Analysing Structure in Complex Networks Using Quality Functions Evolved by Genetic Programming

Fergal Reid
Clique Research Cluster
Complex and Adaptive Systems Laboratory
University College Dublin
Ireland
fergal.reid@gmail.com

Neil Hurley
Clique Research Cluster
Complex and Adaptive Systems Laboratory
University College Dublin
Ireland
neil.hurley@ucd.ie

ABSTRACT

When studying complex networks, we are often interested in identifying structures within the networks. Previous work has successfully used algorithmically identified network structures to predict functional groups; for example, where structures extracted from protein-protein interaction networks have been predictive of functional protein complexes. One way structures in complex networks have previously been described is as collections of nodes that maximise a local quality function. For a particular set of structures, we search the space of quality functions using Genetic Programming, to find a function that locally describes that set of structures. This technique allows us to investigate the common network properties of defined sets of structures. We also use this technique to classify and differentiate between different types of structure. We apply this method on several synthetic benchmarks, and on a protein-protein interaction network. Our results indicate this is a useful technique of investigating properties that sets of network structures have in common.

Categories and Subject Descriptors: G.2.2 [Graph Theory]: Network problems; I.2.8 [Problem Solving, Control Methods, and Search]: Heuristic methods; I.5.1 [Models]: Structural

General Terms: Algorithms, Theory

Keywords: Genetic Programming, Complex Networks, Structure

1. INTRODUCTION

1.1 Structure in Complex Systems

In the study of complex systems, it is sometimes useful to consider the individual interacting parts of a system as a network. By studying the properties of that network, we can sometimes find structure latent in the system. Considering complex systems in this way has yielded insight across many different disciplines. For example, in the biological domain, the interactions of proteins may be considered as a network. Previous work has found that higher level functional groupings – protein complexes – may be discovered algorithmically by looking for network structures in the graph of interacting proteins[1],[2],[18].

There are now many algorithms that seek to uncover structures across networks generally, across many domains. At an abstract level, this search for higher level structure is motivated by similar principles to evolutionary computation – higher level structure is thought likely to exist in complex systems generally, because it is more efficient to evolve large systems from combinations of high level subsystems, rather than from the simplest components. A frequently cited work discussing the connection between these ideas is that of Simon[19].

In the complex networks literature, the problem of uncovering higher level structures is referred to as 'Community Finding'. A diverse set of algorithms, optimised for differing ideas of what specific higher level network structure should be found – and hence what 'community' means – has been developed. These algorithms operate on a network, and return a set of structures. In general, using the language of graph theory, a network is considered as a set of *nodes*, or *vertices*, and a set of *edges* which exist between pairs of nodes. Edges may be directed or undirected, depending on the domain; for simplicity we will consider only undirected networks, though all methods discussed generalise. To illustrate the concept of a network, a subset of a protein-protein interaction network is visualised in Figure 1.

Given a network consisting of nodes and edges, there are many different algorithms which attempt to find different types of embedded structure. Fortunato[7] provides a comprehensive review of the field. Some algorithms seek to find certain motifs in graphs - such as cliques, or percolated cliques; others seek to find structure that best satisfies some generative or statistical model of community; still others define a local 'fitness' objective function on a set of nodes, the function designed to be at a maximum when the nodes form a good community. It this last family of algorithms, which define a local community 'fitness function', that is then maximised for, that we are concerned with in this work.

1.2 Community quality functions

One method of finding communities is that of defining a 'fitness function' (hereafter 'community quality function' to avoid confusion with the term 'fitness function' in evolutionary computation) that operates on a set of nodes from

GECCO'11, July 12–16, 2011, Dublin, Ireland.

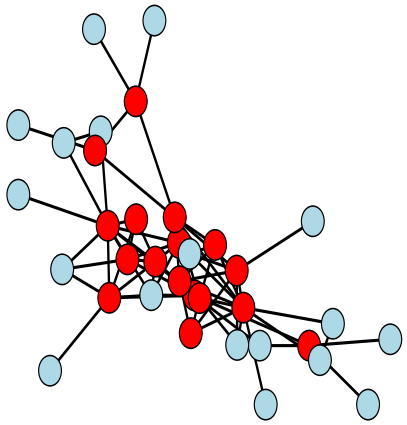

Figure 1: Visualisation of a subset of a protein-protein interaction network[6], where the edges represent the interactions between proteins. The nodes of a particular complex – highly interconnected – are shown in red.

the network, and defines how good a community that set of nodes is. Sets of nodes that locally maximise this quality function can then be found, by a variety of algorithmic search techniques; these groups of nodes will be considered communities. Because the search space of possible combinations of nodes in any non-trivial network is very large, various optimisations and heuristics are employed to find sets of nodes that optimise the quality function. For example, one technique is to use simple greedy methods, which start with some set of nodes, and then repeatedly add to the set the *connected* node which would most increase the quality of the set, until a local maximum for the quality function is reached. More sophisticated search strategies, such as simulated annealing, or genetic search, have also been used.

The quality functions which describe the structures of interest are often arrived at because they correspond to an individual researchers intuition of what 'community' – i.e. higher level structure – should be. Thus a variety of different local quality functions exist - examples include those proposed in [11], [5], [15]. In order to validate the appropriateness of a particular quality function as a community finding method, typically a network is chosen, for which a known set of specific higher level structures, already exists. Then sets of nodes which maximise the community quality function are found algorithmically, and these sets of found structure are then compared against the pre-defined structures. As an example of this approach, one area where important structural data is available are protein-protein interaction networks, where structures found in the network of interactions are benchmarked by evaluating their ability to predict functional protein complexes.

While it is often intended that community finding techniques be broadly applicable across diverse domains, it is perhaps likely that no one type of structure, and hence no one particular quality function, is optimal across every domain. In this research, we examined whether, when given the ground truth communities for a set of network data, it was possible to use a supervised approach to automatically extract information on what an appropriate quality function might be.

1.3 Using genetic search to find quality functions

While previous approaches have used heuristically defined quality functions to find particular structure relevant within a particular domain, it isn't clear that any specific quality function works across all domains. Many different types of fitness function exist, and many ways of maximising them have been proposed. We considered whether it would be possible, given a set of ground truth communities, to search for a quality function that describes structural properties common to those communities. Such a quality function would ideally be at a local maximum for as many of the ground truth communities as possible. Some existing quality functions are relatively simple mathematical combinations of certain defined structural properties of the communities. One example of this is the local quality function of Lancichinetti et al[11], shown in Equation 1. As Lancichinetti et al. write: "Here a community is a subgraph identified by the maximization of a property or fitness of its nodes. We have tried several options for the form of the fitness and obtained the best results with the simple expression"

$$Quality(S) = \frac{k_{in}^S}{(k_{in}^S + k_{out}^S)^\alpha}, \quad (1)$$

This function is a simple function of k_{in}, the number of edges that exist within a given network structure S, and k_{out}, the number of edges that cross from that structure out into the rest of the network. In use, the α parameter is frequently set to 1.0, further simplifying the function.

Our goal in this work is to learn something about the appropriate choice of quality function to describe particular structures in a network, when we have access to a set of known 'ground truth' examples of those structures. We found that genetic programming, as described by Koza[10], provided a very natural representation to allow us to define a search across the space of quality functions, in order to find a quality function that best describes a known set of structures.

1.4 Other evolutionary computation approaches in this domain

Previous work [22] has applied genetic algorithms to find a set of nodes from an e-mail communications database, maximising certain social network quality criteria, in order to extract a useful filtered subset of a larger communications network. Other authors [13], [8] have tackled the problem of finding a set of nodes to maximise a *particular* quality function, using genetic search. There are many ways to find sets of nodes that maximise a given quality function, and indeed, the search space of possible nodes to consider is typically very large, so some optimised search method is generally needed. However, what we are concerned with is searching to determine which quality function we should be

using, given a set of training structures. We are not aware of previous work applying evolutionary computation techniques to the problem of deciding which quality function well describes a set of structures.

2. EXPERIMENTS

2.1 Genetic Programming

As we want a very general method of finding quality functions that can describe sets of known structure, and the space of possible functions to search over is very large, this problem seems a natural fit for the technique of genetic programming. We must first define some space for the quality functions to be drawn from; we choose a grammar for our functions that contains many of the measures we currently use to discuss structure in complex networks, and which would be capable of generating a very general vocabulary of simple quality functions. The structure of the grammar we use to define the possible functions is deliberately simple in this work, for efficiency reasons. However, it allows the generation of simple functions operating on a range of standard network features, and is shown in Figure 2.

```
<expr> ::= <var>  | ( <expr> <op> <expr> )
<op>   ::= + | * | - | / | ^
<var>  ::= InternalEdges | ExternalEdges |
           Diameter | MaxDegree | NumberNodes |
           LargestCliqueSize | ClusterCoefficient |
           AverageShortestPathLength | 1
```

Figure 2: Structure quality function grammar.

The selection of which particular network features to include is somewhat arbitrary; however, we have chosen a range of features used to generally characterise networks, as well as features found in existing quality functions; and there is no reason why in principle this set of features could not be expanded or altered to include other structural features of interest.

The features we chose to characterise structures were:

- *InternalEdges* is the total number of edges within the structure under evaluation, where an internal edge connects a node to another node within the structure.
- *ExternalEdges* is the total number of edges from nodes within the structure that connect to nodes outside the structure.
- The *Diameter* of the structure is the length of the longest path through the structure, without cycles.
- *MaxDegree* of the structure is the degree of the node with largest degree.
- *NumberNodes* simply describes the total number of nodes in the structure.
- *AverageShortestPathLength* is the average of the length of the shortest path between each pair of nodes in the structure, and a famous network statistic[20].
- *LargestCliqueSize* is the size of the size of the largest fully connected subgraph of the structure.
- *ClusterCoefficient* is the average clustering coefficient for the structure, where the cluster coefficient quantifies how interconnected the neighbourhood of each individual node is[21].

Sentences generated from this grammar will yield various quality functions, which can then be evaluated in the context of each particular structure. Each sentence can be represented as a tree, as shown in Figure 3. After Koza[10] we define a range of genetic operations on these trees, such as initialisation, crossover and mutation. By applying these operations repeatedly on a large group, or 'population' of trees, each of which has a certain evaluated fitness, we perform a genetic search.

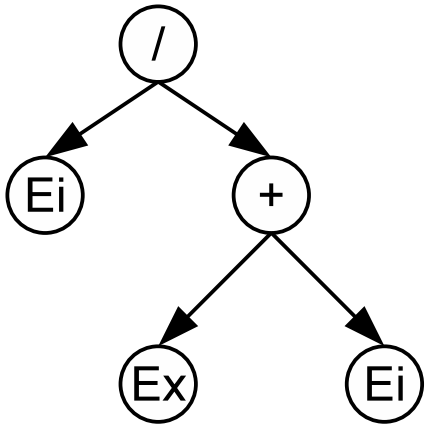

Figure 3: Example tree from the quality function (InternalEdges/(ExternalEdges+InternalEdges)).

2.2 Searching for locally maximum quality functions

With no computational constraints, our ideal approach would be to search the space of possible quality functions, while finding exhaustively, for each quality function, all possible subsets of nodes in the network for which that quality function is a local maximum. We would then evaluate these sets of nodes against the ground truth structures; if the found sets were similar to the ground truths, we would say the quality function was good. However, this exhaustive approach is not computationally feasible in practice. Firstly, for any non-trivial network, there are an intractable number of possible subsets of its nodes. Algorithms which maximise a quality function must thus search for structure in an efficient manner. Previous approaches to find sets of nodes that maximise a given quality function have typically made use of algorithmic optimisations particular to the quality function in question. For example, quality functions have been found that simple greedy methods optimise well; or, frequently, algorithmic optimisation techniques which finely exploit the specific properties of an individual quality function have been employed. However, when searching across the space of possible quality functions, we do not have the opportunity to implement optimisation algorithms tuned to each individual quality function; furthermore, general optimisation methods like genetic algorithms, or simulated annealing, while they work well on some specific quality functions, still fail to optimise well enough in *all* cases to allow

sufficient numbers of quality functions to be evaluated to perform search across the space of quality functions.

In light of these limits, we take measures to simplify our problem. Bearing in mind the success of previous greedy methods in maximising quality functions, the first assumption we make is that if a quality function is a useful description of set of structures, it should be at a local maximum for most of those structures. By a 'local maximum' we mean that removing nodes from, or adding neighbouring nodes to, a structure should not increase the value of the quality function as evaluated on that structure. For example, if the structures we are seeking to identify happen to be unique in the graph because of their clique-like nature, then we would expect that removing nodes from those structures would decrease the size of the largest clique in each structure – whereas adding neighbouring nodes would be unlikely to increase the size of the largest clique. Equally, if the distinguishing feature of the structures we are looking at happens to be that they have a certain relationship between the number of internal and external edges – as is in Equation 1 – we would expect that quality function to be at a maximum for the structures, as opposed to the same structures with 'noise', or random neighbouring nodes added to them. We thus will search for a quality function which will differentiate between the training structure, and the same structure with noise added.

Each specific training structure be somewhat unique, due to the random attributes of the structure of the network in which it is embedded. Finding a quality function which is a local maximum for a single training structure, is not, in itself, that interesting. However, if the set of training structures are generated by the same process, or have structural properties in common for other reasons, then finding a single quality function that is a local maximum for many of them tells us something about what they have in common. We are thus interested in finding a particular quality function which is locally maximum for as many of the training structures as possible.

With these ideas in mind, we cast our problem as follows:

Let $\mathcal{S}$ be a set of training structures i.e. a set of subgraphs of a graph G. Let $\mathcal{Q}$ be the space of all possible quality functions that may be generated through the specified grammar. We can say that a function $q \in \mathcal{Q}$ *identifies* the structure $S \in \mathcal{S}$ from its neighbourhood whenever q has a *local maximum* at S i.e. if $\mathcal{N}(S)$ is the *neighbourhood* of S, define

$$I(q, S, \mathcal{N}) = \begin{cases} 1 & \text{if } q(S) > q(S'), \forall S' \in \mathcal{N}(S) \\ 0 & \text{otherwise} \end{cases}$$

Then, the optimisation problem is to find a q that identifies the most training structures i.e.

$$q = \arg\max_{q' \in \mathcal{Q}} s(q', \mathcal{S}, \mathcal{N}) \tag{2}$$

where

$$s(q, \mathcal{S}, \mathcal{N}) = \sum_{S \in \mathcal{S}} I(q, S, \mathcal{N}) .$$

In order for this problem to be well-defined, it is necessary to precisely define the neighbourhood of S, for instance, by specifying a neighbourhood generation function. In the following, the neighbourhood of S is taken to be all the subgraphs of the network that can be obtained from S by removal of a node in S, or by addition of a node incident to S (i.e. a node that is connected to a some node in S, but not itself in S) .

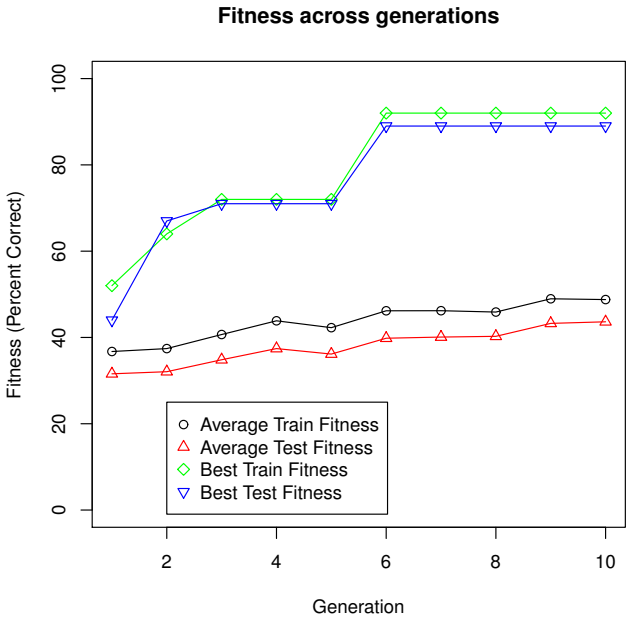

Figure 4: Fitness across generations, for average and best individuals, on train and test data.

In general, the neighbourhood of a structure can be very large, and thus an exact calculation of $s(q, \mathcal{S}, \mathcal{N})$ is time-consuming. Thus to solve (2) by genetic programming, we specify a fitness function that approximates $s(q, \mathcal{S}, \mathcal{N})$. In particular, for each training instance S, let $\mathcal{P}(S) \subseteq \mathcal{N}(S)$, such that $|\mathcal{P}(S)| = k$, be a fixed set of k perturbations of S drawn randomly from the neighbourhood. We define the fitness of q to be

$$f(q) = s(q, \mathcal{S}, \mathcal{P}) . \tag{3}$$

on the assumption that $f(q) \approx s(q, \mathcal{S}, \mathcal{N})$.

Genetic programming is thus applied to find the fittest q, as follows:

Initially, for each training structure, S we generate the set $\mathcal{P}(S)$. We also generate a random population of individual quality functions, bounded within a specific tree depth. We evaluate the fitness of an individual member of our population using (3) by converting the quality function from a tree representation, to a small computer program, which calculates the quality of each original training community, S and each perturbation $P \in \mathcal{P}(S)$. Then, quality functions of a higher fitness are selected for crossover, proportional to their fitness, using roulette wheel selection. A small proportion of individuals are randomly mutated, and this whole process is run for multiple generations. In order to optimise our implementation, we include a small percentage of elites in our selection. We also include a slight bias whereby shorter quality functions are preferred over otherwise equally fit longer functions; this inductive bias is suitable in our domain, where we are anxious to avoid overfitting our training data, and where we are most interested in the shortest quality function that adequately describes our data.

Table 1: Train and test accuracies, and found quality function with highest fitness (or a randomly chosen one, if a tie existed), for each benchmark test.

Benchmark structure	Train Accuracy	Test Accuracy	Found function
Full Clique	100	100	((largestCliqueSize)-(diameter))
Near Clique	92	89	((internalEdges)*(clusterCoefficient))
Ring	90	96	((1.0)/(averageShortestPath))
Star	79	81	((numberNodes)/(diameter))

2.3 Evaluation of method

In order to develop and validate our method, we created a set of benchmark challenges, where we embed known forms of structure into synthetic graphs. Embedding structures with known forms allows us to test the optimisation method in controlled circumstances, and evaluate whether it is finding quality functions that appear insightful, given the specific structures embedded. To create these benchmarks, we first generate a 10,000 node network using the Barabasi-Albert preferential attachment model[4], with $m = 3$.[1] Then, in each experiment, we embed 200 sample structures, half of which are assigned to a training dataset, and half to a test dataset. To make it hard to overfit the structures, each individual structure is of a size drawn uniformly at random from the range $5 - 30$. For efficiency reasons, we generate only 8 randomly perturbed versions of each original structure, and evaluate an individual quality function as correctly describing the individual embedded training structure if it evaluates as higher for that structure, than for all of its perturbed versions. We would have liked to exhaustively search the neighbourhood space; however, the evaluation of each perturbed structure by each quality function is an expensive operation – even after caching the values of the per community structural metrics – which is multiplied by the number of training communities. The size of the neighbourhood space is also very large, which makes meaningful stochastic sampling difficult. However, the small number of perturbations appears to work well in practice, and give good results on the test structures.

In our evaluation, we score the fitness of the quality function as the number of embedded training structures it correctly describes. By running this operation across all 100 training structures, if a quality function scores highly, it is likely to have found a set of graph attributes which are indeed at a local maximum for that type of structure; and thus describes some property the set of structures have in common.

As training structures, we decided to use variants on the following diverse set of network structures.

- *Cliques*, are fully connected subgraphs, wherein every node is connected to every other node. Cliques were introduced[14] as valuable measures of community in a social context (Fig 5).
- *Near-Cliques*, a relaxed version of the clique structure, where some small proportional of the edges are missing. The definition of a clique being sensitive to noise, near-cliques, or quasi-cliques, are often considered as approximations to them.
- *Rings*. A ring is defined as a set of nodes connected by a simple cycle; that is a set of vertices connected in a closed loop (Fig 6).
- *Stars* (or hubs), are defined to be high degree nodes, connected to a set of nodes which are not in turn particularly well connected to each other (Fig7).

We provide visualisations of some of these structures as embedded in a generated network. See Figures 5, 6, 7.

After evolving a quality function that is a local maximum for as many of the training communities as possible, we evaluate the generality of the found quality function by scoring it on 100 held-back embedded test structures. These are evaluated in the same way as the training structures - if the quality function can differentiate many of the diverse test communities from nearby noisy versions of themselves, then the quality function has found a structural property that well characterises the structures.

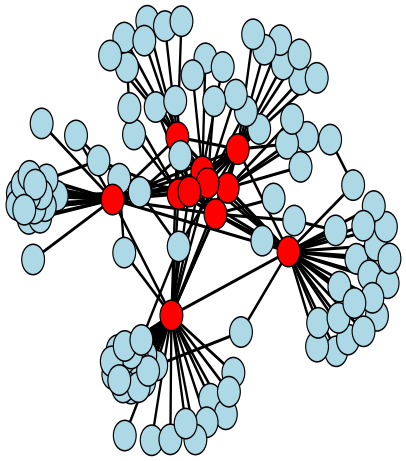

Figure 5: Visualisation of clique benchmark structure, embedded in network. Clique nodes in red.

In our experiments, we use genetic search across 10 generations, with a typical population size of 200 quality functions. These numbers are arbitrarily chosen, but seem to work well in practice; typically the fitness plateaus early in the evolutionary process – but sometimes the maximum quality function is so simple it is found instantly, and sometimes the search space is more complex, and the fitness of the best individual increases over generations. This depends

[1]Supporting materials and data are available at: http://sites.google.com/site/gpnetworkstructure/

on the difficulty of the representing the embedded structure; some structures are more difficult to represent well with the provided grammar terminals than others. One example of a run where the fitness gradually increases over successive generations is shown in Figure 4, which is taken from the search for the *Near-Clique* quality function, as shown in Table 1.

From the results shown in Table 1, the fact that the quality function found differentiates a high number of the different test structures from perturbed versions of themselves, indicates that the quality function has found a way of describing some general property of the embedded structures, as learned from the training structures. Importantly, the evolved quality functions themselves appear to yield insight on the underlying type of structure, that the training examples were generated from.

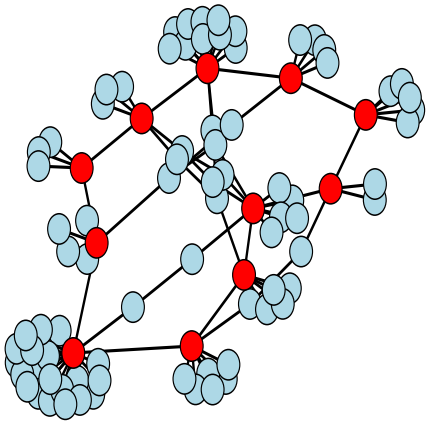

Figure 6: Visualisation of ring benchmark structure, embedded in network. Nodes on the embedded ring shown in red.

2.4 Using quality functions to differentiate between structures

Having observed that, for specific types of structure, we find quality functions that yield insight, and are near a local maximum for many of the structures, we considered whether it was possible to use this technique as a classifier to distinguish different types of embedded structure. If quality functions made up of various complex network features are a useful way of describing structures, then they should be a useful way of differentiating between different types of structure. Specifically, given two sets of different structures, a positive and a negative class, (for example, rings and cliques), and a set of complex network features for each individual structure, could we evolve a function that serves to distinguish between the two classes, by always being a higher value for the positive class? This would also be a harder test of the power of quality functions: Table1 showed that the genetic programming search process evolved quality functions that

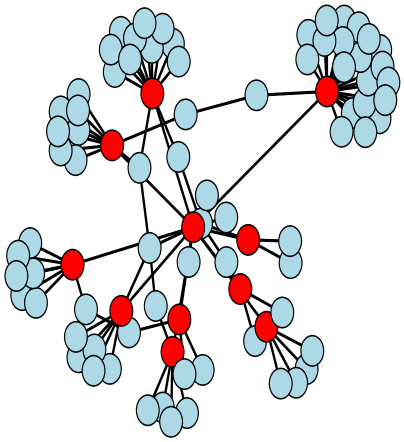

Figure 7: Visualisation of star benchmark structure, embedded in network. Nodes in the star structure in red. Center most node is the center of the star.

were close to local maximum on the unseen structures; but perhaps it was easy to distinguish any type of graph structure from perturbed versions of that structure? If the same method could also be used to evolve a quality function that distinguished between different types of benchmark structure, it would be a further validation.

In order to test this approach by evolving such a function, we again generate random BA complex networks, of size 10,000 and embed randomly generated structures, again of between 5 and 30 nodes in size, in them. Rather than embedding a single type of structure, we embed two separate types – for example, Stars and Cliques – which we then consider as examples of positive and negative class. We then randomly *pair* the positive and negative training structures. We assign the fitness of a particular quality function to be equal to the number of pairs for which the quality function evaluates the positive class higher than the negative class, and then use genetic programming to search the space of possible quality functions for the function that best differentiates between the two classes. For large numbers of training structures, the number of instances where the quality function assigns the positive class a higher value than the negative class is proportional to the area under the ROC curve (AUC), because the AUC, as used generally as a measure of classification accuracy, is equal to the probability that a randomly chosen positive example will be ranked above a randomly chosen negative example[3].

In order to evaluate the usefulness of this technique to distinguish between various types of structure, we performed several experiments where quality functions were evolved that would rate highly for the positive structure, and lowly for the negative structure. We conducted several sets of experiments using combinations of the previously mentioned

Table 2: Classification accuracy for each method in terms of test set Area Under ROC Curve. Also shown, the fittest classification function, and the train and test percentage correct for the evolved function.

Structures	Train%	Test%	AUC	J48 AUC	SVM AUC	Function
Star vs Clique	100	100	1.0	1.0	1.0	(clusterCoefficient)
Star vs Ring	98	93	0.98	1.0	0.98	(diameter)
Clique vs Distinct	72	71	0.69	0.67	0.68	((maxDegree)/(externalEdges))
Near-Clique vs Distinct	77	79	0.80	0.69	0.70	((internalEdges)-(externalEdges))
Clique vs Near-Clique	100	100	1.0	1.0	1.0	(diameter)

introduced benchmark structures. We also introduced another type of structure, to provide a more difficult test of classification – that of distinguishing a 'distinct' clique from an 'indistinct' one. In community finding, there is discussion on communities which are well defined structures (for example, cliques), but which may be well connected to the rest of the network; as contrasted with communities which are 'distinct' from the rest of the network, in that they have few external edges (for their size; models generally provide each node with some small probability of random connections) from them to the rest of the graph. The later view can be seen in the quality function of Lancichinetti et al. (Equation1), where there is a term penalising external edges; however other definitions of community - notably other overlapping community finding algorithms - might not penalise external edges as strongly.

In all experiments, the sets of structure pairs were divided into training and test sets, and after evolving the quality function using the training set, accuracy was evaluated on the test set. To give some indication of how this method was performing as a classifier, we also compared against several standard classification algorithms running on the same set of structure feature vectors. We used the both the decision tree implementation, J48[17] and the default SVM[16] implementation, from the standard WEKA[9] machine learning package. We present the outcomes of the various classification techniques in Table 2. In all cases, we use AUC – area under the ROC curve – on the 50% held-back test dataset as our metric of comparison. We also show our train and test 'percentage correct measures', which are the percentage of instances wherein the quality function evaluated the target class strictly higher. We have not performed any tuning the of the SVM kernel, or decision tree parameters, for the problem in question; we present these results merely to show they appear to be competitive with more general classification approaches.

From these results, this means of using genetic programming to search the quality function space produces results comparable, and sometimes in excess of, the AUC obtained using the default SVM and decision tree methods. This is encouraging, and appears to validate the previously known idea that community quality functions are useful means of describing some of the specific types of network structures we are interested in. It is also of note that as the classification tasks got harder, more complex functions - combinations of the simpler primitives - begin to emerge, despite there being a fitness bias for shorter functions. For example, the function evolved to differentiate *near-cliques* with many connections to the rest of the network, vs near-cliques with fewer connections per node, to the rest of the work $((internalEdges) - (externalEdges))$ is quite useful; externalEdges alone will not suffice to tell them apart, as larger distinct near-cliques, with a smaller fraction of external edges, may still have more external edges than small 'indistinct' cliques. This is not, in itself, remarkable, but shows promise for when similar techniques are applied to applications where the functions describing the classes to be distinguished are not *a priori* known. This means of investigating and distinguishing between structures in a supervised manner has the advantage that the evolved function may be easily examined and interpreted, possibly to gain insight about the problem space; where as the output of other methods is more opaque.

2.5 Biological dataset

Having obtained results indicating search for a local quality functions worked well on synthetic embedded network structures, we examined a biological dataset. We used the interaction data found in the Combined-AP/MS network[6] as our network, and used the protein complexes from the CYC dataset of known complexes as training structures. These complexes have previously been found to be predictable by finding 'communities' in the underlying protein-protein interaction network. We selected half of these complexes, and evolved a quality function that was a local maximum for as many of them as possible. No single quality function was found that was a local maximum for the whole set of complexes; in fact the best quality function was only an approximate local maximum for 37 out of 63 test set structures, and 42 out of 63 training set. It is not surprising that it is hard to find a single quality function that is a perfect local maximum across this dataset; real world biological data is fundamentally noisy; previous benchmarking in work by Lee et al.[12] on this dataset has found only mid range scores (expressed as *normalised mutual information*) across a range of respected community finding algorithms.

The quality function that our search found to best maximise the local maximum criterion was a function of the largest clique size and the cluster-coefficient, specifically: *(largestCliqueSize * Cluster Coefficient)*. While we cannot know that this particular function is objectively good, it appears intuitively similar to the quality function used by the method of finding communities by Greedy Clique Expansion[12], which first finds the largest maximal clique in a network, and then maximises a local density based objective function in the surrounding graph - a method that was found to be more effective than most on this dataset. These results indicate that using search to find a quality function produces sensible results, not just on synthetic benchmark graphs, but also on noisy real world biologicl network data.

This work is supported by Science Foundation Ireland under grant 08/SRC/I1407: Clique: Graph and Network Analysis Cluster.

CYC complexes available: http://wodaklab.org/cyc2008/

3. CONCLUSION

It has previously been established that quality functions are a good way of describing high level network structure of interest. Given this, our work indicates that, for a set of specific structures, we can algorithmically search for a quality function that describes what they have in common. We believe the idea of learning about training structures, by searching for quality functions of their network features, is promising. The primary advantage of supervised search based methods such as this is that they allow us to investigate commonality in identified network structures, without having to rely on a pre-conceived notion of what exactly defines the structure we are interested in. So long as we have a vocabulary with which to describe network structures - a set of complex network measures - this approach allows us to go from only having examples of interesting network data, to learning something about the combinations of network features that define them. Our results on using genetic programming to search for combinations of network features that generate rules to distinguish different structures also shows promise. On our test benchmarks, this general method distinguishes different types of network structure, and gets classification results competitive to existing classification methods applied on the same feature set.

4. FURTHER WORK

It would be possible to generalise this method by adding further complex network measures. Of specific interest would be the incorporation of motif metrics - such as counts of particular graph motifs - and a more general grammar vocabulary to search for combinations of motifs that describe training structures. While Genetic Programming is a powerful representation for this problem, we have not examined in detail the performance of the genetic search, as contrasted with other approaches; further work should investigate this. We believe this type of supervised structure characterisation is an interesting direction in complex systems research; a further step would be to attempt to apply this, or some further development of the ideas, on a real problem setting, where we have many more different types of defined 'ground truth' network structures than we can investigate manually, and a domain need to distinguish between them. Such domains might include finding signatures of fraud in transactional networks, or further classification of structures in more complex biological settings.

5. REFERENCES

[1] B. Adamcsek, G. Palla, I. Farkas, I. Derényi, and T. Vicsek. CFinder: locating cliques and overlapping modules in biological networks. *Bioinformatics*, 22(8):1021, 2006.

[2] M. Altaf-Ul-Amin, Y. Shinbo, K. Mihara, K. Kurokawa, and S. Kanaya. Development and implementation of an algorithm for detection of protein complexes in large interaction networks. *BMC bioinformatics*, 7(1):207, 2006.

[3] D. Bamber. The area above the ordinal dominance graph and the area below the receiver operating characteristic graph. *Journal of mathematical psychology*, 12(4):387–415, 1975.

[4] A. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):509, 1999.

[5] A. Clauset. Finding local community structure in networks. *Physical Review E*, 72(2):26132, 2005.

[6] S. Collins, P. Kemmeren, X. Zhao, J. Greenblatt, F. Spencer, F. Holstege, J. Weissman, and N. Krogan. Toward a comprehensive atlas of the physical interactome of Saccharomyces cerevisiae. *Molecular & Cellular Proteomics*, 6(3):439, 2007.

[7] S. Fortunato. Community detection in graphs. *Physics Reports*, 2009.

[8] A. Gog, D. Dumitrescu, and B. Hirsbrunner. Community detection in complex networks using collaborative evolutionary algorithms. *Advances in Artificial Life*, pages 886–894, 2007.

[9] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. Witten. The WEKA data mining software: An update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.

[10] J. Koza. *Genetic programming: on the programming of computers by means of natural selection*. The MIT press, 1992.

[11] A. Lancichinetti, S. Fortunato, and J. Kertész. Detecting the overlapping and hierarchical community structure in complex networks. *New Journal of Physics*, 11:033015, 2009.

[12] C. Lee, F. Reid, A. McDaid, and N. Hurley. Detecting highly overlapping community structure by greedy clique expansion. *KDD SNA 2010*, 2010.

[13] X. Liu, D. Li, S. Wang, and Z. Tao. Effective algorithm for detecting community structure in complex networks based on GA and Clustering. *Computational Science–ICCS 2007*, pages 657–664, 2007.

[14] R. Luce and A. Perry. A method of matrix analysis of group structure. *Psychometrika*, 14(2):95–116, 1949.

[15] F. Luo, J. Wang, and E. Promislow. Exploring local community structures in large networks. *Web Intelligence and Agent Systems*, 6(4):387–400, 2008.

[16] J. Platt. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods*, pages 185–208. MIT press, 1999.

[17] J. Quinlan. *C4. 5: programs for machine learning*. Morgan Kaufmann, 1993.

[18] R. Sharan, T. Ideker, B. Kelley, R. Shamir, and R. Karp. Identification of protein complexes by comparative analysis of yeast and bacterial protein interaction data. *Journal of Computational Biology*, 12(6):835–846, 2005.

[19] H. Simon. The architecture of complexity. *Proceedings of the American Philosophical Society*, pages 467–482, 1962.

[20] J. Travers and S. Milgram. An experimental study of the small world problem. *Sociometry*, 32(4):425–443, 1969.

[21] D. Watts and S. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, 1998.

[22] G. Wilson and W. Banzhaf. Discovery of email communication networks from the Enron corpus with a genetic algorithm using social network analysis. In *Evolutionary Computation, 2009. CEC'09. IEEE Congress on*, pages 3256–3263. IEEE, 2009.