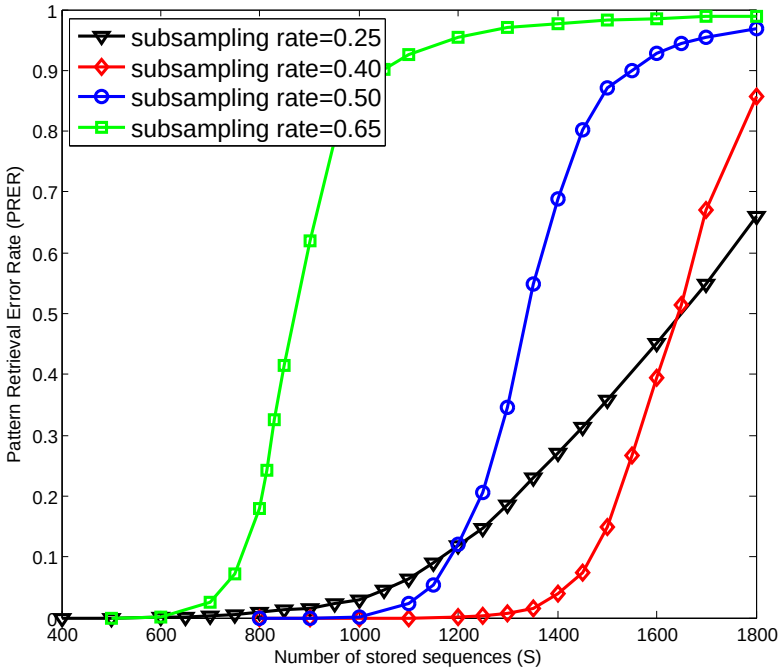

Figure 5-15: Comparison of retrieval performance with respect to subsampling rate. The simulated network is composed of $\chi = 100$ clusters of $l = 64$ fanals. Each sequence to store is of length $L = 100$ and the size of each pattern is $c = 20$. The subsampling rate is 25%, 40%, 50% and 65%, respectively. The time overlapping r is 1. The number of iterations executed in the clique-based layer is 4.

Fig. 5-16 shows optimal parameter settings (subsampling rate together with time overlapping) that maximize the learning diversity, as a function of different size of the dictionary. For a dictionary of very few words, for example, 100 words of size 20 uniformly distributed in a network of 100 clusters of 64 fanals each, words are very likely disjoint, that is, two words do not share any letters. In this case, it is sufficient to provide only one letter to the clique-based layer to retrieve the entire word. The retrieval process becomes similar to a simple lookup table. For the tournament-based layer, we lose incident degree because of the extremely low subsampling rate, so it should be made up by increasing the time overlapping r. On the opposite, when the size of the dictionary increases, the optimal subsampling rate increases as well, and the time overlapping is reduced to 1.

We conclude that, for a sequence that contains some frequently repeated words, or frequently repeated sub-sequences, the temporal dependence can be better exploited

by the decoder than the spatial one. On the other hand, for a sequence that contains distinct words, spatial dimension is then more important than the temporal one.

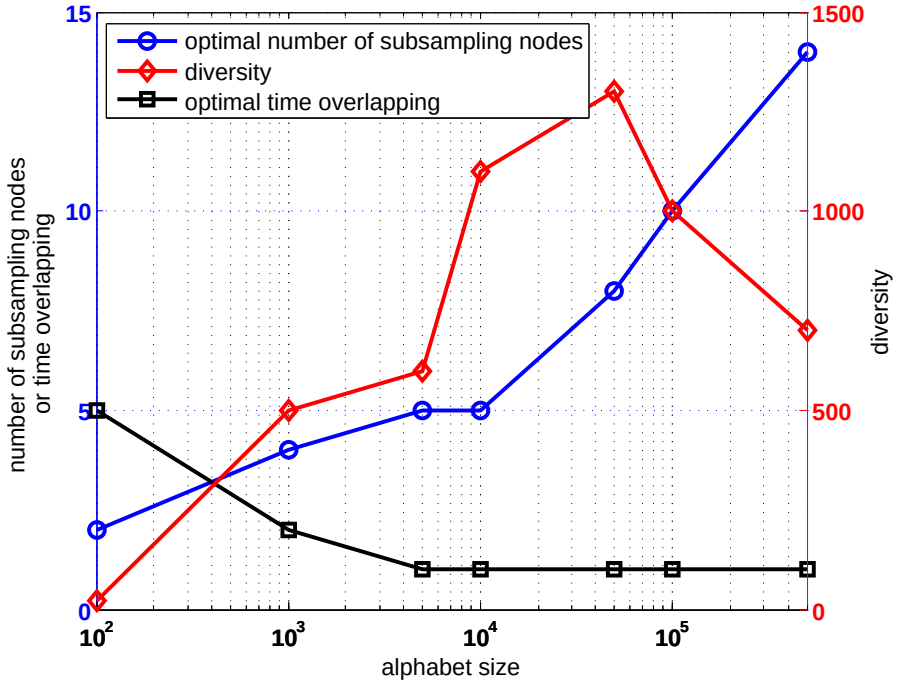

Figure 5-16: Optimal parameter settings for the subsampling rate and time overlapping maximizing the learning diversity. The simulated network is composed of $\chi = 100$ clusters of $l = 64$ fanals. Each sequence to store is of length $L = 100$ and the size of each pattern is $c = 20$. The number of iterations executed in the clique-based layer is 4. The size of the dictionary is varied from 100 to 10^6.

Chapter 6

Conclusion and open problems

6.1 Obtained results

We have proposed some new decoding algorithms in clique-based neural networks and have evaluated their performance. They are fully compatible with classical GB networks. Moreover, in a more generalized structure where the storage and retrieval of a message only involve a small part of the network, those algorithms outperform significantly the originally proposed algorithm, that is, GWTA. We have also extended the range of associative memories in the clique-based model. The network is now able to deal with blurred messages containing approximate information, also messages with binary errors and messages with partial erasures in some fractal-divided cliques, etc. We have shown that the algorithms we have implemented are fully suited to correct these degraded pieces of information.

Then we studied how to reduce smartly redundancy in the structure of cliques in order to maximize information efficiency. We have found that chain of cliques is the most performance-wise choice because of its homogenous connectivity. On the other hand, for the concern of plausibility, bidirectional connections are replaced with unidirectional ones to transform "chains of cliques" into "chains of tournaments". The latter also represents the half of redundancy of the former. We have assessed the performance of chains of tournaments in the application of associative memory, and we have proven that the optimal incident degree r is linked to the logarithm of the

number of fanals per cluster.

We have shown that it is possible for chains of tournaments to store sequential messages, the length of which is variable and only limited by the available neural resource. The key has been the periodic reuse of resource. The decoding in such a structure is error tolerant and anticipatory, which echoes to some biological literature. The efficiency tends asymptotically to 69% when the number of fanals per cluster l tends to infinity, but the divergence is very slow. For a reasonably sized network with the concern for biological plausibility, the efficiency is around 20% to 30%. Also, the optimal configuration that maximizes efficiency corresponds to a network density of 0.5, that implies maximum of entropy. Furthermore, a general framework to store vectorial sequences has been proposed. Single-layered structures with looped or unlooped chains of sparse distributed patterns were proposed and evaluated. It has been shown that these models are of large learning diversity, which could give them the ability to encode the data flows with voluminous information, such as multimedia streams. For example, the network composed of 6400 nodes ($\chi = 100$, $l = 64$) is able to store about $S = 700$ vectorial sequences composed of $L = 100$ patterns of $c = 20$ parallel symbols. Each pattern contains $c \log l + \log_2 \binom{\chi}{c} \approx 188.9$ bits, which corresponds to a total amount of information of approximately 13.2 Mbits.

Finally, we proposed a double layered structure combining the hetero-associative tournaments-based network with the auto-associative clique-based network in order to achieve accurate retrieval. The former reflects the sequential process and the latter emphasizes the co-occurrence of the elements belonging to the same sparse pattern. With the same parameters as above, the network is able to store $S = 1050$ sequences, that is approximately 19.8 Mbits. Nevertheless, the requested number of connections is doubled. Furthermore, combining this model with the subsampling method allows us to extend it to a hierarchical structure, which makes it able to resolve the interference issue with complex sequences. We have shown that the subsampling rate is subject to a tradeoff between different factors. In given situations, the optimal subsampling rates and the corresponding parameters of time overlapping are found via means of simulations.

6.2 Perspectives and open problems

Like most of today's artificial neural networks, our model functions in a synchronous mode. The time involved in our model is discrete. This is due to the binary property of the neuron model that we use. However, in biological neural circuits, there is no central clock. To bring our model closer to biological reality, the spiking neuron model should be considered.

Generally speaking, time is embodied in a temporal message in two ways: temporal order and time duration. By now, only the former aspect has been considered in this work. And this can be regarded as a consequence of the point mentioned in the previous paragraph. It would be thus interesting to introduce the notion of duration in associative memories, which would have utilities to applications such as natural language processing of phoneme sequences.

Our model still possesses some degrees of freedom with respect to several network parameters, in particular threshold θ, which we could make use of in the future. As a matter of fact, in the works we have presented, we often ignored this threshold, which was fixed to zero. However, the threshold plays a primordial role in the correct functioning of the brain. For instance, epilepsy is a common and diverse set of chronic neurological disorders, which result from abnormal, excessive neuronal activity in the brain. Some medications and treatments manage to increase the threshold in order to avoid this excessive activity. In our model, one may imagine implementing a non uniformly space-variant threshold laying over the whole network, which could be different from one cluster to another. The clusters with higher thresholds are thus penalized compared to those with lower ones. At the meantime, the threshold could also vary as long as the sequence progresses or be time-variant, and thus could "guide" the sequence towards appropriate locations.

Moreover, nothing by now enables to switch from one sequence to another to create some sort of "associations" or "discriminations" between sequences. These associations, together with the choice of sequence paths mentioned above, are probably goal-oriented. Here once again, a variable threshold could be useful, since it could

be encoded in the way such that this switching between sequences becomes possible. A speculation could be made that the level of the intelligence would be, at least partially, based on how this kind of switchings or associations are performed in the brain.

Finally, the two-layered hierarchical structure proposed in Chapter 5.6 could be further extended to a multi-layered structure such as [SH07], which describes a hierarchical organization illustrated by a four-level network that isolates letters, words, sentences, and strophes. Particularly, one can imagine a hierarchical network where each level passes a part of subsampled information to the next hierarchically higher level, which then aggregates the received information in the form of super level cliques or chains of tournaments, etc.

Bibliography

[ABGJ13] B. K. Aliabadi, C. Berrou, V. Gripon, and X. Jiang. Storing sparse messages in networks of neural cliques. *IEEE Transactions on neural networks and learning systems*, 2013.

[AGJ13] A. Aboudib, V. Gripon, and X. Jiang. A study of retrieval algorithms of sparse messages in networks of neural cliques. *submitted to Neurocomputing*, 2013.

[AH85] D. Ackley and G. E. Hinton. A learning algorithm for boltzmann machines. *Cognitive Science*, 9(1):147–69, 1985.

[Ali13] B. K. Aliabadi. *Neural networks and sparse information acquisition*. PhD thesis, Télécom Bretagne, June 2013.

[Bad94] A. Baddeley. The magical number seven: still magic after all these years? *Psychological Review*, 101(2):353–56, 1994.

[Bar72] H. B. Barlow. Single units and sensation: a neuron doctrine for perceptual psychology. *Perception*, 1:371–94, 1972.

[Bar03] D. Barber. Learning in spiking neural assemblies. In *In Advances in Neural Information Processing Systems 15*, pages 149–156. MIT Press, 2003.

[BB06] D. S. Bassett and E. Bullmore. Small-world brain networks. *Neuroscientist*, 12(6):512–23, December 2006.

[Ben09] Y. Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009.

[BGD+13] C. Berrou, V. Gripon, O. Dufor, X. Jiang, and B. K. Aliabadi. Codes sur graphes et mémoire cérébrale. *XXIVème Colloque Gretsi*, September 2013.

[BGSH13] B. Boguslawski, V. Gripon, F. Seguin, and F. Heitzmann. Storing non-uniformly distributed messages in networks of neural cliques. *submitted to Neurocomputing*, 2013.

[BS09] E. Bullmore and O. Sporns. Complex brain networks: graph theoretical analysis of structural and functional systems. *Nature Reviews — Neuroscience*, 10:186–98, March 2009.

[BSP11] J. Brea, W. Senn, and J. Pfister. Sequence learning with hidden units in spiking neural networks. In *Proceeding of NIPS*, pages 1422–30, 2011.

[CBP+05] L. Cruz, S. V. Buldyrev, S. Peng, D. L. Roe, B. Urbanc, H. E. Stanley, and D. L. Rosene. A statistically based density map method for identification and quantification of regional differences in microcolumnarity in the monkey brain. *Journal of Neuroscience Methods*, 141(2):321–32, May 2005.

[CGL92] R. L. Coultrip, R. H. Granger, and G. Lynch. A cortical model of winner-take- all competition via lateral inhibition. *Neural Networks*, 5:47–54, 1992.

[Cow00] N. Cowan. The magical number 4 in short-term memory: a reconsideration of mental storage capacity. *Behavioral and Brain Sciences*, 24:87–185, 2000.

[Dra05] D. A. Drachman. Do we have brain to spare? *Neurology*, 64(12):2004–05, June 2005.

[EC01] H. Eichenbaum and N. J. Cohen. From conditioning to conscious recollection: Memory systems of the the brain. 2001.

[Eit99] G. Eitan. *Backtracking algorithms*, chapter 19. Data structures. 1999.

[FÖ3] P. Földiak. *Sparse coding in the primate cortex*, pages 1064–68. The handbook of brain theory and neural networks. 2003.

[FB82] J. A. Feldman and D. H. Ballard. Connectionist models and their properties. *Cognitive Science*, 6:205–54, July 1982.

[FP03] K. J. Friston and C. J. Price. Degeneracy and redundancy in cognitive anatomy. *Trends in Cognitive Sciences*, 7(4), April 2003.

[Fre99] R. M. French. Catastrophic forgetting in connectionist networks: causes, consequences and solutions. *Trends in Cognitive Science*, 3(4):128–35, 1999.

[Fus09] J. M. Fuster. Cortex and memory: emergence of a new paradigm. *Journal of Cognitive Neurosciences*, 21:2047–72, 2009.

[FvEBB+05] R. S. Fisher, W. v. E. Boas, W. Blume, C. Elger, P. Genton, P. Lee, and J. Engel. Epileptic seizures and epilepsy: Definitions proposed by the international league against epilepsy (ILAE) and the international bureau for epilepsy (IBE). *Epilepsia*, 46(4):470–72, April 2005.

[GB11a] V. Gripon and C. Berrou. A simple and efficient way to store many messages using neural cliques. *IEEE Symposium Series on Computational Intelligence*, April 2011.

[GB11b] V. Gripon and C. Berrou. Sparse neural networks with large learning diversity. *IEEE Transactions on Neural Networks*, 22(7), July 2011.

[GB12] V. Gripon and C. Berrou. Nearly-optimal associative memories based on distributed constant weight codes. *Proceedings of Information Theory and Applications Workshop*, pages 269–73, February 2012.

[GJ13] V. Gripon and X. Jiang. Mémoires associatives pour observations floues. *XXIVème Colloque Gretsi*, September 2013.

[GMM01] S. Grossberg, W. Maass, and H. Markram. Introduction: Spiking neurons in neuroscience and technology. *Neural Networks*, 14(6):587, 2001.

[Gri11] Vincent Gripon. *Networks of neural cliques*. PhD thesis, Télécom Bretagne, July 2011.

[Gro02] C. G. Gross. Genealogy of the "grandmother cell". *The Neuroscientist*, 8(5):512–18, 2002.

[GRSG12] V. Gripon, M. Rabbat, V. Skachek, and W. J. Gross. Compressing multisets using tries. *Proceedings of Information Theory Workshop*, pages 647–51, September 2012.

[HB90] S. J. Hanson and D. J. Burr. What connectionist models learn: Learning and representation in connectionist networks. *Behavioral and Brain Sciences*, 13:471–489, 9 1990.

[HB00] G. E. Hinton and A. Brown. Spiking Boltzmann machines. *Advances in Neural Information Processing Systems*, 12, 2000.

[Hop82] J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the United States of America*, 79(8):2554–58, 1982.

[JGB12] X. Jiang, V. Gripon, and C. Berrou. Learning long sequences in binary neural networks. *Proc. Cognitive Science*, July 2012.

[JL07] C. Johansson and A. Lansner. Towards cortex sized artificial neural systems. *Neural Networks*, 20:48–61, January 2007.

[Jon00] E. G. Jones. Microcolumns in the cerebral cortex. *Proc. National Academy of Sciences (PNAS)*, 97(10):5019–21, May 2000.

[Kle86] D. Kleinfeld. Sequential state generation by model neural networks. *Proceedings of the National Academy of Sciences of the United States of America*, 83(24):9469–73, December 1986.

[KPS10] A. Knoblauch, G. Palm, and F. T. Sommer. Memory capacities for synaptic and structural plasticity. *Neural Computation*, 22:289–341, 2010.

[LIKB99] D. K. Lee, L. Itti, C. Koch, and J. Braun. Attention activates winner-take- all competition among visual filters. *Nature Neuroscience*, 2(4):375–81, April 1999.

[Lis99] J. E. Lisman. Relating hippocampal circuitry to function: Recall of memory sequences by reciprocal dentate CA3 interactions. *Neuron*, 22:233–42, 1999.

[LLZ+09] A. M. Leaver, J. V. Lare, B. Zielinski, A.R. Halpern, and J. P. Rauschecker. Brain activation during anticipation of sound sequences. *The Journal of Neuroscience*, 29(8):2477–85, February 2009.

[LMZ+12] C. Di Lanzo, L. Marzetti, F. Zappasodi, F. D. V. Fallani, and V. Pizzella. Redundancy as a graph-based index of frequency specific meg functional connectivity. *Computational and Mathematical Methods in Medicine*, 2012, August 2012.

[LTR05] J. E. Lisman, L. M. Talamini, and A. Raffone. Recall of memory sequences by interaction of the dentate and CA3: A revised model of the phase precession. *Neural Networks*, 18:1191–1201, 2005.

[MC89] M. McCloskey and N. J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *The Psychology of Learning and Motivation*, 23:109–64, 1989.

[MHB05] A. Maurer, M. Herschand, and A. G. Billard. Extended hopfield network for sequence learning: Application to gesture recognition. In *Proceedings of the ICANN*, 2005.

[Mil56] G. A. Miller. The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychological Review*, 63:81–97, 1956.

[MP43] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–33, 1943.

[OF04] B. A. Olshausen and D. J. Field. Sparse coding of sensory outputs. *Current Opin. Neurobiol.*, 14:481–87, August 2004.

[PB04] C. A. Perfetti and D. J. Bolger. The brain might read that way. *Scientific Studies of Reading*, 8(3):293–304, July 2004.

[RHM86] D. E. Rumelhart, G.E. Hinton, and J. L. McClelland. *A general framework for parallel distributed processing*, volume 1 of *Parallel distributed processing: Explorations in the microstructure of cognition*. MIT Press, Cambridge, MA, 1986.

[Rin04] G. Rinkus. A neural model of episodic and semantic spatiotemporal memory. *Proc. of the 26th Annual Conference of the Cognitive Science Society*, pages 1155–60, 2004.

[SH07] J. A. Starzyk and H. He. Anticipation-based temporal sequences learning in hierarchical structure. *IEEE Transaction on Neural Networks*, 18(2), March 2007.

[SHT09] I. Sutskever, G. E. Hinton, and G. Taylor. The recurrent temporal restricted boltzmann machine. *Advances in Neural Information Processing Systems*, 21:1601–08, 2009.

[SK86] H. Sompolinsky and I. I. Kanter. Temporal association in asymmetric neural networks. *Physical Review Letters*, 57:2861–64, 1986.

[SP99] F. T. Sommer and G. Palm. Improved bidirectional retrieval of sparse patterns stored by hebbian learning. *Neural Networks*, 12:281–97, 1999.

[SvS12] C. J. Stam and E. C. W. van Straaten. The organization of physiological brain networks. *Clinical Neurophysiology*, 123(6):1067–87, June 2012.

[SW49] C. E. Shannon and W. Weaver. The mathematical theory of communication. 1949.

[WA90] D. Wang and M. A. Arbib. Complex temporal sequence learning based on short-term memory. *Proceedings of the IEEE*, 78(9), September 1990.

[WBLH69] D. J. Willshaw, O. P. Bunetman, and H. C. Longuet-Higgins. Non-holographic associative memory. *Nature*, 222:960–62, 1969.