A DISTRIBUTED BASIS FOR ANALOGICAL MAPPING

Ross W. Gayler
r.gayler@gmail.com
School of Communication, Arts and Critical Enquiry
La Trobe University
Victoria 3086 Australia

Simon D. Levy
levys@wlu.edu
Department of Computer Science
Washington and Lee University
Lexington, Virginia USA

ABSTRACT

We are concerned with the practical feasibility of the neural basis of analogical mapping. All existing connectionist models of analogical mapping rely to some degree on localist representation (each concept or relation is represented by a dedicated unit/neuron). These localist solutions are implausible because they need too many units for human-level competence or require the dynamic re-wiring of networks on a sub-second time-scale.

Analogical mapping can be formalised as finding an approximate isomorphism between graphs representing the source and target conceptual structures. Connectionist models of analogical mapping implement continuous heuristic processes for finding graph isomorphisms. We present a novel connectionist mechanism for finding graph isomorphisms that relies on distributed, high-dimensional representations of structure and mappings. Consequently, it does not suffer from the problems of the number of units scaling combinatorially with the number of concepts or requiring dynamic network re-wiring.

GRAPH ISOMORPHISM

Researchers tend to divide the process of analogy into three stages: retrieval (finding an appropriate source situation), mapping (identifying the corresponding elements of the source and target situations), and application. Our concern is with the mapping stage, which is essentially about structural correspondence. If the source and target situations are formally represented as graphs, the structural correspondence between them can be described as approximate graph isomorphism. Any mechanism for finding graph isomorphisms is, by definition, a mechanism for finding structural correspondence and a possible mechanism for implementing analogical mapping. We are concerned with the formal underpinning of analogical mapping (independently of whether any particular researcher chooses to describe their specific model in these terms).

It might be supposed that representing situations as graphs is unnecessarily restrictive. However, anything that can be formalised can be represented by a graph. Category theory, which is effectively a theory of structure and graphs, is an alternative to set theory as a foundation for mathematics (Marquis, 2009), so anything that can be mathematically represented can be represented as a graph.

It might also be supposed that by working solely with graph isomorphism we favour structural correspondence to the exclusion of other factors that are known to influence analogical mapping, such as semantic similarity and pragmatics. However, as any formal structure can be represented by graphs it follows that semantics and pragmatics can also be encoded as graphs. For example, some models of analogical mapping are based on labelled graphs with the process being sensitive to label similarity. However, any label value can be encoded as a graph and label similarity cap-

tured by the degree of approximate isomorphism. Further, the mathematics of graph isomorphism has been extended to include attribute similarity and is commonly used this way in computer vision and pattern recognition (Bomze, Budinich, Pardalos & Pelillo, 1999).

The extent to which analogical mapping based on graph isomorphism, is sensitive to different types of information depends on what information is encoded into the graphs. Our current research is concerned only with the practical feasibility of connectionist implementations of graph isomorphism. The question of what information is encoded in the graphs is separable. Consequently, we are not concerned with modelling the psychological properties of analogical mapping as such questions belong to a completely different level of inquiry.

CONNECTIONIST IMPLEMENTATIONS

It is possible to model analogical mapping as a purely algorithmic process. However, we are concerned with physiological plausibility and consequently limit our attention to connectionist models of analogical mapping such as ACME (Holyoak & Thagard, 1989), AMBR (Kokinov, 1988), DRAMA (Eliasmith & Thagard, 2001), and LISA (Hummel & Holyoak, 1997). These models vary in their theoretical emphases and the details of their connectionist implementations, but they all share a problem in the scalability of the representation or construction of the connectionist mapping network. We contend that this is a consequence of using localist connectionist representations or processes. In essence, they either have to allow in advance for all combinatorial possibilities, which requires too many units (Stewart & Eliasmith, in press), or they have to construct the required network for each new mapping task in a fraction of a second.

Problems with localist implementation

Rather than review all the major connectionist models of analogical mapping, we will use ACME and DRAMA to illustrate the problem with localist representation. Localist and distributed connectionist models have often been compared in terms of properties such as neural plausibility and robustness. Here, we are concerned only with a single issue: dynamic re-wiring (i.e., the need for connections to be made between neurons as a function of the source and target situations to be mapped).

ACME constructs a localist network to represent possible mappings between the source and target structures. The network is a function of the source and target representations, and a new network has to be constructed for every source and target pair. A localist unit is constructed to represent each possible mapping between a source vertex and target vertex. The activation of each unit indicates the degree of support for the corresponding vertex mapping being part of the overall mapping between the source and target. The connections between the network units encode compatibility between the corresponding vertex mappings. These connections are a function of the source and target representations and constructed anew for each problem. Compatible vertex mappings are linked by excitatory connections so that support for plausibility of one vertex mapping transmits support to compatible mappings. Similarly, inhibitory connections are used to connect the units representing incompatible mappings. The network implements a relaxation labelling that finds a compatible set of mappings. The operation of the mapping network is neurally plausible, but the process of its construction is not.

The inputs to ACME are symbolic representations of the source and target structures. The mapping network is constructed by a symbolic process that traverses the source and target structures. The time complexity of the traversal will be a function of the size of the structures to be mapped. Given that we believe analogical mapping is a continually used core part of cognition and that all cognitive information is encoded as (large) graph structures, we strongly prefer mapping network setup to require approximately constant time independent of the structures to be mapped.

DRAMA is a variant of ACME with distributed source and target representations.

However, it appears that the process of constructing the distributed representation of the mapping network is functionally localist, requiring a decomposition and sequential traversal of the source and target structures.

Ideally, the connectionist mapping network should have a fixed neural architecture. The units and their connections should be fixed in advance and not need to be re-wired in response to the source and target representations. The structure of the current mapping task should be encoded entirely in activations generated on the fixed neural architecture by the source and target representations and the set-up process should be holistic rather than requiring decomposition of the source and target representations. Our research aims to achieve this by using distributed representation and processing from the VSA family of connectionist models.

We proceed by introducing replicator equations; a localist heuristic for finding graph isomorphisms. Then we introduce Vector Symbolic Architectures (VSA), a family of distributed connectionist mechanisms for the representation and manipulation of structured information. Our novel contribution is to implement replicator equations in a completely distributed fashion based on VSA. We conclude with a proof-of-concept demonstration of a distributed re-implementation of the principal example from the seminal paper on graph isomorphism via replicator equations.

REPLICATOR EQUATIONS

The approach we are pursuing for graph isomorphism is based on the work of Pelillo (1999), who casts subgraph isomorphism as the problem of finding a maximal clique (set of mutually adjacent vertices) in the *association graph* derived from the two graphs to be mapped. Given a graph G' of size N with an $N \times N$ adjacency matrix $A' = a'_{ij}$ and a graph G'' of size N with an $N \times N$ adjacency matrix $A'' = a''_{hk}$, their association graph G of size N^2 can be represented by an $N^2 \times N^2$ adjacency matrix $A = (a_{ih}, a_{jk})$ whose edges encode pairs of edges from G' and G'':

$$a_{ih,jk} = \begin{cases} 1-(a'_{ij}-a''_{hk})^2 & \text{if } i \neq j \text{ and} h \neq k \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

The elements of A are 1 if the corresponding edges in G' and G'' have the same state of existence and 0 if the corresponding edges have different states of existence. Defined this way, the edges of the association graph G provide evidence about potential mappings between the vertices of G' and G'' based on whether the corresponding edges and non-edges are consistent. The presence of an edge between two vertices in one graph and an edge between two vertices in the other graph supports a possible mapping between the members of each pair of vertices (as does the absence of such an edge in both graphs).

By treating the graph isomorphism problem as a maximal-clique-finding problem, Pelillo exploits an important result in graph theory. Consider a graph G with adjacency matrix A, a subset C of vertices of G, and a *characteristic vector* x^C (indicating membership of the subset C) defined as

$$x_i^C = \begin{cases} 1/|C| & \text{if } i \in C \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

where $|C|$ is the cardinality of C. It turns out that C is a maximum clique of G if and only if x^C maximizes the function $f(x) = x^T Ax$, where x^T is the transpose of x, $x \in \Re^N$, $\sum_{i=1}^{N} x_i = 1$, and $\forall i \; x_i \geq 0$.

Starting at some initial condition (typically the barycenter, $x_i = 1/N$ corresponding to all x_i being equally supported as part of the solution), x can be obtained through iterative application of the following equation:

$$x_i(t+1) = \frac{x_i(t)\pi_i(t)}{\sum_{j=1}^{N} x_j(t)\pi_j(t)} \quad (3)$$

where

$$\pi_i(t) = \sum_{j=1}^{N} w_{ij} x_j(t) \quad (4)$$

and W is a matrix of weights, w_{ij}, typically just the adjacency matrix A of the association graph or a linear function of A. The x vector can thus be considered to represent the state of the system's belief about the vertex mappings at a given time, with Equations 3 and 4 representing a dynamical system parameterized by the weights in W. π_i can be interpreted as the evidence for x_i obtained from all the compatible x_j where the compatibility is encoded by w_{ij}. The denominator in Equation 3 is a normalizing factor ensuring that $\sum_{i=1}^{N} x_i = 1$.

Pelillo borrows Equations 3 and 4 from the literature on evolutionary game theory in which π_i is the overall payoff associated with playing strategy i, and w_{ij} is the payoff associated with playing strategy i against strategy j. In the context of the maximum-clique problem, these *replicator equations* can be used to derive a vector x (vertex mappings) that maximizes the "payoff" (edge consistency) encoded in the adjacency matrix. Vertex mappings correspond to strategies, and as Equation 3 is iterated, mappings with higher fitness (consistency of mappings) come to dominate ones with lower fitness.

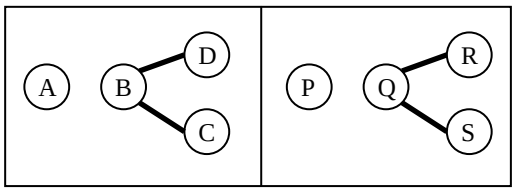

Figure 1. A simple graph isomorphism problem.

Consider the simple graphs in Figure 1, used as the principal example by Pelillo (1999) and which we will later re-implement in a distributed fashion. The maximal isomorphism between these two graphs is {A=P, B=Q, C=R, D=S} or {A=P, B=Q, C=S, D=R}. Table 1 shows the first and last rows of the adjacency matrix for the association graph of these graphs, generated using Equation 1. Looking at the first row of the table, we see that the mapping A=P is consistent with the mappings B=Q, B=R, B=S, C=Q, C=R, C=S, D=Q, D=R, and D=S, but not with A=Q, A=R, A=S, B=P, etc.

	AP	AQ	AR	AS	BP	BQ	BR	BS	CP	CQ	CR	CS	DP	DQ	DR	DS
AP	0	0	0	0	0	1	1	1	0	1	1	1	0	1	1	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
DS	1	0	1	0	0	1	0	0	1	0	1	0	0	0	0	0

Table 1. Fragment of adjacency matrix for Fig. 1.

Initially, all values in the state vector x are set to 0.0625 (1/16). Repeated application of Equations 3 and 4 produces a final state vector that encodes the two maximal isomorphisms, with 0.3 in the positions for A=P and B=Q, 0.1 in the positions for C=R, C=S, D=R, and D=S, and 0 in the others. The conflicting mappings for C, D, R, and S correspond to a saddle point in the dynamics of the replicator equations, created by the symmetry in the graphs. Adding a small amount of noise to the state breaks this symmetry, producing a final state vector with values of 0.25 for the optimal mappings A=P, B=Q, and C=R, D=S or C=S, D=R, and zero elsewhere. The top graph of Figure 4 shows the time course of the settling process from our implementation of Pelillo's localist algorithm.

This example is trivially small. However, the same approach has been successfully applied to graphs with more than 65,000 vertices (Pelillo & Torsello, 2006). It has also been extended to match hierarchical, attributed structures for computer vision problems (Pelillo, Siddiqi & Zucker 1999). Thus, we are confident that replicator equations are a reasonable candidate mechanism for the structure matching at the heart of analogical mapping.

DISTRIBUTED IMPLEMENTATION

The replicator equation mechanism can be easily implemented as a localist connectionist circuit. This is qualitatively very similar to ACME and suffers the same problems due to the localist representation. In this section we

present a distributed connectionist scheme for representing edges, vertices, and mappings that does not suffer from these problems.

Vector Symbolic Architecture

Vector Symbolic Architecture is a name that we coined (Gayler, 2003) to describe a class of connectionist models that use high-dimensional vectors (typically around 10,000 dimensions) of low-precision numbers to encode structured information as distributed representations. VSA can represent complex entities such as trees and graphs as vectors. Every such entity, no matter how simple or complex, is represented by a pattern of activation distributed over all the elements of the vector. This general class of architectures traces its origins to the tensor product work of Smolensky (1990), but avoids the exponential growth in dimensionality of tensor products. VSAs employ three types of vector operator: a multiplication-like operator, an addition-like operator, and a permutation-like operator. The multiplication operator is used to associate or bind vectors. The addition operator is used to superpose vectors or add them to a set. The permutation operator is used to quote or protect vectors from the other operations.

The use of hyperdimensional vectors to represent symbols and their combinations provides a number of mathematically desirable and biologically realistic features (Kanerva, 2009). A hyperdimensional vector space contains as many mutually orthogonal vectors as there are dimensions and exponentially many almost-orthogonal vectors (Hecht-Nielsen, 1994), thereby supporting the representation of astronomically large numbers of distinct items. Such representations are also highly robust to noise. Approximately 30% of the values in a vector can be randomly changed before it becomes more similar to another meaningful (previously-defined) vector than to its original form. It is also possible to implement such vectors in a spiking neuron model (Eliasmith, 2005).

The main difference among types of VSAs is in the type of numbers used as vector elements and the related choice of multiplication-like operation. Holographic Reduced Representations (Plate, 2003) use real numbers and circular convolution. Kanerva's (1996) Binary Spatter Codes (BSC) use Boolean values and elementwise exclusive-or. Gayler's (1998) Multiply, Add, Permute coding (MAP) uses values from $\{+1, -1\}$ and elementwise multiplication. A useful feature of BSC and MAP is that each vector is its own multiplicative inverse. Multiplying any vector by itself elementwise yields the identity vector. As in ordinary algebra, multiplication and addition are associative and commutative, and multiplication distributes over addition.

We use MAP in the work described here. As an illustration of how VSA can be used to represent graph structure, consider again the optimal mapping {A=P, B=Q, C=R, D=S} for the graphs in Figure 1. We represent this set of mappings as the vector

$$A*P+B*Q+C*R+D*S \tag{5}$$

where A, B, C, ... are arbitrarily chosen (random) vectors over $\{+1, -1\}$ and $*$ and $+$ represent elementwise vector multiplication and addition respectively. For any mapped vertex pair X=Y, the representation Y of vertex Y can be retrieved by multiplying the mapping vector $(X*Y+\mathrm{K})$ by X, and vice-versa. The resulting vector will contain the representation of Y plus a set of representations not corresponding to any vertex, which can be treated as noise; e.g.:

$$\begin{aligned} &A*(A*P+B*Q+C*R+D*S) \\ &= A*A*P+A*B*Q+A*C*R+A*D*S \\ &= P+A*B*Q+A*C*R+A*D*S = P+noise \end{aligned} \tag{6}$$

The noise can be removed from the retrieved vector by passing it through a "cleanup memory" that stores only the meaningful vectors (A, B, C, D, P, Q, R, S). Cleanup memory can be implemented in a biologically plausible way as a Hopfield network that associates each meaningful vector to itself (a variant of Hebbian learning). Such networks can reconstruct the original form of a vector from a highly

degraded exemplar, via self-reinforcing feedback dynamics.

Note that although the vectors depicted in Equations 5 and 6 appear complex they are just vector values like any other. From the point of view of the implementing hardware all vectors are of equal computational complexity. This has profound implications for the resource requirements of VSA-based systems. For example, the computational cost of labelling a graph vertex with a simple attribute or a complex structure is exactly the same.

Our Model

Our goal is to build a distributed implementation of the replicator Equations 3 and 4 by representing the problem as distributed patterns of fixed, high dimension in VSA such that the distributed system has the same dynamics as the localist formulation. As in the localist version, we need a representation x of the evolving state of the system's belief about the vertex mappings, and a representation w of the adjacencies in the association graph.

In the VSA representation of a graph we represent vertices by random hyperdimensional vectors, edges by products of the vectors representing the vertices, and mappings by products of the mapped entities. It is natural to represent the set of vertices as the sum of the vectors representing the vertices. The product of the vertex sets of the two graphs is then identical to the sum of the possible mappings of vertices (Equation 7). That is, the initial value of x can be calculated holistically from the representations of the graphs using only one product operation that does not require decomposition of the vertex set into component vertices. For the graphs in Figure 1:

$$\begin{aligned} x &= (A+B+C+D)*(P+Q+R+S) \\ &= A*P+A*Q+\mathrm{K}+B*P+B*Q+\mathrm{K}+D*R+D*S \end{aligned} \tag{7}$$

For VSA it is natural to represent the set of edges of a graph as the sum of the products of the vertices connected by each edge. The product of the edge sets of the two graphs is identical to a sum of products of four vertices. This encodes information about mappings of edges, or equivalently, about compatibility of vertex mappings. That is, one holistic product operation applied to the edge sets is able to encode all the possible edge mappings in constant time no matter how many edges there are.

The reader may have noticed that the description above refers only to edges, whereas Pelillo's association graph also encodes information about the mapping of non-edges in the two graphs. We believe the explicit representation of non-edges is cognitively implausible. However, Pelillo was not concerned with cognitive plausibility. Since our aim here is to reproduce his work, we include non-edges in Equation 8. The distributed vector w functions as the localist association matrix W. For the graphs in Figure 1:

$$\begin{aligned} w &= (B*C+B*D)*(Q*R+Q*S)+ \\ &\quad (A*B+A*C+A*D+C*D)*(P*Q+P*R+P*S+R*S) \\ &= B*C*Q*R+B*C*Q*S+B*D*Q*R+B*D*Q*S \\ &\quad +A*B*P*Q+A*B*P*R+\mathrm{K}+A*B*R*S \\ &\quad +A*C*P*Q+A*C*P*R+\mathrm{K}+A*C*R*S+\mathrm{K}+C*D*R*S \end{aligned} \tag{8}$$

The terms of this sum correspond to the nonzero elements of Table 1 (allowing for the symmetries due to commutativity). With x and w set up this way, we can compute the payoff vector π as the product of x and w. As in the localist formulation (Equation 4), this product causes consistent mappings to reinforce each other. Evidence is propagated from each vertex mapping to consistent vertex mappings via the edge compatibility information encoded in w. (The terms of Equation 9 have been rearranged to highlight this cancellation.)

$$\begin{aligned} \pi &= x*w \\ &= (A*P+A*Q+\mathrm{K})*(A*P*B*Q+A*P*B*R+\mathrm{K}) \\ &= B*Q+B*R+P*B+Q*P*B*R+\mathrm{K} \end{aligned} \tag{9}$$

Implementing the update of x (Equation 3) is more challenging for the VSA formulation. As in the localist version, the idea is for corresponding vertex mappings in x and π to reinforce each other multiplicatively, in a kind of multiset intersection (denoted here as $\wedge$): if $x=(k_1A*P+k_2B*Q+k_3B*R)$ and $\pi=(k_4A*P+k_5B*Q)$ then $x \wedge \pi$ equals $(k_1k_4A*P+k_2k_5B*Q)$, for non-negative weights k_1, k_2, k_3, k_4, and k_5.

Because of the self-cancellation property of the MAP architecture, simple elementwise multiplication of x and π will not work. We could extract the k_i by iterating through each of the pairwise mappings $(A*P, A*Q, \mathrm{K}, D*S)$ and dividing x and π elementwise by each mapping, but this is the kind of functionally localist approach we argue is neurally implausible. Instead, we need a holistic distributed intersection operator. This can be construed as a special case of lateral inhibition, a winner-takes-all competition, which has traditionally been considered a localist operation (Page, 2000; Levy & Gayler, in press).

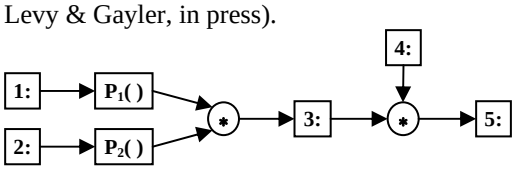

Figure 2. A neural circuit for vector intersection.

To implement this intersection operator in a holistic, distributed manner we exploit the third component of the MAP architecture: permutation. Our solution, shown in Figure 2, works as follows: **1:** and **2:** are registers (vectors of units) loaded with the vectors representing the multisets to be intersected. $\mathbf{P_1(\ )}$ computes some arbitrary, fixed permutation of the vector in **1:**, and $\mathbf{P_2(\ )}$ computes a different fixed permutation of the vector in **2:**. Register **3:** contains the product of these permuted vectors. Register **4:** is a memory (a constant vector value) pre-loaded with each of the possible multiset elements transformed by multiplying it with both permutations of itself. That is, $4: = \sum_{i=1}^{M} X_i * P_1(X_i) * P_2(X_i)$, where M is the number of items in the memory vector (**4:**). To implement the replicator equations the clean-up memory **4:** must be loaded with a pattern based on the sum of all the possible vertex mappings (similar to the initial value of the mapping vector x).

To see how this circuit implements intersection, consider the simple case of a system with three meaningful vectors X, Y, and Z where we want to compute the intersection of $k_1 X$ with $(k_2 X + k_3 Y)$. The first vector is loaded into register **1:**, the second into **2:**, and the sum $X*P_1(X)*P_2(X)+Y*P_1(Y)*P_2(Y)+Z*P_1(Z)*P_2(Z)$ is loaded into **4:**. After passing the register contents through their respective permutations and multiplying the results, register **3:** will contain

$$P_1(k_1 X) * P_2(k_2 X + k_3 Y)$$
$$= k_1 k_2 P_1(X) * P_2(X) + k_1 k_3 P_1(X) * P_2(Y)$$

Multiplying registers **3:** and **4:** together will then result in the desired intersection (relevant terms in bold) plus noise, which can be removed by standard cleanup techniques:

$$(\mathbf{k_1 k_2 P_1(X) * P_2(X)} + k_1 k_3 P_1(X) * P_2(Y)) *$$
$$(\mathbf{X * P_1(X) * P_2(X)} + Y * P_1(Y) * P_2(Y) + Z * P_1(Z) * P_2(Z))$$
$$= \mathbf{k_1 k_2 X} + noise$$

In brief, the circuit in Figure 2 works by guaranteeing that the permutations will cancel only for those terms X_i that are present in both input registers, with other terms being rendered as noise.

In order to improve noise-reduction it is necessary to sum over several such intersection circuits, each based on different permutations. This sum over permutations has a natural interpretation in terms of sigma–pi units (Rumelhart, Hinton & McClelland, 1986), where each unit calculates the sum of many products of a few inputs from units in the prior layer. The apparent complexity of Figure 2 results from drawing it for ease of explanation rather than correspondence to implementation. The intersection network of Figure 2 could be implemented as a single layer of sigma–pi units.

COMPARING THE APPROACHES

Figure 3 shows the replicator equation approach to graph isomorphism as a recurrent neural circuit. Common to Pelillo's approach and ours is the initialization of a weight vector w with evidence of compatibility of edges and non-edges from the association graph, as well as the computation of the payoff vector π

from multiplication ($*$) of x and w, the computation of the intersection of x and π ($\wedge$), and the normalization of x ($/\Sigma$). The VSA formulation additionally requires a cleanup memory (c) and intersection-cleanup memory ($c_\wedge$), each initialized to a constant value.

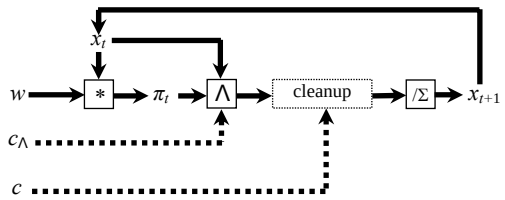

Figure 3. A neural circuit for graph isomorphism.

Figure 3 also shows the commonality of the localist and VSA approaches, with the VSA-only components depicted in dashed lines. Note that the architecture is completely fixed and the specifics of the mapping problem to be solved are represented entirely in the patterns of activation loaded into the circuit. Likewise, the circuit does not make any decisions based on the contents of the vectors being manipulated. The product and intersection operators are applied to whatever vectors are present on their inputs and the circuit settles to a stable state representing the solution.

To demonstrate the viability of our approach, we used this circuit with a 10,000-dimensional VSA to deduce isomorphisms for the graphs in Figure 1. This example was chosen to allow direct comparison with Pelillo's results. Although it was not intended as an example of analogical mapping, it does directly address the underlying mechanism of graph isomorphism. Memory and processor limitations made it impractical to implement the main cleanup memory as a Hopfield net (10^8 weights), so we simulated the Hopfield net with a table that stored the meaningful vectors and returned the one closest to the noisy version. To implement the intersection circuit from Figure 2 we summed over 50 replicates of that circuit, differing only in their arbitrary permutations. The updated mapping vector was passed back through the circuit until the Euclidean distance between x_t and x_{t-1} differed by less than 0.001. At each iteration we computed the cosine of x with each item in cleanup memory, in order to compare our VSA implementation with the localist version; however, nothing in our implementation depended on this functionally localist computation.

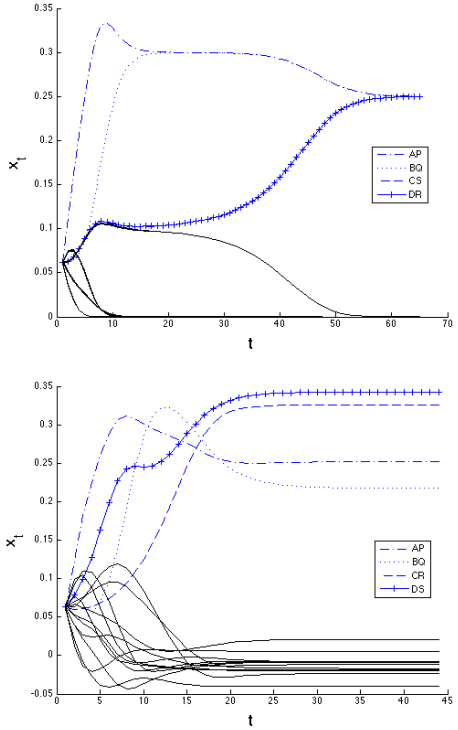

Figure 4. Convergence of localist (top) and VSA (bottom) implementation.

Figure 4 compares the results of Pelillo's localist approach to ours, for the graph isomorphism problem shown in Figure 1. Time (iterations) t is plotted on the abscissa, and the corresponding values in the mapping vector on the ordinate. For the localist version we added a small amount of Gaussian noise to the state vector on the first iteration in order to keep it from getting stuck on a saddle point; the VSA version, which starts with a noisy mapping vector, does not suffer from this problem. In both versions one set of consistent vertex mappings (shown in marked lines) comes to dominate the other, inconsistent mappings

(shown in solid lines) in less than 100 iterations.

The obvious difference between the VSA and localist versions is that the localist version settles into a "clean" state corresponding to the characteristic vector in Equation 2, with four values equal to 0.25 and the others equal to zero; whereas in the VSA version the final state approximates this distribution. (The small negative values are an artifact of using the cosine as a metric for comparison.)

CONCLUSIONS AND FUTURE WORK

The work presented here has demonstrated a proof-of-concept that a distributed representation (Vector Symbolic Architecture) can be applied successfully to a problem (graph isomorphism) that until now has been considered the purview of localist modelling. The results achieved with VSA are qualitatively similar to those with the localist formulation. In the process, we have provided an example of how a distributed representation can implement an operation reminiscent of lateral inhibition, winner-takes-all competition, which likewise has been considered to be a localist operation. The ability to model competition among neurally encoded structures and relations, not just individual items or concepts, points to promising new directions for cognitive modelling in general.

The next steps in this research will be to demonstrate the technique on larger graphs and investigate how performance degrades as the graph size exceeds the representational capacity set by the vector dimensionality. We will also investigate the performance of the system in finding subgraph isomorphisms.

Graph isomorphism by itself does not constitute a psychologically realistic analogical mapping system. There are many related problems to be investigated in that broader context. The question of what conceptual information is encoded in the graphs, and how, is foremost. It also seems reasonable to expect constraints on the graphs encoding cognitive structures (e.g. constraints on the maximum and minimum numbers of edges from each vertex). It may be possible to exploit such constraints to improve some aspects of the mapping circuit. For example, it may be possible to avoid the cognitively implausible use of non-edges as evidence for mappings.

Another area we intend to investigate is the requirement for population of the clean-up memories. In this system the clean-up memories are populated from representations of the source and target graphs. This is not unreasonable if retrieval is completely separate from mapping. However, we wish to explore the possibility of intertwining retrieval and mapping. For this to be feasible we would need to reconfigure the mapping so that cleanup memory can be populated with items that have been previously encountered rather than items corresponding to potential mappings.

We expect this approach to provide fertile lines of research for many years to come.

SOFTWARE DOWNLOAD

MATLAB code implementing the algorithm in (Pelillo, 1999) and our VSA version can be downloaded from tinyurl.com/gidemo

ACKNOWLEDGMENTS

We thank Pentti Kanerva, Tony Plate, and Roger Wales for many useful suggestions.

REFERENCES

Bomze, I. M., Budinich, M., Pardalos, P. M., & Pelillo, M. (1999) The Maximum Clique Problem. In D.-Z. Du & P. M. Pardalos (Eds.) *Handbook of combinatorial optimization. Supplement Volume A* (pp. 1-74). Boston, MA, USA: Kluwer Academic Publishers.

Eliasmith, C. (2005). Cognition with neurons: A large- scale, biologically realistic model of the Wason task. In G. Bara, L. Barsalou, & M. Bucciarelli (Eds.), *Proceedings of the 27th Annual Meeting of the Cognitive Science Society.*

Eliasmith, C., & Thagard, P. (2001). Integrating structure and meaning: A distributed

model of analogical mapping. *Cognitive Science*, *25*, 245-286.

Gayler, R. (1998). Multiplicative binding, representation operators, and analogy,. In K. Holyoak, D. Gentner, & B. Kokinov (Eds.), *Advances in analogy research: Integration of theory and data from the cognitive, computational, and neural sciences* (p. 405). Sofia, Bulgaria: New Bulgarian University.

Gayler, R. W. (2003). Vector Symbolic Architectures answer Jackendoff's challenges for cognitive neuroscience. In Peter Slezak (Ed.), *ICCS/ASCS International Conference on Cognitive Science* (pp. 133-138). Sydney, Australia: University of New South Wales.

Hecht-Nielsen, R. (1994). Context vectors: general purpose approximate meaning representations self- organized from raw data. In J. Zurada, R. M. II, & B. Robinson (Eds.), *Computational intelligence: Imitating life* (pp. 43-56). IEEE Press.

Holyoak, J., & Thagard, P. (1989). Analogical mapping by constraint satisfaction. *Cognitive Science*, *13*, 295- 355.

Hummel, J., & Holyoak, K. (1997). Distributed representations of structure: A theory of analogical access and mapping. *Psychological Review*, *104*, 427-466.

Kanerva, P. (1996). Binary spatter-coding of ordered k-tuples. In C. von der Malsburg, W. von Seelen, J. Vorbrüggen, & B. Sendhoff (Eds.), *Artificial neural networks (Proceedings of ICANN 96)* (pp. 869-873). Berlin: Springer-Verlag.

Kanerva, P. (2009). Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors. *Cognitive Computation*, *1*, 139-159.

Kokinov, B. (1988). Associative memory-based reasoning: How to represent and retrieve cases. In T. O'Shea & V. Sgurev (Eds.), *Artificial intelligence III: Methodology, systems, applications* (pp. 51-58). Amsterdam: Elsevier Science Publishers B.V. (North Holland).

Levy, S. D., & Gayler, R. W. (in press). "Lateral inhibition" in a fully distributed connectionist architecture. In *Proceedings of the Ninth International Conference on Cognitive Modeling (ICCM 2009)*. Manchester, UK.

Marquis, J.-P. (2009). Category theory. In E. N. Zalta (Ed.), *The Stanford Encyclopedia of Philosophy (Spring 2009 Edition)*, http://plato.stanford.edu/archives/spr2009/entries/category-theory/

Page, M. (2000). Connectionist modelling in psychology: A localist manifesto. *Behavioral and Brain Sciences*, *23*, 443-512.

Pelillo, M. (1999). Replicator equations, maximal cliques, and graph isomorphism. *Neural Computation*, *11*, 1933-1955.

Pelillo, M., Siddiqi, K., & Zucker, S. W. (1999). Matching hierarchical structures using association graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *21*, 1105-1120.

Pelillo, M., & Torsello, A. (2006). Payoff-monotonic game dynamics and the maximum clique problem. *Neural Computation*, 18, 1215-1258.

Plate, T. A. (2003). *Holographic reduced representation: Distributed representation for cognitive science*. Stanford, CA, USA: CSLI Publications.

Rumelhart, D. E., Hinton, G. E., & McClelland, J. L. (1986). A general framework for parallel distributed processing. In D. E. Rumelhart & J. L. McClelland (Eds.), *Parallel distributed processing: Explorations in the microstructure of cognition. Volume 1: Foundations* (pp. 45-76). Cambridge, MA, USA: The MIT Press.

Smolensky, P. (1990). Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence*, *46*, 159-216.

Stewart, T., & Eliasmith, C. (in press). Compositionality and biologically plausible models. In M. Werning, W. Hinzen, & E. Machery (Eds.), *The Oxford handbook of compositionality*. Oxford, UK: Oxford University Press.