HAL
open science

Implementing Relational-Algebraic Operators for Improving Cognitive Abilities in Networks of Neural Cliques

Ala Aboudib, Vincent Gripon, Baptiste Tessiau

To cite this version:

Ala Aboudib, Vincent Gripon, Baptiste Tessiau. Implementing Relational-Algebraic Operators for Improving Cognitive Abilities in Networks of Neural Cliques. COGNITIVE 2015 : the 7th International Conference on Advanced Cognitive Technologies and Applications, Mar 2015, Nice, France. pp.36 - 41. hal-01216082

HAL Id: hal-01216082

https://hal.science/hal-01216082

Submitted on 19 Oct 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Implementing Relational-Algebraic Operators for Improving Cognitive Abilities in Networks of Neural Cliques

Ala Aboudib, Vincent Gripon and Baptiste Tessiau
Télécom Bretagne - Electronics Department
Brest cedex 3, France
Emails: ala.aboudib@telecom-bretagne.eu, vincent.gripon@telecom-bretagne.eu, baptiste.tessiau@ens-rennes.fr

***Abstract*—Associative memories are devices capable of retrieving previously stored messages from parts of their content. They are used in a variety of applications including CPU caches, routers, intrusion detection systems, etc. They are also considered a good model for human memory, motivating the use of neural-based techniques. When it comes to cognition, it is important to provide such devices with the ability to perform complex requests, such as union, intersection, difference, projection and selection. In this paper, we extend a recently introduced associative memory model to perform relational algebra operations. We introduce new algorithms and discuss their performance which provides an insight on how the brain performs some high-level information processing tasks.**

Keywords–Cognitive modeling; artificial neural networks; relational algebra; associative memory; simulated annealing

I. Introduction

Associative memories are special types of memories that are capable of high-speed content-based mapping between input queries and outputs. This type of storage differs from classical index-addressable memories in that no explicit address is needed to search for stored data. In order to efficiently provide this associative functionality to memories, different methods of content structuring are required. Artificial neural networks (ANNs) are known to be such an adapted medium to implementing associative memorization. Their design is inspired from neo-cortical neural mechanisms in mammalian brains, believed to be knowledge-associative biological neural networks [1].

Many ANN models, differing in topology and functionality, were proposed to act as associative memories. Famous examples include the Perceptron [2], self-organizing maps [3], Hopfield networks [4] and Boltzmann machines [5]. A new model was proposed recently by Gripon and Berrou in [6] and generalized by Aliabadi et al. in [7]. This model relies on sparse coded patterns stored as cliques and is based on Hebian learning [8]. This sparse coding approach resembles the ones introduced by willshaw [9] and Palm [10], with an added explicit mapping between stored messages and their representation in the network.

Typically, an associative memory is able to retrieve a previously stored piece of information given partial content, a sort of erasure-retrieval property. However, human ability to handle more complex queries suggests that their representation of information should be able to perform other operations. Relational algebra gives a formal framework to introduce complex operations on tuples of stored messages, including union, intersection, difference, projection and selection. In this paper, we aim at extending the model introduced in [7] to process these operations.

Most of these operations are common as far as human cognition is concerned. For example, selection aims at retrieving the list of all messages that match a given probe (e.g., listing all city names you know that start with the letter 'b'). The union operation can be used for merging date while intersection and difference operation are useful for comparing contents of several memories or memory regions. Our main motivation is to show that existing neural-network-based architectures for associative memories are easily adapted in order to handle these complex queries. There have been several works addressing the relational problem and its biological plausibility such as [11] and [12], which were more focused on inference and relational learning.

The rest of this paper is organized as follows: in Section II, we introduce the associative memory model extended in this paper. Sections III, IV, V and VI describe how to perform resp. union, intersection, difference and projection using these models. In Section VII, we explain how to handle the more complex selection operator. For this operator, we introduce a novel algorithm using simmulated annealing. Simulation results are provided in Section VIII when independently identically uniformly distributed messages are considered. Section IX is a conclusion.

II. The memory model and relational algebra

A. The associative memory model

First, we introduce the associative memory model proposed by Gripon and Berrou in [6] and then extended in [7]. Let us consider the finite alphabet $\mathcal{A}$ made of integers between 1 and ℓ. we define a blank character $\perp \notin \mathcal{A}$ and $\overline{\mathcal{A}} = \mathcal{A} \cup \{\perp\}$. We are interested in storing words made of χ characters over $\overline{\mathcal{A}}$. We call such a word a *message* $m = m_1 m_2 \dots m_\chi$. Blank characters represent the absence of a character at a given position, such that depending on their number, messages can be regarded as sparse vectors.

An associative memory is a device capable of storing messages and then retrieving them given partial knowledge about some of their nonblank characters. To implement this device, the authors of [6] propose to use a symmetric, binary, χ-partite neural network composed of $n = \chi \cdot \ell$ vertices that we shall refer to as *units*. The authors of [6] explain

that units should be considered analogous to cortical micro-columns believed to be the computational building-blocks of the cerebral neo-cortex [13] [14]. This network can be split into χ *clusters*, each containing the same number ℓ of units.

We denote by $[n]$ the set of integers beween 1 and n, let us then index each cluster of the network by an integer in $[\chi]$. We also index units of a given cluster by integers in $[\ell]$. As a result, each unit in the network is uniquely addressed giving a couple (i, j) where i is the index of a cluster and j the index of the unit within this cluster.

We define a function f that maps each message m into a set of couples (i, j) as follows:

$$f : m \mapsto \qquad \{(i,j) \big| i \in [\chi], j \in [\ell] \text{ and } m_i = j\} \tag{1}$$

where m_i refers to the i-th character of m.

The network topology is entirely captured by an adjacency matrix W of size $\chi \cdot \ell$ such that:

$$w_{(i,j)(i',j')} = \begin{cases} 1 & \text{if } (i,j) \text{ and } (i',j') \text{ are connected} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

To store a message $m = m_1 m_2 \dots m_\chi$ in such a network, all unit pairs in $f(m)$ are connected pairwise according to 2 forming a clique in the underlying graph. It is worth noticing that cliques corresponding to different messages could sometimes overlap and share connections. Given a set of messages $\mathcal{M}$, we denote by $W(\mathcal{M})$ the adjacency matrix obtained after storage of all messages in $\mathcal{M}$.

A subset of $f(m)$ is called a *partial input* associated with the message m. The task of an associative memory is then: given a partial input of $m \in \mathcal{M}$, retrieve m using $W(\mathcal{M})$.

To retrieve a message from a partial input, an iterative algorithm is performed. Retrieval algorithms and techniques have been discussed in detail in [15].

It has been shown in [7] that performance of this structure as an associative memory mainly depends on a density parameter defined as the ratio of the number of connections in the network to the total number of possible connections. Under the hypothesis that messages contain exactly c non-blank characters uniformly distributed over $\overline{\mathcal{A}}$, density can be approximated by the following equation:

$$d = 1 - \left(1 - \frac{c(c-1)}{\chi(\chi-1)l^2}\right)^M \tag{3}$$

In this paper, we aim at extending the functionality of these associative memories, which we shall call Clustered Clique Networks (CCNs), to cover more general problems defined in relational algebra.

B. Connections to relational algebra

In relational algebra, operators are defined on *relations* (sets of tuples). A set of attributes is associated with each relation. Then, each tuple is defined as a set of instances of these attributes. A CCN in this respect can be viewed as a relation. Each cluster represents an attribute and units within each cluster are instances of that attribute (attribute values). A clique connecting units is equivalent to a tuple. In the following sections, we are going to propose algorithms and methods for implementing some relational operators on relations defined in the form of CCNs.

III. Union

Defined in terms of the set theory, the union of a collection of sets $S^1, S^2, ..., S^n$ is a set $S^\cup$ containing all distinct elements in this collection. It can be described as follows:

$$S^\cup = S^1 \cup S^2 \cup ... \cup S^k \tag{4}$$

$$= \{x | x \in S^1 \ \vee \ x \in S^2 \ \vee \ ... \ \vee \ x \in S^k\} \tag{5}$$

where $\cup$ is the set union operator and $\vee$ is the Boolean OR function.

In the context of memory storage, union is used to combine the contents of several memories or memory partitions into a single one while avoiding the redundancy resulting from the same data-word being stored multiple times. An example of this is merging the contents of two folders on a computer. This task is straightforward when using a classical indexed memory since messages do not overlap.

Suppose that we have two CCNs with the same dimensions $W(\mathcal{M}_1)$ and $W(\mathcal{M}_2)$ that we wish to merge in a single network $W^\cup$ of the same dimensions. We define this operation as follows:

$$w^\cup_{(i,j)(i',j')} = \begin{cases} 1 \text{ if } & w(\mathcal{M}_1)_{(i,j)(i',j')} = 1 \vee \\ & w(\mathcal{M}_2)_{(i,j)(i',j')} = 1 \\ 0 \text{ otherwise} \end{cases} \tag{6}$$

Following intuitively from this definition is the fact that upon applying the union operation, information is conserved in the new network. That is, if a clique exists in either $W(\mathcal{M}_1)$ or $W(\mathcal{M}_2)$, it would also exists in $W^\cup$. Hence we can rewrite $W^\cup$ as $W^\cup(\mathcal{M})$ with $\mathcal{M} = \mathcal{M}_1 \cup \mathcal{M}_2$. The problem here is that combining memories in this fashion can cause a significant growth in the density of $W^\cup(\mathcal{M})$, leading possibly to dramatically low performance in terms of retrieval error rates of stored messages. More formally, if d^1 is the density of $W(\mathcal{M}_1)$ and d^2 is the density of $W(\mathcal{M}_2)$, then the density of $W^\cup(\mathcal{M})$ denoted $d^\cup$ is given by:

$$d^\cup = 1 - (1 - d^1)(1 - d^2) \tag{7}$$

and thus $d^\cup$ is greater than both d^1 and d^2. This means that while retrieval error rates may be optimal for $W(\mathcal{M}_1)$ and $W(\mathcal{M}_2)$, the network resulting from their union might suffer from some degeneration in performance because of the increased density. Therefore, union should be done only if the resulting network performance is acceptable. The relationship between retrieval error rates and density are presented in details in [7] and [15].

According to (6) all possible connections in the networks should be tested during the union operation. So, given $\frac{\chi(\chi-1)\ell^2}{2}$ possible connections [7] in each network, the average-case complexity of such process is $\Theta(\chi^2\ell^2)$.

The phenomenon of degenerated memorization efficiency caused by the increased density is not uncommon in the brain. For example, learning and recalling a new word in a foreign language is typically not a difficult task. However, trying to learn a dozen of new words at the same time might turn out to be much more challenging comprising many memorization errors and confusions and even mixing syllables of different words. New words can be better learned by training and experience, which is partially due to the association of these words with other memories. We suggest that this process, in

terms of CCNs is equivalent to adding more clusters to the network such that more units can be added to existing cliques, which lowers its density and increases performance.

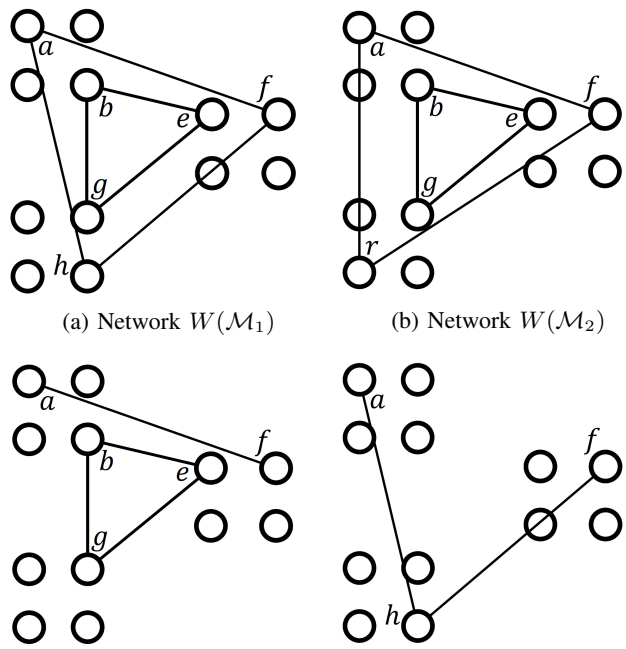
IV. Intersection

The intersection among several sets $S^1, S^2, ..., S^k$ is a set $S^\cap$ containing only those elements that $S^1, S^2, ..., S^k$ have in common:

$$S^\cap = S^1 \cap S^2 \cap ... \cap S^k \tag{8}$$

$$= \{x | x \in S^1 \ \wedge \ x \in S^2 \ \wedge \ ... \ \wedge \ x \in S^k\} \tag{9}$$

where $\cap$ is the set intersection operator and $\wedge$ is the Boolean AND function.



(a) Network $W(\mathcal{M}_1)$ (b) Network $W(\mathcal{M}_2)$

(c) Network $W^\cap$ (d) Network $W^\setminus$

Figure 1. Intersection and Difference between two CCNs. $W^\cap$ shown in 1c is the network resulting from the intersection of $W(\mathcal{M}_1)$ and $W(\mathcal{M}_2)$. $W^\setminus$ shown in 1d is the difference of $W(\mathcal{M}_1)$ from $W(\mathcal{M}_2)$.

When applied to relations, intersection serves in extracting common tuples between two or more tables having the same number and types of attributes. This is done easily when such a database is stored in an indexed memory. One way to implement intersection between two CCNs is by keeping only those connections that happen to exist in the exact same place in both networks. So, given two CCNs $W(\mathcal{M}_1)$ and $W(\mathcal{M}_2)$ of the same type and dimensions, we can define their intersection $W^\cap$ as follows:

$$w^\cap_{(i,j)(i',j')} = \begin{cases} 1 \text{ if } & w(\mathcal{M}_1)_{(i,j)(i',j')} = 1 \wedge \\ & w(\mathcal{M}_2)_{(i,j)(i',j')} = 1 \\ 0 \text{ otherwise} \end{cases} \tag{10}$$

The average-case complexity of this operation is given by $\Theta(\chi^2\ell^2)$ for the same reason as in the union. The density $d^\cap$ of $W^\cap$ as a function of d^1 and d^2 (the densities of $W(\mathcal{M}_1)$ and $W(\mathcal{M}_2)$ respectively) is given by:

$$d^\cap = d^1.d^2 \tag{11}$$

We notice from (11) that $d^\cap$ is always lower than d^1 and d^2 given that densities have their values in the interval [0,1]. Thus, we guarantee that no density explosion occurs as in union.

Some problems might still occur when performing this operation, as depicted in Figure 1 where four simple identical networks are considered each with a total number of units $n = 12$ grouped in $\chi = 3$ clusters with a message size of $c = \chi$. Network $W(\mathcal{M}_1)$ shown in Figure 1a contains two cliques afh and beg and network $W(\mathcal{M}_2)$ in Figure 1b also contains two cliques afr and beg. We notice that only the clique beg is common between these two networks. By applying the intersection operation as in (10) we obtain $W^\cap$ shown in Figure 1c, which contains the common clique beg as expected but also contains the edge af, which is an undesirable result, because af does not represent a complete message (tuple). We shall call af a *residual* edge. As a consequence, $W^\cap$ could possibly contain more connections than a an ideal intersection network $W^\cap(\mathcal{M})$ with $\mathcal{M} = \mathcal{M}_1 \cap \mathcal{M}_2$. This increase in density due to residual edges is expected to deteriorate performance [7].

V. Difference

The difference of two sets S^1 and S^2, which can also be called that relative complement of S^2 with respect to S^1, is a set $S^\setminus$ that contains only those elements of S^1 that are not in S^2:

$$S^\setminus = S^1 \setminus S^2 = \{x | x \in S^1 \text{ and } x \notin S^2\} \tag{12}$$

where $\setminus$ is the set difference operator which is not commutative so that $S^1 \setminus S^2 \neq S^2 \setminus S^1$.

So, the difference between two database tables is the set of tuples in the first one that do not exist in the other, which is also an easy-to-implement operation in classical memory systems. A simple method of implementing difference between two CNNs $W(\mathcal{M}_1)$ and $W(\mathcal{M}_2)$ is by instantiating a new memory $W^\setminus$ containing only connections in $W(\mathcal{M}_1)$ that do not exist in $W(\mathcal{M}_2)$. That is:

$$w^\setminus_{(i,j)(i',j')} = \begin{cases} 1 \text{ if } & w(\mathcal{M}_1)_{(i,j)(i',j')} = 1 \wedge \\ & w(\mathcal{M}_2)_{(i,j)(i',j')} = 0 \\ 0 \text{ otherwise} \end{cases} \tag{13}$$

The average-case complexity of this operation is also given by $\Theta(\chi^2\ell^2)$. Intuitively, the density $d^\setminus$ of the new network $W^\setminus$ is always less than or equal to the density d^1 of $W(\mathcal{M}_1)$. So, if d^1 is well controlled, we would have no problems with the performance of $W^\setminus$. The density $d^\setminus$ is given by the following relationship:

$$d^\setminus = d^1.(1 - d^2) \tag{14}$$

As in the case of intersection, the difference operation defined in (13) processes data down on the level of individual connections not on the level of cliques. This causes the problem depicted in Figure 1d. In this figure, $W^\setminus$ is the network resulting from applying $W(\mathcal{M}_1) \setminus W(\mathcal{M}_2)$. Ideally, we wish that $W^\setminus = W^\setminus(\mathcal{M})$ with $\mathcal{M} = \mathcal{M}_1 \setminus \mathcal{M}_2$,i.e., a network that contains only the clique afh. However, according to (13), we only get two edges ah and fh because af is a common edge between $W(\mathcal{M}_1)$ and $W(\mathcal{M}_2)$ and thus eliminated by the difference operation. This represents a loss of information since the network $W^\setminus$ no more stores the message

corresponding to the clique afh. We shall call this undesirable effect *erosion*. Actually, no method is yet available for getting an ideal intersection or an ideal difference between CCNs. We consider the methods proposed in this paper as approximations to the real operations.

VI. Projection

In relational algebra, projection is defined as a unary operator Π applied to a tuple R in order to produce a new tuple R^{Π} consisting of k attributes $\{r_1, r_2, ..., r_k\}$, which is a subset of the attributes originally contained in R. This can be written as follows:

$$R^{\Pi} = \Pi_{r_1, r_2, ..., r_k}(R) \tag{15}$$

A network W^{Π} is said to be a projection of a network W on a given set of attributes, if it contains only a subset of the attributes of W ,i.e., W^{Π} contains only a subset of the clusters of W.

Clearly, this operation is very easy to implement in CNNs with a constant time complexity. Moreover, the density d^{Π} of the resulting network is equal to the density of the original network given that connections are uniformly distributed within the latter network:

$$d^{\Pi} = d \tag{16}$$

VII. Selection

A. Problem definition

In relational algebra, selection or restriction is a unary operator applied to a relation R^1 and returns another relation R^2. The latter relation contains all tubles in R^1 whose attribute values satisfy a propositional formula φ. This can be transcribed as follows:

$$R^2 = \sigma_{\varphi}(R^1) \tag{17}$$

We argue in this paper that a mechanism similar to selection might be used by the brain for thinking and memorization. A typical such request would be, for instance, to name all scientific authors one knows whose names start with an 'a'. We aim at using the model proposed in [7] as a substrate for this selection process.

The selection algorithm we shall present here runs continuously, giving multiple (possibly redundant) answers one after another. This appears to us being behaviorally similar to what humans could produce facing a similar query. The process that makes us avoid redundancy is called short-term/working memory. Once its buffer is full or overcrowded, repetitions of the same word/name can occur.

The requirements that our selection algorithm is meant to meet are the following:

1) Determine the sub-graph G of potentially interesting units.
2) In sub-graph G, find all cliques of size c.

Finding a maximum clique in a graph (or equivalently) a minimum cover is a know NP-complete problem. Many algorithms and heuristics were proposed to give approximate solutions to this problem in medium-sized graphs. Examples of these algorithms are [16] and [17] that make use of the simulated annealing principle introduced in [18] and [19], which is a probabilistic meta-heuristic method for locating the global optimum of a given function. Another known method widely used in applications such as computational chemistry is the BronKerbosch algorithm [20], which can efficiently find maximal cliques in an undirected graph.

We propose to use an adaptation of the simulated annealing algorithm proposed by Geng et al. in [16]. We also use their same objective function to evaluate our solutions.

B. The proposed selection algorithm

Suppose we have a CCN denoted by W and a partial input message m containing $q \leqslant c$ known nonblank attributes. The selection operator consists in finding all stored cliques made of a set of units containing $f(m)$. In order to perform this search efficiently, it is sufficient to restrict the search to the subgraph made of only the units connected to units in $f(m)$. We shall call this subgraph G_m. We denote its adjacency matrix by A_m.

For simplicity of representation, we refer to each unit of G_m by an integer index k or s where $k, s \in \{0, 1, ..., n' - 1\}$, n' being the number of units in G_m. Our objective now is to find all the cliques in G_m that have a size (number of units) of $c' = c - q$.

We consider the following optimization problem: We define ρ as a permutation of units in G_m (ρ is an array of size n' containing all unit indexes of G_m as its elements). For a given ordering of elements in ρ, we consider the first c' elements of ρ as indexes of units in G_m acting as a potential solution (a clique). So, by permuting ρ's contents we can get a different candidate solution. The objective function used to evaluate these solutions is given by:

$$F(G_m, \rho) = \sum_{k=0}^{c'-2} \sum_{s=k+1}^{c'-1} (1 - a_{\rho[k],\rho[s]}) \tag{18}$$

where $a_{\rho[k],\rho[s]}$ is an element of the adjacency matrix W. As $F(G, \rho) = 0$ when a clique is found, the goal is to find permutations that minimize this function.

The algorithm is applied to G_m as follows:

Step 1: Parameter initialization.
Initial temperature T_1, end temperature T_2, current temperature $t = T_1$ and cooling coefficient α. Set the initial permutation as $\rho[k] = k, k \in \{0, 1, ..., n' - 1\}$. Random setting of permutation is also possible.

Step 2: Compute $F(G_m, \rho)$ and terminate if it evaluates to zero.

Step 3: Randomly choose two integer indexes u and w of ρ such that $u \in \{0, 1, ..., c' - 1\}$ and $w \in \{c', c' + 1, ..., n' - 1\}$.
Condition 1:
If $\rho[w]$ has more or the same number of connections with $\{\rho[0], \rho[1], ..., \rho[c' - 1]\}$ than $\rho[u]$ has, then $\rho[w]$ and $\rho[u]$ are swapped and thus a new permutation ρ' is obtained.
Condition 2:
If $\rho[u]$ has more connections with $\{\rho[0], \rho[1], ..., \rho[c' - 1]\}$ than $\rho[w]$ has, then the index w is rejected and we go back to step 3. but if w has already been rejected for more than $8n'$ times consecutively as defined in [16], then

$\rho[w]$ and $\rho[u]$ are swapped and a new permutation ρ' is obtained.

Step 4: Compute $F(G_m, \rho')$ and terminate if it evaluates to zero.

Step 5: Compute $\Delta F = F(G_m, \rho') - F(G_m, \rho)$. If $\Delta F \leqslant 0$, then accept the new permutation ρ' by setting $\rho = \rho'$. Otherwise accept ρ' with probability $p = e^{\frac{-\Delta F}{t}}$.

Step 6: Update current temperature as $t = \alpha t$. If $t < T_2$ terminate the algorithm; otherwise, go back to step 3.

A single run of this selection algorithm is meant to find one clique ,i.e., one answer. So, in order to output more answers this algorithm should be repeated many times.

VIII. Results

A. Selection

In order to test the proposed selection algorithm, we used a CCN with $n = 3240$ units grouped in $\chi = 15$ clusters. 15000 randomly generated messages of $c = 10$ characters were stored giving a network density d of about 0.13. We used the same initial and end temperatures as in [16] for the simulated annealing algorithm; 100 and 0.001, respectively. We set the cooling coefficient to 0.996. To construct a selection query message, one already stored message is randomly selected of which 9 characters were erased (set to $\perp$) giving an input query message with only one known non-blank character.

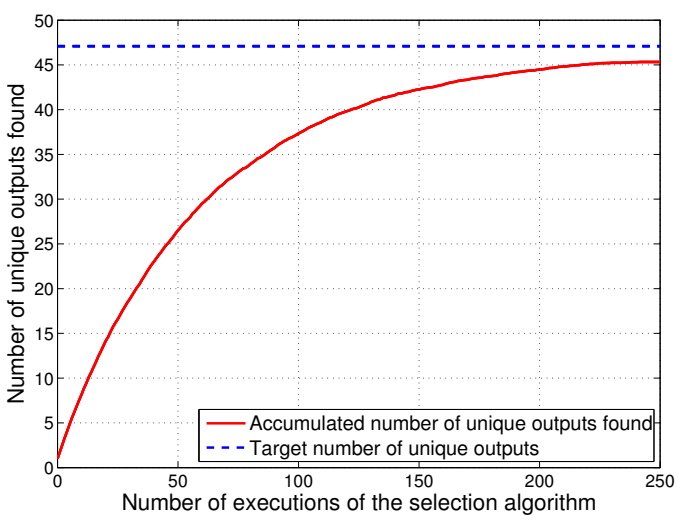


Figure 2. Average accumulated number of unique outputs as a function of the number of iterations in a network of $n = 3240$, $\chi = 15$, $c = 10$, 15000 stored messages. The input message used has only one non-blank character.

As described in Section VII, the algorithm we propose might give redundant outputs when executed several times. Figure 2 shows how fast unique outputs were found as a function of the number of executions of the selection algorithm. Curves in the figure are averaged over 100 identical experiments.

An interesting property of the resulting curve is that most unique answers are obtained during earlier executions of the algorithm. So, referring back to Figure 2, about 45 unique answers out of 47 possible ones are found by the 250th execution, 40 answers by the 125th execution and about 30 answers by the 60th. In other words, about 89% of answers were obtained when only 50% of total executions were achieved and 67% of answers after 24% of total executions. This is a natural result for a selection by replacement experiment where all answers have an equal chance of being chosen at each execution.

We suggest that such result bears some qualitative resemblance to the way human beings memorize lists of mental objects where it is common that the last few items turn out to be more difficult and time consuming to recall because of the distraction caused by redundant answers coming to mind and other phenomena.

B. Intersection and difference

A comparison among average complexities of some operators when applied to CCNs and two other known data structures (ordered lists and binary search trees) storing sparse messages of the form $m = m_1 m_2 \ldots m_\chi$ is provided in Table I. An interesting observation is the fact that the order of complexity of operators using a CCN is close to that of a binary search tree given that setting $\ell \gg \chi$ is preferable in practice for a higher network capacity [7]. However, the complexity of union, intersection and difference of ordered lists is lower by a factor of χ than that of CCNs while the complexity of insertion is ℓ^2/χ higher.

TABLE I. COMPLEXITY OF SOME RELATIONAL OPERATORS IN SEVERAL TYPES OF DATA STRUCTURES.

	CCN	ordered list	binary search tree
Insertion(Storing)	$\Theta(\chi^2)$	$\Theta(\chi\ell^2)$	$\Theta(\chi \log(\ell))$
Union	$\Theta(\chi^2\ell^2)$	$\Theta(\chi\ell^2)$	$\Theta(\chi\ell^2 \log(\ell))$
Intersection	$\Theta(\chi^2\ell^2)$	$\Theta(\chi\ell^2)$	$\Theta(\chi\ell^2 \log(\ell))$
Difference	$\Theta(\chi^2\ell^2)$	$\Theta(\chi\ell^2)$	$\Theta(\chi\ell^2 \log(\ell))$

IX. Conclusion and Future Work

In this paper, we have introduced some methods for applying certain algebraic-relational operators on a new class of neural-network-based associative memories we call CCNs. We argued that the process of recalling a list of items (which can also be mapped to more general memorization tasks) in the brain can be behaviorally assimilated to the selection operation known to relational algebra. We proposed an algorithm for implementing this process using the principle of simulated annealing. Then, we showed that the results we got have some resemblance to what might be obtained by a human subject in terms of redundancy.

We have also demonstrated that CCNs can be used as classic data-structures by approximating operators such as union, intersection, difference and projection. We saw that the implementation of union and its related density explosion problem raised the question as to how the brain organizes information with high correlation or high density. Two possible mechanisms the brain might be using are forgetting rarely used “data” (by the decay of synaptic weights) and tagging pieces of correlated data with different contextual information. Similar mechanisms might be integrated in CCNs by allowing connections to have real values with a decay parameter and by providing contextual tagging in the form of CCNs existing on

a separated level of a hierarchy of networks. Another possible solution is to design networks with dynamic sizes to prevent exceeding a maximum allowed density.

ACKNOWLEDGMENT

This work was supported by the European Research Council under the European Union's Seventh Framework Program (FP7/2007-2013) / ERC grant agreement n 290901.

REFERENCES

[1] J. R. Anderson and G. H. Bower, Human associative memory. Psychology press, 2013.

[2] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain." Psychological review, vol. 65, no. 6, 1958, p. 386.

[3] T. Kohonen, "Self-organized formation of topologically correct feature maps," Biological cybernetics, vol. 43, no. 1, 1982, pp. 59–69.

[4] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," Proceedings of the national academy of sciences, vol. 79, no. 8, 1982, pp. 2554–2558.

[5] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, "A learning algorithm for boltzmann machines," Cognitive science, vol. 9, no. 1, 1985, pp. 147–169.

[6] V. Gripon and C. Berrou, "Sparse neural networks with large learning diversity," Neural Networks, IEEE Transactions on, vol. 22, no. 7, 2011, pp. 1087–1096.

[7] B. K. Aliabadi, C. Berrou, V. Gripon, and J. Xiaoran, "Storing sparse messages in networks of neural cliques." IEEE transactions on neural networks and learning systems, vol. 25, no. 5, 2014, pp. 980–989.

[8] D. O. Hebb, The Organization of Behavior. John Wiley, 1949.

[9] D. J. Willshaw, O. P. Buneman, and H. C. Longuet-Higgins, "Non-holographic associative memory." Nature, 1969.

[10] G. Palm, "On associative memory," Biological Cybernetics, vol. 36, no. 1, 1980, pp. 19–31.

[11] H. Blockeel and W. Uwents, "Using neural networks for relational learning," ICML-2004 Workshop on Statistical Relational Learning and its Connection to Other Fields, 2004, pp. 23–28.

[12] J. E. Hummel and K. J. Holyoak, "A symbolic-connectionist theory of relational inference and generalization." Psychological review, vol. 110, no. 2, 2003, p. 220.

[13] E. G. Jones, "Microcolumns in the cerebral cortex," Proceedings of the National Academy of Sciences, vol. 97, no. 10, 2000, pp. 5019–5021.

[14] V. B. Mountcastle, "The columnar organization of the neocortex." Brain, vol. 120, no. 4, 1997, pp. 701–722.

[15] A. Aboudib, V. Gripon, and X. Jiang, "A study of retrieval algorithms of sparse messages in networks of neural cliques," COGNITIVE 2014, The Sixth International Conference on Advanced Cognitive Technologies and Applications, 2014, pp. 140–146.

[16] X. Geng, J. Xu, J. Xiao, and L. Pan, "A simple simulated annealing algorithm for the maximum clique problem," Information Sciences, vol. 177, no. 22, 2007, pp. 5064–5071.

[17] X. Xu and J. Ma, "An efficient simulated annealing algorithm for the minimum vertex cover problem," Neurocomputing, vol. 69, no. 7, 2006, pp. 913–916.

[18] S. Brooks and B. Morgan, "Optimization using simulated annealing," The Statistician, 1995, pp. 241–257.

[19] V. Černỳ, "Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm," Journal of optimization theory and applications, vol. 45, no. 1, 1985, pp. 41–51.

[20] C. Bron and J. Kerbosch, "Algorithm 457: finding all cliques of an undirected graph," Communications of the ACM, vol. 16, no. 9, 1973, pp. 575–577.