ComPlexUs
INFORMATION TECHNOLOGY MODELLING
Complexus 2006;3:108–118 DOI: 10.1159/000094193
Published online: ■■■

Emergent Group Level Selection in a Peer-to-Peer Network

David Hales

Department of Computer Science, University of Bologna, Bologna, Italy

This work was partially supported by the EU within the 6th Framework Programme under contract 001907 (DELIS).

Key Words
Design of complex systems · Robustness · Network dynamics · Emergent structure

Abstract
Many peer-to-peer (P2P) applications benefit from node specialization: for example, the use of supernodes, the semantic clustering of media files or the distribution of different computing tasks among nodes. We describe simulation experiments with a simple selfish re-wiring protocol (SLAC) that can spontaneously self-organize networks into internally specialized groups (or 'tribes'). Peers within the tribes pool their specialisms, sharing tasks and working *altruistically* as a team – or 'tribe' – even though their individual behaviour is selfish. This approach is scalable, robust and self-organizing. These results have implications and applications in many disciplines and areas beyond P2P systems.

Fax +41 61 306 12 34
E-Mail karger@karger.ch
www.karger.com

KARGER

© 2006 S. Karger AG, Basel
1424–8492/06/0033–0108
$23.50/0
Accessible online at:
www.karger.com/cpu

David Hales
Department of Computer Science, University of Bologna
Mura Anteo Zamboni 7, IT–40127 Bologna (Italy)
Tel. +39 051 1998 0461, Fax +39 051 209 4510,
E-Mail dave@davidhales.com

Simplexus

Peer-to-peer (P2P) networks such as Kazaa, Gnutella or Bittorrent now account for more than a third of everything that happens on the Internet. As subcommunities within the larger Internet, where members interact with others who share their interests – in downloading music, for example, or exchanging other kinds of files – P2P networks give the Internet a 'social' structure that is every bit as real as the social structure in the physical world, and just as useful. In principle, such networks work like tightly knit communities and make the resources of anyone available to all.

In general, each member of a P2P group downloads 'client' software that acts as an interface through which users interact. Each user is linked to a handful of others, who are their immediate neighbours in the network. When a user wants a particular file, they request it from their neighbours, who can supply the file, if they have it, or pass on the request to their own neighbours. In this way, the file may eventually be found and returned to whoever requested it. But today's file sharing, of course, may only hint at a universe of more sophisticated future applications. P2P networks might help people pool their resources to provide a data backup service, for example, or provide the shared infrastructure for distributed computing. In this sense, P2P networks probably represent the forerunners of a far more 'social' Internet, in which users will be able to cooperate and coordinate their activities far more effectively.

But making it work in a stable and resilient way won't be so easy. As David Hales of the University of Bologna points out in this paper, social cooperation, whether it is among social bacteria, chimpanzees or Internet users, always co-exists in a very uneasy way with potential social dysfunctions associated with cheating. Within a P2P network, for example, users have incentives to cheat by downloading files while never helping to supply them to oth-

1 Introduction

Open peer-to-peer (P2P) networks (in the form of applications on top of the Internet) have become very popular for file sharing applications (e.g. Kazaa[1], Gnutella[2], Bittorrent[3]). However, can such technology be applied to other computing tasks? For example consider a system in which some nodes have lots of free storage, some high bandwidth and others non-firewalled connections to the network.

Those nodes could cooperate to provide a data back-up service – something that no individual node could provide. Obviously, in such a situation, if there is a demand for a back-up service we would wish the nodes to, somehow, get together and provide the service – but how? One solution (and currently, it would seem, the only viable one for deployable applications) is to code the process of specialization, coordination and cooperation into the protocol directly for each different kind required. So for example, where semantic clustering of media files is required for file sharing, protocols exist that implement it[4] [1]. Where systems require supernodes [2], again, these are implemented directly. There are two problems with this approach; firstly, for every kind of specialization required a programmer must envisage this a priori, design a protocol, then implement and test it. Secondly, since this process is complex enough on its own, it is generally assumed that nodes will follow the protocol – it is rare to find protocols robust to node failure, noise or malicious behaviour, such as free riding, although this is, to a certain extent, true within the BitTorrent system [3].

Additionally, it is also rare that nodes can spontaneously change their specialism if they come to recognize that they might be able to do better following a different role. The specialism of the node tends to be hard-coded or relies on user level switches. This kind of approach limits the ability of the system to automatically adapt to changing task scenarios – however, see Montresor [2] in which supernodes are dynamically allocated to improve performance.

Ideally, we would like a more general approach that could be applied to a range of different task domains with minimal tuning. We would like the approach to offer dynamic specialization and respecialization if nodes come to recognize they could do better playing another role and have the ability to do so or if the task domain changes requiring different kinds of skills to be combined. In addition, we want the system to be able to deal with free riders and errant or malicious nodes but also to support altruistic cooperation between specialists when this is required for job completion. Finally, we require this to be as scalable, self-organizing and robust as possible.

In this paper we do not claim to have addressed all these issues to the level of deployment; what we propose is, we claim, the beginnings of an approach that may allow us to address these issues. In the simulated scenarios so far implemented our results are very encouraging and we plan to continue this line of work.

In the following sections, we state our assumptions concerning behaviour in open P2P systems, then we introduce the SLAC algorithm in general terms. We follow by formulating a minimal task domain scenario called the SkillWorld, to which we wish to subject a simulated P2P network running SLAC. We then describe how we apply SLAC within SkillWorld and present some experiments and results. We interpret the results and describe a 'typical history' in the SkillWorld.

At the end of the paper we summarize what we have observed and what it means.

ers. Several years ago, studies of the Gnutella network found just this problem – well over half its users were 'free riders' and never contributed their resources to the network. When many free ride, of course, those who cooperate naturally lose out and have the incentive to become free riders themselves. Consequently, free riders potentially undermine the cooperation on which social function depends.

Yet the main point of Hales's paper is more optimistic – that an imaginative approach to engineering P2P networks may yet provide means for overcoming such problems. In human societies, it is often the credible threat of punishment that tends to deter cheaters. Such mechanisms may ultimately also be useful in the P2P context. But Hales suggests that it may also be possible to achieve cooperation in the absence of punishment by designing the basic rules by which a network operates so that cooperation can be sustained, even if some users do try to cheat. Hales and colleagues have already shown how this can be achieved in the simple context of file-sharing networks. Here he demonstrates that the same simple mechanism can also be extended to help promote cooperative behaviour in more sophisticated scenarios of the kind that may ultimately make P2P networks more generally useful.

In a file-sharing network, users would like to find files they want and download them from others. Each person controls their links to others and can change these links as they like. They can also control whether or not they are cooperative. A user might be altruistic and happily respond to others' requests for files, or be non-altruistic (like a lot of greedy Gnutella users) and never lift a finger to help anyone else. In previous studies, Hales has demonstrated that it is surprisingly easy, in this context, to engineer cooperation by supplying P2P users with a simple strategy for self-improvement. The idea is to let pairs of users occasionally compare their performance – the fraction of times their re-

[1] The Gnutella home page: http://www.gnutella.com.
[2] The Kazaa home page: http://www.kazaa.com.
[3] The BitTorrent home page: http://www.bittorrent.com. See also Cohen [3] for a description of the way BitTorrent works.
[4] For example see the MLdonkey system: http://mldonkey.org.

We claim that the results indicate a process that has possibly profound implications and applications beyond just P2P systems.

2 Behavioural Assumptions in Open Networks

How do nodes behave in open P2P networks? Of course, the simple answer is, assuming nodes are autonomous: *anyway they like to behave!*

Given this fact, how then do we proceed to devise protocols that will lead to desired system-level functions? Obviously, we have to begin by making assumptions about the *likely behaviour* of other nodes in the network. Such assumptions should be as realistic as possible but also simple enough to be practically computable and transferable between a number of domains. Assumptions made here are essentially the axioms of a kind of mini *social theory* which then informs the design of peer software.

Many approaches (often unconsciously) inherit assumptions from previous social sciences (e.g. economics, sociobiology, sociology). For example, if we assume nodes will behave '*rationally*' in the context of *classical game theory*, then we compute 'Nash equilibrium'; inheriting our assumptions from game theory which is a body of knowledge assuming perfect rationality and perfect information. The basic approach is to assume that all individuals have perfect knowledge of the game being played and all possible outcomes along with infinite computational time and common knowledge that all individuals are the same in these respects. Given these assumptions it is sometimes possible to analytically derive the 'Nash equilibria' of the game being played. The idea is that given the previous classical assumptions any system will find and stay in a Nash equilibrium. However, it is unclear that such assumptions hold in dynamic open P2P networks and the derivation of such equilibria within dynamic topologies and changing populations is currently beyond state-of-the-art analytical techniques.

In the context of sociobiological models [4,5], which are based on the evolution of behaviours of interacting animals over time, the assumption is that behaviours (or strategies) reproduce in proportion to their average fitness (utility or score) such that fitter behaviours become more numerous over time. Additionally such models assume that mutation in the form of random changes in behaviour also takes place. This *evolutionary game theory* approach allows for an ecology of behaviours to evolve over time. In addition, there is no requirement that agents have perfect rationality or perfect information – just enough, such that better performing strategies tend to increase in the population. For biological systems this occurs via Darwinian evolution where utility equates to fitness. However, P2P networks do not evolve in a Darwinian fashion. Nodes do not reproduce and it is unclear what 'fitness' means in this context.

We have shown in recent work that results from evolutionary models *can* be applied in networks if we allow nodes the ability to 'copy and re-wire' within the network to improve their own situation [6–9]. This latter innovation demonstrates it is possible to import work originally modelled in a conventional evolutionary framework into a dynamic network model. Nevertheless, in the absence of any deductive proof of the equivalence of evolution and the re-wire rules it is necessary to implement and test previous mechanisms to determine if the properties of interest can be carried over into networks.

Summary of assumptions concerning open P2P networks:

(1) Nodes are in the network for what they can get out of it.

(2) Nodes modify their behaviours to improve their individual benefit.

(3) Nodes have limited knowledge about other peers and the network in general.

quests actually get answered successfully – and to let those doing worse copy the behaviour of those doing better; that is, to mimic their behavioural stance (altruistic or non-altruistic) and also to copy their links to other users.

It is plausible to assume that if users have ready access to such a 'copy and rewire' tactic, they will probably use it. After all, it's an easy way, selfishly, to improve your own performance. What is remarkable is that users' greed, if channelled and expressed in this particular fashion, leads quickly to the emergence of lots of altruism and cooperation. Here is how it works, as Hales has discovered in extensive simulations. Suppose everything starts out at random, with users mixed in their altruism and linked together in some haphazard way. Then naturally, some clusters of users quite by chance will happen to have more altruists than others. Members of these altruistic groups will generally do better than average, as their neighbours help them to find files. Consequently, as users go about comparing themselves to others and trying to copy their way to improvement, many will tend to become altruistic and to link themselves into this altruistic cluster, making it grow. In short, copying and rewiring, inspired solely by greedy self-interest, makes altruistic clusters grow and spread at the expense of less altruistic clusters, all through an evolutionary competition between groups with different levels of altruism.

If this seems a little too good to be true, it is, and Hales has also found that such 'tribes' of cooperators cannot last. In his simulations, he allowed for occasional 'mutations' – for the possibility that users can change their behaviour and links to others quite at random. As a result, an altruistic tribe will eventually get 'infected' by a cheater, someone who stays linked with the group, but who turns non-altruistic. Cheaters immediately make a killing, gaining from the altruism of all their neighbours, without spending any effort to help them.

The first assumption would appear to be plausible within open P2P networks. In the currently popular file sharing networks the majority of uses download and run peer client software (and hence join the network) in order to get something (e.g. to download a movie or a music file). It certainly is true that some people would join for other reasons. For example, a user may join to feel 'part of an online community' [10] or to distribute only their own content – not downloading. Some could aim to damage the functionality of the network by distributing malicious content. However, we argue that neither of these motivations informs the majority of the nodes. In any case, most functions would be *enhanced* by purely altruistic behaviour (such as distributing content without downloading) and we *conjecture* that there are at least as many pure altruistic as pure malicious nodes in working networks.

The second assumption is more problematic – *who says nodes within a given P2P network change behaviours to improve their benefit?* Our argument here is rather speculative – if not conjectural. We start from the assumption of autonomy and argue that the function of peer client software is *ultimately* under the control of the user. For example, users may change operating system or client software settings (e.g. limiting upload speeds), download new versions of a peer client (e.g. incorporating ways to improve download success and rates) or simply hack their own code if they have the required skills. Of course, a hacked client can be distributed to others if it appears to have desirable properties and will tend to be adopted if it delivers those properties to others. We therefore claim that currently, this kind of process is occurring at the user level – via the adoption of various clients and the control of various node-level settings. The problem hidden in this assumption is that the space of available behaviours that each user can choose from varies over time and is also dependent on the knowledge of the user, the kind of network connection, form of operating system and many other related factors. However, we note that similar assumptions have provided some insight into human sociocultural phenomena at least as complex as the sociocultural phenomena of P2P systems [11].

Perhaps a more plausible way of thinking about the second assumption is to interpret the space of all available clients in a given P2P domain as the space of behaviours a user can select from – that is, a user may change clients programs, say from edonkey to BitTorrent because edonkey is too slow for the content they required. The user has in fact changed protocol and network completely – but that need not matter to them, and in fact the interpretation then is an ecology of different networks with users switching between them.

Alternatively, the assumption that behaviour can change regularly within a single network can be seen as a design proposal to be incorporated into a new protocol, rather than an interpretation of existing protocols.

The third assumption would appear to be a necessary one in any large and highly dynamic system – it is not practical or possible to collate accurate global statistics in most such systems.

3 The SLAC Algorithm

In previous work we showed how a simple 'copy and re-wire' rule (or protocol or algorithm) could produce high levels of cooperation within simulated P2P networks performing collective tasks. We named this algorithm 'SLAC' because it uses selfish link and behaviour adaptation to produce cooperation. We showed that nodes in a network emerged cooperation when playing the single-round prisoner's dilemma game, under, what we argue, are plausible assumptions about the kinds of behaviour we find in P2P systems. We also demonstrated that the same results could carry over into a more realistic file-sharing P2P task domain [6].

One cheater attracts others, and together they ultimately undermine the group. So tribes naturally come and go. Even so, however, Hales found that they form so quickly that the overall level of cooperation remains high on average – it's just not always the same users who are cooperating.

In a real P2P network, of course, no 'algorithm' will completely control users' behaviour. They can be altruistic or non-altruistic at will, and can even hack into the client software and make it do anything they like. So it remains to be seen if the copy and re-wire tactic, if made available in a network, really would promote high levels of cooperation. But in the present paper, Hales tackles another issue on the road to the practical engineering of P2P cooperation by exploring how the copy and rewire strategy might be adapted to more delicate applications that move beyond simple file sharing. He does this in the context of a virtual world that he calls 'SkillWorld'.

File sharing networks require only one skill – the ability to supply a file. In more diversified settings, P2P users might want to accomplish tasks of many different kinds, and so require a range of skills from their network partners. SkillWorld models this situation. Within it, as in a file-sharing network, each of N nodes links to a handful of neighbours. Each can be either altruistic or non-altruistic, but each user now also has a 'skill' type, selected from one of five different possibilities. In SkillWorld, users get assigned tasks that they need to complete, and to do so, unless they luckily have just the right skill, need to locate a neighbour in the network who both has that skill and is willing to share it. In simulations of such a network, Hales keeps track of the costs and benefits of users' behaviour by giving one 'utility' point to anyone who gets one of their tasks completed. Meanwhile, the person who had the relevant skill and did the work has to pay out 0.25 utility points (helping others isn't free).

```
DO periodically forever
  select a random node j from the network
  compare utility of this node (i) with node j
  IF utility of j is higher (Uj >= Ui)
    drop all current links (clear view of i)
    copy links of node j (copy view from j to i)
    add link to j (add to view i a link to j)
    copy behavioural strategy of j
    with a low probability (mutation rate 1)
      drop all current links (clear view of i)
      add a link to a randomly chosen node
    with a low probability (mutation rate 2)
      change the behavioural strategy randomly
   END IF
END DO
```

Fig. 1. The generic SLAC algorithm. Each node executes this algorithm.

The basic algorithm assumes that peer nodes have the freedom to change behaviour (i.e. the way they handle and dispatch requests to and from other nodes) and drop and make links to nodes they know about. In addition, it is assumed nodes have the ability to discover other nodes randomly from the network, compare their performance against other nodes and copy the links and (some of) the behaviours of other nodes.

As discussed above, we assume that nodes will tend to use their abilities to selfishly increase their own utility in a greedy and adaptive way (i.e. if changing some behaviour or link increases utility then nodes will tend to select it).

Over time nodes engage in some activity and generate some measure of utility *U* (this might be number of files downloaded or jobs processed etc., depending on the domain).

Periodically, each node (*i*) compares its performance against another node (*j*), randomly selected from the population. If $Ui \leq Uj$ node *i* drops all current links and copies all node *j* links and adds a link to *j* itself. Also, periodically, and with low probability, each node adapts its behaviour and links in some randomized way using a kind of 'mutation' operation. Mutation of the links involves removing all existing links and replacing them with a single link to a node randomly drawn from the network. Mutation of the behaviour involves some form of randomized change – the specifics being dictated by the application domain (see later).

Previous 'tag' models, from which SLAC was developed [8,9,12], have indicated that for good scalability properties the rate of mutation applied to the links needs to be higher than that applied to the behaviour, by about one order of magnitude. In the context of the algorithm shown in figure 1 this means that 'mutation rate 1' >> 'mutation rate 2'.

When applied in a suitably large population, over time, the algorithm follows a kind of evolutionary process in which nodes with high utility tend to replace nodes with low utility with nodes periodically changing behaviour and moving in the network. However, as will be seen, this does not lead to the dominance of selfish behaviour, as might be intuitively expected, because a form of incentive mechanism emerges via a kind of ostracism in the network. The process can also be viewed as a kind of 'cultural group selection' process (see later discussion).

Given the inherent cost to helping, it seems unlikely that nodes in this network could possibly manage to cooperate. It won't generally pay to be altruistic, as you do the work but get no benefit. But Hales found in his simulations that the 'copy and re-wire' strategy of node self-improvement can also build and sustain cooperation in this more demanding setting. He started SkillWorld with all users set randomly to be altruistic or not. Randomly, he gave them skills from the five possibilities and some links to other users. Users were then selected, again at random, and given jobs to do demanding one of the skills. As before, Hales allowed mutations to rarely and randomly alter a user's skill set, altruistic stance and network links. He then monitored how the fraction of jobs completed successfully changed with time.

In general, the only way a node can get a job completed, if it doesn't have the required skill, is to pass it to an altruistic neighbour who does. Hence, a high rate of success would imply the emergence of large islands or tribes of altruistic nodes, which is precisely what Hales found. By the time every node had, on average, handled a few tens of requests, the level of completed jobs had reached 90%. Interestingly, however, it only worked for networks having more than about 1,000 nodes, presumably because network size increases the likelihood that an altruistic group will form somewhere, by chance, and then grow by attracting others. In essence, the formation of altruistic tribes faces a 'nucleation' barrier much like that which makes some liquid solutions 'metastable', though the solid crystals grow quite quickly once seeded.

In his section 5.2, Hales offers a detailed look at how precisely such cooperation comes about. At first, cheaters prosper, taking advantage of altruists. But altruists leave these groups and begin forming cooperative groups, which then grow (see his fig. 4). Anyone within this group, if they search for a skill, will generally find a

4 The SkillWorld Scenario

In order to determine if the SLAC approach can support specialization within tribes we construct an abstract and minimal simulated task domain that requires nodes to perform specialized tasks cooperatively in order to satisfy their individual needs. We call the task domain SkillWorld and it is an adaptation of a sociologically inspired scenario originally given in Hales [13].

The SkillWorld consists of a population of N nodes. Each node may have zero or more links (up to a maximum of 20) to other nodes. Links are undirected such that the entire population can be considered as a undirected graph G with each vertex being a node and each edge being a link. Each vertex (or node) is composed of three state variables – a 'skill type' $s \varepsilon \{1,2,3,4,5\}$, an 'altruism flag' $a \varepsilon \{0,1\}$ and a satisfaction score or 'utility' $u \varepsilon$ R (where R is a positive real number).

Periodically, with uniform probability, a node i is selected from the population N. A 'job' J is then generated marked with a randomly chosen skill sJ. The skill is selected, again randomly with uniform probability, from the domain $\{1,2,3,4,5\}$. Job J is then passed to node i. If node i possesses the correct matching skill (i.e. if $s_i = sJ$) then node i may process the job itself without any help from other nodes. For successfully processing a job J the receiving node gains one unit of credit: $u \leftarrow u + 1$.

This process of generating and passing jobs to nodes represents user-level requests for services – such as, for example, searching for a particular file, performing some processing task or storing some data. In the SkillWorld we do not represent the actual jobs to be done, rather we represent the skill required to perform the job. In our minimal scenario, each job only requires one skill to be completed.

But what if node i receives a job for which it does not have the correct skill (i.e. if $s_i \neq sJ$)? In this case i passes the job request to each neighbour in turn until all have been visited or one of them, j, agrees to process the job J. A neighbour j will only agree to process J if its skill matches ($s_j = sJ$) and the altruism flag is set ($a_j = 1$). If j does agree to process the job then this *costs* j a quarter unit of utility ($uj \leftarrow uj - 0.25$) yet *increases* the utility of i by one unit ($ui \leftarrow ui + 1$).

What this means is that node i looks for an altruistic neighbour with the correct skill to process job J. If i finds such a neighbour (j) it increases its utility *as if* it had completed the job itself whereas j *decreases its utility*. This reflects the notion that j is altruistically processing J for the benefit of i and that users are happy when jobs submitted to their nodes are completed but are not happy when jobs from other nodes use their node resources with no immediate benefit to themselves.

5 SLAC in the SkillWorld

We apply the SLAC algorithm within SkillWorld by making the node skill types and the altruism flags into evolvable state variables such that they are copied from more successful nodes (based on utility) and mutated occasionally with low probability.

Although SLAC has previously been demonstrated as successful in promoting cooperation in both a prisoner's dilemma playing scenario [7] and a simple file-sharing scenario [6, 9] it has not yet been applied within a scenario requiring intra-group (or tribe) specialization *in addition* to altruism. We are therefore asking a lot from a simple algorithm: to self-organize the population into altruistic yet internally specialized tribes that pass and process jobs using their various skills.

The SkillWorld is the simplest scenario we could think of that captures a process of specialization for this initial investigation. We have a small number of skills (five in these simulations) and we only pass jobs to immediate neighbours. Each node and job is related to a single skill only (rather than a subset of skills which would seem neighbour willing to supply it. These successful, cooperative groups draw in new members and come to dominate the population, while at the same time evolutionary pressure also leads these groups to have well-mixed sets of skills, improving their ability to carry out tasks. But success again sets the stage for ultimate demise, as these cooperative tribes offer prime ground for cheaters, who can prosper at the tribe's expense. Consequently, as Hales puts it, history in SkillWorld is 'the history of the formation, growth and destruction of tribes'.

The demise of cooperative tribes is a peculiar feature of the 'copy and re-wire' rule of node self-improvement, either in the file-sharing context, or in SkillWorld. Yet the perpetually dynamic nature of the cooperation to which it leads may hold hidden advantages in helping the network to evolve solutions to new challenges 'on the fly'. If the needs of the network were to suddenly shift so that most jobs required, say, just two particular skills, a new tribe would quickly emerge with just this set of skills. The inherent cooperation set up between different groups gives the entire population a kind of adaptive talent for pattern recognition, as those groups tuned to present conditions naturally grow and displace others.

The copy and re-wire strategy of node self-improvement supports the emergence of cooperation in a flexible and even 'intelligent' way, and appears to be a promising route for engineering the self-organization of socio-technological communities. As Hales points out, the tribes are in no way 'pre-programmed' into the simulation; they emerge quite on their own as nodes simply try to get what they want. The tribes emerge, in effect, as a crude form of 'culture' that helps nodes coordinate themselves to handle tasks they could not handle alone. Further work will be required to see how this particular technique fares in the context of the full range of human behaviour likely to be encountered in any real-world P2P scenario, but it is hard to imagine that similar approaches, inspired by familiarity with the dynamics of cooperation in biological and social settings, will not become increasingly relevant to the engineering of a socially healthy Internet.

Mark Buchanan

more realistic). Also we assume nodes can change skills at will (randomly via mutation). This latter assumption might not hold if skills relate to physical or unchangeable characteristics of nodes like storage or bandwidth for example. However, at this stage we leave more realistic scenarios with multi-hop passing and more complex skill set arrangements to future work.

In order to measure the success of SLAC we take a simple measure – the proportion of submitted jobs that are completed. We can infer that a network in which the majority of jobs submitted are completed is sustaining internally cooperative and specialized tribes since the only way to complete most jobs is for nodes to pass them to altruistic neighbours with required skills.

5.1 Some Experiments and Results

Initially we ran a set of simulation experiments in which we initialized all nodes in the population with uniformly randomly selected skills, altruism flags and links. We experimented with a number of network sizes determining for each how many cycles before the high performance was achieved. A single cycle is the time unit by which all nodes will have executed the SLAC algorithm at least once, on average. In one cycle 10N jobs are submitted to randomly chosen nodes.

In order to measure the success of SLAC we take a simple measure – the percentage of submitted jobs that are completed (PCJ). We can infer that a network in which the majority of jobs submitted are completed is sustaining internally cooperative and specialized groupings (or tribes) since the only way to complete most jobs is for nodes to pass them to altruistic neighbours with the required skills.

We categorized ‘high performance’ as a PCJ > 90%, we found that in the simulations this was the highest stable value reached, and ran simulations until this value was obtained – recording the number of cycles required. Hence, if SLAC was working well in the SkillWorld we would hope that within a small number of cycles the PCJ would become high.

We used a mutation rate of 0.001 on skill type *s* and altruism flag *a* (shown in fig. 1 as ‘mutation rate 2’). Mutation on the links (shown in fig. 1 as ‘mutation rate 1’) was an order of magnitude higher (0.01). We carry over this assumption – that the mutation rate on the links should be higher than that on the ‘strategies’ – from previous experimental work comparing several different scenarios and models [9]. We fixed the maximum number of links between nodes to 20. Links are undirected and therefore symmetric. If an operation results in a

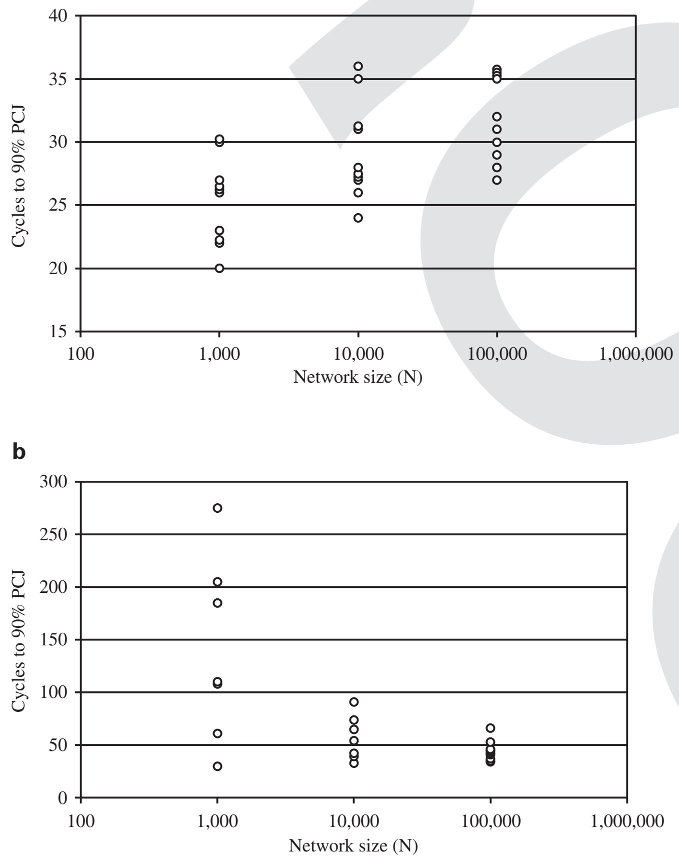

Fig. 2. a Number of cycles to high performance for different network sizes. When PCJ > 90% this means that over 90% of all jobs submitted to nodes are completed. Note: Overlapping circles have identical values. **b** Number of cycles to high performance for different network sizes when all nodes are initialized selfish (*a* = 0). This can be compared to the random initialization results in chart **a**. Note that there is a reverse scaling cost here. The results for N = 1,000 are worse than shown since three outliers at about 1,000 cycles are not shown here.

node requiring a new link and it already has the maximum then a random link is discarded by the node and the new link accepted. Using this method nodes never refuse new links but may often lose old ones. This adds to the noisy and dynamic nature of the scenario.

Figure 2 shows results from 30 individual simulation runs. Each point is a different run showing the first cycle at which the PCJ $> 90\%$. As can be seen, high performance is attained within a few tens of cycles even for networks of size $N = 10^5$. Notice that there appears to be a very slight upward trend in cycles as N increases; however, this is negligible – the results therefore indicate close to *zero scaling cost*. This highly desirable property was also evidenced in a previous application of SLAC to a simulated fire-sharing scenario [6]. Figure 2b shows results under the same conditions except that all nodes are initialized to be selfish ($a = 0$). This gives a kind of 'worst case scenario' as far as altruism is evolving. It is important to show that the system can escape from this, since this demonstrates that even if a complete failure of node altruism should occur (either through chance or malicious attacks) then the system can recover relatively quickly. We notice here the reverse scaling properties that we originally noticed and analyzed in a previous 'tag' model [12]. Essentially, with bigger populations, there is more likelihood of the chance formation of a small altruistic tribe. This then goes on to 'seed' the population with altruism.[5]

Interestingly, it was found that for populations where $N < 1{,}000$ high performance was *not* produced even when runs were extended to several thousand cycles. Intuitively this is consistent with the 'group selection' hypothesis concerning how SLAC operates. With small populations there are not enough nodes to form enough competing groups (or tribes) so evolution cannot operate at the group level.

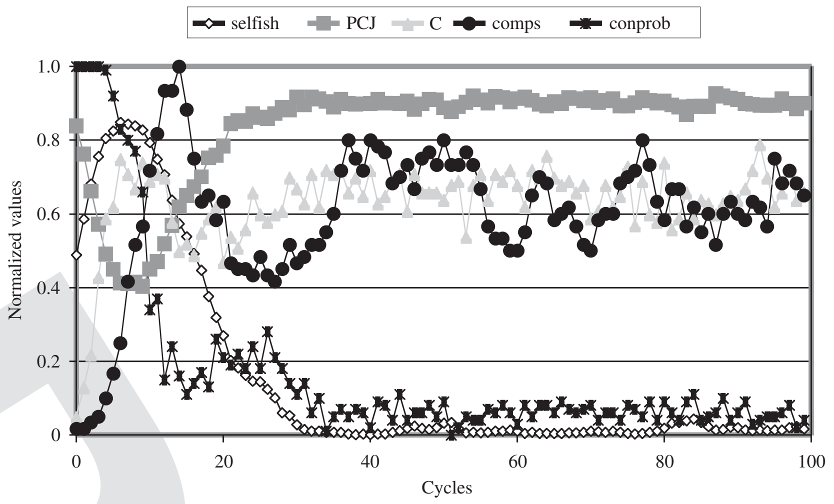

Fig. 3. The time series of a typical single run in SkillWorld (N = 1,000). Shown are the number of selfish nodes as a proportion of the entire population (selfish), the PCJ, the clustering coefficient (C), the number of components in the population (comps, which is normalized by dividing by 60) and the average probability that a route exists between any two nodes (conprob).

5.2 History in the SkillWorld – Tribal Dynamics

One way to convey the dynamics of a typical SkillWorld simulation run is to describe a typical 'history' in narrative form – this method is sometimes used in computational sociology, particularly in work with artificial societies [14, 15] carried over from more traditional sociological methods of explanation. In the rest of this section we give such a 'typical history'. Although we will make general points we will also refer to a specific single simulation, run given in figure 3, to illustrate our analysis.

Initially, the SkillWorld is a random graph, all nodes are connected via a few hops and clustering is low. Skills and altruism are randomly scattered. Very quickly, the graph breaks into a population of many disconnected components because nodes quickly re-wire themselves to better performing nodes.

The better performing nodes are initially the non-altruists who exploit their groups (or tribes) selfishly. However, this is a non-sustainable strategy since this exploitation causes nodes to leave their exploited tribes and join tribes in which there is less exploitation – nodes in tribes with less exploiters in them do better (higher utility) because they are cooperating as a team. The tribes dominated by non-altruists quickly 'wither away' as nodes leave. When no nodes are left then the tribe no longer exists – in this way *tribes die*, even though *nodes do not die*. This emergent property of the birth and death of tribes lays the ground for evolution to operate the group (tribe) level.

Figure 3 indicates the above process occurring in the first 10 cycles or so. Notice that the number of selfish nodes peaks, and the PCJ bottoms out, at about cycle 10. The number of components (i.e. tribes) increases in the early phase peaking just before cycle 20 (representing a peak of 60 components).

Altruistic tribes function well and grow as more nodes join, new tribes are occasionally formed as nodes randomly, through mutation, split from a tribe. As altruistic tribes grow larger they eventually

[5] See Hales [12] for a more detailed explanation of this reverse-scaling cost including the beginnings of an analytical treatment.

become 'infected' or 'invaded' by a non-altruist node – either by mutation of an existing member node or the entering of a new node to the tribe. When this happens the tribe is quickly destroyed via dispersion since a non-altruist will exploit the tribe selfishly and this will lead to many more nodes quickly copying that node until the tribe 'dies' because all nodes leave it – because a tribe dominated by selfish nodes gives lower utility to *all nodes* within it than one dominated by altruists.

Figure 3 shows, from about cycle 20 onward, the above process occurring. A decrease in the number of components (comps) and an increase in completed jobs (PCJ) are correlated with a decrease in the number of selfish nodes (selfish). This is because altruistic tribes grow in size – reducing the total number of components (comps) and reducing selfish nodes (selfish). By about cycle 30 selfishness is very low and completed jobs (PCJ) reach a high level. Notice that the dynamic nature of the formation and dissolution of the tribes is reflected in the variation of the number of components over time (comps) after PCJ goes high.

History in the SkillWorld is the history of the formation, growth and destruction of tribes. From the simple rules of the SLAC algorithm an *evolutionary process emerges at the tribal or group level*. Essentially one can think of this evolution as the competition between tribes to retain nodes to continue to exist. This process is in constant flux due to mutation and movement, no equilibrium state is attained and no tribe lasts forever. As long as new altruistic tribes are created at least as rapidly as they are destroyed altruism can survive.

Figure 4 shows a small detail of snapshots of the population over time (space does not permit full size snapshots). As can be seen, tribes quickly emerge and grow, producing various structures and sizes with internally specialized nodes.

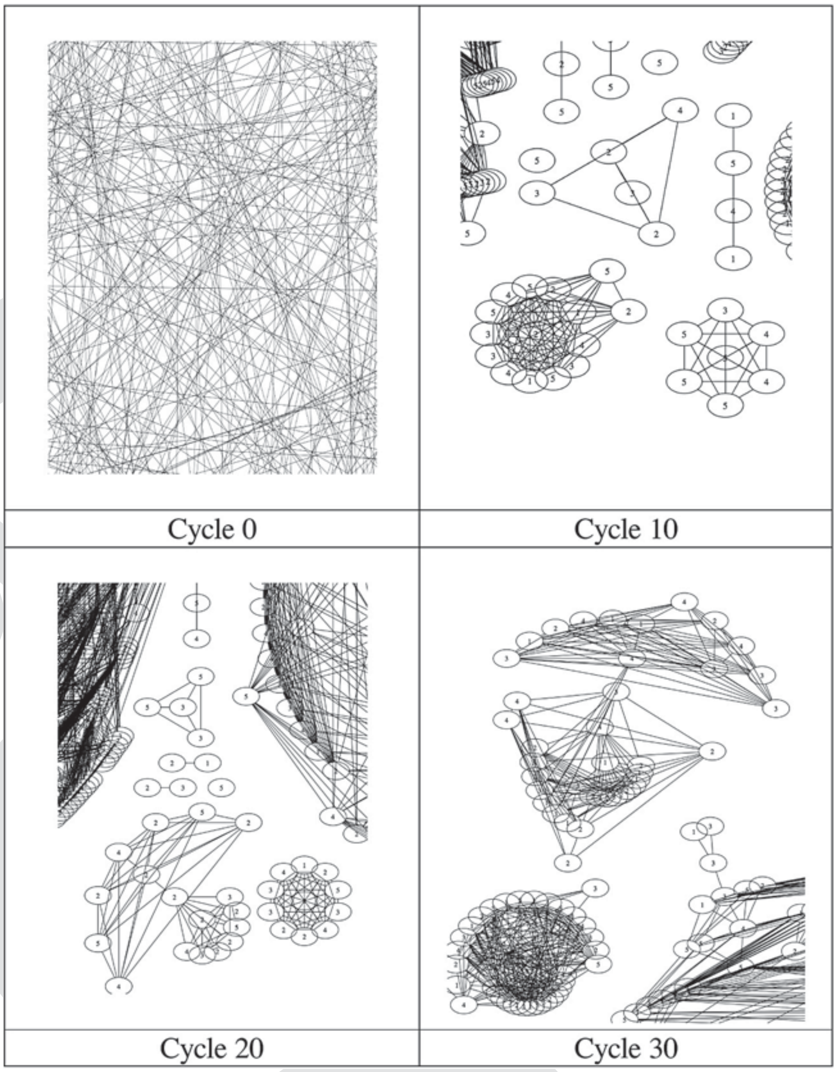

Fig. 4. Details showing just a small part of the entire population for the same typical run as show in figure 3. From an initially random graph disconnected components (we call tribes) emerge with internal specialization and rich structure. The numbers in the nodes represent the node skill type.[6]

5.3 Tribal Structures

Within the SkillWorld, tribes with different structures and skill mixes will support different levels of utility – a highly connected tribe with an even mix of skills would produce better results than a tribe missing some skill. Hence, selection at the tribe level (group selection) will tend to operate to structure the tribes into more optimal structures of skill types. We would therefore expect to see tribes composed of nodes possessing each skill type linked together such that a node receiving a job can either process it directly or will be directly linked to a node with the appropriate skill willing to do the job. In the SkillWorld then, we have tribe level selection not only operating to control selfishness but also to tune the internal (organizational) structure of the tribe.

[6] Full-sized pictures can be found at http://www.davidhales.com/esoa05pics.

We find this particularly exciting since we believe that by increasing the complexity of the task domains and giving nodes a little more freedom to hop more than one link within their tribe it should be possible to evolve tribes with *complex organizational structures* tuned to performing in the given task domain. Moreover, since the tribes are constantly evolving they should be able to *change their structure dynamically to address a change in the task domain.*[7]

6 Conclusion

We have demonstrated that the SLAC algorithm can be applied in a scenario (the SkillWorld) requiring node specialization in addition to the suppression of selfish behaviour. When the algorithm is executed the network quickly divides into competing 'tribes' (disconnected components). An evolutionary process then emerges at the group level selecting efficient internally specialized tribes – which deliver high levels of service with respect to user-submitted jobs at the nodes.

We adapted the SkillWorld scenario from a previous model developed for the purposes of social scientific theorizing [12, 13]. The previous 'tag-based' model relied on mean-field mixing (with no population structure) and followed a conventional evolutionary process.

Our belief that the SLAC algorithm works via a kind of group selection occurring at the level of the 'tribe' gave us the a priori expectation that it would select tribes that could perform well in the SkillWorld. In this sense, *dare we claim the beginning of a 'proto-theory'* allowing us to make some modest qualitative predictions? More generally, we claim that this paper demonstrates concretely within a dynamic network the emergence of what has been termed a 'meta-state transition' within evolution [16]. It has been argued that the emergence of life itself and major steps in biological evolution (e.g. multicellular organisms) and social evolution (e.g. large complex societies) occur over such meta-state transitions. In this context we advance our results as possibly of great theoretical insight.

It is important to understand that the concept of *the 'tribe' is actually a theoretical construct* we use to help to explain and understand the *emergent phenomena* produced by the SLAC algorithm over time. The tribes are not 'programmed' into the nodes a priori but rather emerge from the interplay of task domain, interaction and the SLAC algorithm. We use the concept of 'tribes' because we believe it to be valuable in beginning to understand, control and theorize about what is occurring in SLAC networks. However, since the tribes are emergent we do not *begin* with a 'theory of tribes' rather we observe, experiment and induce knowledge about them. As discussed below, this does not preclude, but, in fact, should support the formation of an analytical theory – we hope.

Since the nodes do not die or model genetic operators, the tribe level selection process can be viewed as a kind of artificial *cultural group selection process*. What is quite extraordinary is that *such a simple node level algorithm (SLAC) based on a few plausible assumptions about preferential attachment can lead to such complex and useful group level evolutionary dynamics*.

A key issue, however, is that, although SLAC is simple, to implement the dynamics is complex and currently it is not known how analytical tools can be applied to truly understand, predict and prove the properties of SLAC. So far the only 'proofs' we have are in the form of 'existence proofs' demonstrated by empirical analysis of simulation runs. Such 'proofs' are not watertight and can always be questioned given anomalous results from future simulation studies (rather like experiments in the natural sciences). We have some confidence in the general results from SLAC-like algorithms, however (such as those based purely on 'tags'), since there have been a number of replications of those results from multiple independent implementations using different languages, machines and programmers [17]. However, none of this offers predictive insight into the process as a good analytical model would. What we currently have is a kind of 'toolbox' of algorithmic heuristics that appear to be reasonably robust over some minimal task domains and scenarios.

However, currently, the only way to apply these methods to new domains is to simulate and experiment – copying and adapting heuristics that worked previously in similar domains. Perhaps this is not so far away from the edit/compile/debug cycle of good old-fashioned software engineering (GOFSE). This could bode well for future progress.

[7] Further experiments not detailed here demonstrate that even when all skills in the population are initialized to the same single type – the network quickly adapts into an even skill spread due to mutation on the skills and selection at the tribe level.

Acknowledgements

Thanks go to Mark Jelasity for pointing out some of the recent models that bear close comparison to this one, also, along with Ozalp Babaoglu and Alberto Montresor, for writing clear and readable papers about P2P systems that have helped me in beginning this line of work.

References

1 Handurukande S, Kermarrec A-M, Le Fessant F, Massoulié L: Exploiting semantic clustering in the eDonkey P2P network. 11th ACM SIGOPS European Workshop (SIGOPS), Leuven, 2004.
2 Montresor A: A Robust Protocol for Building Superpeer Overlay Topologies. 4th International Conference on Peer-to-Peer Computing (P2P'04), 2004, pp 202–209.
3 Cohen B: Incentives Build Robustness in BitTorrent (2003). 1st Workshop on the Economics of Peer-2-Peer Systems, Berkley, 2003. Available at http://www.sims.berkeley.edu/ research/conferences/p2pecon/.
4 Trivers R: The evolution of reciprocal altruism. Q Rev Biol 1971; 46: 35–57.
5 Maynard-Smith J: Evolution and the Theory of Games. Cambridge, Cambridge University Press, 1982.
6 Hales D: From Selfish Nodes to Cooperative Networks – Emergent Link Based Incentives in Peer-to-Peer Networks. Proc 4th IEEE International Conference on Peer-to-Peer Computing (P2P2004). Los Alamitos, IEEE Computer Soc Press, 2004.

7 Hales D: Self-Organising, Open and Cooperative P2P Societies – from Tags to Networks. 2nd Workshop on Engineering Self-Organising Applications (ESOA 2004), LNCS 3464, pp. 123–137. Berlin, Springer, 2005.
8 Hales D: Change Your Tags Fast! – a Necessary Condition for Cooperation? Workshop on Multi-Agents and Multi-Agent-Based Simulation (MAMABS'04). Berlin, Springer, 2005.
9 Hales D, Edmonds B: Applying a socially-inspired technique (tags) to improve cooperation in P2P Networks. IEEE Trans Syst Man Cybern A, 2005;35:385–395.
10 Strahilevitz L: Charismatic code, social norms, and the emergence of cooperation on the file-swapping networks. Virginia Law Rev 2003; 89. http://ssrn.com/abstract = 329700.
11 Binmore K: Just Playing, Game Theory and the Social Contract. Cambridge, MIT Press, 1998, vol 2.
12 Hales D: Cooperation without space or memory: tags, groups and the Prisoner's dilemma; in Moss S, Davidsson P (eds): Multi-Agent-Based Simulation. Lecture Notes in Artificial Intelligence. Berlin, Springer, 1979, pp 157–166.
13 Hales D: Evolving specialisation, altruism and group-level optimisation using tags; in Sichman JS, Bousquet F, Davidsson P (eds): Multi-Agent-Based Simulation II. Lecture Notes in Artificial Intelligence. Berlin, Springer, 2002, vol 2581, pp 26–35.
14 Epstein JM, Axtell R: Growing Artificial Societies – Social Science from the Bottom Up. Cambridge, MIT Press, 1996.
15 Axelrod R: A model of the emergence of new political actors; in Gilbert GN, Conte R (eds): Artificial Societies. London, UCL Press, 1995.
16 Heylighen F: Evolution, Selfishness and Cooperation. J Ideas 1992; 2/4: 70–76.
17 Edmonds B, Hales D: Replication, replication and replication – some hard lessons from model alignment. J Artif Soc Soc Simul 2003; 6/4.