White Rose Research Online

eprints@whiterose.ac.uk
https://eprints.whiterose.ac.uk
Universities of Leeds, Sheffield and York

Deposited via The University of York.

White Rose Research Online URL for this paper:
https://eprints.whiterose.ac.uk/id/eprint/1528/

Proceedings Paper:
Austin, J, Turner, A, Turner, M et al. (1999) Chemical structure matching using correlation matrix memories. In: NINTH INTERNATIONAL CONFERENCE ON ARTIFICIAL NEURAL NETWORKS (ICANN99), VOLS 1 AND 2. 9th International Conference on Artificial Neural Networks (ICANN99), 07-10 Sep 1999 IEEE, EDINBURGH, pp. 619-624.

Reuse
Items deposited in White Rose Research Online are protected by copyright, with all rights reserved unless indicated otherwise. They may be downloaded and/or printed for private study, or other acts as permitted by national copyright laws. The publisher or other rights holders may allow further reproduction and re-use of the full text version. This is indicated by the licence information on the White Rose Research Online record for the item.

Takedown
If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing eprints@whiterose.ac.uk including the URL of the record and the reason for the withdrawal request.

UNIVERSITY OF LEEDS
University of Sheffield
UNIVERSITY of York

Chemical Structure Matching using Correlation Matrix Memories

J. Austin, A. Turner, M. Turner, K. Lees
Advanced Computer Architectures Group,
Department of Computer Science, University of York,
York, YO10 5DD, UK
email: austin@cs.york.ac.uk.

ABSTRACT

This paper describes the application of the Relaxation By Elimination (RBE) method to matching the 3D structure of molecules in chemical databases within the frame work of binary correlation matrix memories. The paper illustrates that, when combined with distributed representations, the method maps well onto these networks, allowing high performance implementation in parallel systems. It outlines the motivation, the neural architecture, the RBE method and presents some results of matching small molecules against a database of 100,000 models.

INTRODUCTION

The maintenance of large chemical structure databases, possibly containing many millions of small molecules, and the development of techniques for rapidly querying these databases plays a key role in the process of ligand design in the pharmaceutical industry. This is the generation of small molecules that bind strongly with key receptor sites on biologically important proteins so as to inhibit or alter their activity. The aim of the design process is to discover molecules which are both readily synthesised and which deliver the required medicinal action without producing unwanted side effects, i.e., the drugs should work and be safe. Since the synthesis and experimental testing of potential ligands is both time-consuming and very expensive the ability of computational techniques to reduce the number of molecules that need to be considered experimentally is very attractive to the pharmaceutical industry.

There are two forms that the querying of a database may take. When the 3D structure of a receptor site on a protein is known the task is to search the database to locate potential ligands that are complementary to the site, i.e., they may 'dock' there. Often the 3D structure of the receptor site is unknown, but molecules that bind there are known through experiments. In these instances additional candidate ligands can be retrieved on the basis of their similarity to these known molecules.

A number of different techniques for assessing the complementary and similarity of molecular structures are currently in routine operational use in the pharmaceutical industry e.g. [12],[13],[14]. Early approaches were underpinned by the conventional 2D bond diagram whereby each molecule is described in terms of a planar arrangement of atoms and bonds. This representation remains central to many chemical information systems today. More recently search techniques have been extended to handle 3D chemical databases employing the 'ball and stick' representation of molecules (for a review see [14]). However, there has been an increasing realisation that both simple bond diagrams and ball and stick models are limited since they are inadequate for conveying the space requirements of molecules and certain important properties at the molecular surface which influence the strength of protein-ligand interaction. As such, there has been a shift in emphasis to pattern matching methods based upon shape descriptors such as surface positions, normals and curvatures, and properties, such as electrostatic potential and lipophilicity, and a corresponding increase in interest in optimization techniques for maximising the similarity (or complementarity) of molecules represented in this way e.g.[15],[16],[17].

Unfortunately, this new research has met with only limited success. The reason is that when these more sophisticated descriptions are employed the search space tends to be large and complex and this causes problems. Specifically, conventional optimization techniques (e.g. gradient descent, simulated annealing [18], genetic algorithms [19], the EM algorithm [20]) tend to get stuck in local minima because they only perform local searches in a small proportion of the search space.

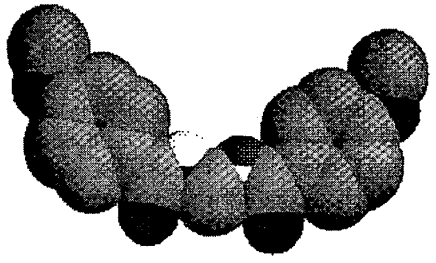


Fig. 1 A typical small molecule based on van der Waals surface.

Our objective has been to introduce a framework for searching in large chemical databases on the basis of 3D shape and surface properties (an example is

shown in Fig. 1) that delivers a technique that is both fast and avoids local minima. Moreover, we aim to provide a method that maps efficiently onto highly parallel systems based on neural networks.

For the moment we restrict ourselves to similarity assessment. The problem of assessing the similarity of two molecules is abstracted as one of 3D relational graph matching and our development originally followed probability theory [9]. In this paper a different mapping of the method is provided that allows it to be run on a large number of binary neural correlation matrix memories.

In general, the novel feature of the approach is the use of a new method for optimising the similarity function delivered by the probablistic approach. Rather than use conventional optimization methods hindered by local minima, or exhaustive search methods (e.g.[21],[22]) compromised by speed, we exploit a radically different third approach which takes into account all of the search space but allows processing to be achieved in a reasonable time without seriously undermining representational power. The basis of the method was outlined in the 1998 IEE Neural Network Conference[9].

In this paper we concentrate on the implementation of the method in neural correlation matrix memories. The first section presents a functional outline of the method, methods. Following this we describe the implementation in Correlation Matrix Memories as combined within the AURA technology and evaluate the implementation in comparison to a conventional implementation [9] and describe the speed on a conventional workstation.

BACKGROUND TO AURA

The AURA methods are designed to implement systems that manipulate uncertain information [2] using correlation matrix memories. They are based on methods developed from associative memories and neural networks. In particular, the approach exploits the pattern matching abilities of neural networks (specifically threshold logic and distributed representations) using simple one shot training methods similar to that used in associative memories. The information store is based on a simple one layer neural network structure, called a correlation matrix memory (CMM) that utilises binary weights.

The AURA technology has grown out of binary neural networks developed in the early 1970's, and inherits their characteristics of ease of implementation, high-speed performance, and simple structure [6].

AURA has been developed for use in rule based systems [6] where it has the ability to deal with large rule sets in small amounts of time, for use in time series prediction [3], image databases [4], string matching problems such as address databases and for general pattern matching problems based on the k-NN method [5]. In addition, the technology has unique parallel processing capabilities [2] which allow it to deal with multiple simultaneous data inputs.

OUTLINE OF THE AURA METHOD

As described above, the AURA methods are based on correlation matrix memories. These networks have the advantage of one shot learning, i.e. data need only be presented once for the network to learn an association between two items. The CMMs used in AURA are typically binary, that is they have binary weights and binary inputs and outputs. If a CMM, M, has input vectors I and output vectors O, then training such a CMM to form an association between one instance of I and O requires the following process:

(1) Before M is used, initialise all elements to zero.

(2) Take the outer product between I and O, to produce a matrix M' of the same dimensions as M as follows: $M' = IO^T$

(3) Form the bit wise logical OR of M with M', as follows: $M = M \vee M'$

This process is repeated for all training examples.

To recall a stored output pattern, O, for a given input pattern, I, the following operations are used:

(1) Obtain a raw output pattern, R by: $R = MI$

(2) Threshold R to obtain a binary pattern, in this case by applying a fixed threshold:

```
For all i
   If R_i ≥ W  then o_i=1
   else o_i=0
endfor
```

The threshold W is obtained by knowing the bit density in the query input I. If the query input must exactly match the stored item, then W is chosen to equal the number of bits set in I. If the query input need only match by $X\%$ then the threshold is set to $I \times X\%$.

In the ideal case the output pattern O will contain only a single pattern that was trained against the given input pattern, I. Unfortunately, if I is incomplete, then it is likely that more than one trained example will respond. This is represented by the superposition of the responding patterns in the output pattern O. To allow these patterns to be separated we restrict each trained output pattern to a fixed density, and apply a method called MBI [7] to identify the multiple matches. MBI has been designed to operate very quickly on large datasets.

In practical problems we also use a fixed density input

pattern (each trained pattern has the same number of bits set to one). When used in combination with multiple memories, this allows us to identify if an input query exactly matches a stored example (described fully in [2]).

Typical input data is made up of a number of data items which must be combined and applied to the memory to find a match. This is achieved by first converting the data items to fixed weight binary strings then either superimposing the separate strings or concatinating the strings. The former removes any order in the data being presented and is useful when dealing with cardinal data such as found in rule evaluation systems. The latter preserves the order in the data and is useful in ordinal data streams as for example in text matching.

The speed of the CMMs derives from the use of only very sparse access to memory and from their use of binary weights, allowing simple implementation in dedicated hardware [1]. In practice, an input vector, *I*, will contain only a small percentage of bits set. To perform the matrix multiplication for recall requires summation of the matrix lines selected by those bits set to one in *I*. The process of summing the lines is supported well in conventional software and very well in dedicated hardware.

THE RBE FRAMEWORK

This section outlines the RBE method more fully described in [1]. The development given here is aimed at showing how the method maps onto an array of correlation matrix memories.

The approach is characterised by a number of *models* representing the stored molecules, and *data* representing the molecule to be matched. Furthermore, the data is made up of a number of *data points* and the model made up of a number of *model points*.

In our example, for simplicity of description, each model contains the same number of *model* points. Each point in the *model* is numbered and characterised by a set of measurements. For simplicity one measurement per *model* point is used in our example, as well as the distance between the points. In practice both the number of points in the *model* and in the *data* can vary. The problem addressed here is a conventional optimization problem, where it is necessary to find out which *model* best fits the *data* points and which point within a given *model* should be optimally assigned to a point on the *data* molecule.

To understand how optimization is achieved, consider *N data points*, each which contain an initial unary measurement which represent data from a given molecule to be matched. The task to be undertaken is to match these points to a set of *models*, *M*, and find the model that best fits. Each *model* contains a set of *model points, P,* which also contain a single measurement giving the distance to the centeroid of the molecule. The data points are distributed over the *data* molecule at a regular distance from each other, and the inter-point distance, $D_{i,j}$ is given between all *data* points *i*, and *j*, and for all *model* points.

As described below the method commences by using the distance to the centeroid to get an initial hypothesis of the matching models, which returns a large number of matches. With this data to hand, The approach introduces the inter-point distances and applies RBE to prune the search space, through iterative update.

Initialisation.

The first stage aims to obtain a context-free set of potential models at each data point. For this the approach utilises the unary measurements at the data points to generate the list of initial candidate matches.

```
For each data point i in N
   For each model point, j, in M
      Find all points in the model, j
      matches the measurement at the
      data point.
      Generate a list Lj of this data.
endfor
```

Fig.2 shows the state of a simple three point example, after the initialisation stage. In this example, point N_3 is the data point being updated, all nodes present lists of initial matches.

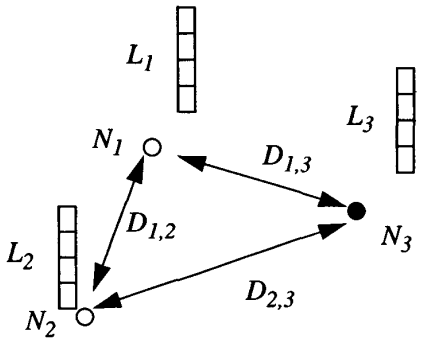


Fig. 2. The state of the system after initialisation

Search.

Step 1, finding the support

The next stage applies knowledge of inter-point distances. It visits each data point *i*, and for each checks the knowledge held at other data points *j* to find any support for the models at *i*.

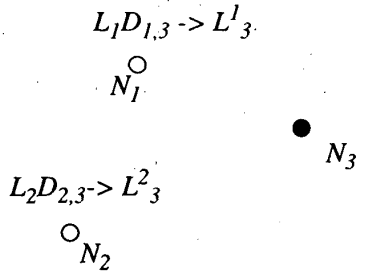


Fig. 3. Computation of support to node N_3

```
For each data point, i, in N.
  For all j, visit each data point,
  j, in N (not including i)
    Calculate the distance, D_{i,j},
    between the two points, i and j.
    Find which of the current mod-
    els, L'_j, support points at the
    distance D_{i,j}.
    (This results in a support vector, L^j_i that shows
    how the data point j supports the data point i,
    given the list of points in models at j and the
    distance Di,j. Note, this is where the CMM is
    used later.)

  endfor
endfor
```

Step 2, fusing the support.

This looks at each data point and fuses the support for its models given from other nodes.

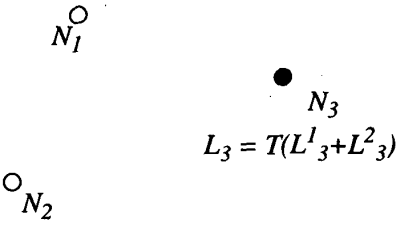


Fig. 4. Calculation of new models supported at node N_3 given support at the other nodes.

```
For each point, i, in N.
  Sum the support for all model
  points, L^j_i to get the raw support
  L_i^raw at the node i.
  Threshold L_i^raw at a level T to get
  the binary model support vector,
  L_i.
endfor
```

Step 3, pruning the support.

This stage performs the elimination of models that have little support. In this case it is performed over all data points and all models at each. In effect this is by setting T to the correct value.

```
Over all points p in N×M
  Find the models with the least
  support, and delete them.
endfor
```

Step 4, halting condition.

```
This stops the iterations when all
    models have a support that ex-
    ceeds a threshold, T.
```

Step 5, iteration.

```
Repeat from step 1.
```

Step 6, back check.

The final stage is to find the dominant models that exist in the final labelings, extract these and compare them with the input data.

Note that the approach given here uses quite soft support information based on inter-point distances. The benefit of this is that it is fast, and will not remove any models that could match. At the end of processing each node will have information on which models are supported. Some of these will be ambiguous due to the simple distance constraint used. These can be eliminated by using conventional geometric matching on, what is now, a small set of candidate models.

The iteration continues until only one or a number of models remain with the same support. Following the cessation of iterations the set of lists of model points at each node, L, must be examined to determine which nodes are supported, and this should be compared against the original models to determine if any models are totally supported - i.e. if all the points for that model are to be found in L, distributed across the nodes N in a rational way. Although the method uses binary support information, the original formulation uses a probablistic framework which was then described in binary terms, but not implemented in CMMs [1].

The process halts when the support at all nodes fails to change. In practice, this may not be the lowest energy state of the system, in that a large number of nodes may remain with high support. In this case a 'kick start' is given to the node with the highest "ambiguity" of support, by increasing T at that node, effectively removing the least supported model at that node.

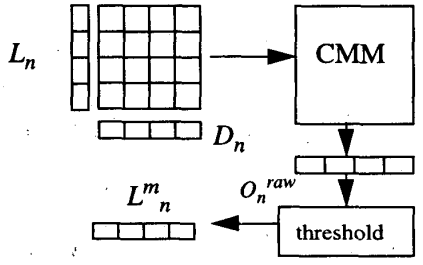
In practice, all neighbours of a given node are selected to provide information to up date that node. It is possible to use a subset of the neighbours, but this leads to slower convergence.

The approach uses a process of removing points that do not get support from other nodes (step 3). The motivation for this is based on the observation that it is simpler and more reliable to eliminate all models that have no support, and to let this knowledge propagate, than to select models that have the highest support as found in other relaxation based methods [11].

USING THE CMM

The CMMs are used to store information concerning "which points support which models".

At each data point there is a list of models which have model points that match the current data point. In the search stage, these models are combined with the inter-point distance on different points of the molecule and used to find which models could match. A search must be made of the mode database to find models that are supported. The input of the CMM is a 2D ma-



trix shown above, which codes the currently supported models, L_j, against the inter-point distance of interest, $D_{j,i}$ (see Fig. 2). This is input to the CMM which then looks up the models that match and outputs a raw vector O_i^{raw} which expresses this. This vector is then thresholded at a level Y to obtain a binary vector giving the models supported at data point j from data point i, given as L^j_i. This information is sent to the data point currently being evaluated where it is combined as given above. The threshold level, Y, is determined from the number of bits set in the input to the CMM, which is controlled by the number of currently matching models. In practice Y can be reduced as explained in section The parameterisation of the memory is derived from analysis of CMM storage ability [10].

The encoding of the list of models L to a binary format suitable for the CMM is simple, but the reverse process is not. For this reason the halting criterion in section 4, step 4 must be changed to one which is easy to compute when dealing with binary encodings of L. The criterion used is that the system be in a stable state, even after a 'kick start'.

DISTRIBUTED SUPERIMPOSED CODE REPRESENTATION

To ensure efficient use of resources the method uses distributed representations. This uses a n in m bit superimposed binary code and is used widely in AURA. In practice, if you have 10^6 molecules to match in the database and about 50 points on each molecule, then using unary representations (one bit in m), the output of the CMM, M, will be M_{size}, 10^6 in size. The input to the CMM is based on M_{size} multiplied by a unary encoded binary representation of the distance between any two data points, which is typically D_{size}, 30 in size. This leads to a CMM of the order of $(10^6)^2 \times 30$, or 3×10^{13} binary bits. This is clearly too large to implement (3.4 Tbytes). To overcome this the distributed coding is used, and relies on the fact that typically, at *some* data points, very few models will be activated after initialisation. Thus, the list labelling all points in the models M, can be compressed using a distributed representation without loss of too much information at those, i.e. by using 2 bits set in a binary vector size M_{size}' where $M_{size}' << M_{size}$, and then superimposing the model point codes into one vector of size M_{size}'. In practice as long as some bits remain unset in the distributed code at some nodes, the system will converge.

COMPARATIVE RESULTS

The method given above has been implemented using the AURA C++ library developed at York, and compared with the BINARY, non-CMM based method (binary version) described in [1]. Results in Table 1

Table 1. Comparative results.

	BINARY	AURA	AURA
No. Mols	100	100	10,000
Sample size.	50	50	50
Time (seconds)	245	1.8	14.9
P	keep=0.9	Y=0.3, W=4	Y=0.3 W 4

show a single query molecule tested against a small data set of 100 molecules from the Cancer Institute Database, and for a database of 10,000 molecules. The sample size is the number of points selected on the van de Waals (at 0.3 Å sampling) for the query. The time is the time to perform the query(does not include data base build time, which is 26.6 seconds for 100 molecules) The P values are the optimal parameters needed in the method. The 'keep' value is the fraction of molecules retained at each node after each itteration. The Y value is the threshold used in the

CMM method, and W value is the number of bits set in the binary vectors used to label each molecule. Our results show that the method recovers the same molecules as the 'binary' implementation described in [1].

PARALLEL IMPLEMENTATION

The method is designed to be implemented on dedicated implementations of CMMs such as the PRESENCE hardware developed in our group [2]. It is anticipated that a 5 times speed up is achievable using one PRESENCE processor node, and that good scale-up should be possible with more processing nodes.

CONCLUSIONS

This paper has described the use of a CMM based approach for chemical structure matching. The technique exploits a relaxation by elimination method that operates effectively in the AURA framework. The approach is scalable and is simple to implement in parallel. It can incorporate a wide variety of data types. Our current work is extending the basic framework to incorporate more molecular information, and mapping the method on to a parallel implementation based on the PRESENCE hardware.

ACKNOWLEDGEMENTS

This work was, in part, supported by EPSRC grant number GR/K51662 in collaboration with Glaxo-Wellcome Ltd. and Silicon Graphics Ltd.

REFERENCES

[1] M Turner and Austin, 1998, *A CMM architecture for fast structural matching, Neural Networks*, Submitted.

[2] J Austin, J V Kennedy, K Lees, 1995, *The Advanced Uncertain Reasoning Architecture*, Weightless Neural Network Workshop, University of Kent, Canterbury, UK.

[3] D Kustrin, J Austin and A B Sanders, 1997, *Application of correlation matrix memory matrices in high frequencies asset allocation*, Ed. M Niranjan, Fifth International Conference on Artificial Neural Networks, Cambridge, UK, IEE.

[4] S Alwis and J Austin, 1997, *A novel architecture for trade mark image retrieval systems, The Challenge of Image retrieval*, A Symposium and workshop on image retrieval, Newcastle, UK

[5] P Zhou and J Austin, 1998, *A Binary Correlation Matrix Memory k-NN classifier*, International Conference on Artificial Neural Networks, Sweden.

[6] J Austin, 1995, *Distributed Associative Memories for High Speed Symbolic Reasoning*, International Journal of Fuzzy Sets and Systems, pp223-233, Vol. 82

[7] R. Filer and J Austin, 1995, *Using CMM for Inferencing in Expert Systems*, ADT 95, Brunel University.

[8] I Aleksander and T J Stonham, *Guide To Pattern Recognition Using Random-Access Memories*, Computers And Digital Techniques, Vol. 2, Issue. 1, p 29-40.

[9] M Turner and J Austin. *A neural network technique for chemical graph matching*, Fifth International Conference on Artificial Neural Networks, Cambridge, UK, IEE.

[10] M Turner and J Austin. 1997, *Matching Performance of Binary Correlation Matrix Memories*, Neural Networks, Vol. 10, No. 9, pp. 1637-1648.

[11] Hancock E.R. and Kittler J., *Discrete relaxation, Pattern Recognition*, 23, pp.711-733, 1990.

[12] (Anonymous), *Daylight Theory Manual*, Daylight Chemical Industries, 1995.

[13] van Geerestein V.J., Perry N.C., Grootenhuis P.D.J. and Haasnoot C.A.G., *3D database searching on the basis of ligand shape using the SPERM prototype method*, Tetrahedron Computer Methodology, V 3, N 6c, pp. 595-613, 1990.

[14] Willett P., *Three-dimensional chemical structure handling*, Research Studies Press Ltd., 1991.

[15] Blaney F., Finn P., Phippen R. and Wyatt M., *Molecular surface comparison: application to drug design*, Journal of Molecular Graphics, V11, pp. 98-105, 1993.

[16] Dean P.M, Callow P. and Chau P.I., *Molecular recognition: blind-searching for regions of strong structural match on the surfaces of two dissimilar molecules*, Journal of Molecular Graphics, V6, pp. 28-35, 1988.

[17] , Good A.C., Hodgkin E.E. and Richards W.G., *Similarity screening of molecular data sets*, Journal of Computer-Aided Molecular Design, V6, pop. 513-520, 1992.

[18] Kirkpatrick S., Gelatt C.D. and Vecchi M.P., *Optimization by simulated annealing*, Science, V220, pp. 671-680, 1983.

[19] Goldberg D.E., *Genetic algorithms in search optimization and machine learning*, Addison-Wesley, 1989.

[20] Dempster N.M. and Laird D.B., *Maximum-likelihood from incomplete data via the EM algorithm*, J. Royal Statistical Soc. Ser. B (methodological), V39, pp. 1-38, 1977.

[21] van Geerestein V.J., Perry N.C., Grootenhuis P.D.J. and Haasnoot C.A.G., *3D database searching on the basis of ligand shape using the SPERM prototype method*, Tetrahedron Computer Methodology, V 3, N 6c, pp. 595-613, 1990.

[22] Wallace A.C., Borkakoti N. and Thornton J.M., *TESS: A geometric hashing algorithm for deriving 3D coordinate templates for searching structural databases. Application to enzyme active sites*, Protein Science, V6, pp. 2308-2323, 1997.