/N-60
[illegible]
43035
p. 42

A CLASS OF DESIGNS FOR A SPARSE DISTRIBUTED MEMORY

Louis A. Jaeckel

July 1989

Research Institute for Advanced Computer Science
NASA Ames Research Center

RIACS Technical Report 89.30

NASA Cooperative Agreement Number NCC 2-408 and NCC 2-387

RIACS

Research Institute for Advanced Computer Science
An Institute of the Universities Space Research Association

A CLASS OF DESIGNS FOR A SPARSE DISTRIBUTED MEMORY

Louis A. Jaeckel

Research Institute for Advanced Computer Science
NASA Ames Research Center

RIACS Technical Report 89.30
July 1989

Abstract. This report describes a general class of designs for a Sparse Distributed Memory. I show that Kanerva's original design and the *selected-coordinate design*, described in a previous report, are related, and that there is a series of possible *intermediate designs* between those two designs. In each such design the set of addresses that activate a memory location is a sphere in the address space. We can also have *hybrid designs*, in which the memory locations may be a mixture of those found in the other designs. In some applications the bits in the read and write addresses that will actually be used might be mostly zeros; that is, the addresses might lie on or near a hyperplane in the address space. I describe a *hyperplane design*, which is adapted to this situation, and I compare it to an adaptation of Kanerva's design. To study the performance of these designs, I compute the expected number of memory locations activated by both of two addresses. The hardware embodiments for the selected-coordinate design, described in the previous report, can be modified for the various designs in this report. More generally, if the addresses are not distributed uniformly in the address space, it may be possible to define the memory locations in a way that is better adapted to the actual distribution of the addresses. These designs are the subject of a recent patent application.

Work reported herein was supported in part by Cooperative Agreement NCC 2-408 and NCC 2-387 between the National Aeronautics and Space Administration (NASA) and the Universities Space Research Association (USRA).

A CLASS OF DESIGNS FOR A SPARSE DISTRIBUTED MEMORY

INTRODUCTION

In a previous report (Jaeckel, 1989), I described a new design for a Sparse Distributed Memory, called the *selected-coordinate design*, and I compared it to Kanerva's (1988) original design. In this report I will define a more general class of possible designs, which includes both of the above designs. Since I will draw upon the ideas and the terminology found in the earlier report, this report may be viewed as a sequel to it.

The designs described in this report are like Kanerva's design in that they consist of a large number of memory locations, called *hard locations*, each of which may be activated by many different addresses in a very large address space. The difference between these designs is in how the set of addresses that activate a particular hard location is defined. Other aspects of these designs, such as the counters at the hard locations and the method of storing and retrieving data in the memory, are the same as in Kanerva's design.

Let the address space S be the set of all n-dimensional binary vectors. S is the set of all possible read and write addresses. In any of the designs described in this report, a memory location can be described or represented by giving the set of read or write addresses that would activate it. Conversely, for each possible read or write address, there is a corresponding subset of memory locations, consisting of those locations that would be activated by that address.

In Kanerva's design, a memory location is activated by any read or write address that is within a fixed Hamming distance of the address of the memory location. In the selected-coordinate design, each memory location is defined by specifying ten (or some other number of) *selected coordinates* (bit positions in the address vectors) and a set of corresponding *assigned values*, consisting of one bit for each selected coordinate. A memory location is activated by an address if, for all of the location's selected coordinates, the corresponding bits in the address vector match the respective assigned values, regardless of the other bits in the address vector.

In this report I will show that Kanerva's original design and the selected-coordinate design are related, and that there is a series of possible designs intermediate between the two. In the selected-coordinate design and in each such *intermediate design*, the set of addresses that activate a memory location is a sphere in S, the address space. However, the center of the sphere is not a point in S, as in Kanerva's design; it is a point in the n-dimensional vector space Σ in which S is embedded.

For each $q < n$, and for a given radius for the spheres, there is an intermediate design. Each memory location in such an intermediate design is defined by specifying a set of q

selected coordinates and corresponding assigned values, as in the selected-coordinate design; these determine the center of the sphere of addresses that would activate the location. A common radius is chosen for the spheres so that each memory location is activated by a desired proportion of the addresses in S. The result is that a memory location is activated by a read or write address if, among the location's selected coordinates, at least a certain number of the corresponding components of the address vector match the assigned values. If $q = n$, we have Kanerva's design, and if q is small and a memory location is activated only if the address bits match the assigned values for all of the selected coordinates, we have the selected-coordinate design.

The performance of the various designs may be studied by computing the expected number of hard locations in the *access overlap*, that is, the set of memory locations activated by both of two addresses. The size of the access overlap is a function of the Hamming distance between the two addresses. I will show how to compute this quantity for the intermediate designs.

It is also possible to construct a ***hybrid design***, in which each memory location may have a different value of q and a different radius. The proportion of addresses that would activate a memory location need not be the same for all locations.

I will also show how the various hardware embodiments for the selected-coordinate design, described in Jaeckel (1989), can be modified for these intermediate and hybrid designs.

In some applications, the read and write addresses that we will actually encounter may not be distributed uniformly throughout S. For example, the bits in the addresses used might be mostly 0's, together with a small percentage of 1's; that is, the addresses might lie on or near a hyperplane in S. I will describe a design, called the ***hyperplane design***, that is adapted to this situation. In this design, a memory location is defined by three (or some other number of) selected coordinates, and it is activated by an address if the components of the address vector are 1's for all of the location's selected coordinates. I then show how the hardware embodiments for the selected-coordinate design can be modified — and simplified — for the hyperplane design.

This design is then compared with a modification of Kanerva's design. In this ***adapted Kanerva design***, the addresses of the hard locations are chosen from the same subset of S that contains the addresses to be encountered — in this case a hyperplane in S. We can compare the two designs by computing the size of the access overlap for each. As in Jaeckel (1989), we can then compute estimates of comparative memory capacities for these two designs.

We can also define intermediate and hybrid hyperplane designs, analogous to those mentioned above. The hardware embodiments can be further modified for these designs. I will also show that the adapted Kanerva design above is the same as one of these intermediate designs.

Finally, I discuss in more general terms the possibility that the addresses that we will actually encounter in a particular application will not be distributed uniformly throughout S — for example, the hyperplane situation above. Keeler (1988) has suggested modifying Kanerva's design by choosing the addresses of the hard locations at random from the same frequency distribution that the read and write addresses are expected to follow. I will discuss some modifications that can be made in the various other designs. It may be possible to choose the information defining the hard locations in a way that is better adapted to the actual distribution of the addresses. No changes in the hardware embodiments are required, since they may be used with any choice of hard locations.

The designs and hardware embodiments in this report and in Jaeckel (1989) are the subject of a patent application and a continuation of that application, filed in 1988 and 1989. The descriptions of the designs and hardware embodiments in this report are based on material in those documents.

To make the exposition more concrete, I will generally assume, as in Jaeckel (1989), that n, the length of the address vectors, is 1000, that one million hard locations are implemented, and that about 1/1000 of them are activated by a given read or write address. These numbers could of course be varied.

INTERMEDIATE AND HYBRID DESIGNS

Although the rule for activating memory locations in the selected-coordinate design is very different from the activation rule in Kanerva's design, if we view these designs geometrically, we can see that they are related. In Kanerva's design the set of read or write addresses that would activate a memory location is a sphere in S centered at the address of the memory location. In the selected-coordinate design, a memory location is activated by an address if, for all of the location's selected coordinates, the corresponding bits in the address vector match the respective assigned values. In that case (assuming $n = 1000$ and ten selected coordinates for each memory location), the subset of S representing a memory location, that is, the set of read or write addresses that would activate it, is actually a sphere, whose center is a point in Σ, the 1000-dimensional vector space in which S, the address space of binary vectors, is embedded.

The center of this sphere is a vector in Σ whose components agree with the assigned values for each of the ten selected coordinates defining the subset, and have the value $\frac{1}{2}$ for each of the other 990 coordinates. The radius of the sphere is 495, where distances are measured using L1 distance. (Since the L1 distance between any two points in S is the same as the Hamming distance, it will be convenient to use L1 distance here.) Any point in S (a vector of 0's and 1's) is at least $990 \cdot \frac{1}{2} = 495$ away from this center point, due to the 990 coordinates for which the component of the center vector is $\frac{1}{2}$. Therefore, those points in S that agree with the center point on all ten selected coordinates are exactly 495 away, and all other points in S are more than 495 from the center. Thus the subsets representing memory locations may be thought of as spheres.

Viewing the two designs in this way, we can see that there is a series of possible *intermediate designs* with Kanerva's design at one end and the selected-coordinate design at the other. For any fixed $q < n$ (n is assumed to be 1000), I will define an n-dimensional vector in Σ to be the center of a sphere in S by selecting q coordinates, assigning values of 0 or 1 to each selected coordinate, and letting all of the other components of the vector have the value $\frac{1}{2}$ (to represent indifference to whether the value of a bit in an address vector is 0 or 1). I will then choose the radius of the sphere so that the sphere contains a given proportion of the points in S, say, about 1/1000 of them. The radius of the sphere, using L1 distance, is $R = (1000 - q)\cdot\frac{1}{2} + r$, where the first term is to account for the $1000 - q$ coordinates for which the center vector has the value $\frac{1}{2}$, and the second term, r, is chosen so that in the q-dimensional subspace of S generated by the q selected coordinates, which I will call T, about 1/1000 of the points are within r of the q-dimensional vector defined by the assigned values for the selected coordinates. The sphere in S with the above center and radius R consists of those points in S for which the q-dimensional Hamming distance between the q assigned values and the corresponding q components of the point is less than or equal to r; the values of the bits for the other coordinates are free to be either 0 or 1. In other words, if we project S into T, a point in S is within R of the center of the sphere in S if and only if its projection in T is within r of the vector of assigned values. Since 1/1000 of the points in T are within r of that vector, and since each point in T represents the same number of points in S as does every other point in T, it follows that the sphere in S contains about $1/1000$ of the points in S.

For example, if $q = 100$, the radius of the sphere would be $900 \cdot \frac{1}{2} + 34 = 484$, since in a 100-dimensional binary vector space a sphere of radius 34 contains about $1/1000$ of that space. (See Kanerva, 1988.)

The numbers used above are examples; the address space could have any dimension n, and the volumes of the spheres could be any desired proportion of the space, or at least any proportion for which there are spheres of that volume. Since S is discrete, there are only a finite number of possible values for r, so we may have to approximate the desired volume of the spheres. (If q is small, there are few choices for r.)

The intermediate design for a given q and a given r is defined by considering all of the spheres of the kind described above, for those given values. That is, all of the spheres have the same radius R, and the center of each sphere is defined as above, based on its q selected coordinates and the corresponding assigned values. Each such sphere in S represents a potential memory location, in the sense that the points in the sphere constitute the set of addresses that would activate the location. In other words, a read or write address activates a memory location if, among the location's selected coordinates, at most r of the corresponding components of the address vector do not match the assigned values. This is equivalent to activating the location if for at least $q - r$ of the selected coordinates, the components of the address vector do match the assigned values. The hard locations would then be chosen at random from the set of all potential memory locations. The number of such spheres, that is, the number of potential memory locations, is

$$\binom{1000}{q} \cdot 2^q ,$$

where the first term is the number of ways of choosing q selected coordinates and the second term is the number of ways of choosing q assigned values. If $q = n$, we have Kanerva's design, and if q is small and $r = 0$, we have the selected-coordinate design. (It is the condition $r = 0$ that characterizes the selected-coordinate design.)

Suppose we have chosen a value of q, and also a value of r so that each of the spheres representing the potential memory locations contains about $1/1000$ of S. Then, for a given address x, that is, a point in S, the class of potential memory locations activated by it is the class of spheres containing it. I will show that this class consists of about $1/1000$ of the total number of potential memory locations. For any given set of q selected coordinates, let A be the q-dimensional binary vector space of all possible sets of assigned values for those selected coordinates. Let x' be the vector in A defined by the q components of x corresponding to the selected coordinates. Consider the sphere in A with center x' and radius r. Because of the way r was chosen, this sphere contains about 1/1000 of the points in A; that is, 1/1000 of the possible vectors of assigned values are within r of x'. Each vector in this sphere in A defines a potential memory location that is activated by x. Therefore, since this argument applies to any choice of selected coordinates, about 1/1000 of the potential memory locations are activated by x. The fact that the proportion of memory locations activated by an address is the same as the proportion of addresses that would activate a memory location can also be seen from the general result derived in Jaeckel (1989), p. 18. Thus, since the hard locations are chosen at random, a point in S, acting as an address, will activate about 1/1000 of the hard locations, more or less. The actual number will be different for different addresses.

One way to construct an embodiment of an intermediate design is to have an address decoder for each hard location, similar to those described in Jaeckel (1989), p. 7-8 for Kanerva's design. In the intermediate design, an address decoder for a hard location would have q inputs for the q selected coordinates defining that hard location. The assigned values and the radius r would be stored in the address decoder. Given a read or write address, it would compute the q-dimensional Hamming distance between the assigned values for those q coordinates and the corresponding components of the address vector. In other words, it would count the number of selected coordinates for which the component of the address vector does not match the assigned value. If that distance is less than or equal to r, the radius of a q-dimensional sphere containing 1/1000 of the q-dimensional subspace T, the location would be activated. For example, if $q = 100$, the location would be activated if the q-dimensional distance is less than or equal to 34. The components of the address vector for the other coordinates are not used in determining whether to activate the location.

Given two addresses x and y in S with Hamming distance $d(x,y) = d$ between them, we need to know the size of the *access overlap*, that is, the number of potential memory locations that are activated by both points. Since the hard locations are a random sample of the potential memory locations, the expected number of hard locations in the access overlap will be proportional to this number. As in Kanerva (1988) and Jaeckel (1989), the performance of the system may be judged by the size of the access overlap as a function of $d(x,y)$. Select a set of q coordinates. This may be done in $\binom{1000}{q}$ ways. Let k be the number of the q selected coordinates that are among the coordinates on which x and y differ. Since the components of x and y differ on d coordinates and agree on $1000 - d$ coordinates, the number of ways of selecting q coordinates so that k has a given value is

$$\binom{d}{k} \cdot \binom{1000-d}{q-k} .$$

The first term is the number of ways of choosing k coordinates from among d, and the second is the number of ways of choosing the others.

For a given set of q selected coordinates, let A be the space of all possible vectors of assigned values as above, and let x' and y' be the q-dimensional vectors in A defined by the components of x and y, respectively, corresponding to the selected coordinates. The q-dimensional Hamming distance between x' and y' is k, the quantity defined above. Among the potential memory locations defined by the points in A (for the given set of selected coordinates), the number that are activated by both x and y is the number of points in A that are within r of both x' and y'. This number is the volume of (the number of points in) the intersection of the two q-dimensional spheres of radius r whose centers are x' and y'. Let $V(q,r,k)$ be the volume of this intersection, where $k = d(x',y')$. A formula for this volume is derived in Kanerva (1988). The total number of potential memory locations activated by both x and y, for all sets of q selected coordinates, is therefore

$$\sum_k \binom{d}{k} \cdot \binom{1000-d}{q-k} \cdot V(q,r,k) \ .$$

If q and d are large, this sum may be approximated as follows: If we think of the q coordinates as being selected at random, then k is a random variable with a hypergeometric distribution, and the expected value of k is $qd/1000$. For large q and d, k will be near its expected value with high probability; therefore, a rough approximation to the sum above is

$$\binom{1000}{q} \cdot V\left(q, r, \frac{qd}{1000}\right) \ .$$

If we express this number as a fraction of the number of locations activated by x, which is $1/1000$ of the total number of potential memory locations, we have

$$\frac{1000\cdot V\left(q,r,\frac{qd}{1000}\right)}{2^q} .$$

Approximate values of this fraction for various values of d may be found in the middle column of Table 1.3 of Kanerva (1988), p. 24, which corresponds to spheres whose volume is $1/1000$ of the space. Since the rows in that table are labeled by the distance relative to the dimension of the space, the row corresponding to a given value of d is the one labeled with the value of $\frac{qd}{1000} \div q = \frac{d}{1000}$. This is the same row in the table that we would use to find the access overlap in Kanerva's design for $n = 1000$ and a given d. Thus we can see from the table that for large q and d the performance of the intermediate design is close to that of Kanerva's design, in the sense that if two addresses are a distance d apart, then the relative number of locations activated by both points is similar in the two designs.

These intermediate designs can be constructed by modifying the various hardware embodiments described in Jaeckel (1989) for the selected-coordinate design. As mentioned above, each hard location could have an address decoder with q inputs, which would compute a q-dimensional Hamming distance, that is, the number of selected coordinates for which the assigned value disagrees with the corresponding component of the address vector, and activate the location if this number is less than or equal to r. Alternatively, the address decoder could count the number of matches between the assigned values and the corresponding components of the address vector, and activate the location if this number is greater than or equal to $q - r$. Note that since the complexity of these address decoders depends only on q, they can be fairly simple devices even if n, the dimension of the address space, is very large.

For the *first embodiment* of the selected-coordinate design, described in Jaeckel (1989), p. 35, we can modify the address module as follows: The two parts of the hard-address unit would remain the same; two 256-bit data words for each hard location, containing the assigned-value and selected-coordinate information in the same format as before, would be stored there. The logic unit would be somewhat more complex. As before, it would perform an exclusive-or (XOR) of the reference (read or write) address and the word containing the assigned values, and then perform a logical AND of the result of the XOR and the word containing the selected coordinates. Among the resulting 256 bits, there would be 1's for those selected coordinates for which the assigned-value bit disagrees with the corresponding reference-address bit; all of the other bits would be 0's. But instead of simply testing these bits for 0's, the logic unit would add them and compare the sum to r; if the sum is less than or equal to r, the hard location would be activated. The logic unit would thus require a means to add the bits. The *Stanford prototype* (Flynn et al., 1988), a small-scale embodiment of Kanerva's design, includes a set of adders to compute such a sum quickly; similar circuitry could be used here. Alternatively, the logic unit could count the number of matches between these bits and activate the location if that number is greater than or equal to $q - r$. In that case, the bits to be added would have to be such that there is a 1 for each selected coordinate for which the corresponding bits match. This could be done either by storing the complements of the assigned values in the hard-address unit, or by having an element that computes the complement of the XOR, instead of the XOR.

The address module of the *second embodiment* (Jaeckel, 1989, p. 38) can be modified as follows: The hard-address unit would store the selected-coordinate and assigned-value information as before, using two bytes for each selected coordinate for each hard location. It would thus require $2q$ bytes of memory for each hard location, instead of 20. For each selected coordinate, the logic unit would compare the assigned-value bit to the corresponding bit of the reference address, as before. But instead of activating a memory location only if all of the bits match, it would, for each hard location, count the number of selected coordinates for which the assigned-value bit disagrees with the corresponding reference-address bit. If this number is less than or equal to r, the hard location would be activated. Or, as above, the logic unit could count the number of matches and activate the hard location if this number is greater than or equal to $q - r$.

Finally, we could have a *hybrid design*, in which different hard locations could have different values of q and r, that is, different numbers of selected coordinates and different radii. The Hamming radius r for each hard location would depend on the value of q for that location, and also on the desired number of the addresses in S that would activate the location. This number of addresses need not be the same for all hard locations; we may want some locations to be activated by more addresses and others by fewer addresses. Such a design would combine the characteristics of the various intermediate designs. The value of q for some of the hard locations could be equal to n; if $q = n$ for all of the hard locations and if they all have the same value of r, the design would be the same as Kanerva's design. To compute the expected number of hard locations in the access overlap for two addresses, we would use the formulas above to find the expected number for each combination of values of q and r used in the design, and then take the sum over all of these combinations of q and r.

This hybrid design is the most general of the designs described in this report. It includes as special cases Kanerva's design, the selected-coordinate design, the intermediate designs, and all of the various hyperplane designs in the following sections.

The modified hardware embodiments described above could be further modified to allow for the possibility of different values of q and r for different hard locations. For example, if each hard location has an address decoder, each address decoder could have its own values for q and r. In the first embodiment of the intermediate design, described above, the address module would have to store the value of r (or $q - r$) for each hard location. In the second embodiment of the intermediate design above, the address module would have to store the value of q for each hard location so that the logic unit would know how many selected coordinates to check. It would also have to store the value of r (or $q - r$) for each hard location. (If for all q, all of the hard locations with a given value of q are to have the same value of r, then in either embodiment we could store q for each hard location and also store a table containing the value of r (or $q - r$) for each value of q.)

THE HYPERPLANE DESIGN

Suppose that in a particular application, all of the binary vectors actually used as read or write addresses are such that only a small proportion of their bits, say approximately 10%, are 1's and the rest are 0's. This situation is somewhat analogous to a model of a neural network in which there are, say, 1000 input neurons, about 10% of which are firing at any one time. Thus, an input vector would be a point in S, with a value of 1 for a coordinate meaning that the corresponding excitatory input neuron is firing, and a value of 0 meaning that it is not firing. The proportion of input neurons firing at any one time might be held to about 10% by the action of inhibitory neurons. Two input vectors would be considered similar to each other if many of the neurons represented as firing by one vector were also shown as firing by the other, that is, if their logical AND contained many 1's. This concept of the input to a neural network, and the hyperplane design described below, bear some resemblance to Marr's (1969) model of the cerebellum.

The set of binary vectors in S that contain a fixed number of 1's among their components (e.g., 100 out of 1000) forms a hyperplane in S. Thus, the addresses we expect to encounter in this situation lie on or near a hyperplane in S, rather than being distributed throughout S, as was assumed for the previous designs. Although I will sometimes assume for simplicity that the addresses are restricted to a hyperplane, the results below that are based on this assumption will be approximately true if all of the addresses are at least near a given hyperplane. The general situation in which the addresses are not distributed uniformly in S will be discussed in a later section.

The *hyperplane design* is a design for a Sparse Distributed Memory that is adapted to this situation. To describe the design, I will designate a collection of subsets of S to represent potential memory locations; that is, each of these subsets is the set of read or write addresses that would activate a potential memory location. I will assume that $n = 1000$, that about 10% of the bits in each address are 1's, that one million of the potential memory locations, chosen at random, are implemented as hard locations, and that each memory location is to be activated by about 1/1000 of the addresses. As with the previous designs, these numbers could be varied.

In the next section I will describe an adaptation of Kanerva's design to this situation, and I will compare it to the hyperplane design by computing, for each design, the size of the access overlap for two addresses.

I will define a subset of S by selecting three of the 1000 coordinates. The subset consists of all points in S that have the value 1 for all three of the selected coordinates, and any values for the other coordinates. The memory location represented by such a subset is activated by any address vector whose components for all three of the location's selected coordinates are 1. (This is somewhat analogous to an associative memory in which all of the stored patterns containing a given three items are recalled whenever the stimulus contains those three items.) The number of subsets of this type is

$$\binom{1000}{3} = 166{,}167{,}000 \ .$$

A random sample of the potential memory locations would be implemented as hard locations. These subsets are like the spherical subsets in the selected-coordinate design, except that there are only three selected coordinates for each subset, and the assigned values for the selected coordinates are always 1.

For any binary vector x in S, let h(x) be the number of 1's in x. Let C be the set of all points in S with h(x) = 100. C is a hyperplane in S, consisting of all 1000-dimensional binary vectors containing 100 1's and 900 0's. The total number of points in C is $\binom{1000}{100}$. If we assume that all of the read and write addresses must lie in C, then it is easy to compute the proportion of the addresses that would activate a memory location. If a vector in C lies in a given subset of the kind described above, then its components for the three selected coordinates defining the subset are 1, and it contains 97 other 1's distributed among the remaining 997 coordinates. Since the number of ways of placing 97 1's among those 997 coordinates is $\binom{997}{97}$, this is the number of points in C that lie in the subset. Therefore, the proportion of the addresses in C that would activate the memory location is .000973, or just under 1/1000.

Note that I am using three selected coordinates here because I assumed that about 10% of the bits in the address vectors are 1's, and that a location is to be activated by about 1/1000 of the addresses. If these numbers are changed, I would use a different number of selected coordinates to define a memory location.

For any x in S, the number of potential memory locations activated by x, that is, the number of subsets it belongs to, is

$$\binom{h(x)}{3} ,$$

since this is the number of ways of choosing three coordinates for which the components of x are all 1. If x is in C, that is, if $h(x) = 100$, this number is .000973 of the total number of potential memory locations. This is exactly the same as the proportion found above. As with the intermediate designs, the equality of these proportions can also be seen from the general result in Jaeckel (1989), p.18. If we assume that there are one million hard locations chosen at random, then the expected number of hard locations activated by an address x in C is

$$\frac{1{,}000{,}000}{\binom{1000}{3}} \cdot \binom{100}{3} = 973 \ .$$

We do not have to restrict the addresses to a single hyperplane; instead, we could require only that all of the addresses lie within a certain distance of C. Then, if $h(x) > 100$, the proportion of memory locations activated by x would be greater than the amount found above, and if $h(x) < 100$, the proportion would be less. In this case the design would differ from the other designs considered so far, in that the number of potential memory locations activated by an address would not be the same for all addresses. But even in this case the general result in Jaeckel (1989), p. 18, shows that the proportion of the memory locations activated by a "typical" address is the same as the proportion of the addresses that would activate a "typical" memory location.

We will see that in this design the size of the access overlap for two address vectors depends on the number of 1's in the logical AND of the two vectors, so I will use that as a measure of their similarity, rather than using the Hamming distance. I will begin by showing how the two are related.

For any two points x and y in S, let $A(x,y)$ be the number of 1's in the logical AND of x and y, that is, the number of coordinates for which both x and y are 1. Since the Hamming distance $d(x,y)$ is the sum of the number of coordinates for which x is 1 and y is 0, of which there are $h(x) - A(x,y)$, plus the number for which y is 1 and x is 0, of which there are $h(y) - A(x,y)$, we have:

$$\begin{aligned} d(x,y) &= [h(x) - A(x,y)] + [h(y) - A(x,y)] \\ &= h(x) + h(y) - 2\cdot A(x,y) \ . \end{aligned}$$

Therefore,

$$A(x,y) = \frac{1}{2}\cdot[h(x) + h(y) - d(x,y)] \ .$$

If $h(x)$ and $h(y)$ are both approximately 100, then

$$A(x,y) \approx 100 - \frac{d(x,y)}{2} \ ,$$

or

$$d(x,y) \approx 200 - 2\cdot A(x,y) \ .$$

If both x and y are in C, these equations are exact. Thus, for the points in S with which we are concerned, A and d are closely related, and I will use A as a measure of similarity instead of d.

I will now show that for two given points in S, such as a write address x and a read address y, the number of potential memory locations activated by both of them is a function of $A(x,y)$. A memory location is activated by both x and y if the components of both vectors are 1 for all three of the selected coordinates defining that memory location. Since $A(x,y)$ is the number of coordinates for which both x and y are 1, the number of ways of choosing three coordinates for which both are 1 is

$$\binom{A(x,y)}{3} \ .$$

Hence this is the number of potential memory locations activated by both x and y, that is, the size of the access overlap. If there are one million hard locations chosen at random, the expected number of hard locations activated by both x and y is

$$\frac{1{,}000{,}000}{\binom{1000}{3}} \cdot \binom{A(x,y)}{3} .$$

The third column in Table 1 in the next section gives this expected number for several values of A(x,y). Note that x and y here may be any points in S.

The various hardware embodiments described in Jaeckel (1989) for the selected-coordinate design can easily be adapted to the hyperplane design. Since the hyperplane design is like the selected-coordinate design, except that each memory location is defined by three (or some other number of) selected coordinates and the assigned values are always 1, any embodiment for the selected-coordinate design can be used for the hyperplane design without modification, as long as it can function with hard locations defined by the required number of selected coordinates. Moreover, if we are interested only in the hyperplane design, those embodiments can be simplified, because they would not have to store assigned values or perform an exclusive-or. For example, if we have an address decoder for each hard location, each address decoder would have one input for each selected coordinate, to receive the corresponding bit in the read or write address; it would perform a logical AND of those bits, and activate the location if all of them are 1's. As in the earlier designs, these address decoders would be simple devices even if n is very large.

We can modify the *first embodiment* of the selected-coordinate design (Jaeckel, 1989, p. 35), by omitting the assigned-value part of the hard-address unit and the XOR part of the logic unit. Since we can think of the hyperplane design as having all assigned values equal to 1, we do not need to store them. Since the exclusive-or of the reference (read or write) address with a vector of 1's is the complement of the reference address, one way to implement the design is to take the complement of the reference address at the beginning of a read or a write operation and hold that vector in the reference-address register. Then we can proceed as in the embodiment for the selected-coordinate design: For each hard location, the logic unit would perform a logical AND of the complemented reference address with the 256-bit mask from the hard-address unit (a vector containing 1's at the positions of the selected coordinates and 0's for the other coordinates). If all 256 of the resulting bits are 0's, the hard location is to be activated. An alternative implementation is this: Instead of complementing the reference address, the complements of the selected coordinate masks (that is, 0's for the selected coordinates and 1's for the others) could be stored in the hard-address unit. Then, for each hard location, the logic unit would perform a logical OR of the reference address with the complemented mask; if all of the 256 resulting bits are 1's, the location is to be activated.

The *second embodiment* of the selected-coordinate design (Jaeckel, 1989, p. 38) can be simplified as follows: If we use two bytes to store the number of each selected coordinate, the selected-coordinate information for each hard location can be stored in six bytes in the hard-address unit (assuming three selected coordinates per hard location). Since we do not need to store assigned values, we can use addresses consisting of as many as $2^{16} = 65,536$ bits. The operation of the logic unit would be like that for the selected-coordinate design, except that it would not have to separate the assigned value from the number of the selected coordinate, and it would compare bits in the reference address to 1, rather than to an assigned value.

COMPARISON OF DESIGNS

In this section I will compare the hyperplane design to an adaptation of Kanerva's design to the hyperplane case.

In the situation where $h(x)$ is approximately 100 for all possible addresses, neither Kanerva's design nor the selected-coordinate design, as they are described in Jaeckel (1989), would perform well. This is because $A(x,y)$ for two such addresses chosen at random would be about 10 (see below), which corresponds to a Hamming distance of about 180. (The maximum Hamming distance between two such addresses is about 200.) For Kanerva's design, using the parameters used in Jaeckel (1989), the expected number of hard locations in the access overlap if $d(x,y) = 180$ is 119 (Kanerva, 1988, Table 7.1, p. 63), and for the selected-coordinate design it is 133 (by the formula in Jaeckel, 1989, p. 20). In either case, this number is more than one tenth of the number of hard locations activated by a single address. The result is a poor signal-to-noise ratio: If all of the addresses are on or near the hyperplane C, and if we try to recover a data word written at address x by reading from the memory at that same address, and if we assume that the other write addresses are randomly distributed on or near C, then the vector of sums computed during the read operation will contain about 1000 copies of the data word to be recovered, along with over 100 copies of each of many other stored words, whose write addresses would be at a distance of about 180 from the read address. On the other hand, if the addresses are distributed uniformly throughout S, in which case the expected distance between two random addresses is 500, then, in either of these designs, the vector of sums would contain, on the average, only one copy of each of the other stored data words. See Table 1 of Jaeckel (1989), p. 22. Consequently, in the present situation, the "noise" produced by even a modest number of stored data words could overwhelm the data word to be recovered. If we read at an address near, but different from, the write address x, the problem is even worse, because the sums would contain fewer copies of the data word to be recovered.

In principle, the signal-to-noise ratio in Kanerva's design could be improved by decreasing the activation radius of the hard locations. (The equivalent to this in the selected-coordinate design would be to increase the number of selected coordinates defining each hard location.) But since the hard locations are chosen at random, the effect of this — unless we increase the total number of hard locations — would be to greatly reduce the expected number of hard locations activated by a single address.

Because of the random choice of hard locations, it would be difficult to recover the stored data reliably if each data word is written to only a small number of memory locations.

In order to have a design like Kanerva's to compare to the hyperplane design, I will describe an adaptation of Kanerva's design to the situation here. (The hyperplane design may be thought of as an adaptation of the selected-coordinate design to this situation.) One way to modify Kanerva's design to adapt it to this situation is to restrict the addresses of the hard locations to the part of the address space that contains the read and write addresses that we will actually encounter, as has been suggested by Keeler (1988), p. 321-24. Since more of the hard locations will then be relatively close to the read and write addresses actually used, the activation radius can be reduced, and the memory should be better able to discriminate between addresses that are near each other. The activation radius and the size of the access overlap will depend on the distribution of the addresses of the hard locations.

I will define the ***adapted Kanerva design*** as follows: As before, C is the hyperplane consisting of all points in S with $h(x) = 100$. The number of points in C is $\binom{1000}{100}$. I will assume for simplicity that all of the read and write addresses to be used lie in C, and that they are distributed uniformly in C.

The set C is also used as the set of addresses of potential memory locations in this design. A random sample of these points will be chosen to be the addresses of the hard locations. A memory location at z in C is activated by any read or write address within a given Hamming distance D of z; that is, the subset of addresses in C that represents the memory location is a sphere with radius D centered at z. Conversely, the set of addresses of potential memory locations activated by a read or write address x in C is a sphere with radius D centered at x. Therefore, the number of addresses that activate a given memory location is the same as the number of potential memory locations that are activated by a given address. D is chosen so that these spheres will contain a desired number of points. I will show how to compute volumes of spheres and intersections of spheres in C, so that we can compute the size of the access overlap for this design. We can then compare the memory capacity of this adapted Kanerva design to that of the hyperplane design.

Choose a point x in C to be the center of a sphere in C. A sphere centered at x is the set of points within a given distance D of x, that is, the set of all z in C such that $d(x,z) \leq D$. (Since the Hamming distance between any two points in C is even, we may assume that D is even.) Since $A(x,z) = 100 - d(x,z)/2$, we can describe the sphere as the set of all z such that $A(x,z) \geq A$, where $A = 100 - D/2$. For any $\alpha \geq A$, the number of points z in C for which $A(x,z) = \alpha$ is

$$\binom{100}{\alpha} \cdot \binom{900}{100-\alpha} ,$$

where the first term is the number of ways of placing α 1's among the 100 coordinates for which the component of x is a 1, and the second term is the number of ways of placing $100 - a$ 1's among the 900 coordinates for which the component of x is a 0. The number of points in the sphere is therefore

$$\sum_{a \geq A} \binom{100}{a} \cdot \binom{900}{100-a} .$$

If $A = 20$ (or $D = 160$), the volume of the sphere is .001070 of C; that is, the sphere contains about 1/1000 of the points in C. Thus, if there are one million hard locations with addresses chosen at random in C, and if a read or write address x in C activates all hard locations whose address z is such that $A(x,z) \geq 20$, then, on the average, an address in C will activate about 1070 hard locations. To compare the designs, I will use the value $A = 20$ to define the activation threshold, that is, the size of the spheres, for the memory locations in the adapted Kanerva design. I use this value so that, for this design and for the hyperplane design, a similar number of hard locations would be activated by an address in C.

Given two read or write addresses x and y in C, with $A(x,y) = U$, I will compute the number of potential memory locations activated by both addresses. That is, I will find the number of points z in C such that both $A(x,z) \geq 20$ and $A(y,z) \geq 20$. For convenience, we can rearrange the order of the coordinates so that for the first U coordinates, the components of both x and y are 1's; for the next $100 - U$ coordinates, x is 1 but y is 0; for the $100 - U$ coordinates following those, x is 0 and y is 1; and for the remaining $800 + U$ coordinates, both x and y are 0. The vectors would then look like this:

```
           U       100 - U    100 - U      800 + U
   x:  11...11  11.....11  00.....00  00...........00
   y:  11...11  00.....00  11.....11  00...........00
           α         β          γ            δ
```

The number of ways of choosing a point z in C with α 1's in the first block of coordinates, β in the second, γ in the third, and δ in the fourth, where $\alpha + \beta + \gamma + \delta = 100$, is

$$\binom{U}{\alpha} \cdot \binom{100-U}{\beta} \cdot \binom{100-U}{\gamma} \cdot \binom{800+U}{\delta} ,$$

where the first term is the number of ways of placing α 1's among U coordinates, and so on. The number of points in the access overlap, that is, in the intersection of the spheres about x and y, is found by summing this expression over all permissible combinations of values for α, β, γ, and δ for which

$$A(x,z) = \alpha + \beta \geq 20$$

and

$$A(y,z) = \alpha + \gamma \geq 20 .$$

Thus the size of the access overlap is a function of A(x,y). If there are one million hard locations, then the expected number of hard locations in the access overlap is found by multiplying the number of points in the intersection by

$$\frac{1{,}000{,}000}{\binom{1000}{100}} .$$

Some representative values are given in the fourth column of Table 1 below.

The second column in the table gives the Hamming distance d(x,y) corresponding to each value of A(x,y), assuming that x and y are in C. The third column gives the expected numbers for the hyperplane design; these numbers are valid for the given values of A(x,y) whether or not x and y are in C. In the fourth column, however, x and y are assumed to be in C.

TABLE 1

Expected number of hard locations activated by both x and y

A(x,y)	d(x,y)	Hyperplane design	Adapted Kanerva
100	0	973	1070
90	20	707	486
80	40	494	284
60	80	206	95
40	120	59	25
20	160	6.9	3.9
10	180	0.72	1.06
0	200	0	0.17

In Jaeckel (1989) some estimates of comparative memory capacities and signal-to-noise ratios were computed for Kanerva's design and the selected-coordinate design, under some assumptions as to the randomness of the data words written to the memory. If the same assumptions are made here, with respect to the hyperplane design and the adapted Kanerva design, then the formulas derived in that report can be applied to the two designs here, and we can compute similar estimates for these designs. I will assume for this comparison that for both designs, all addresses lie in C.

First I will show that if two addresses x and y are chosen at random in C, then the expected value of $A(x,y)$ is 10. For each of the 1000 coordinates, there is a 1/10 chance that the component of x is 1, and, independently, a 1/10 chance that the component of y is 1. Therefore, the probability that both are 1 is 1/100. So the expected number of coordinates for which both are 1 is 1/100 of 1000, which is 10.

In the notation of Jaeckel (1989), t is the number of data words written to the memory, the x_i are the write addresses for those data words, x_1 is the write address of the data word to be recovered by reading from the memory, y is the read address, and λ_1 is the expected number of hard locations in the access overlap of x_1 and y. I assume that y is near x_1, and that for the other write addresses, $A(x_i,y)$ is near 10. Using the variance formulas derived in Jaeckel (1989), p. 29, and the values given in Table 1 above for $A(x,y) = 10$, we find that the approximate variance of the noise for the hyperplane design is

$$\lambda_{1,h} + 1.24(t - 1) ,$$

and for the adapted Kanerva design it is

$$\lambda_{1,a} + 2.18(t - 1) ,$$

where the subscripts h and a denote the two designs here.

We can now estimate comparative memory capacities for the two designs. One way to do this is to compute, for a given value of $A(x_1,y)$, the maximum number of data words that can be stored in the memory, for which there is at least a 99% chance of correctly recovering a given stored data bit. For example, using the method given in Jaeckel (1989), p. 30-31, it can be shown that if $A(x_1,y) = 80$, then the maximum number for the hyperplane design is about 35,854 stored data words, while for the adapted Kanerva design the maximum is about 6,686. That is, under these conditions, more than five times as many data words can be stored in the hyperplane design, compared to the adapted Kanerva design, with the same probability of recovering a stored data bit.

INTERMEDIATE AND HYBRID HYPERPLANE DESIGNS

We saw earlier that there is a series of intermediate designs between Kanerva's design and the selected-coordinate design. Similarly, in the present situation, where I assume that the addresses are restricted to the set C, we can describe a series of intermediate designs related to the hyperplane design and the adapted Kanerva design. For any fixed q and threshold value A, we can define a collection of subsets of C to represent the potential memory locations for an *intermediate hyperplane design* as follows: Define a subset by choosing q selected coordinates; the subset consists of all points in C that have the value 1 for at least A of the selected coordinates, and any values for the other coordinates. The number A is chosen so that the subset will contain a desired proportion of the points in C. (As in the other designs, not all proportions are possible, due to the discreteness of the address space, so we may have to approximate the desired proportion.) The memory location represented by such a subset would be activated by any address in the subset, that is, whenever the components of the address vector for at least A of the selected coordinates are 1's. If we think of the selected coordinates as having assigned values of 1, these subsets are like the spherical subsets in the earlier intermediate designs. The number of potential memory locations is $\binom{1000}{q}$. A random sample of them would be implemented as hard locations.

If $q = A = 3$, we have the above example of the hyperplane design. And if we view the adapted Kanerva design above somewhat differently, we can see that the adapted Kanerva design is the same as the intermediate hyperplane design with $q = 100$ and $A = 20$. That is, the subsets representing its potential memory locations fit the description given above, for these values of q and A. Any z in C is the address of a potential memory location in the adapted Kanerva design; the location is activated by any address x in C for which $A(x,z) \geq 20$. Given a memory location with address z, the corresponding subset in the intermediate hyperplane design is defined by choosing the 100 coordinates for which the value of z is 1 to be the 100 selected coordinates defining a subset of the kind described above; if $A = 20$, that subset is the same as the set of points that activate the location in the adapted Kanerva design whose address is z. Conversely, any subset in the intermediate hyperplane design with $q = 100$ and $A = 20$ corresponds to a potential memory location in the adapted Kanerva design; the address of that location is the vector in C whose components are 1's for the selected coordinates and 0's for the others. Therefore, the collection of subsets representing potential memory locations in the adapted Kanerva design is the same as the collection of subsets above for $q = 100$ and $A = 20$.

We can therefore analyze the intermediate design corresponding to a given q and A just as was done above for the adapted Kanerva design. We can find the number of points in the subsets defined above, and also the size of the access overlap for two given addresses, by methods similar to those used above for the adapted Kanerva design.

Note that q could be greater than 100, but in view of the relatively poor performance of the adapted Kanerva design, as shown in Table 1, it is unlikely that large values of q would be useful, at least in this context.

We could also construct ***hybrid hyperplane designs***, like the hybrid designs described earlier, in which different hard locations could have different values for q and for A. The value of A for a hard location would depend on q and on the desired number of addresses that would activate the location; this number of addresses need not be the same for all locations. Computation of the expected number of hard locations in the access overlap for such a design would be done as before, by combining the results for the various values of q and A used in the design.

In an earlier section I showed how the various hardware embodiments for the selected-coordinate design could be modified for the intermediate and hybrid designs related to that design.

The corresponding embodiments for the hyperplane design, which are described above, can be modified in an analogous way for the intermediate and hybrid hyperplane designs described in this section. The embodiments for these designs would determine whether to activate a hard location by counting the number of 1's in the bit positions in the reference address corresponding to the selected coordinates; if that number is greater than or equal to A, the location would be activated.

CORRELATED ADDRESSES AND CHOICE OF HARD LOCATIONS

In all of the designs considered so far, the hard locations have been chosen at random from a set of potential memory locations. This choice is based on the assumption that the read and write addresses will be distributed randomly throughout S (or, in the hyperplane case, in or near C). However, in some applications the addresses that will actually be encountered will lie in a subset of S, such as a hyperplane, or will have some nonuniform frequency distribution over S. In such cases we could say that the addresses are *correlated*. In this situation it may be inefficient to have the hard locations scattered at random, without regard to the nature of the addresses to be encountered. Some hard locations might be activated very often, while others are rarely activated. The data stored in them might be redundant or uninformative. Moreover, if the addresses encountered tend to be bunched in certain regions in S, such as in the hyperplane C, then the average distance between two such addresses may be much less than $n/2$, the average distance if the addresses are random. Consequently, as in the hyperplane case, the "noise" due to data words stored at other addresses might make it difficult to recover a desired data word.

I assume that each data word is written to the memory once, as in Kanerva's original design; that is, I assume that no "retraining" is done to reduce the noise caused by interference among the stored data words. In any of the designs it should be possible to improve the response of the memory, after data have been stored in it, by testing the memory and then altering the stored data so that when the memory is read from at certain addresses, the response will be closer to the desired response. If this is done, the assumptions of randomness that underlie the memory capacity and signal-to-noise ratio computations in Jaeckel (1989) would not apply.

A modification of Kanerva's design to fit this situation has been suggested by Keeler (1988), p. 321-24. He suggests that if the read and write addresses are expected to follow a known nonuniform frequency distribution, then the addresses of the hard locations should be chosen at random based on the same frequency distribution. If the frequency distribution is not known, it may be possible to estimate it. The hard locations would then have a higher density in the regions of S where more of the read and write addresses will lie. The activation radius would have to be adjusted so that a desired proportion of the hard locations would be activated by an address. Because of the higher density of hard locations near the read and write addresses, the radius can be reduced without resulting in too few locations being activated by an address. And with a smaller activation radius, the memory can discriminate between addresses that are closer to each other. For example, in the adapted Kanerva design above, the spheres have a Hamming radius of 160.

A similar modification can be made in the selected-coordinate design and in the related intermediate and hybrid designs, if the frequency distribution of the addresses to be encountered is known or can be estimated. One possibility is to choose the selected coordinates for a hard location at random as before, and then to assign values for the selected coordinates based on the joint frequency distribution of the components of the address vectors corresponding to those selected coordinates. This may be derived from the distribution of the addresses. A possible alternative is to assign combinations of values for the selected coordinates that are *less* likely to occur in the addresses, but that will occur sometimes. Hard locations activated by such relatively unlikely patterns may be more useful in that they may make certain regions of the address space "stand out", so that the memory can make finer distinctions between similar patterns. The hyperplane design above is an example of this; although most of the bits in the addresses are 0's, the assigned values are all 1's.

Instead of choosing the selected coordinates for a hard location at random, it may be better to choose nonrandom sets of selected coordinates, together with appropriate assigned values, that are more useful for discriminating among the addresses to be encountered. For example, if the values of the bits in the address vectors for two of the coordinates are highly correlated, and if we use both of those coordinates as selected coordinates for the same hard location, we should probably assign them combinations of values that are relatively unlikely to occur. Otherwise, using both coordinates as selected coordinates for the same hard location would be somewhat redundant.

Thus, it may be possible to define the hard locations in a way that is better adapted to the actual distribution of the addresses. Moreover, the hardware embodiments described above and in Jaeckel (1989) may be used in these cases. Since the user can choose any desired combination of selected coordinates and (except for the hyperplane embodiments) assigned values for any of the hard locations, and enter that information into the address decoders or the hard-address units, no changes in the hardware are required.

Thanks to Pentti Kanerva and Mike Raugh for reading and commenting on earlier drafts of these reports.

REFERENCES

Flynn, M. J., Kanerva, P., Ahanin, B., Bhadkamkar, N., Flaherty, P., & Hickey, P. (1988). Sparse Distributed Memory Prototype: Principles of Operation. Technical Report CSL-TR-87-338, Computer Systems Laboratory, Stanford University.

Jaeckel, L. A. (1989). An Alternative Design for a Sparse Distributed Memory. RIACS Technical Report 89.28.

Kanerva, P. (1988). *Sparse Distributed Memory*. MIT Press, Cambridge, Mass.

Keeler, J. D. (1988). Comparison Between Kanerva's SDM and Hopfield-type Neural Networks. *Cognitive Science, 12,* 299-329.

Marr, D. (1969). A Theory of cerebellar cortex. *Journal of Physiology, 202,* 437-470.