Project Number 001907

DELIS

Dynamically Evolving, Large-scale Information Systems

Integrated Project

Member of the FET Proactive Initiative **Complex Systems**

Deliverable D5.4.1

Application of Motif Analysis to Artificial Evolving Networks

Start date of the project:	January 2004
Duration:	48 months
Project Coordinator:	Prof. Dr. math. Friedhelm Meyer auf der Heide Heinz Nixdorf Institute, University of Paderborn, Germany
Due date of deliverable:	December 2005
Actual submission date:	January 2006
Dissemination level:	PU – public
Work Package 5.4:	Multi-Scale Topology Evolution in Natural and Artificial Networks
Participants:	Universtitat Pompeu Fabra (UPF), Barcelona, Spain Universita di Bologna (UniBO), Italy Telenor R&D (Telenor), Norway
Authors of deliverable:	Sergi Valverde (svalverde@imim.es) Ricard V. Solé (ricard.sole@upf.edu) David Hales (hales@cs.unibo.it) Ozalp Babaoglu (babaoglu@cs.unibo.it) Stefano Arteconi (arteconi@cs.unibo.it) Geoffrey Canright (geoffrey.canright@telenor.com)

Abstract

This report comprises the complete D5.4.1 deliverable as specified for workpackage WP5.4 in Subproject SP5 of the DELIS (Dynamically Evolving Large-scale Information Systems) Integrated Project.

The essential goal of the DELIS project is to understand, predict, engineer and control large evolving information systems. In this workpackage we wish to tie together, hitherto, independent research lines on evolving network "form" and "function". Both in naturally found networks (e.g., biological and software networks) and artificial networks (e.g., dynamic peer-to-peer networks) there appear to be various levels of selection and evolution and interesting relationships between form and function. It is becoming increasingly clear that form does not follow directly from function (or vice versa) in many observed evolving networks. What then is the relationship and how can it be detected and characterised?

We summarise two main lines of work, the first section covers detailed analysis of motifs and structures in software graphs based on class relationships. Here predictive analytical models have been extracted from empirical analysis of code. In addition to giving deep insights into the nature of software evolution at the code-programmer level, these models offer the possibility of predicting or sensing automatically if a particular code-base could need re-factoring without knowing the specifics, semantics, or function of the code itself - hence offering the possibility of generic tools.

The second part details some initial motif analysis of peer-to-peer protocols based on evolutionary algorithms that support cooperation between selfish adaptive nodes in a network. We were surprised to find the motif profiles match closely protein structure networks. Additionally we gained interesting and potentially very useful insights on the relationship between form and function in such networks. Similarly, we found that it may be possible, automatically, to detect if a peer-to-peer network is malfunctioning or under malicious attack without knowing the specifics of the application level function. Again, these results point to the possibility of highly generic and useful tools[1].

[1] Most papers produced within DELIS are available from the DELIS website as DELIS Technical Reports. Where this is the case references are appended with the DELIS Tech Report number in square brackets. This indicates the paper was produced within the DELIS project, not some other project.

Contents

1 Motif Evolution in Software Development

1.1 Introduction

How do Scale-Free (SF) nets originate? There are a number of well-identified processes leading to SF structure. Most of them rely in a growing network displaying some rules of preferential attachment of new nodes. However, it has been suggested that a sparse SF network can actually result from an underlying optimization process in which efficient communication at low cost is involved [19]. But the most interesting implications from SF architecture are related to their high robustness against random node failure, together with a high level of fragility when hubs fail. In other words, information transfer keeps working in an efficient way when a randomly chosen node fails but typically degrades when a highly connected component fails. Such observation has been shown to have immediate implications for reliable network architecture. Since systems sensitivity to component failure is a fundamental problem in any area of engineering, it is important to recognize how network topology will influence systems performance.

An example of such type of distribution is found in the large-scale architecture of any C++ software system (see fig.1 and fig. 2A), where the hubs (i.e., classes participating in many different relationships) can be appreciated, together with a large number of elements having a single connection [19]. Such scale-free graphs are the result of multiplicative processes. An example is the so called preferential attachment rule [1]. But heterogeneous architectures can also be a consequence of a pressure towards achieving good communication at a low cost. What is more important in our context: rules of tinkering, such as duplication of existing nodes plus rewiring, are able to generate such heterogeneous graphs. These rules are known to be operating in biological evolution, and the emergence of the protein interaction network seems to provide a clever example ([15]; [13]). It is less clear that such type of evolutionary rules apply in software development.

A careful analysis of network motifs (see fig. 1C) in real software maps [17] has shown that this is actually the case. Although engineers are building a system with a predefined purpose, they make extensive use of copy-and-paste. The modular architecture of many parts of a large program, and in particular the organization into classes makes easy to reuse pre-existing pieces that already have a desired set of properties, followed by a convenient set of modifications. It has been shown that, looking at both large-scale and small-scale features (such as network motifs) it is not difficult to explain the most relevant features of software graphs by means of an extremely simple model of duplication and rewiring process. Actually, available data from software development seems to be consistent with a simple, multiplicative process of network evolution. Such class of growth has been dubbed growing network with copying (GNC). In this framework ([9]) the network grows through a blind process of duplication and rewiring.

1.2 Software Evolution

In the GNC model, the network grows by introducing a single node at a time. This new node links to m randomly selected target node(s) with probability p as well to all ancestor nodes of each target, with probability q. The discrete dynamics follows a rate equation:

$$L(N+1) = L(N) + \frac{m}{N}\left\langle \sum_{\mu} (p + qj_{\mu}) \right\rangle \tag{1}$$

where L and N are the number of links and nodes, respectively. The second term in the right-hand side describes the copying process, where the average number of links added is given by$p + qj_{\mu}$. The μ index refers to the node μ, to be selected uniformly from among the N elements. Assuming a continuum approximation, the number of links is driven by the following differential equation:

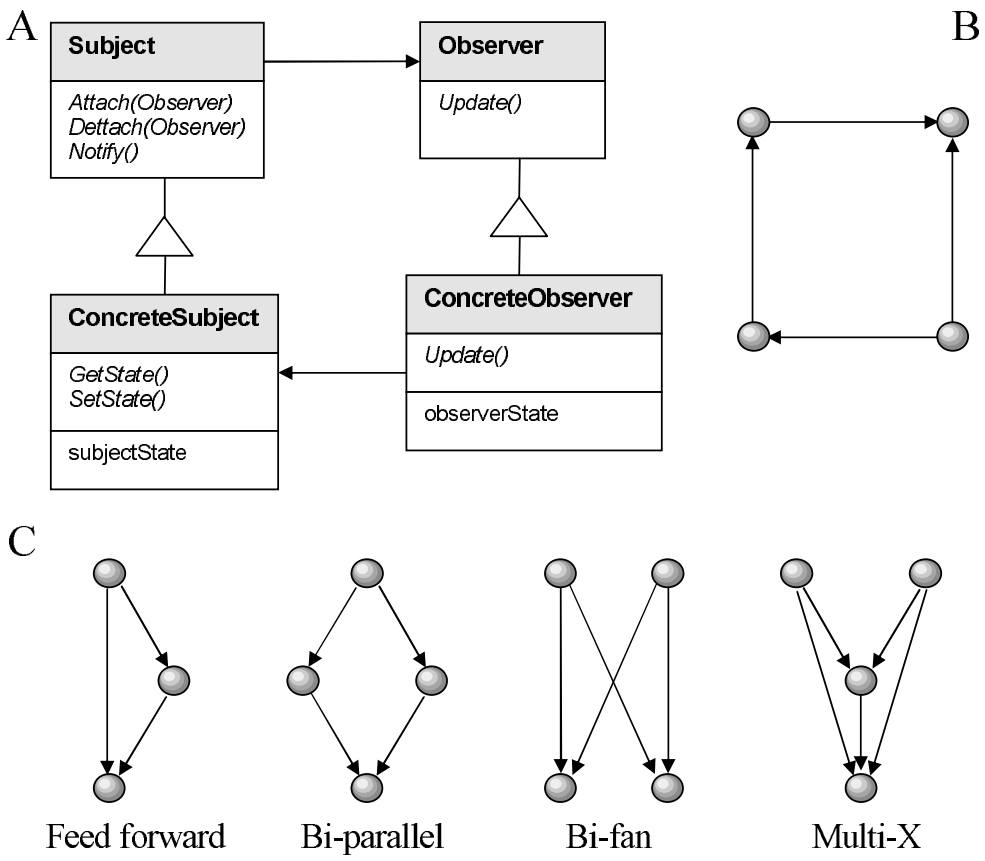

Figure 1: (A) An example of design pattern described in standard UML notation is compared with (B) its corresponding software network. The observer pattern is commonly used to maintain consistency between related objects (i.e., Subject and Observer) and it is fully described in [3]. The bare-bones structure of this pattern is described by the little square subgraph in the right. Graph nodes and directed links represent classes and collaborations, respectively. (C) Some of the most common motifs found in software networks. We can use the degree distribution $P(k)$ to predict the abundances of software motifs.

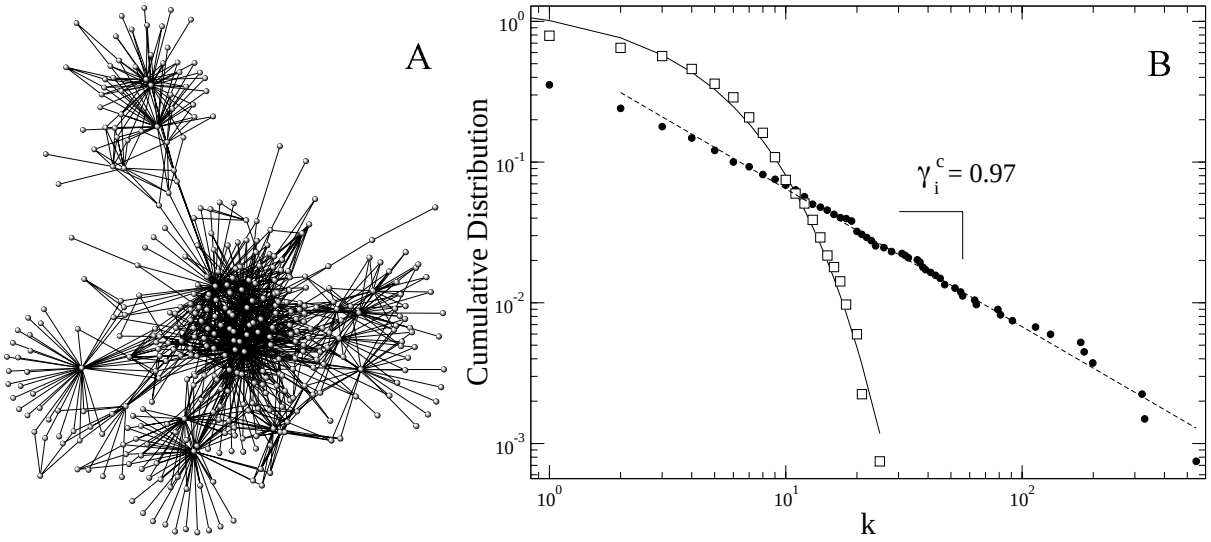

Figure 2: (A) Largest connected component of the XFree86 project at 15/05/1994 (with $N = 393$) displays scale-free behavior. In (B), the cumulative distributions $P_{i>}(k)$ and $P_{o>}(k)$ are shown for a more recent version of XFree86 with $N = 1299$ (not shown here). The power-law fit of the in-degree distribution (black dots) yields $P_i(k) \sim k^{-\gamma_i^c - 1}$ with $\gamma_i^c = 0.97 \pm 0.01$ while the out-degree distribution is exponential (white squares).

$$\frac{dL}{dN} = mp + mq\frac{L}{N} \tag{2}$$

The asymptotic growth of the average total number of links depends on the extent of copying defined by the product mq. In particular, logarithmic growth is recovered when mq=1 and $L(N) = mpN\ \log N$. This corresponds to a marginal situation separating a domain of linear growth (mq<1) to a domain of exponential growth (2>mq>1). Interestingly, for mq=1 the GNC model predicts a power-law in-degree distribution $P_i(k) \approx k^{-\gamma_i}$ with exponent γ_i= 2 and an exponential out-degree distribution $P_o(k)$, independently of copying parameters m, p and q.When looking at the time dynamics of software development, we found that they exhibit the patterns predicted by GNC scenarios, close to the *mq=1* regime. They include the sparseness, the asymmetries found in the in- and out-degree distributions (see fig. 2B), the small worldness and the time dependent logarithmic growth (see fig. 3A). Indeed, the sparseness seen in software maps is likely to result from a compromise between having enough dependencies to provide diversity and complexity (which require more links) and evolvability and flexibility (requiring less connections). Here we have uneven, but detailed information of the process of software building. In this context, different software projects developments display specific patterns of growth.

Specifically, the number of nodes N grows with time following a case-dependent functional form $N = \Phi(t)$. Using $dL/dt = (dL/dN)(d\Phi/dt)$ we have:

$$\frac{dL}{dN} = \left[mp + mq\frac{L}{\Phi(t)}\right]\dot{\Phi} - 1 \tag{3}$$

with a general solution

$$L(t) = e^{mq\int(\Phi\,\dot{\Phi})^{-1}dt}\left[mp\int e^{-mq\int(\Phi\,\dot{\Phi})^{-1}dt}\,\dot{\Phi} - 1dt + \Gamma\right] \tag{4}$$

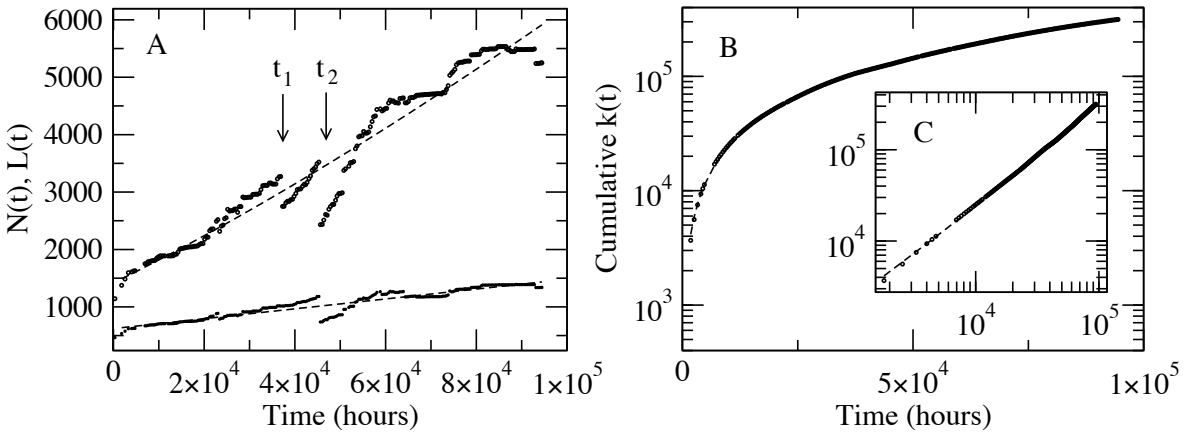

Figure 3: (A) The top curve shows the comparison between the time evolution of $L(t)$ in XFree86 between 16/05/1994 and 01/06/2005 (points) and the GNC prediction (dashed line) assuming logarithmic growth (mq=1). We also assume that system size $N(t) = N_0 + at$ evolves linearly in time (see bottom curve). We observe an anomalous growth pattern followed by a discontinuity (here indicated as t_1 and t_2). Notice how t_2 signals a discontinuity both in $L(t)$ and $N(t)$, while discontinuity t_1 only takes place in $L(t)$. (B) Comparison between time evolution of the cumulative average degree in XFree86 during the same time period as in (A) and the analytic prediction. (C) The inset shows the same data as in (B) but in a double logarithmic plot. The fitting parameters are: $N_0 = 622.17 \pm 10.92$, $a = 0.0086 \pm 0.0002$, $L_0 = 1419.8 \pm 4.1$, and $mp = 2.20 \pm 0.01$. Time is measured in hours.

where Γ is a constant. Using a linear law growth (which is not uncommon in software development), i.e. $N(t) = N_0 + at$, and assuming mq=1, we have:

$$L(t) = (N_0 + at)\left[mp \log\left(\frac{N_0 + at}{N_0}\right) + \frac{L_0}{N_0}\right] \tag{5}$$

Our analysis of available data sets of software development (see fig. 3) support this scenario ([18]). This agreement suggests that, beyond the specific details of the development process and (what is more unexpected) the specific performed function, topological patterns emerge from the constraints imposed by the rules of software growth. In this context, the studies presented here indicate that function is adapted to the emerging architecture through the process, instead of being responsible for the final pattern.

In addition, the previous model provides further validation of our prediction for the number of appearances of a particular motif in any software network ([17]). Given a particular motif with n nodes, g edges and maximum in-degree s, the following expression gives the average number $< G >$ of motif instances found in a software network with N nodes:

$$\langle G \rangle \approx N^{n-g+s-\gamma_i+1} \tag{6}$$

Using a set of 83 software systems we have estimated a scaling exponent $\gamma_i \approx 2.09 \pm 0.06$ for the in-degree distribution. Notice how this estimation is in remarkable agreement with the prediction of the GNC model.

1.3 Summary

Previous studies indicate that network motifs act like fingerprints of development processes. In this context, our recent analysis of software motifs has shown that both motif abundances and heterogeneous software topology are the consequence of simple tinkering rules of software development. Then, we do not require the assumption of conscious and optimal design in order to explain the scale-free behavior of software. Inspired by the above findings, we have successfully modeled the evolution of software architecture. The scale-free invariant observed in software systems enabled us to predict the growth of software systems, where the number of links is constrained to follow a logarithmic trend. This simple model also explains the typical in-degree and out-degree asymmetry observed in real software architectures. In addition, deviations from the logarithmic trend might anticipate costly design changes (i.e., refactorings).

2 Motifs and Anti-motifs in Evolving Peer-to-Peer Networks

2.1 Introduction

From an engineering point of view, our aim is to produce distributed systems with many of the desirable properties of living systems (as discussed in detail in D5.1.1). These properties include, robustness yet adaptability, scalability, self-healing and self-management. We have developed a number of distributed Peer-to-Peer (P2P) protocols, based on a a socially inspired evolutionary algorithm, that display, at least some, of these characteristics.

In a previous deliverable (D5.2.1) we discussed how we translated the evolutionary algorithm, based on social tags [14], into a copy and re-wire P2P protocol that promoted cooperation between connected nodes even when they had incentives to behave selfishly. We also discussed how the protocol structures the population into competing 'tribes' that, through a group-like selection process, lead to socially beneficial behaviour even when the individual nodes behave in an essentially selfish way - following a local greedy optimisation algorithm.

We tested the protocol by having nodes play the Prisoner's Dilemma game - a canonical game for exploring situations in which collective interests and individual interests diverge. We called this protocol SLAC (Selfish Link Adaptation for Cooperation) and applied it to a simulated file-sharing scenario demonstrating it had the ability to control the outbreak of selfish behaviour by nodes (downloading without uploading - so called "leeching") [6].

Intriguingly, we noticed that the copy and re-wire rules used in the protocol were very similar to the kinds copy and re-wire algorithms found to produce networks with similar structural properties to those found in software graphs [18]. Hence human constructed software graphs appear to follow a copy and re-wire process at some level (as discussed in the previous sections).

Although the SLAC algorithm performs well for certain task domains it results in networks with many disconnected components. Certain kinds of task require fully connected networks, for example a broadcast task that requires a single node to send a message to all nodes in the network, collective spam filtering [8] or improving distributed hash table (DHT) performance [10].

In order to address this limitation of SLAC we modified the protocol such that the copying and re-wiring of links follows a probabilistic rule. In this way there is a probability that old links are retained when nodes re-wire (move) within the network. We called this new protocol SLACER (Selfish Link Adaptation Excluding Rewiring) [5].

Figure 4 shows the pseudocode outline of the active thread of the SLACER protocol[2]. Each node periodically selects another node from the network at random and copies it's links and strategy if it has a utility that is equal to or higher than it's own. A strategy is some application level behaviour

[2]Since we have covered this algorithm in detail in a previous deliverable (D5.2.1) and elsewhere [6] we do not go into detail here since we wish to focus on the motif analysis of the produced networks.

Active thread:	Passive thread:

i ← this node
do forever:
Engage in application task
update i.Utility
Periodically (compare utility):
  j ← GetRandomNode()
  j.GetState(i)
  if i.Utility ≤ j.Utility
  CopyStatePartial(j)
  Mutate(i)
  Utility ← 0.0 (reset utility)
	j ← this node
	do forever:
	sleep until a request received
	GetState(i) – send j state to node i:
	  Send j.Utility to i
	  Send j.Links to i
	  Send j.Strategy to i

Function CopyStatePartial(j):	**Function Mutate(i):**

i.Strategy ← j.Strategy
drop each link from i with prob. W
for each link in j.Links:
  i.addLink(link)
	with prob. M mutate i.Strategy
	with prob. MR mutate i.Links:
	  drop each link with prob. W
	  i.addLink(SelectRandomNode())

Figure 4: The SLACER protocol pseudocode. Note that when W=1 SLACER collapses into the SLAC protocol. For an overview of the protocol see the text but for more detail see [5, 4].

or algorithm and utility is some application level indication of nodes individual performance. For example, in a file-sharing scenario the strategy might be the amount of bandwidth to devote to servicing other nodes requests and the utility might be the download rate for the individual node.

With low probability, after copying another node, the copied links and strategy are mutated - changed in some randomised way. Mutation on the links involves dropping each one with high utility (W) and then adding a link to a randomly chosen node form the network. Again mutation operations on the strategy would be application specific, for example, for a file-sharing application the amount of bandwidth used to service other nodes requests would be changed randomly.

When the link drop probability W=1 (see figure 4) then SLACER collapses to the previous SLAC protocol producing highly cooperative yet disconnected networks. However, when W is slightly reduced, SLACER produces networks in which almost all nodes are members of a giant connected component in which all neighbours are cooperative. Further, we found that almost all pairs of nodes can find cooporative routes linking them (i.e. routes that pass through only cooperative nodes) - an important property for addressing the limitations of the SLAC.

Interestingly, SLACER produced networks that were small-world like, with a low average path length between nodes and a high clustering coefficient. It also retained the desirable file-like properties of scalability, adaptability and robustness - meaning that if nodes are removed or links broken the network quickly readjusts back into a cooperative state.

Although we have analysed some of the global properties of SLACER networks and how they evolve (such as average path length and clustering coefficient) we have not yet considered motif level analysis in which the frequency of subgraph structures within the networks is considered. This kind of analysis is of value since it allows us to compare the structure of the produced networks with other networks as well as characterise the evolution over time of the of the SLACER networks themselves.

In the next section we briefly introduce the notion of network motifs and in the following sections we show the results obtained from the analysis of some SLAC and SLACER P2P networks. We also

compare those results with previous analysis of naturally occurring and engineered networks. *We were rather intrigued by our results which showed a strong resemblance between the P2P networks and certain naturally occurring protein structures.*

2.2 Mofits and Anti-motifs

Networks are often characterised using average global measures, such as average path length and clustering coefficient. Although valuable such measures rarely give a picture of the detailed structure of the networks. This means that networks with different topologies can have identical global average measurements. Hence, in order to further understand and classify natural and artificial networks new methods have been proposed.

Recently, researchers working with complex networks (both natural and artificial) have begun to analyse and characterise them using more sophisticated topological techniques and one of these approaches is so called "motif analysis" [11].

By breaking the network down into all possible n node subgraph patterns and counting them it is possible to compare those counts against randomly generated networks with the same characteristics (number of nodes and in / out degree links). Then, where certain n node subgraph patterns are significally more prevelant than in the random case, these are considered mofits of the network. Additionally, although less discussed in the literature, n node subgraph patterns that are under-represented in the network have been termed anti-motifs [12] and are of equal value in characterising network structure.

Obviously, for large subgraph sizes the number of possible motifs becomes large but for smaller sizes (3 and 4 nodes) it is possible computationally to search, even large, networks for all occurrences efficiently. Figure 5 shows all possible three node subgraphs for directed graphs. Note, that for non-directed three node subgraphs there would be only two possible subgraphs (shown as id78 and id238 in figure 5).

2.3 Subgraph Ratio Profiles

The P2P networks produced by SLAC and SLACER are undirected in the sense that all links are bidirectional. So for the purposes of analysis we search for all undirected four node subgraphs (tetrads). Figure 6 shows the six possible undirected tetrads.

In order to analyse the P2P networks we used a subgraph ratio profile (SRP) method [12]. This approach is particularly useful for analysis of the P2P networks since traditional motif analysis methods using z-scores are not network size invariant for non-directed tetrads and this makes comparison with networks of different sizes difficult.

For a given network N the SRP is a normalised vector of Δ_i values:

$$SRP_i = \frac{\Delta_i}{\sqrt{\sum_i \Delta_i^2}} \tag{7}$$

The vector elements, one for each of the six tetrads, are calculated based on the abundance of each tetrad i relative to randomly generated networks, To avoid large values as an artefact of very small occurrences of tetrads in both the real and random networks the value $\varepsilon = 4$ is added to the the denominator.:

$$\Delta_i = \frac{Nreal_i - < Nrand_i >}{Nreal_i + < Nrand_i > + \varepsilon} \tag{8}$$

A given SRP can be graphed producing a curve which characterises the tetrad motifs and anti-motifs visually. Figure 7 (taken from [12]) shows the SRP curves of a number of natural and engineered networks grouped into similar so-called "superfamlies".

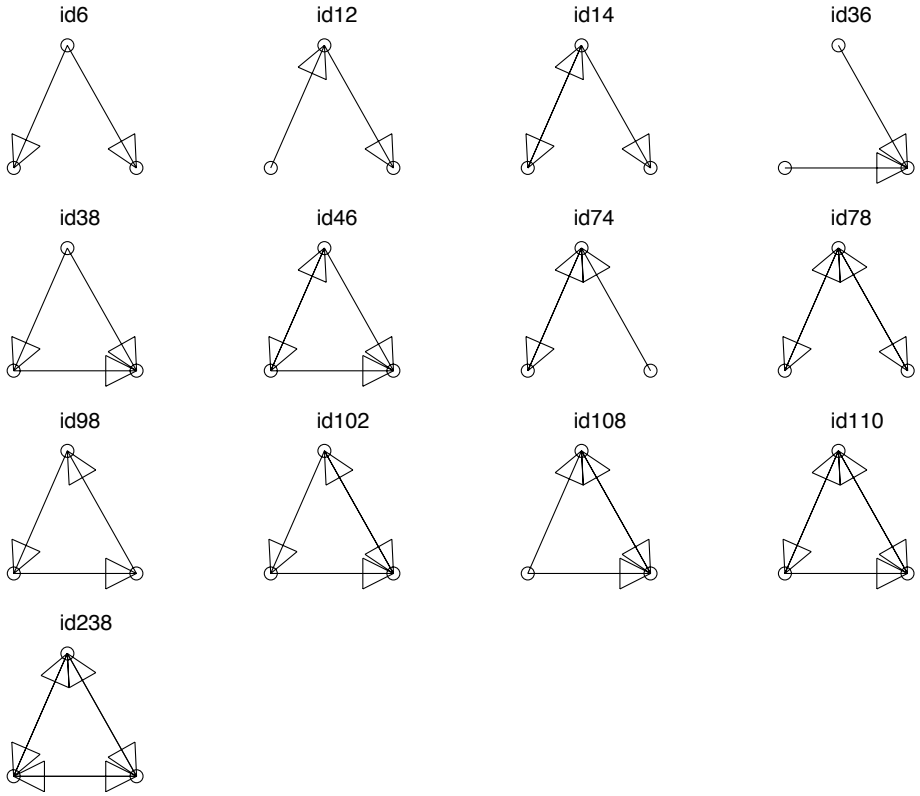

Figure 5: All thirteen possible three node directed subgraphs (taken from [20]). The id is obtained by representing the subgraph as an adjacency matrix structured as a binary integer extracted by concatenation of the rows of the matrix. In this way any size of subgraph can be given a unique id which specifies the structure completely.

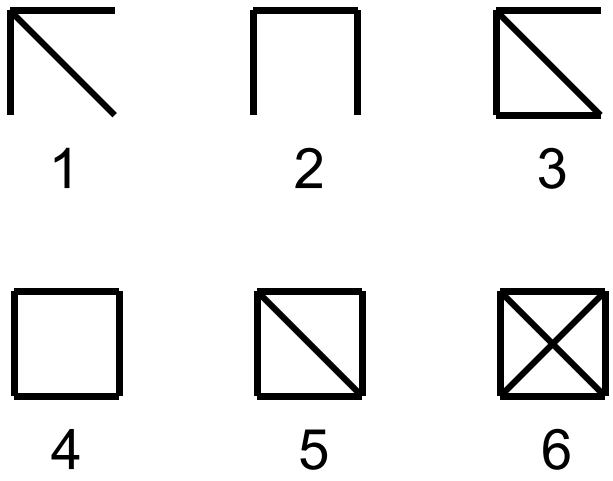

Figure 6: All six possible four node undirected subgraphs (tetrads). Nodes are not shown but should be assumed at the end of each line. The adjacency matrix derived id's are not shown but the tetrads are ordered by ascending value of them.

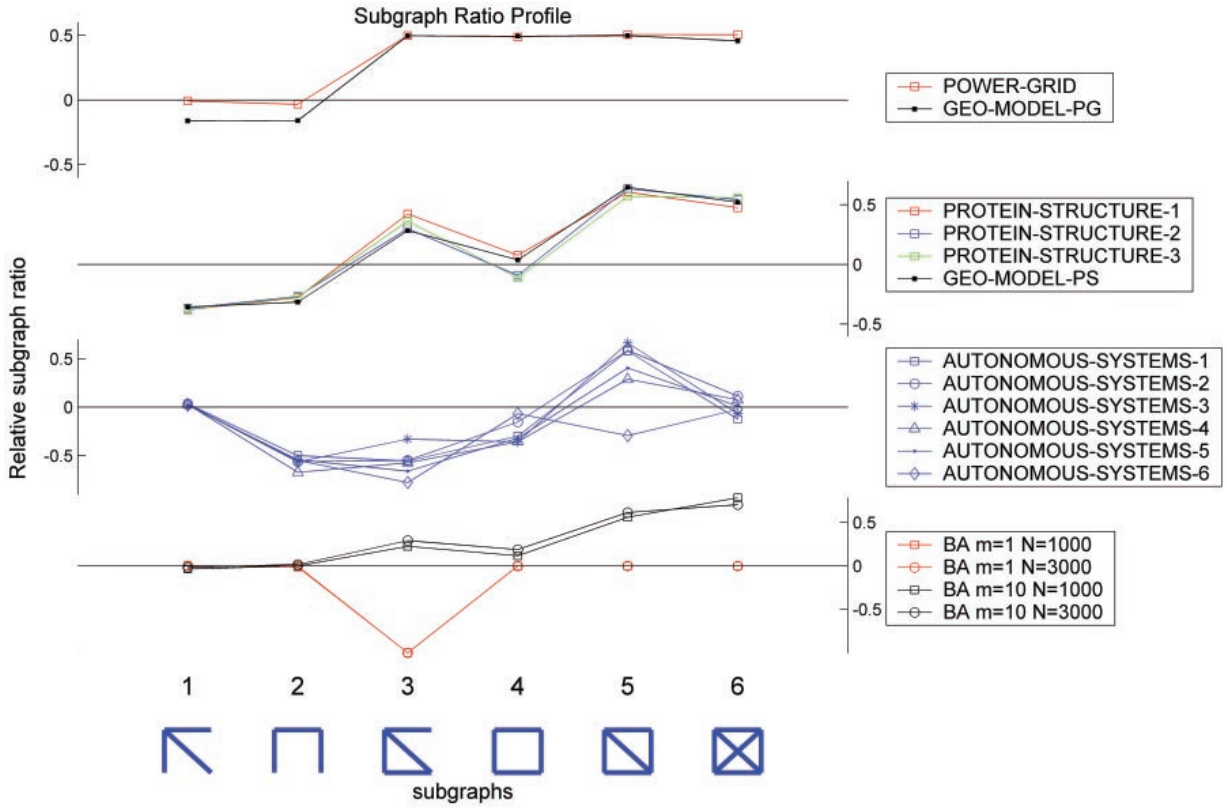

Figure 7: The subgraph ratio profiles (SRP) for a number of different undirected networks both natural and engineered (this figure taken from [12]). Here they are shown grouped into so-called "superfamilies" based on SRP similarity. The networks are as follows: (i) The electrical power grid of the western United States (POWERGRID N = 4941, E = 6594) and a geometric model with similar clustering coefficient (GEO-MODELPG N = 5000, E = 7499). (ii) Networks of secondary-structure elements adjacency for several large proteins [structure based on the PDB database (www. rcsb.org/pdb/); the proteins (and their PDB ID) were 1A4J, an immunoglobulin (PROTEINSTRUCTURE-1 N = 95, E = 213); 1EAW, a serine protease inhibitor (PROTEIN-STRUCTURE2 N = 53, E = 123); and 1AOR, an oxidoreductase (PROTEINSTRUCTURE-3 N = 99, E = 212)] and a geometric model with similar clustering coefficient (GEO-MODEL-PS N = 53, E = 136). (iii) The Internet at the autonomous system level (www.cosin.org) (AUTONOMOUS-SYSTEMS 1 to 6; N = 3015, 3522, 4517, 5357, 7956, 10515; E = 5156, 6324, 8376, 10328, 15943, 21455). (iv) Networks grown according to the preferential attachment BA model [1] with m = 1 or m = 10 edges per new node (BA m = 1, 10; N = 1000, 3000, 1000, 3000; E = 1000, 3000, 9901, 29901). Note for further details on data sources see the original article from which this figure is taken [12]

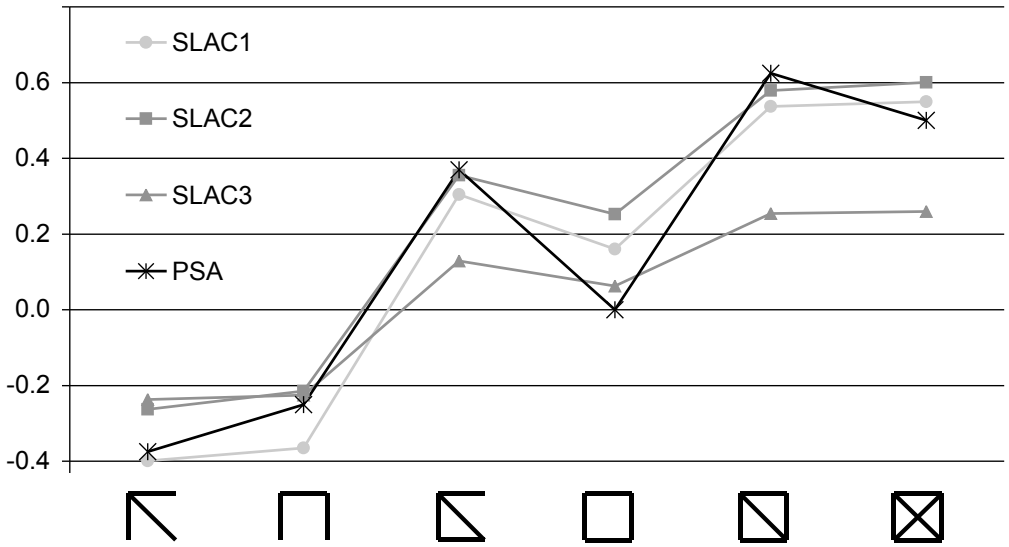

Figure 8: The subgraph ratio profiles (SRP) of SLAC P2P networks at different stages in their evolution to cooperation - network size $N = 500$, edges $E \approx 3500$. In each case three time ordered network snapshots are shown. The networks were initialised with random topologies and all nodes as non-cooperative. The first snapshot is taken immediately before any cooperation has been attained, the second during the rapid outbreak of cooperation and the final snapshot is taken after stable cooperation is attained. *Note the similarity to the protein-structure superfamily shown in figure 7 - PSA shows the average of the protein structure results, given for comparison.*

2.4 Analysing SLAC and SLACER P2P Networks

Figures 8 and 9 show SRP's for both SLAC and SLACER P2P networks at different stages in their evolution [3]. In each case three time ordered network snapshots are shown. The application task that generated the node level utility required for the protocols was to periodically play the Prisoner's Dilemma (PD) game with a randomly chosen neighbour and accumulate average payoff. The period of game playing was one order of magnitude higher than the period used by the SLACER protocol as shown in figure 4. This means that on average ten games of PD would be played between SLACER invocations.

As previously discussed, the difference between SLAC and SLACER is that the link drop probability (W) is set to a value lower than 1 (in this case we used W = 0.9 for SLACER). All nodes in the network stored a pure PD strategy (either to cooperate or defect) - representing the application behaviour. The mutation probabilities were set to M = 0.001 and MR = 0.01. We fixed the maximum in / out degree of each node to 20 links. All links are undirected and hence symmetric. The protocol perseveres the symmetry of links between nodes at all times.

The networks were initialised with random topologies and all nodes as non-cooperative (defect strategy). The first snapshot is taken immediately before any cooperation has been attained, the second during the rapid outbreak of cooperation and the final snapshot is taken after stable cooperation is attained. We do not show the initial network topology since this would be a flat line along

[3] We used the freely available "mfinder" software tool for identifying the subgraphs in the P2P networks [21], the P2P networks themselves were implemented in the open source Peersim environment. The code for the SLACER protocol, in addition to Peersim itself and tutorial materials are available from the Peersim website [22]. The random sampling service required by SLACER is provided by the Newscast protocol [7]. Peersim was initially developed within the BISON project [24]

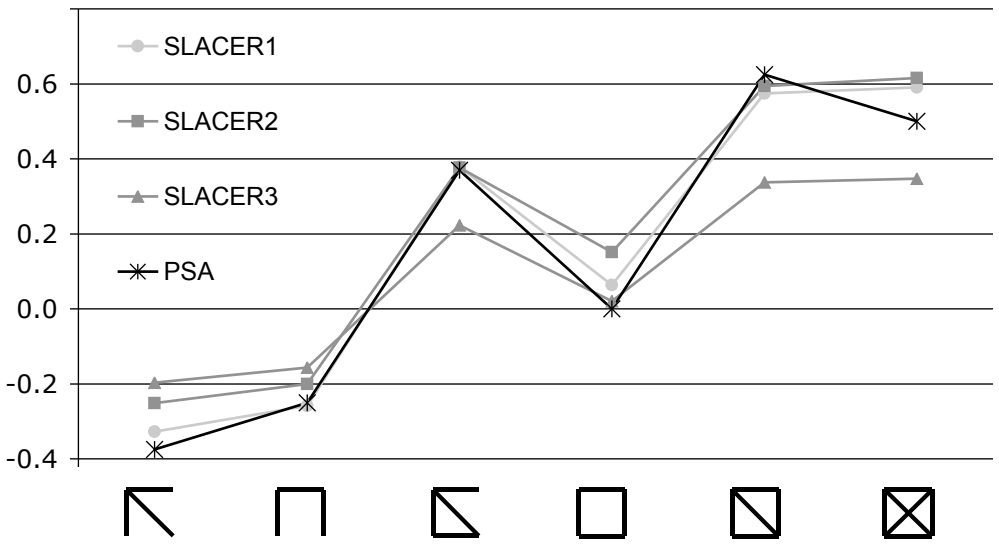

Figure 9: The subgraph ratio profiles (SRP) of the SLACER P2P networks at different stages in their evolution to cooperation compared to the protein structure averages (PSA) - results for SLACER were obtained in the same was as described for figure 8

the x-axis (being a random network). However, we have found that from *any* initial topology the network evolves to the same topology immediately before cooperation breaks-out [5] so we would expect the shown results to look the same no matter what the starting topology.

Notice that the curves in figures 8 and 9 follow a similar time evolution for both SLAC and SLACER: immediately before cooperation (snapshot 1) the curve already has a very similar shape to the final curve, during the outbreak of cooperation (snapshot 2) the curve tends to move upward slightly (less anti-motifs 1 and 2, but more motifs 3 to 6), then, after stable cooperation is attained (snapshot 3) the curve tends to flatten (with all points moving towards the x-axis). Motifs 1 and 2 are under represented (anti-motifs) and motifs 3 to 6 are over represented (motifs) but with a large dip for motif 4 - almost close to zero (identical to the random occurrence) when cooperation has stabilised (snapshot 3).

The SRP curves shown in figures 8 and 9 can be directly compared to those shown in figure 7 and we have copied the average of the protein-structure curves onto these figures, labelled as PSA. We note that the nearest super-family is composed mainly of protein-structure networks and a geographical growth model. This characteristic anti-motif representation in 1 and 2 and dip at 4 is reproduced in the P2P networks. However, notice that motif 6 dips slightly in the protein-structure networks but this is not reflected in the P2P networks. Here the P2P networks look more like the BA models [1] shown at the bottom of figure 7 or the power-grid networks shown at the top. Notice also that the early snapshots of SLAC look most like the protein-structure networks - interestingly this is before cooperation has stabilised. The snapshot 1 networks are taken before *any* cooperation has formed in the networks and can therefore only be the result of random copy and rewire since all utilities are then equal because all nodes use the defect strategy in the PD.

This suggests some rather intriguing new possibilities *and* re-enforces existing findings. Firstly, that the motifs and anti-motifs evident in the P2P SRP's initially result from a randomised process rather than the specific function of cooperation *but* that this randomised process is *itself* an artefact of an algorithm designed for cooperation. That is, an apparently random process can produce the same SRP but that this does not mean the underlying process is not driven by a specific function.

SLACER appears to work *because* a randomised re-wiring process draws the network towards a

topology that eventually supports cooperation. We have discussed in detail this multi-stage process elsewhere [5]. What we see is a complex interplay of function and topology formation leading to a phase transition in cooperation with negative scaling properties (i.e. the bigger the network the quicker it happens).

Another interesting result is that the different snapshots show significant differences in the SRP, such that for a given (P2P network) SRP it would be be possible to predict if it was in a stable cooperative mode or not - without considering the strategies stored in the nodes. This could possibly be very useful, since it indicates a way to detect if the network is functioning cooperatively or at which stage in the evolution to cooperation it has reached based purely on structural characteristics. Since we have observed SLACER to perform other kinds of cooperative task (not just P2P) and produce networks with the same topologies we speculate that potentially the final snapshot 3 SRP signature could characterise many possible application domains that were functioning under SLACER cooperatively (correctly), *even if the specifics of the applications themselves were complex or unknown or unknowable from a snapshot.* If this were the case, then a SRP curve could provide a very powerful fingerprint of the global collective functioning of a SLACER supported application (even one's not yet written!). Also a deviation from the cooperative fingerprint could indicate malfunctioning or network attacks.

Additionally, as stated previously, the only difference between the SLAC and SLACER protocols is the link drop probability (W). In SLACER this is lower than one (0.9) and this produces a larger dip in motif 4 - nearer to the protean-structure networks. It would be interesting to calculate SRP's for different values of W to see how this changes the curve.

2.5 Summary

We have briefly summarised some existing P2P protocols that we have designed to promote cooperation - system level utility - when nodes behave selfishly with local information. We performed a motif analysis by calculating and plotting subgraph ratio profiles as curves. We compared this to existing work showing curves for various natural and engineered networks. We did not expect to find the P2P networks to match any of the existing graphs but were intrigued to find that the P2P networks appeared very similar to a family of protein-structure curves. However, should we be surprised? The P2P algorithms were inspired by an evolutionary tagging algorithm [14] that produces robust and desirable life-like properties such as scalability, robustness and self-repair. It therefore may not be so surprising that the resulting topologies come to resemble naturally occurring networks with similar properties. In any case, we are excited by the discovery.

Rather than design an algorithm for constructing a particular network *form* [1] we have designed the P2P protocols for a specific *function* - to suppress the individual selfish behaviour of nodes for the collective good of the network without the need for complex or sophisticated reputation systems, repeated reciprocal interactions or centralised control. Perhaps this function has some kind of universal applicability - if so, could this throw any light on protein networks and their role in cell level computation? Is there really any linkage at all or just superficial similarity?

One interesting idea that this work suggests is that although networks may appear to be constructed from a random process of copy and re-wire this could be an *artefact of an underlying functional process* - which in certain phases reduces to a random process but nevertheless harnesses the functionally of the properties produced by that process.

Another potentially very useful line of work (for P2P engineering) would be to explore our hunch that the cooperative fingerprint shown in the SRP will hold over many possible application domains, hence providing a method for *monitoring the network for serious malfunctions or malicious attacks that degrade collective performance, generically, without having the know the application specifics.* This would be consistent with the thesis of [12] which is that perhaps the super-families identify some very general underlying functional characteristics even though we might not know the specifics.

Our initial work has raised a number of interesting questions which we hope to pursue in future work.

3 Conclusion

It appears that motif analysis gives deep insights into general properties of evolving information processing graphs.

What is becoming evident is that in many evolving networks, both natural, artificial and functionally engineered, the relationship between form and function is not simple. Rather than function dictating form or vice versa, it appears that form and function mutually constrain each other but only at a very generic level. Neither form nor function is master or slave.

Tinkering heuristics appear prevalent in this relaxed yet constrained relationship between form and function. In both the software evolution processes and the peer-to-peer protocols studied here, copy and re-wire heuristics tinker [16, 2] with the network but ultimately support functional properties. Perhaps "constrained tinkering" could be another word for "design" from an evolutionary perspective.

What is remarkable is that, at the very basic level of the simple motifs that we have explored, certain kinds of generic functional properties appear to be deducible and, moreover, can be compared and categorised with simple measures. Rather than the generic level being a problem it offers the great potential of producing generally applicable tools, techniques and understanding to a wide range of applications and phenomena.

Science, or understanding, appears to wrestle with the task of identifying function *from* form whereas design or engineering traverses the opposite direction - from function *to* form. In both cases the road between them is far from direct but somehow they meet on a tinkered landscape.

References

[1] Barabási, Albert-László and Albert, Reka (1999) Emergence of Scaling in Random Networks, *Science 286, 509*

[2] Edelman, G. M. and Gally, J. A. (2001) Degeneracy and complexity in biological systems. *Proc. Nat. Ac. Sc. 98(24):13763-13768.*

[3] Gamma, E., Helm, R., Johnson, R. and Vlissides, R, (1994) *Design Patterns Elements of Reusable Object-Oriented Software*, Addison-Wesley, New York.

[4] Hales, D.; Arteconi, S.; Babaoglu, O. (2005) SLACER: randomness to cooperation in peer-to-peer networks. In *Proceedings of the 1st International Conference on Collaborative Computing: Networking, Applications and Worksharing*, Workshop on Stochasticity in Distributed Systems (STODIS'05), IEEE Computer Society Press. [DELIS-TR-0119]

[5] Hales, D. and Arteconi, S. (2005) Friends for Free: Self-Organizing Artificial Social Networks for Trust and Cooperation. Submitted to IEEE Intelligent Systems Special Issue on Self-management through self-organization in information systems. Available: http://arxiv.org/abs/cs.MA/0509037. [DELIS-TR-0196]

[6] Hales, D. and Edmonds, B. (2005) Applying a socially-inspired technique (tags) to improve cooperation in P2P Networks. *IEEE Transactions in Systems, Man and Cybernetics - Part A: Systems and Humans, 35(3):385-395.* [DELIS-TR-0111]

[7] Jelasity, M., Montresor, A. and Babaoglu, O. (2005) Gossip-based aggregation in large dynamic networks. *ACM Trans. Comput. Syst., 23(1):219-252*

[8] Kong, J. S., Boykin, P. O, Rezei, B., Sarshar, N. and Roychowdhury, V. (2005) Let your cyberalter ego share information and manage spam. Available as pre-print: http://xxx.lanl.gov/abs/physics/0504026.

[9] Krapivsky, P. L. and Redner, S. (2005) Network growth by copying. *Physical Review E, 71, 036118.*

[10] Marti, S., Ganesan, P. and Garcia-Molina, H. (2005) DHT Routing Using Social Links In *Peer-to-Peer Systems III: IPTPS 2004, Revised Selected Papers, LNCS 3279* Springer.

[11] Milo, R., Shen-Orr, S., Itzkovitz, S., Kashtan, N., Chklovskii, D. & Alon, U. (2002) Network Motifs: Simple Building Blocks of Complex Networks *Science, 298:824-827*

[12] Milo, R., Itzkovitz, S., Kashtan, N., Levitt, R., Shen-Orr, S., Ayzenshtat, I., Sheffer, M. & Alon, U. (2004) Superfamilies of designed and evolved networks *Science, 303:1538-42.*

[13] Pastor-Satorras, R., Smith, E. D. and V. Solé, R. (2003) "Evolving Protein Interaction Networks through Gene Duplication", *Journal of Theoretical Biology, 222, pp. 199*

[14] Riolo, R.; Cohen, M.; Axelrod, R. (2001) Evolution of cooperation without reciprocity. *Nature 414, pp. 441-443.*

[15] Solé, R., Pastor-Satorras, R., Smith, E. D. and Kepler, T. (2002). A Model of Large-Scale Proteome Evolution. *Advances in Complex Systems 5, 43.*

[16] Solé, R. Ferrer-Cancho, R. Montoya, J. M. and Valverde, S. (2002) Selection, Tinkering and Emergence in Complex Networks, *Complexity, 8(1):20–33.*

[17] Valverde, S and Solé, R. (2005) Network Motifs in Computational Networks: A Case Study in Software Architecture. *Physical Review E 72, 026107.* [DELIS-TR-0206]

[18] Valverde, S and Solé, R. (2005) Logarithmic Growth Dynamics in Software Networks. *Europhysics Letters 72 (5)* [DELIS-TR-0207]

[19] Valverde, S., Ferrer-Cancho, R. and Solé, R. (2002) Scale-Free Networks from Optimal Design. *Europhysics Letters 60, pp. 512-517.*

[20] Motif Dictionary,
available at: http://www.weizmann.ac.il/mcb/UriAlon/groupNetworkMotifSW.html

[21] mfinder network motif detection tool software,
available at: http://www.weizmann.ac.il/mcb/UriAlon/groupNetworkMotifSW.html

[22] Peersim Peer-to-Peer Simulator, http://peersim.sf.net

[23] Graphviz graph visualisation software, http://www.graphviz.org/

[24] The BISON Project, http://www.cs.unibo.it/bison