Fifty-first Annual Allerton Conference
Allerton House, UIUC, Illinois, USA
October 2 - 3, 2013

Sparse Structured Associative Memories as Efficient Set-Membership Data Structures

Vincent Gripon
Electronics Department
Télécom Bretagne, Brest, France
vincent.gripon@ens-cachan.org

Vitaly Skachek
Institute of Computer Science
University of Tartu, Estonia
vitaly.skachek@ut.ee

Michael Rabbat
Electrical and Computer Engineering
McGill University, Montréal, Canada
michael.rabbat@mcgill.ca

***Abstract*— We study the use of sparse structured associative memories as a memory-efficient and computationally-efficient data structure for representing a set of elements when one wishes to perform set-membership queries and some errors (false positives) are tolerable. Associative memories, when viewed as representing a set, enjoy a number of interesting properties, including that set membership queries can be carried out even when the input (query element) is only partially known or is partially corrupted. The associative memories considered here (initially proposed in [Gripon and Berrou, 2011]) encode the set in the edge structure of a graph. In this paper we generalize this construction to encode the set in the edge structure of a hypergraph. We derive bounds on the false positive rates (the probability that the associative memory erroneously declares that an element is in the stored set when it, in fact, was not). Interestingly, the proposed structures enjoy many of the same properties as Bloom filters (e.g., they have zero false negative rate, the time to perform an insert and lookup does not depend on the number of elements stored, and the false positive rate can be reduced by using additional memory for storage), while also offering the properties of associative memories (allowing for queries on partial or corrupted inputs).**

I. Introduction

A wide variety of applications require the ability to quickly determine whether a given element is a member of a pre-specified subset of possible elements. For example, in network security and intrusion detection, a packet must be quickly tested to see if it matches patterns of malware or known attacks [1]. Similarly, queries in database systems can be resolved or approximated set representations and operations (e.g., counting) [2], [3].

In general, we consider the problem of representing a subset S of a finite universe of elements $\mathcal{U}$ with $|\mathcal{U}| = n$ and $|S| = m < n$. The aim is to represent S using a data structure which can quickly test whether a particular element $u \in \mathcal{U}$ is a member of S (i.e., in $\Theta(\log_2(n))$ time), while not requiring substantial memory. Observe that storing S as a sorted array would require $\Theta(m \log_2(n))$ bits of memory and each query would have a complexity of $\Theta(\log_2(m) \log_2(n))$ time.

Bloom filters [4], and their variations [5], are a popular randomized data structure for representing sets in order to efficiently perform set membership queries when some errors are tolerable. They store the subset S in such a way that set membership queries have no false negatives (i.e., if $u \in S$, then the filter always recognizes u as being in S), but false positives may occur (i.e., an element $u \in \mathcal{U}$, $u \notin S$ may be falsely declared to be a member of S). Specifically, Bloom filters represent S using an array of b bits and k hash functions $h_1, \ldots, h_k$, with each h_j mapping elements of $\mathcal{U}$ to $[b] \stackrel{\text{def}}{=} \{1, \ldots, b\}$. The empty set $S = \emptyset$ is represented by setting all bits to 0. To add an element $x \in S$, the bits indexed by $\{h_j(x)\}_{j=1}^k$ are set to 1. Once a bit is set to 1, it remains set at 1 regardless of whether other elements of S also hash to the same bit. After all elements $x \in S$ have been inserted, to test whether an element x' is in S, one checks the status of the bits indexed by $\{h_j(x')\}_{j=1}^k$; if all bits are set to one, then we declare $x' \in S$, and otherwise we declare $x' \notin S$. Clearly, as more elements are inserted in the data structure, more bits will be set to 1, leading to more false positives. The false positive rate can be controlled by appropriate choice of the number of bits b and the number of hash functions k. In particular, the false positive rate is guaranteed to be no more than $\epsilon > 0$ if

$$b \geq m \cdot \log_2(e) \cdot \log_2(1/\epsilon) \quad \text{bits} \tag{1}$$

and $k = \lceil (b/m) \log_2(e) \rceil$ hash functions are used, regardless of the size of the universe. Insertions and queries both have complexity of $\Theta(k \log_2(n))$ time if the hash functions are evaluated one after the other, or $\Theta(\log_2(n))$ if the hash functions can be evaluated in parallel.

In this paper we consider an alternative to Bloom filters for representing sets. Our approach builds on an associative memory construction proposed by Gripon and Berrou [6] which can be viewed as sparse, structured binary graphical models with binary-valued edge-potentials. We refer to these as *sparse, structured associative memories* (SSAMs). SSAMs enjoy many of the same properties as Bloom filters. Insertions and queries can be performed in $\Theta(\log_2(n))$ time. They never produce false negatives, and their false positive rate can be reduced through the use of additional bits of memory. In addition, SSAMs enjoy the error-correcting properties of associative memories; that is, they can be used to test if an element x' is in the stored set, even if a portion of the binary representation of x' has been erased or corrupted. On the other hand, the analysis of SSAMs assumes that the set S being represented is drawn uniformly from all subsets of m elements from $\mathcal{U}$.

The contributions of this article are as follows. First, we present detailed error bounds on the false positive rate for

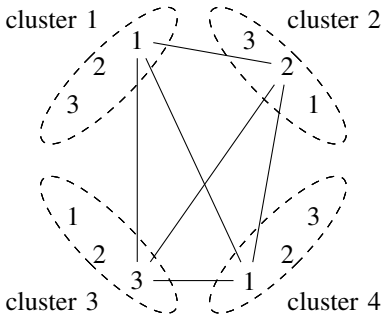

Fig. 1. Illustration of the storing process of the message 1231 in a graph using the principles described in [6].

SSAMs. The SSAM model described in [6] represents the set S in terms of the edges of a graph (see Section II for details). In this paper we present an extension of the SSAM model to hypergraphs and demonstrate how this leads to improved error rates, at the cost of higher storage. We show that using a q-regular hypergraph is more efficient (in terms of the amount of memory required) when the number of messages m to be stored is in the regime $\Theta(n^{\frac{q}{\log_2(n)^{1/q}}})$.

II. Graph-based Sparse Structured Associative Memories

This section describes the construction of SSAMs following the description in [6]. Recall that each element $x \in \mathcal{U}$ can be represented using $\log_2(n)$ bits. Equivalently, each element can be represented as a string of c symbols from an alphabet $\mathcal{A} = [l]$ as long as $c\log_2(l) \geq \log_2(n)$. For an element $x \in \mathcal{U}$, let $w_i(x) \in \mathcal{A}$, $i \in [c]$, denote the ith symbol in its representation as a string of c symbols in $\mathcal{A}$.

An SSAM is represented as a simple undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, and in particular, the set $S = \{x_1, \ldots, x_m\} \subset \mathcal{U}$ of elements to be stored is encoded in the edge set of the graph. Let c and l be positive integers such that $c\log_2(l) \geq \log_2(n)$. The graph $\mathcal{G}$ has $|V| = cl$ vertices; more precisely, $\mathcal{G}$ is c-partite with each partition containing l vertices. Let $v_{i,j}$, $i \in [c]$, $j \in [l]$, denote the jth vertex in the ith partition. Initially the graph is empty; i.e., $\mathcal{E} = \emptyset$. Then, for each element $x \in S$, we add edges between all distinct pairs of vertices $(v_{i,w_i(x)}, v_{i',w_{i'}(x)})$, where $i, i' \in [c]$, $i \neq i'$. Once an edge has been added to the graph for one element $x \in S$, it remains in the graph (i.e., nothing changes if another element $x' \in S$ also includes this edge). Observe that, after all elements have been added, each element $x \in S$ corresponds to a clique in $\mathcal{G}$. Let

$$C(x) = \big\{(v_{i,w_i(x)}, v_{i',w_{i'}(x)}) \in \mathcal{V}\times\mathcal{V} \colon 1 \leq i, i' \leq c,\, i \neq i'\big\}$$

denote the edges comprising the clique corresponding to $x \in \mathcal{U}$. Then, after adding all elements in S to the SSAM we have that $\mathcal{E} = \bigcup_{i=1}^{m} C(x_i)$. See Figure 1 for an illustration.

Now, given the graph $\mathcal{G}$ representing the set S, the SSAM declares that a given element x' is in S if all edges $(v_{i,w_i(x')}, v_{i',w_{i'}(x')})$, $1 \leq i, i' \leq c$, $i \neq i'$, are in $\mathcal{E}$; i.e., if $C(x') \subseteq \mathcal{E}$. Clearly, there are no false negatives, since $x \in S$ implies that $C(x) \subseteq \mathcal{E}$. However, there may be false positives. In particular, as $|S|$ grows, more edges are added to $\mathcal{E}$ and the false positive rate will increase. We provide bounds on the false positive rate in the following section.

Before proceeding, we note that representing $\mathcal{G}$ requires $\binom{c}{2}l^2$ bits since $\mathcal{G}$ is c-partite and each partition contains l vertices. The operations of adding an element to the SSAM and querying an element for set membership both have complexity of $\Theta(\binom{c}{2})$ time since they involve setting or checking the status of $\binom{c}{2}$ bits.

III. False Positive Rate

Next we derive bounds on the false positive rate of the SSAMs described in the previous section. We assume that the set $S = \{x_1, \ldots, x_m\}$ is comprised of m random elements $x_i \in \mathcal{U}$ (equivalently, m words in $[l]^c$) drawn uniformly and independently. Let $\mathbb{P}$ denote the corresponding probability measure. To avoid trivialities, we assume that $m > 1$.

Recall that there are $\binom{c}{2}l^2$ possible edges that may be in the graph. Each time a word is inserted, one edge is added (if it isn't already present) between each partition of $\mathcal{G}$. Therefore, after inserting m elements, the probability that a particular edge has not been added is

$$\left(1 - \frac{1}{l^2}\right)^m. \tag{2}$$

The expected density (the proportion of edges present in $\mathcal{G}$ after all m elements are inserted) is thus

$$\rho = 1 - \left(1 - \frac{1}{l^2}\right)^m. \tag{3}$$

Our main result characterizes the error rate when, for a fixed density, we let the size of the network grow as the number of elements inserted increases.

Theorem 1: Let ρ and K be fixed constants satisfying $0 < \rho < 1$ and $0 < K < \frac{2}{3}$, and let l and c be positive integers satisfying $c\log_2(l) \geq \log_2(n)$ and chosen such that

$$l \sim \rho^{-K(c-1)/2} \tag{4}$$

as $c \to \infty$. Then

$$\mathbb{P}\big(C(x) \subset \mathcal{E} \big| x \notin S\big) \leq \exp\{-\Theta(c)\}. \tag{5}$$

Remark 2: Note that the condition (4) states that binary messages are split into submessages whose number, c, and length, $\log_2(l)$, are of the same order of magnitude. The constant K is a scaling factor, the role of which will become clear in the sequel. It also follows from (3) that the number of messages m to be stored is of the order of $\rho^{-K(c-1)}$.

Theorem 1 states that the false positive rate decays exponentially in c, the number of partitions in $\mathcal{G}$, as long as c and l are chosen to satisfy (4), which dictates the rate at which the size of the network should grow as a function of the number of elements which have been added. To prove Theorem 1 we need to bound the probability that $\mathcal{G}$ contains a clique $C(x)$ for some element $x \notin S$. For two elements, $x, x' \in \mathcal{U}$, let $\mathsf{d}(x, x')$ denote the Hamming distance between $\boldsymbol{w}(x)$ and $\boldsymbol{w}(x')$, where $\boldsymbol{w}(x) = (w_1(x), w_2(x), \ldots, w_c(x))$ is the representation of the element using c symbols from

the alphabet $\mathcal{A} = [l]$. Hence $\mathsf{d}(x, x')$ is a non-negative integer between 0 and c. With slight abuse of notation, for a set $S \subset \mathcal{U}$, let $\mathsf{d}(x, S) = \min_{x' \in S} \mathsf{d}(x, x')$. The proof of Theorem 1 hinges on the following lemma.

Lemma 1: Let ρ and K be fixed constants satisfying $0 < \rho < 1$ and $0 < K < \frac{2}{3}$, and let l and c be positive integers satisfying $c \log_2(l) \geq \log_2(n)$ and chosen such that

$$l \sim \rho^{-K(c-1)/2} \tag{6}$$

as $c \to \infty$. Let $\delta \in [c]$ be an integer. Then, for x_0 such that $\mathsf{d}(x_0, S) = \delta$,

$$\mathbb{P}\big(C(x_0) \subset \mathcal{E} \big| x_0 \notin S\big) \leq \exp\{-\Theta(c)\}. \tag{7}$$

Proof: [Sketch of proof] For the elements $\{x_1, \ldots, x_m\}$ comprising the set S, let $C_i = C(x_i)$ denote the corresponding clique. Let us also write $C_0 = C(x_0)$ for the element $x_0 \notin S$. Without loss of generality, suppose that $\mathsf{d}(x_0, x_1) = \delta$, and suppose that $\boldsymbol{w}(x_0)$ and $\boldsymbol{w}(x_1)$ differ in the last δ symbols, where $\boldsymbol{w}(x) = \big(w_1(x), \ldots, w_c(x)\big)$. To simplify notation, let us further assume (without loss of generality) that C_1 is the set of all edges between pairs of vertices in the set

$$\{v_{1,1}, v_{2,1}, \ldots, v_{c,1}\},$$

and C_0 is the set of all edges between pairs of vertices in the set

$$\{v_{1,1}, v_{2,1}, \ldots, v_{c-\delta,1}, v_{c-\delta+1,2}, \ldots, v_{c,2}\}.$$

In other words, C_0 differs from C_1 in edges involving the last δ vertices. Let $\mathcal{E}_\Delta = C_0 \setminus C_1$.

Since the expected probability that a specific edge is present in $\mathcal{G}$ is ρ, and since $(v_{1,1}, v_{c-\delta+1,2}) \notin C_1$, we have

$$\mathbb{P}\big((v_{1,1}, v_{c-\delta+1,2}) \in \mathcal{E} \big| C_1\big) \leq \rho.$$

Next, consider the event that the edge $(v_{1,1}, v_{c-\delta+1,2}) \in \mathcal{E}$, and let C_2 be a clique that contains it. Then the edge $(v_{1,1}, v_{c-\delta+2,2})$ is in $\mathcal{E}$ if it either belongs to C_2 or if it belongs to a clique different from C_1 and C_2. For the first case, we have

$$\mathbb{P}\big((v_{1,1}, v_{c-\delta+2,2}) \in C_2 \big| C_1, (v_{1,1}, v_{c-\delta+1,2}) \in C_2\big) = \frac{1}{l}.$$

For the second case, similar to above, we obtain

$$\mathbb{P}\big((v_{1,1}, v_{c-\delta+2,2}) \in \mathcal{E} \big| C_1, (v_{1,1}, v_{c-\delta+1,2}) \in C_2, (v_{1,1}, v_{c-\delta+2,2}) \notin C_2\big) \leq \rho.$$

Continuing along this line of reasoning, we can bound the probability that each of the edges in $\mathcal{E}_\Delta$ are in $\mathcal{E}$. Let

$$\mathcal{V}_{\text{both}} = \{v_{1,1}, v_{2,1} \ldots, v_{c-\delta,1}\},$$

and let

$$\mathcal{V}_{\text{diff}} = \{v_{c-\delta+1,2}, v_{c-\delta+2,2}, \ldots, v_{c,2}\}.$$

Observe that $\mathcal{E}_\Delta$ is exactly

$$\mathcal{E}_\Delta = (\mathcal{V}_{\text{both}} \times \mathcal{V}_{\text{diff}}) \bigcup (\mathcal{V}_{\text{diff}} \times \mathcal{V}_{\text{diff}} \setminus \{(v_{i,j}, v_{i,j}) : v_{i,j} \in \mathcal{V}_{\text{diff}}\}).$$

Therefore, there are

$$|\mathcal{E}_\Delta| = (c-\delta)\delta + \frac{\delta(\delta-1)}{2} = \delta\left(c - \frac{\delta+1}{2}\right)$$

edges in $\mathcal{E}_\Delta$, and we obtain that

$$\mathbb{P}\big(\mathcal{E}_\Delta \subset \mathcal{E} \big| C_1\big) \leq \underbrace{\left(\rho + \frac{\delta(c - \frac{\delta+1}{2}) - 1}{l}\right)}_{\rho'}^{\delta(c - \frac{\delta+1}{2})}.$$

Recall that, by assumption, ρ is a fixed constant and $l = \exp\{\Theta(c)\}$. Thus, for sufficiently large c, we have

$$\mathbb{P}\big(C_0 \subset \mathcal{E} \big| C_1\big) \leq (\rho')^{\delta(c - \frac{\delta+1}{2})} \sim \rho^{\delta(c - \frac{\delta+1}{2})},$$

where ρ' approaches ρ as $c \to \infty$ since, from (6), l grows exponentially as $c \to \infty$.

Now, fix $x_0 \in \mathcal{U}$ (equivalently, fix the clique C_0 or the word $\boldsymbol{w}_0 = \boldsymbol{w}(x) \in [l]^c$). For a random element $x_1 \in \mathcal{U}$,

$$\mathbb{P}\big(\mathsf{d}(x, x_1) = \delta > 0\big) \leq \binom{c}{\delta} \cdot \left(\frac{1}{l}\right)^{c-\delta} \left(\frac{l-1}{l}\right)^{\delta}.$$

Since $\frac{l-1}{l} < 1$, it follows that the joint probability that $C_1 \subset \mathcal{E}$, $C_0 \subset \mathcal{E}$, and $\mathsf{d}(C_0, C_1) = \delta$ is at most

$$\binom{c}{\delta} \cdot \left(\frac{1}{l}\right)^{c-\delta} \rho^{\delta(c - \frac{\delta+1}{2})}.$$

There are m elements $x_1, \ldots, x_m$ in S, and we assume each is drawn independently and uniformly from $\mathcal{U}$. Therefore, applying the union bound, the probability that at least one x_i is at Hamming distance δ from x_0 is at most

$$m \cdot \binom{c}{\delta} \cdot \left(\frac{1}{l}\right)^{c-\delta} \rho^{\delta(c - \frac{\delta+1}{2})}.$$

Since

$$l \sim \rho^{-K(c-1)/2},$$

and since $m \sim \log(\frac{1}{1-\rho}) l^2$ (cf. Equation (3)), we conclude that the probability that there is an element x_i in S at distance δ from x_0 and that $C_0 \subset \mathcal{E}$ is bounded from above by

$$m \cdot \binom{c}{\delta} \cdot \left(\frac{1}{l}\right)^{c-\delta} \rho^{\delta(c - \frac{\delta+1}{2})} < \log\left(\frac{1}{1-\rho}\right) \cdot \binom{c}{\delta} \rho^{K\frac{(c-1)(c-\delta-2)}{2} + \delta c - \frac{\delta(1+\delta)}{2}}.$$

There are l^c possible ways to choose x_0. By the union bound, we obtain that the probability of creating a false positive clique $C_0 \in \mathcal{E}$ at distance δ from one of the cliques $C_1, \ldots, C_m$ corresponding to the set S is upper bounded by

$$\begin{aligned} & l^c \cdot \log\left(\frac{1}{1-\rho}\right) \cdot \binom{c}{\delta} \rho^{K\frac{(c-1)(c-\delta-2)}{2} + \delta c - \frac{\delta(1+\delta)}{2}} \\ &= \log\left(\frac{1}{1-\rho}\right) \cdot \binom{c}{\delta} \rho^{K\frac{(-\delta-2)(c-1)}{2} + \delta c - \frac{\delta(1+\delta)}{2}} \\ &= \log\left(\frac{1}{1-\rho}\right) \cdot \rho^{\log_\rho 2 \cdot h_2(\delta/c) \cdot c - K\frac{(\delta+2)(c-1)}{2} + \delta c - \frac{\delta(1+\delta)}{2}}, \end{aligned}$$

where $h_2(\cdot)$ denotes the binary entropy function.

Since $0 < \rho < 1$, to show that this last expression goes to zero exponentially quickly as $c \to \infty$, it suffices to show that the following holds:

$$\log_\rho 2 \cdot h_2(\delta/c) \cdot c - K\frac{(\delta+2)(c-1)}{2} + \delta c - \frac{\delta(1+\delta)}{2} \geq \frac{\delta c}{12}. \tag{8}$$

[Note that the right hand side of (8) approaches infinity for any $\delta > 0$ as $c \to \infty$.]

It remains to show that (8) holds. Since $K < \frac{2}{3}$ and since $1 \leq \delta \leq c$, we have

$$\begin{aligned} &-K\frac{(\delta+2)(c-1)}{2} + \frac{11}{12}\delta c - \frac{\delta(1+\delta)}{2} \\ &> -(\frac{(\delta+2)c}{3} + \frac{11}{12}\delta c - \frac{\delta(\delta+1)}{2} \\ &\geq \frac{7}{12}\delta c - \frac{2}{3}c - \frac{\delta^2}{2} - \frac{\delta}{2} \\ &\geq \frac{\delta c}{12}, \end{aligned}$$

where the last inequality holds based on the following reasoning: if $\delta = \Omega(c)$ (and $\delta \leq c$ by definition), then the two dominating terms are $\frac{7}{12}\delta c > \delta^2/2$, and the other two terms are small; if $\delta = o(c)$, then $\frac{6}{12}\delta c > \frac{2}{3}c$ (since $\delta \geq 2$) and the two other terms are small compared with $\frac{1}{12}\delta c$. Moreover, for any δ,

$$\log_\rho 2 \cdot h_2(\delta/c) \cdot c + \frac{c\delta}{12}$$

approaches infinity as $c \to \infty$. Therefore, (8) holds, and so we have shown that

$$\mathbb{P}\big(C(x_0) \subset \mathcal{E} \big| x_0 \notin S\big) \leq \exp\{-\Theta(c)\}$$

as $c \to \infty$. ∎

The proof of Theorem 1 now follows by bounding the false positive rate over all possible Hamming distances.

Proof: [Proof of Theorem 1] Lemma 1 gives that the probability of introducing a clique corresponding to an element x_0 at a distance $\mathsf{d}(x_0, S) = \delta$ from S decays exponentially quickly as $c \to \infty$. Taking the union bound over all distances $\delta = 1, 2, \ldots, c$ gives that the probability of introducing a false positive clique into the graph is upper bounded by

$$\sum_{i=1}^{c} \exp\{-\Theta(c)\} = \exp\{-\Theta(c)\}.$$

∎

IV. Extension to Hypergraphs

The SSAMs introduced in Section II are based on a graph construction. Specifically, each element x_i inserted into the memory is represented as a clique in a c-partite graph. In this section we generalize this construction so that the set S is represented by the edge structure of a hypergraph. Specifically, let us consider a q-regular hypergraph $\mathcal{H}$ with the same set of vertices $\mathcal{V}$ as above, and with each hyperedge being a set of q vertices for some positive integer q between 2 and c. Let

$$V(x) = \{v_{1,w_1(x)}, v_{2,w_2(x)}, \ldots, v_{c,w_c(x)}\}$$

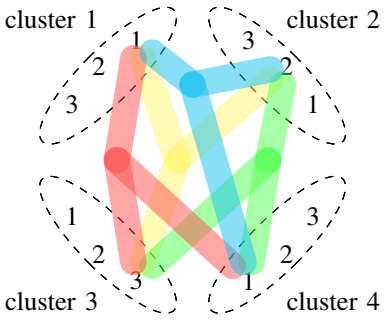

Fig. 2. Illustration of the storing process of the message `1231` in a tridimensional hypergraph ($q = 3$).

denote the vertices implicated by the element $x \in \mathcal{U}$. For $q > 2$, when inserting x into the associative memory, we add all hyperedges corresponding to subsets of q vertices in $V(x)$. In general this corresponds to a hyperclique containing $\binom{c}{q}$ hyperedges. The case $q = 2$ is equivalent to the graph construction described in Section II. Another interesting case is $q = c$, for which there is a bijection between the sets of input messages and the possible networks. Figure 2 shows an example for $c = 4$, $l = 3$, and $q = 3$.

Since $\mathcal{H}$ is still c-partite, and there are l vertices in each partition, and since $\mathcal{H}$ is also q-regular, the memory required to store such a hypergraph is $\binom{c}{q} l^q$ bits. Similar to the graph-based SSAM, it is clear that the hypergraph SSAM (H-SSAM) will not produce false negatives. As more elements are inserted, there will be more hyperedges in the network, and so the false positive rate increases.

The false positive rate for H-SSAMs can be computed using similar arguments to those as in Section III. Here we sketch the main ideas. The probability that a given hyperedge is in $\mathcal{H}$ after inserting m elements is now

$$\rho = 1 - \left(1 - \frac{1}{l^q}\right)^m.$$

Thus, for a fixed density ρ, the number of messages stored should grow like

$$m \approx \log\left(\frac{1}{1-\rho}\right) l^q.$$

The probability that a random message x_0 is accepted by the network is

$$p_e = \rho^{\binom{c}{q}},$$

which is the probability that the corresponding $\binom{c}{q}$ hyperedges are in network after having inserted the m elements. The expected number of messages accepted by the network is

$$m_e = p_e l^c = \exp\left\{\binom{c}{q}\log(\rho) + c\log(l)\right\}.$$

Thus, $m_e = o(1)$ if and only if $\binom{c}{q}\log(\rho) + c\log(l) \to \infty$. Given $c \to \infty$ and fixed $0 < \rho < 1$, a sufficient condition is that

$$\log(l) \sim \frac{K}{c}\binom{c}{q}\log\left(\frac{1}{\rho}\right),$$

where $K < 1$ is a constant. Equivalently, it is sufficient to have

$$l \sim \rho^{-\frac{K}{c}\binom{c}{q}}.$$

Now, let us re-express the number of bits, L, used to represent $\mathcal{H}$ in memory in terms of ρ and K. Since there are $\binom{c}{q}l^q$ possible hyperedges, each of which is either present or not in $\mathcal{H}$, we have that

$$L \sim \binom{c}{q}\rho^{-K\binom{c-1}{q-1}}.$$

Clearly, L could be reduced by using some compression technique, especially if ρ is far from 0.5.

Since $m = \log(\frac{1}{1-\rho})\rho^{-K\binom{c-1}{q-1}}$ and $n = l^c = \rho^{-K\binom{c}{q}}$, it follows that $m = o(n)$. It can be shown that the number of bits required to represent a random set S of size $|S| = m$ of elements drawn from a universe of size $|\mathcal{U}| = n$ is lower bounded by the entropy of such a set which scales as

$$\begin{aligned} H(S) &\sim n\log_2(n) - (n-m)\log_2(n-m) \\ &\sim m\log_2(n) \end{aligned}$$

as $n \to \infty$. Clearly, since $n = l^c$ and $l \geq 2$, we have $n \to \infty$ as $c \to \infty$. Observe that the ratio of L to $H(S)$ obeys

$$\begin{aligned} \frac{L}{H(S)} &\sim \frac{\binom{c}{q}\rho^{-K\binom{c-1}{q-1}}}{K\binom{c}{q}\log_2(\frac{1}{\rho})\log(\frac{1}{1-\rho})\rho^{-K\binom{c-1}{q-1}}} \\ &\sim \frac{\log(2)}{K\cdot\log(\rho)\cdot\log(1-\rho)} \end{aligned}$$

as $c \to \infty$. This ratio is minimum, and hence the H-SSAMs are most efficient, when $\rho = 0.5$. In this case, when $K \to 1$, the memory required by H-SSAMs is within a factor of $1/\log(2) \approx 1.44$ of the optimal value.

Fixing $\rho = 0.5$ and fixing $q \geq 2$, we can derive the number of messages m that gives the best efficiency. It is equal to $\log(2)2^{K\binom{c-1}{q-1}}$. Recall that $n = 2^{K\binom{c}{q}}$. Combining these expressions gives that the corresponding H-SSAM is most efficient (i.e., the number of bits required to represent the H-SSAM is closest to the lower bound) when the number of messages stored is $m = \log(2)n^{\frac{q}{c}} = \Theta(n^{\frac{q}{\log_2(n)^{1/q}}})$.

V. Comparison With Other Set Implementation Techniques

Let us fix $\rho = 0.5$, and recall that, in this case, $m = \Theta(2^{\binom{c-1}{q-1}})$ and $\log_2(n) = \Theta(\binom{c}{q})$. Table 1 compares the complexities of various set implementation techniques expressed as a function of c. In these comparisons we have assumed that each approach is implemented on a machine which performs operations on up to $\log_2(l)$ bits in constant time.

Recall that a Bloom filter is guaranteed to have a false positive rate of no more than ϵ if

$$b \geq m\cdot\log_2(e)\cdot\log_2(1/\epsilon) \quad \text{bits}$$

and $k = \lceil(b/m)\log_2(e)\rceil$ hash functions are used. For the given density $\rho = 0.5$, the false positive rate of the H-SSAM is $2^{-\binom{c}{q}}$. Thus, to achieve a comparable false positive rate while storing the same number of messages, a Bloom filter requires $\Theta(\binom{c}{q}2^{\binom{c-1}{q-1}})$ bits of memory, which is of the same order as that required by H-SSAMs. Hence, Bloom filters and H-SSAMs are equivalent from the perspective of complexity.

VI. Conclusion

This paper proposes the use of sparse structured associative memories (SSAMs) as a mechanism for efficient representation for sets when one needs to perform set membership queries and some errors are tolerable. The basic construction represents the set of elements in terms of the edge structure of a graph. We also propose an extension where the set is encoded in the structure of a hypergraph (H-SSAMs), and we show that this approach has advantages in terms of reduced error rate in appropriately large regimes.

In comparison to Bloom filters, SSAMs and H-SSAMs have the advantage that they can be used in situations where error correction is required. For example, if the queried element x is represented as a string of c symbols from an alphabet of size l, if some symbols are erased or corrupted, Bloom filters are no longer applicable (since the entire description of x is needed for hashing). Associative memories, on the other hand, are designed to allow for recovery of corrupted or missing data, and their ability to correct for errors is studied in [6], [7].

SSAMs and H-SSAMs may be of interest for use in distributed applications, e.g., by storing the elements of the adjacency matrix/tensor at different agents. In this manner, no single agent could independently determine if an element is in the set, and they must cooperate when responding to queries. This may be of interest in distributed databases with sensitive content.

By using hash functions to map elements to random bit indices, Bloom filters provide universal compression of sets in the sense that the false positive rates hold regardless of how the elements of S are drawn. However, the analysis in this paper is conducted under the assumption that elements are drawn uniformly and independently. We are currently investigating the performance of SSAMs under more general distributions on S. One simple way to circumvent correlation is to append the binary representation of each element x with additional bits which are drawn uniformly and independently; of course, this reduces the correlation but also increases the amount of memory required.

Finally, we note that associative memories with structures similar to SSAMs have recently been shown to enjoy nice fault tolerance properties [8], in the sense that the error rates are not severely impacted if the graph structure is perturbed (i.e., edges are randomly added or removed). Achieving this fault tolerance requires a slightly more sophisticated, iterative lookup procedure for performing queries.

In general there are many interesting similarities between Bloom filters and SSAMs, and establishing more concretely the connection between these two structures is the subject of future work.

TABLE I

COMPARISON OF THE COMPLEXITIES OF VARIOUS SET IMPLEMENTATION TECHNIQUES.

Implementation	Average complexity		Worst case complexity		Memory complexity
	Insert	Query	Insert	Query	
Naive array	$\Theta(\binom{c}{q})$	$\Theta(\binom{c}{q}2^{\binom{c-1}{q-1}})$	$\Theta(\binom{c}{q})$	$\Theta(\binom{c}{q}2^{\binom{c-1}{q-1}})$	$\Theta(\binom{c}{q}2^{\binom{c-1}{q-1}})$
Hash table	$\Theta(\binom{c}{q})$	$\Theta(\binom{c}{q})$	$\Theta(\binom{c}{q})$	$\Theta(\binom{c}{q}2^{\binom{c-1}{q-1}})$	$\Theta(\binom{c}{q}2^{\binom{c-1}{q-1}})$
Trie	$\Theta(\binom{c}{q})$	$\Theta(\binom{c}{q})$	$\Theta(\binom{c}{q})$	$\Theta(\binom{c}{q})$	$\Theta(\binom{c}{q}\binom{c-1}{q-1}2^{\binom{c-1}{q-1}})$
Self-balancing binary search tree	$\Theta(\binom{c}{q}\binom{c-1}{q-1})$	$\Theta(\binom{c}{q}\binom{c-1}{q-1})$	$\Theta(\binom{c}{q}\binom{c-1}{q-1})$	$\Theta(\binom{c}{q}\binom{c-1}{q-1})$	$\Theta(\binom{c}{q}\binom{c-1}{q-1}2^{\binom{c-1}{q-1}})$
Bloom filter	$\Theta(\binom{c}{q})$	$\Theta(\binom{c}{q})$	$\Theta(\binom{c}{q})$	$\Theta(\binom{c}{q})$	$\Theta(\binom{c}{q}2^{\binom{c-1}{q-1}})$
H-SSAM	$\Theta(\binom{c}{q})$	$\Theta(\binom{c}{q})$	$\Theta(\binom{c}{q})$	$\Theta(\binom{c}{q})$	$\Theta(\binom{c}{q}2^{\binom{c-1}{q-1}})$

ACKNOWLEDGEMENTS

This work was funded in part by the European Research Council (ERC AdG 2011 NEUCOD), the Natural Sciences and Engineering Research Council of Canada (NSERC), and the *Fonds Québécoos de la recherche sur la nature et les technologies* (FQRNT).

REFERENCES

[1] A. Papadogiannakis, M. Polychronakis, and E. P. Markatos, "Improving the accuracy of network intrusion detection systems under load using selective packet discarding," in *Proceedings of the Third European Workshop on System Security*, ser. EUROSEC '10, New York, NY, USA, 2010, pp. 15–21.

[2] C. S. Lin, D. C. P. Smith, and J. M. Smith, "The design of a rotating associative memory for relational database applications," *ACM Trans. Database Syst.*, vol. 1, pp. 53–65, Mar. 1976.

[3] A. Rajaraman and J. Ullman, *Mining of Massive Datasets*. Cambridge University Press, 2012.

[4] B. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[5] A. Broder and M. Mitzenmacher, "Network applications of Bloom filters: A survey," *Internet Mathematics*, vol. 1, no. 4, pp. 485–509, 2005.

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

[7] ——, "Nearly-optimal associative memories based on distributed constant weight codes," in *Proceedings of Information Theory and Applications Workshop*, San Diego, CA, USA, February 2012, pp. 269–273.

[8] F. Leduc-Primeau, V. Gripon, M. Rabbat, and W. Gross, "Fault-tolerant associative memories based on clustered graphs," May 2013, submitted.