A Neural Supergraph Matching Architecture

Stefan Klinger
Advanced Computer Architectures Group
Department of Computer Science
University of York
York, YO10 5DD, UK
E-mail: klinst@cs.york.ac.uk

Jim Austin
Advanced Computer Architectures Group
Department of Computer Science
University of York
York, YO10 5DD, UK
E-mail: austin@cs.york.ac.uk

***Abstract*— A neural supergraph matching architecture is introduced based on relaxation labeling and the minimum common supergraph of pairs of graphs. The system is implemented on correlation matrix memories and is efficient in constructing this supergraph. We test the effectiveness of this graphical cluster representation on two different sets of graphs.**

I. Introduction

The concept of a minimum common supergraph $MCS(g_1, g_2)$ was introduced by Bunke *et. al.* [1] as a the smallest graph that has both graphs g_1 and g_2 as its subgraphs. In the same paper, the authors linked the measure of the minimum graph edit distance between two graphs g_1 and g_2 to the size of the $MCS(g_1, g_2)$. They also proved that the $MCS(g_1, g_2)$ acts as a mean graph of g_1 and g_2 because it minimizes the graph edit distances between g_1 and g_2.

The notion of the minimum common supergraph was later extended to the weighted minimum common supergraph $WMCS(G)$ [2] for representing a cluster of similar weighted structural patterns $G = (g_1, ..., g_n)$. Unfortunately, the exact computation of the $WMCS(G)$ is exponential in the number of vertices and in the number of graphs. An approximate algorithm based on the pairwise computation of the weighted minimum common supergraph was suggested to circumvent this problem. This method depends on the order θ of the graphs in the cluster G and the resulting $WMCS(G)$ is no longer guaranteed to be optimal, i.e. it could contain more than the minimum number of vertices and edges. Furthermore, the resulting weights on some of the vertices and edges can also be suboptimal. A threshold p is subsequently applied to extract the common structural elements from the set of graphs G by removing all vertices and edges that have a weight below p. This way, it is possible to separate the data representing a graphical object from any noise introduced by the system.

We propose to present a different approximation algorithm for the generation of the WMCS based on relaxation labeling and correlation matrix memories.

II. Definitions

The following are definitions given by Bunke *et. al.* [2] and are reproduced here for the purpose of completeness. A maximum common subgraph $mcs(g_1, g_2)$ of two graphs g_1 and g_2 is a subgraph of both g_1 and g_2 and has among all those subgraphs, the maximum number of vertices. A supergraph of graphs g_1 and g_2 is a graph that has both g_1 and g_2 as its subgraphs. The minimum common supergraph $MCS(g_1, g_2)$ of two graphs g_1 and g_2 is a supergraph that has among all possible supergraphs, the minimum number of vertices. The difference between g_2 and a subgraph g_1, $g_2 - g_1$ is calculated by deleting g_1 from g_2 together with the edges connecting g_1 with the remaining graph. These edges are referred to as the embedding of g_1 in g_2, $E = emb(g_1, g_2)$. The minimum common supergraph $MCS(g_1, g_2)$ of a pair of graphs g_1 and g_2 is computed as follows:

$$MCS(g_1, g_2) = mcs(g_1, g_2) \cup_{E1} (g_1 - mcs(g_1, g_2)) \cup_{E2} (g_2 - mcs(g_1, g_2)) \quad (1)$$

where $E_1 = emb(mcs(g_1, g_2), g_1)$ and $E_2 = emb(mcs(g_1, g_2), g_2)$.

Furthermore, a weighted graph for storing the occurrences of vertices and edges in a cluster is defined as $g = (V, \lambda, E, \epsilon, \alpha, \beta)$ where

- V is the set of vertices
- $\lambda : V \rightarrow N^+$ assigns positive weights to the vertices
- $E \subseteq V \times V$ is the set of edges
- $\epsilon : E \rightarrow N^+$ assigns positive weights to the edges
- $\alpha : V \rightarrow L$ assigns attributes to the vertices
- $\beta : E \rightarrow L$ assigns attributes to the edges

In addition, we denote $N(v_i) \subseteq V \backslash v_i$ as the set of adjacent vertices of vertex v_i, i.e. $(v_i, v_j) \in E, \forall v_j \in N(v_i)$.

The Weighted Common Supergraph $WCS(G)$ of a set of graphs $G = (g_1, ..., g_n)$ is a weighted graph g such there exist subgraph isomorphisms from g to $g_i, \forall g_i \in G$. If there exists no other weighted common supergraph of G with fewer vertices than g, then g is called the Weighted Minimum Common Supergraph $WMCS(G)$ of G. The weights of the vertices and edges are equivalent to their number of occurrences within the maximum common substructures in the cluster G.

The above definition of $WMCS(G)$ is exponential in the number of vertices and number of graphs in the cluster G. In order to reduce the complexity of the problem, Bunke *et. al.* suggested an approximate method based on the pairwise computation of the $WMCS(G)$ using the weighted maximum common subgraph $wmcs(g_1, g_2)$ of a pair of graphs [2]. Given two weighted graphs g_1 and g_2, the $WMCS(g_1, g_2)$ can be computed as follows:

$$WMCS(g_1, g_2) = wmcs(g_1, g_2) \cup_{E1} (g_1 - wmcs(g_1, g_2)) \cup_{E2} (g_2 - wmcs(g_1, g_2)) \quad (2)$$

where $E_1 = emb(wmcs(g_1, g_2), g_1)$ and $E_2 = emb(wmcs(g_1, g_2), g_2)$. The $wmcs(g_1, g_2)$ can be calculated by any standard mcs algorithm. In their study, the authors applied the algorithm of McGregor [3]. The computation of $WMCS(G)$ depends on a chosen order θ and is calculated for every pair $(WMCS, g_i), \forall i \in (2, ..., n-1)$ using (2) where the $WMCS$ is initially set to g_1. The influence of the order on the accuracy was reduced by altering the sequence θ using random shuffles. However, no significant improvements with increasing number of shuffles were observed and therefore we do not consider it for this study.

III. Neural Supergraph Algorithm

There are several potential problems associated with the algorithm described above. First of all, due to its approximate nature, the number of vertices and edges and their corresponding weights might not be optimal because only pairwise common substructures in a certain order are included in the generation of the $WMCS(G)$. Secondly, the mcs of two graphs is not necessarily unique and the weights are only increased on one chosen mcs from the set of available maximum common subgraphs. Finally, the mcs problem is NP-complete [4], which means, that the exact computation might have exponential time order growth in the worst case.

Here, we present methods for overcoming the limitations of the last two problems by using an efficient neural graph matching algorithm.

A. Relaxation By Elimination

Our algorithm is based on the relaxation labeling technique [5]. The general idea behind relaxation labeling is to iteratively update vertex correspondences of two graphs based not only on the unary measurements, but to take into account the edge attributes in the contextual neighborhood as well. Relaxation labeling is composed of two categories, discrete and probabilistic. Discrete relaxation [6] is realized by computing a local consistency measure for possible vertex assignments and replacing the current correspondence in case of an improvement in the matching criterion. In probabilistic relaxation [7] the feasible correspondences are weighted to indicate a belief in each of the current vertex assignments. The probabilities of the correspondences are iteratively updated in a way that maximizes the *a posteriori* probability of an individual match. Unlike probabilistic and discrete relaxation methods that operate similarly to hill climbing optimization, our Relaxation By Elimination (RBE) [8] technique initially keeps all plausible solutions and iteratively removes the most unlikely candidates.

Consider two graphs, $g_1 = (V_1, \lambda_1, E_1, \epsilon_1, \alpha_1, \beta_1)$ and $g_2 = (V_2, \lambda_2, E_2, \epsilon_2, \alpha_2, \beta_2)$. Each vertex $v_i \in V_1$ of graph g_1 has a list of potential candidates $C_i \subseteq V_2$ that denote the plausible vertex associations between g_1 and g_2. The list is initialized based on matching vertex attributes, i.e. we add $v_x \in V_2$ to the set C_i if

$$\|\alpha_1(v_i) - \alpha_2(v_x)\| < \epsilon \quad \forall v_i \in V_1, v_x \in V_2 \quad (3)$$

for some defined tolerance value ϵ.

The lists of candidates C_i are pruned iteratively by removing implausible candidates that have a low support from neighboring vertex candidates C_j where $v_j \in N(v_i)$. Here, we count the number of neighboring vertices that have at least one candidate that is consistent with the current assignment $c_{ix} \in C_i$. We define the discrete binary consistency measure $h(\beta_1(v_i, v_j), \beta_2(c_{ix}, c_j))$ as

$$h(\beta_1(v_i, v_j), \beta_2(c_{ix}, c_j)) = \begin{cases} 1, & \text{when } \|\beta_1(v_i, v_j) - \beta_2(c_{ix}, c_j)\| < \epsilon \\ 0, & \text{otherwise} \end{cases} \quad (4)$$

The support function $S(c_{ix}) = \sum_{j \in N(v_i)} h(\beta_1(v_i, v_j), \beta_2(c_{ix}, c_j))$ sums the number of consistent binary relations for the current candidate assignment $c_{ix} \in C_i$. By defining an appropriate threshold γ, we can eliminate all vertex associations that do not exceed this threshold. We remove the candidate c_{ix} if $S(c_{ix}) < \gamma$. This process is repeated for each vertex until a stopping criterion is met, e.g. all remaining candidates $c_{ix} \in C_i$ have support $S(c_{ix}) \geq \gamma, \forall c_{ix} \in C_i$.

B. Supergraph algorithm

We now transform this algorithm into a supergraph method. The binary consistency measure $h(\beta_1(v_i, v_j), \beta_2(c_{ix}, c_j))$ is equivalent to the current number of neighboring vertex correspondences in the neighborhood of vertex v_i that are consistent with a given vertex candidate c_{ix} of vertex v_i. The threshold γ specifies the minimum support required for any given candidate c_{ix}. The number of consistent vertex mappings in this common subgraph is at least $\gamma + 1$, assuming that such a subgraph does exist. Note that this subgraph is not necessarily optimal because we only consider binary relationships $\beta_1(v_i, v_j), \forall v_i, v_j \in V_1$ where $e(v_i, v_j) \in E_1$ and only estimate an upper bound for the edges $e(v_j, v_k) \in E_1, v_k \neq v_i$ [8].

The algorithm starts by setting $\gamma = 1$ and runs the conventional RBE algorithm until a stable state has been reached, i.e. $S(c_{ix}) \geq \gamma, \forall (v_i, c_{ix}) \in C^\gamma$. We denote $C^\gamma = v_i \times C_i, \forall v_i \in V_1$ as the set of all vertex candidate pairs with support of at least γ. The threshold γ is incremented by 1 when the RBE algorithm has reached a stable state. This process is repeated until no more candidates remain, i.e. $C^\gamma = \emptyset$. The penultimate set $C^{\gamma-1}$ contains the set W of all possible subgraphs remaining that have at least γ consistent vertex pairs. Note that this value represents an upper bound on the size of the mcs and its actual size might be smaller.

Since we now have the set of all feasible subgraphs remaining, we could use any mcs algorithm to determine one of the subgraphs with the maximum number of vertices. However,

this would disregard some of the vertices and edges that are included in the set W. Alternatively, we could treat the entire set W as the collection containing all of the maximum elements among the two graphs g_1 and g_2. In this case, we increment the weight on each vertex v_i that has $C_i \neq \emptyset$. Similarly, the edge weights $\epsilon(e(v_i, v_j))$ are incremented whenever $\beta_1(v_i, v_j) = \beta_2(c_{ix}, c_{jx}), c_{ix} \in C_i, c_{jx} \in C_j$. In this study, we denote the algorithm that increases the weights on one mcs only as neural(1), while the procedure that takes all remaining vertices and edges into account as neural(2).

For the purpose of calculating the embedding $emb(g_1, g_2)$, we need to determine a single representative mcs from the set W. It is a well-known fact that the mcs problem of two graphs can be cast to a maximum clique problem of the association graph g_a of g_1 and g_2 [9]. The association graph g_a is created on the vertex set $C^{\gamma-1}$ and two vertices (v_i, c_{ix}) and (v_j, c_{jy}) are adjacent whenever $(v_i, v_j) \in E(g_1)$ and $(c_{ix}, c_{jy}) \in E(g_2)$ and $\beta(v_i, v_j) = \beta(c_{ix}, c_{jy})$. Furthermore, all vertex pairs are connected if they are disconnected in both graphs, i.e. $(v_i, v_j) \notin E(g_1)$ and $(c_{ix}, c_{jy}) \notin E(g_2)$. Here, we use the Bron-Kerbosch maximal clique algorithm [10] because of its straight-forward implementation.

Once we have found a $mcs(g_1, g_2)$, we can apply (2) in order to determine the $WMCS(g_1, g_2)$. As in the original paper, we separate information from noise by applying a threshold p to obtain the $WMCS_p(G)$.

C. Neural Architecture

The process described above has the potential of fast and efficient implementation using an architecture of inter-connected Correlation Matrix Memories (CMMs) [11]. A CMM is a simple binary associative neural network that offers quick training and highly flexible and fast search capability. The CMM has been used as a match engine in a number of successful applications, e.g. symbolic reasoning in the AURA (Advanced Uncertain Reasoning Architecture) approach [12] and post code matching.

The list of candidates C_i can be represented as a binary array. Furthermore, if measurements are discretized, then the support function can be executed through the use of bitwise operations on binary arrays. The bit vector of current candidates C_i of vertex v_i is used as an input to the processing of evidence counts for candidates $C_{N(v_i)}$ of adjacent query vertices $N(v_i)$. This process can be performed for each query node candidate list C_i in parallel. The use of CMMs also allows the sharing of rows in memory by multiple binary patterns. This enables efficient use of memory at the expense of introducing false positive results. By superimposing the set of vertices from more than one model graph in the candidate list C_i, multiple graph correspondences can be matched in parallel. In order to keep the number of false positives to a minimum, we ensure that all single vertex patterns are orthogonal to each other.

D. Complexity

The maximum common subgraph problem is known to be NP-complete [4]. The Bron-Kerbosch maximal clique algorithm is an optimal method which means it will experience exponential order of time growth in the worst case. However, it might be much faster in the average case. For example, Wilf [13] has shown that the maximum independent set problem has sub-exponential time complexity of $O(n^{logn})$ in the average case. Relaxation procedures replace this problem with a polynomial-time algorithm, however, this guarantees to find solutions that are only locally optimal. On the other side, Relaxation By Elimination determines the set of all plausible solutions above a defined threshold. However, further processing is required to find the optimal solution from the set of feasible graph intersections. The RBE method has a worst-case time complexity of $O(|V_1|^2|V_2|^2)$ [8].

IV. Experiments

We propose two experiments based on simple, planar Attributed Relational Graphs.

The first experiment is run on graphs extracted from pictures using the plex grammar tool [14]. This tool generates large collections of similar images based on the defined grammar. Each component of the picture is represented by a vertex. On each vertex the area of the bounding box of the component is used as the attribute. The area is normalized between 0 and 99 for all components and two vertices have the same attribute if the difference between their areas is below 5. Each edge is assigned the distance between the two vertices as its attribute. As done for the area, the distances are normalized between 0 and 99 and two edge attributes are considered to be identical if their discrepancy is below 5. In the first test case, we create the graphs of the pictures, completely connect all vertices and set the edge attributes to the respective distances between the vertices. In the second test case, we only assign edge attributes if the distance between the vertex pairs is below 20. The remaining vertex pairs remain connected, however, their edge distance is set to null. A sample of plex pictures showing men, ships and houses is depicted in Fig. 1.

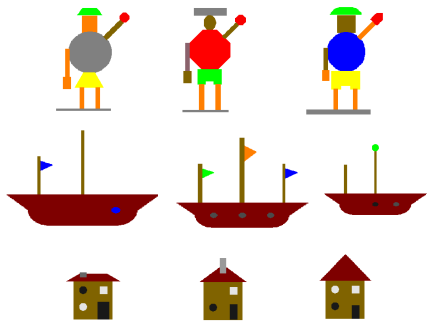

Fig. 1. Examples of images generated by the plex grammar tool

The quality of the created cluster representative graph

$WMCS(G)$ is evaluated using the following entropy function [2] :

$$E = -\sum_{i=1}^{t} \frac{w_i}{n} \cdot log\frac{w_i}{n} - \sum_{i=1}^{t}\sum_{j=1}^{t} \frac{w_{ij}}{n} \cdot log\frac{w_{ij}}{n} \qquad (5)$$

where n is the number of graphs in the cluster, t is the number of vertices of the $WMCS(G)$, w_i is the weight of the vertex i and w_{ij} is the weight of edge $e(i,j)$. The minimum of E is obtained if the cluster contains only isomorphic graphs, while the maximum is achieved if there is no overlap between any pairs of graphs in the cluster. We create clusters for three different classes of patterns using 10 random patterns per class. We test the quality of the $WMCS(G)$ by evaluating the differences in entropy when a new pattern from the same class or from different classes is added to it. The tests are carried out for 100 ships, 100 houses and 100 men drawn randomly from the plex grammar tool. The classification rates based on the minimum entropy variations for the original algorithm and our neural methods are shown in tables I, II and III, respectively.

TABLE I
CLASSIFICATION RATES WITH CONVENTIONAL MCS ALGORITHM

	Global			Local		
	man	house	ship	man	house	ship
man	99	1	0	80	20	0
house	2	95	3	0	100	0
ship	6	16	78	0	0	100

TABLE II
CLASSIFICATION RATES WITH NEURAL(1) ALGORITHM

	Global			Local		
	man	house	ship	man	house	ship
man	100	0	0	82	0	18
house	1	96	3	0	100	0
ship	5	10	85	2	4	94

TABLE III
CLASSIFICATION RATES WITH NEURAL(2) ALGORITHM

	Global			Local		
	man	house	ship	man	house	ship
man	100	0	0	95	0	5
house	1	99	0	0	100	0
ship	7	14	79	25	2	73

These results indicate that in some instances the neural algorithm achieves better results than the conventional algorithm, however, in most cases it misclassifies more patterns than the exact method. The findings also seem to show that the men form more cohesive clusters using global distances, while the house and ship prefer local distances as their edge features. This is probably due to the fact that the ship and houses have a smaller number of vertices and these form different local patterns that are shared by the men, therefore fewer vertices and edges are added to the cluster graph.

We now run a similarity search using the representative cluster graph $WMCS_p(G)$ for each of the three local clusters as the target g_1. The database of model graphs is made up of the 300 graphs used in the previous experiment containing 100 ships, 100 houses and 100 men. The representative cluster graph is obtained by thresholding the $WMCS(G)$ at different thresholds p ranging from 1 to 9. The value of p specifies the minimum number of occurrences of a vertex or an edge within the cluster in order to consider it to be part of the cluster graph. We run the RBE algorithm introduced earlier [8] as the similarity search and we apply the $Willshaw(\gamma)$ [11] threshold during the elimination stage. This approach retains all candidates $c_{ix} \in C_i$ that have a support of their neighbor candidates greater than γ, i.e. $S(c_i) > \gamma$. Here, we vary γ using values between 1 and $|V_{WMCS_p(G)}|$.

We compare the effectiveness of the search using the precision and recall values of the returned data sets. Precision is defined as the fraction of correct graphs of the cluster found (c) over the total number of graphs retrieved (n), i.e. $P = \frac{c}{n}$. Recall is defined as the fraction of correct graphs retrieved (c) over the total number of graphs of this cluster contained in the database (C), i.e. $R = \frac{c}{C}$. The total number of graphs from all clusters in the database are referred to as N. We evaluate the search effectiveness using the Guner-Henry (GH) score [15] that is defined as:

$$GH = \left(\frac{a(3 \cdot N + n)}{4 \cdot n \cdot C}\right)\left(1 - \frac{n - c}{N - C}\right) \qquad (6)$$

We apply cutoff values of $P \geq 0.5$ and $R \geq 0.05$ [16] because searches that results in values below these thresholds represent poor effectiveness. We discard searches that do not achieve these thresholds and denote them by the shorthand D. The resulting highest GH scores together with the corresponding precision and recall values for the similarity searches for the original $WMCS_p(G)$ algorithm are depicted in table IV.

TABLE IV
BEST GH SCORES OF THE ORIGINAL $WMCS$ ALGORITHM

	Global				Local			
	p	GH	P	R	p	GH	P	R
man	0.6	0.91	1.00	0.64	0.3	0.94	0.99	0.83
house	0.9	0.99	1.00	0.98	0.9	0.98	1.00	0.94
ship	0.2	0.76	1.00	0.05	0.2	0.53	0.67	0.26

Note that we do not apply any similarity metric based on the size of the mcs. The reason for this is that most of those metrics normalize the score based on the difference in size between the two graphs. However, at low thresholds p the graph $WMCS_p(G)$ is relatively large, so larger graphs with the same size of the mcs are ranked higher than smaller graphs. The converse is true for large thresholds p. Here, we are only interested in those graphs that share a minimum common substructure with the $WMCS_p(G)$ that is at least of the size given by the threshold γ.

We apply the same set of rules to the neural supergraph algorithm. In the first instance, the neural(1) algorithm determines the mcs between the two graphs of the cluster before incrementing the weights on the vertices and edges of that graph intersection. In the second case, the neural(2) method increments the weights on all vertices and edges remaining after the neural match algorithm has terminated. The results of both methods are shown in tables V and VI, respectively.

TABLE V
BEST GH SCORES OF THE NEURAL(1) ALGORITHM

	Global				Local			
	p	GH	P	R	p	GH	P	R
man	0.4	0.90	0.94	0.89	0.6	0.85	0.91	0.86
house	0.9	0.88	0.91	0.98	0.8	0.98	1.00	0.94
ship	0.2	0.77	1.00	0.07	D	D	D	D

TABLE VI
BEST GH SCORES OF THE NEURAL(2) ALGORITHM

	Global				Local			
	p	GH	P	R	p	GH	P	R
man	0.4	0.92	1.00	0.68	0.3	0.86	0.91	0.84
house	0.8	0.86	0.89	0.98	0.9	0.79	1.00	0.14
ship	0.8	0.35	0.50	0.77	D	D	D	D

The best mean GH scores for the similarity search using each of the 10 cluster graphs of the three classes as targets are shown in table VII below.

TABLE VII
BEST MEAN GH SCORES OF 10 MCS SIMILARITY SEARCHES

	Global			Local		
	GH	P	R	GH	P	R
man	0.89	0.97	0.69	0.76	0.86	0.76
house	0.82	0.99	0.31	0.79	0.99	0.18
ship	0.66	0.86	0.15	0.72	0.93	0.11

Judging from these results, it seems there is an advantage in using a cluster graph representation for retrieving similar patterns belonging to the same class of patterns. All $WMCS_p(G)$ algorithm achieve a better effectiveness in terms of the GH scores than the similarity search using single queries for the man and house classes. The ship class is best classified by the exact $WMCS_p(G)$ algorithm with global distances, while using local distances the 10 single query similarity searches prevail. In the latter instance, the neural algorithms do not even achieve the minimum precision and recall cutoff values.

The second experiment clusters chemical graphs that possess a common activity. The term activity is referred to a property of a molecule that enables it to alter or inhibit the function of a given target protein. Based on the structure-similarity principle [17], we assume that similar activities of molecules are based on similar structural properties. For this experiment, we use the ATP data set made up of 10,930 molecules supplied by Evotec OAI (http://www.evotecoai.com). The database contains 10,000 diverse structures drawn randomly from the supplier database of Evotec OAI. The remaining 930 molecules are known actives that have similar binding properties and were also extracted from different sources. We randomly select 10 active compounds from the available 930 active structures. These compounds are depicted in Fig. 2.

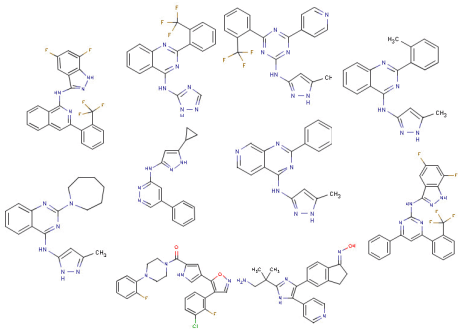

Fig. 2. Active Structures from the ATP data set

We denote the atoms as graph vertices and set their atomic numbers as the vertex attribute. In line with the first experiment, we apply both global and local binary features. The global edge attributes are set to the number of bonds separating two atoms in the shortest path, while the local edge features are limited to bond distances up to a maximum number of bonds of 3. There exists only one cluster because the non-active structures are diverse in their structural properties by definition. Therefore, we only conduct the similarity searches by applying the different thresholds p as in the first experiment. The precision and recall values of the retrievals with the highest GH scores of the three $WMCS$ algorithms are displayed in table VIII.

TABLE VIII
PRECISION AND RECALL VALUES FOR THE SIMILARITY SEARCH USING THE WMCS

	Global				Local			
	p	GH	P	R	p	GH	P	R
original	0.7	0.50	0.60	0.22	0.5	0.50	0.64	0.11
neural(1)	0.7	0.43	0.55	0.06	0.5	0.48	0.59	0.16
neural(2)	0.6	0.42	0.54	0.09	D	D	D	D

For comparison, we run a similarity search for each of the queries contained in the cluster of active molecules. As before, retrievals that result in precision or recall values below the given thresholds are discarded. The average scores of acceptable retrievals are shown in table IX.

Again, there is some value added to the retrieval by using a structural cluster representation. We avoid the execution of searches that are later discarded due to bad precision and recall values.

TABLE IX

MEAN PRECISION AND RECALL VALUES FOR 10 SIMILARITY SEARCHES

	Global				Local			
	GH	P	R	D	GH	P	R	D
mcs	0.46	0.59	0.07	5	0.40	0.52	0.05	7

On the other side, we require a method to set the threshold p in order to get a common substructure $WMCS_p(G)$ that discriminates well between active and inactive molecules in our database. At high thresholds p, this structure is likely to be very small because it represents those structural patterns found in most of the cluster graphs. Small structures are more likely to be contained within any molecule, so we end up with a high recall and low precision values for our similarity search. The opposite is true for small values of p. Here, the cluster graph emphasizes the unique elements of the structures contained within that group. The result is a high precision and low recall of the search. Therefore, in order to achieve an acceptable retrieval performance, the medium-sized structures at intermediate threshold levels p should be used as the query for the search. These findings are in line with another study [18]. One possible way to determine the most suitable cluster graph would be to run an information gain analysis on a small test set [19]. This will become part of a further study.

The results of both experiments indicate that the effectiveness of the neural graph matcher is generally worse compared to the original $WMCS(G)$ algorithm. We compare the efficiency by measuring the time to generate the cluster graph. All algorithms are written in Visual C++ .NET and run on a AMD Athlon XP2000+ PC with MS Windows 2000. The execution times are depicted in table X.

TABLE X

AVERAGE TIME (IN S) TO BUILD A $WMCS(G)$ CLUSTER GRAPH

	Exact $WMCS(G)$	Neural $WMCS(G)$
plex (global)	0.7	0.6
plex (local)	2.1	1.4
molecule (global)	14.2	12.7
molecule (local)	1,435	1,074

These times show the generation of the cluster graph is generally faster using the neural $WMCS(G)$ algorithm. The durations do not differ significantly in these tests because we only use 10 small graphs from each of the classes to create the approximate mean graph of that class. In addition, an exact mcs algorithm is required after the elimination stage to remove all ambiguous assignments for determining the embedding $emb(g_1, g_2)$. The execution time is slowest for local distances because in this case the association graph is very dense since many edges in the factor graphs g_1 and g_2 share the null edge attribute.

V. CONCLUSION

This study introduces a neural method for generating a mean graph of a cluster of similar structural patterns. In general, the cluster representation shows great promise in achieving better similarity search performances than using single structural patterns as targets. The neural algorithm increases the efficiency of the original method at the expense of precision and recall of the search results.

ACKNOWLEDGEMENT

This study was undertaken as an EPSRC CASE studentship no GR/P03292/01 with the support of Evotec OAI.

REFERENCES

[1] H. Bunke, X. Jiang, and A. Kandel, "On the minimum common supergraph of two graphs," *Computing*, vol. 65, no. 1, pp. 13–25, 2000.
[2] H. Bunke, P. Foggia, C. Guidobaldi, and M. Vento, "Graph clustering using the weighted minimum common supergraph," *Lecture Notes in Computer Science*, vol. 2726, pp. 235–246, 2003.
[3] J. McGregor, "Backtrack search algorithms and the maximal common subgraph problem," *Software Practice and Experience*, vol. 12, pp. 23–34, 1982.
[4] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York: W.H. Freeman, 1979.
[5] A. Rosenfeld, R. Hummel, and S. Zucker, "Scene labelling by relaxation operations," *IEEE Trans. Syst., Man, Cybern.*, vol. 6, pp. 400–433, 1976.
[6] E. Hancock and J. Kittler, "Discrete relaxation," *Pattern Recognition*, vol. 23, pp. 711–733, 1990.
[7] J. Kittler and E. Hancock, "Probabilistic relaxation," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 3, no. 1, pp. 29–51, 1989.
[8] M. Turner and J. Austin, "Graph matching by neural relaxation," *Neural Computing and Applications*, vol. 7, pp. 238–248, 1997.
[9] H. Barrow and R. Burstall, "Subgraph isomorphism, matching relational structures and maximal cliques," *Information Processing Letters*, vol. 4, pp. 83–84, 1976.
[10] C. Bron and J. Kerbosch, "Finding all cliques of an undirected graph," *Communications of the ACM*, vol. 16, no. 9, pp. 575–577, 1973.
[11] D. Willshaw, O. Buneman, and H. Longuet-Higgins, "Non-holographic associative memory," *Nature*, vol. 222, pp. 960–962, 1969.
[12] J. Austin, "Distributed associative memories for high-speed symbolic reasoning," *Fuzzy Sets and Systems*, vol. 82, pp. 223–233, 1995.
[13] H. Wilf, *Algorithms and Complexity*. New Yersey: Prentice-Hall, 1986.
[14] M. Hagenbuchner and A. Tsoi, "The traffic policeman benchmark," in *European Symposium on Artificial Neural Networks*, ser. ISBN 2-9600049-9-X, M. Verleysen, Ed. D-Facto, April 1999, pp. 63–68.
[15] O. Guner, *Pharmacophore Perception, Development and Use in Drug Design*. La Jolla, CA, USA: International University Line, 2000, p. 194.
[16] J. Raymond and P. Willet, "Effectiveness of graph-based and fingerprint-based similarity measures," *Journal of Computer-Aided Molecular Design*, vol. 16, pp. 59–71, 2002.
[17] A. Johnson and G. Maggiora, *Concepts and Applications of Molecular Similarity*. New York: Wiley, 1990.
[18] A. Bender, H. Mussa, R. Glen, and S. Reiling, "Molecular similarity searching using atom environments, information-based feature selection, and a naive bayesian classifier," *Journal of Chemical Information and Computer Sciences*, vol. 44, pp. 170–178, 2004.
[19] J. Quinlan, "Introduction of decision trees," *Machine Learning*, vol. 1, pp. 81–106, 1986.