Fashioning social simulations into engineering tools – the case of cooperation on P2P networks[1]

David Hales[2] (dave@davidhales.com) and Bruce Edmonds[3] (bruce@cfpm.org)

Abstract

A sequence of simulation models is presented that applies, develops and explores a self-organizing cooperation-producing technique that does not require reciprocity even when subunits behave egotistically (selfishly). The models progress from the domain of social theory (group based cooperation) to an engineering-oriented domain (peer-to-peer file sharing). For each model we analyze the results of extensive computer simulations and draw conclusions concerning the domains of applicability of the technique. Based on these (and other) experiences, we sketch a generalized approach that develops and applies complex emergent techniques to real engineering problems – an approach which puts greater emphasis on hypothesizing and experimentation.

Keywords: Self-Organization, Tags, Peer-to-Peer systems, Networks, Prisoner's Dilemma.

1. Introduction

In this paper we start with some hypotheses that originate from social science. To be precise, the hypotheses are about coordination and cooperation mechanisms discovered in agent-based simulations designed to explore problems suggested in the social sciences. These hypotheses suggest mechanisms for the coordination of peer-to-peer (P2P) networks. These hypotheses are tested in a sequence of simulations from abstract to more applied. The results of these experiments suggest that these mechanisms could be successfully applied within implemented P2P protocols, but they require more testing and experimentation before they could be applied. The earlier, more abstract simulations are important because they embody the hypotheses which the later, more concrete, simulations rely upon. In this sense the more abstract simulations act as the theory for the later simulations.

The paper proceeds as follows. We start by explaining a little of the context in terms of agent-based social simulation and the motivation behind the application in P2P networks, namely the 'tragedy of the commons' that can occur in P2P networks. The main body of the paper is a description of a sequence of three models and their results: the 'TagWorld' to show the basic mechanisms of 'tags' that is used to attack

[1] This work partially supported by the EU within the 6th Framework Programme under contract 001907 (DELIS).
[2] Dept. of Computer Science, University Of Bologna, Italy.
[3] Centre for Policy Modelling, Manchester Metropolitan University, UK.

the target problem; the ‘NetWorld’ model which shows how the basic approach is transferred to that of P2P networks; and the ‘FileWorld’ model which is a more realistic model of P2P file sharing. The paper ends with a general discussion including related work and some tentative methodological conclusions.

1.1 Agent-based social simulation

Social Scientists have always been interested in complex, self-organizing, emergent systems because their target is naturally occurring societies that display these properties. Historically social theoreticians had to rely on “macro-scale” mathematical models in which a few aggregate variables were related. Although not explicitly included, the idea was that, these simplified models represented the aggregation of individual (micro) behaviors based on some, generally un-stated, individual and local behavior rules.

One major problem with such models is that the relationship between the micro and the macro behaviors is often opaque. This is evidenced by references to unspecified guiding processes such as “hidden hands” or universal behavioral assumptions such as “universal rationality”[4]. Such models don’t *model* emergence so much as *assume it* without specifying precisely the individual behaviors that would bring it about. Consequently much work within this area is of little use for the engineer who wants to build self-organizing systems. They need to know explicitly how to program the subunits in their system.

More recently, through the use of increasingly available computational resources, Social Scientists and Computer Scientists have been working together to explore the relationships between micro behaviors and macro emergent properties using “simulated social worlds” comprising many individuals (or agents) following their own individual behavior rules (or programs). Such work is often termed Agent-Based Simulation (ABS), Multi-Agent-Based Simulation (MABS) or Artificial Societies [8, 9, 10, 15, 18, 25]. These areas have already produced a wealth of published computational models of complex, self-organizing, emergent systems. The general approach is to reverse engineer some property of interest that is believed to exist in naturally occurring societies in order to understand it (i.e. not so much to just *implement* but to *augment, test and develop* social theories “*in silico*”[5]). By explicitly modeling the agent level behaviors and

[4] It should be stated that authors often state that such “shortcuts” don’t indicate real processes but stand-in for some kind of self-correcting underlying process of heterogeneous individual and local behavioral rules.
[5] A term often used to underline that this stage and form of theory development is not empirical in the sense of comparing results to a naturally occurring target societies – though that may happen at a later stage.

observing the resultant social level behaviors there is no need to assume *a priori* what kinds of emergent properties the system will have – these can be obtained by observation of the model output[6].

Here then, it would seem, is a developing body of work that can be used by engineers of self-organizing systems. Isn't it, then, simply a matter of dipping into the literature, selecting a relevant technique or model and deploying that technique within an application domain? Unfortunately the answer to this is a resounding *no!* There are a number of issues causing problems. Firstly, the models and techniques presented in this literature are not oriented towards engineering applications - the scenarios given are often highly abstract (e.g. with populations interacting in mean-fields and / or interacting in very limited ways). The domains of application are influenced by social scientific questions of *understanding* rather than engineering questions of *performance and robustness*. Finally, it is rare that such work can offer conclusive or even suggestive evidence for necessary (or even sufficient) conditions for the presented phenomena of interest. Generally the work presents a set of existence proofs for given stated parameters that demonstrate some phenomena. In some cases we may get a sensitivity analysis in which we can see a range of parameter values supporting the phenomena. On the whole however, because of the high dimensionality of even the simplest of such models, exhaustive sensitive analysis is not feasible. So, to summarize, we have three main problems in applying such models: domain of application – i.e. *do something useful*; the aim of application – i.e. *make it work well* and the conditions of application – *i.e. make it work over a range of plausible conditions*. Tackling each of these will require significant modification of, and experimentation with, existing ABS techniques when attempting to use it for engineering real working systems.

Or basic approach is to move, via a succession of models, from an initial abstract model to the actual application by introducing piecemeal refinements, while preserving the phenomena of interest. In addition to discovering interesting new phenomena along the way, this approach is cumulative, and, given intermediate models are published, collaborative – allowing different researchers to build on, refine and check existing work.

In this paper we take a previously published model that demonstrates cooperation without reciprocity (initially in a Prisoners Dilemma (PD) scenario without space) and develop three further models that move the cooperation mechanisms closer to an application domain (peer-to-peer file sharing). We do not claim

[6] When relying on the output of a model in this way (i.e. without *a priori* specification of what will come out) it is best to align and compare a number of similar models in order to be more certain that the results are not an artifact of some apparently minor implementation issue (or even just a plain bug). We have argued this elsewhere [7].

that we have reached a level where a real application could be deployed yet and we outline the open issues and next stages that will inform a possible next set of models towards this goal.

1.2 The application problem – the 'tragedy of the commons' in P2P networks

In order to put our succession of models into context we outline here the existing initial model and our desired target application. In this section we will not give a detailed specification of either (see later sections) but rather the reasons for choosing these points of entry and exit and the general requirements of the translation process from one to the other.

We begin by observing that P2P file sharing systems using unstructured overlay type networks have become very popular over recent years [1, 11]. At any one time millions of users (peers) may be running, connected over the Internet. To a certain extent these applications are self-organizing in that peer software is provided freely for downloading with users deciding when to download and execute these peers on their hardware. In this sense there is no centralized administration or control and the systems are essentially open, there is nothing stopping users from modifying and releasing new updated peer software that can then be executed and join the system. Also users can choose to either share nothing or only low quality files. Indeed, organizations that perceive a threat from the free distribution of files[7] may purposely flood such networks with low quality, erroneous or poor quality files. Given these latter considerations and empirical evidence [1] it can be shown that a major problem in such applications is to develop mechanisms that discourage selfish behavior and encourage altruistic behavior. This would mean, somehow, limiting the effect of selfish or malicious peers and encouraging good behavior in the form of sharing high quality files.

The problem of limiting the effects of "commons tragedies" [16] in P2P systems is common to many possible applications (such as the sharing of processing power, passing of messages and performing remote operations) so attempting to apply new techniques would appear to be a worthwhile enterprise.

We take a technique from the ABS literature that appears to solve a general commons tragedy and progressively modify the domain, aim and conditions of the given model until we have something nearer to a file-sharing scenario.

2. The TagWorld Model – How tags work

We begin with a model given by Hales [12] that we will briefly summarize here. The model presents a mechanism that generates high levels of cooperation among selfish bloundedly rational optimizers playing

[7] Such organisations might include traditional content distributors or regional censorship authorities.

single-round games of the Prisoner's Dilemma game (PD). It is claimed that the single round PD game captures certain kinds of interactions in real social worlds. The employed mechanism – the "tag" – structures the population into competing groups that emerge and dissolve, over time, as part of the evolutionary process. In the remainder of this section, firstly, we brief introduce the PD game, then the "tag" concept, then the model and results.

2.1 The Prisoner's Dilemma

The Prisoner's Dilemma (PD) game captures a scenario in which there is a contradiction between collective and self-interest. Two players interact by selecting one of two choices: Either to "cooperate" (C) or "defect" (D). For the four possible outcomes of the game players receive specified payoffs. Both players receive a reward payoff (R) and a punishment payoff (P) for mutual cooperation and mutual defection respectively. However, when individuals select different moves, differential payoffs of temptation (T) and sucker (S) are awarded to the defector and the cooperator respectively. Assuming that neither player can know in advance which move the other will make and wishes the maximize her own payoff, the dilemma is evident in the ranking of payoffs: $T > R > P > S$ and the constraint that $2R > T + S$. Although both players would prefer T, only one can attain it. No player wants S. No matter what the other player does, by selecting a D move a player ensures she gets either a better or equal payoff to her partner. In this sense a D move can't be bettered since playing D ensures that the defector cannot be suckered. This is the so-called "Nash" equilibrium for the single round game. It is also an evolutionary stable strategy for a population of randomly paired individuals playing the game where reproduction fitness is based on payoff. So the dilemma is that if both individuals selected a cooperative move they would both be better off but both evolutionary pressure and game theoretical "rationality" result in a tendency towards defection. The PD game was extensively investigated in [2].

2.2 Tags

Tags are markings or social cues that are attached to individuals (agents) and are observable by others [17]. Often represented in models by a single number or a bit string; they evolve like any other trait in a given evolutionary model. The key point is that the tags have no direct behavioral implication for the agents that carry them. But through indirect effects, such as the restriction of interaction to those with the same tag value, they can evolve from initially random values into complex ever changing patterns that serve to structure interactions. A number of tag models have been applied to social dilemma type scenarios [14, 24, 26] but only the TagWorld [12] model appears to produce cooperation in the single-round PD game (i.e. interactions with strangers) and we envisage this would be something that would be beneficial in our target P2P application.

2.3 Description of TagWorld

In [12] a model is presented of agents playing the PD in pairs in a population with no topological structure (other than tag based biasing of interaction). The model is composed of very simple agents. Each agent is represented by a small string of bits. On-going interaction involves pairs of randomly selected agents playing a single round of PD. Agent bits are initialized at random. One bit is designated as the PD strategy bit: agents possessing a “1” bit play C but those possessing a “0” bit play D. The other bits represent the agents’ tag. These bits that have no direct effect on the PD strategy selected by the agent but they are observable by all other agents. Below is an outline of the simulation algorithm used:

```
LOOP some number of generations
    LOOP for each agent (a) in the population
        Select a game partner agent (b) with the same tag (if possible)
        Agent (a) and (b) invoke their strategies and get appropriate payoff
    END LOOP
    Reproduce agents in proportion to their average payoff
    Apply, with low probability, mutation to tag and strategy of each reproduced agent
END LOOP
```

Figure 1. Pseudo code algorithm for main loop of the model

Agents are selected to play a single-round of PD not randomly but based on having the same tag string. If an agent can find another with the same tag string it will play PD against that agent. If it cannot then it plays against some randomly chosen partner. Agents are reproduced probabilistically in proportion to average payoff they received (using roulette wheel selection).

2.4 Results from TagWorld

Extensive experimentation varying a number of parameters showed that if the number of tag bits is high enough (we found 32 tag bits for a population of 100 agents to be sufficient with a mutation rate of

0.001 and PD payoffs of T=1.9, R=1, P=S=0.0001[8]) then high levels of cooperation quickly predominated in the population[9].

More interesting still, if all the agents are initially set to select action D (as opposed to randomly set) then the time required to achieve a system where C actions predominate is found to monotonically decrease as population size increases. This is an inverse scaling phenomena: *the more agents, the better*. Additionally the fact that the system can recover from a state of total D actions to almost total C actions (under conditions of constant mutation) demonstrates high robustness. The tag-based model produces an efficient, scalable and robust solution – based on very simple individual learning methods (modeled as reproduction and mutation).

2.5 How Tags Work

We have described this model because it seems to offer up a method for achieving three important properties in a simple (PD) task domain: efficiency, scalability and robustness. But how do tags produce this result? The key to understanding the tag process is to realize that agents with the identical tags can be seen as forming a sort of "interaction group". This means that the population can be considered to be partitioned into a collection of groups. If a group happens to be entirely composed of agents selecting action C (a cooperative group) then the agents within the group will outperform agents in a group composed entirely of agents selecting action D (a selfish group). This means that individuals in cooperative groups will tend to reproduce more than those in selfish groups. However, if an agent happens to select action D within a cooperative group then it will individually outperform any C acting agent in that group and, initially at least, any other C acting agent in the population – remember the T payoff is 1.9 but the best a C acting agent can do is R = 1.

However, by others copying such an agent (i.e. the agent reproducing copies of itself) the group becomes very quickly dominated by D acting agents and therefore the relative advantage of the lone D acting agent is lost – the group *snuffs itself out* due to the interaction being kept within the group. So by selecting the D action an agent destroys its group very quickly (remember groups are agents all sharing an identical tag). Figure 2 visualizes this group process in a typical single run. Each line on the vertical axis represents a unique tag string. Groups composed of all C action agents are shown in light gray (Coop), mixed groups of C and D agents are dark gray and groups composed of all D are black.

[8] P and S were set to the same small value for simplicity. If a small value is added to P (enforcing T > R > P > S) results are not significantly changed.
[9] If tags are removed from the model and pairing for game playing is completely random then the population quickly goes to complete defection (the Nash equilibrium for the single-round PD).

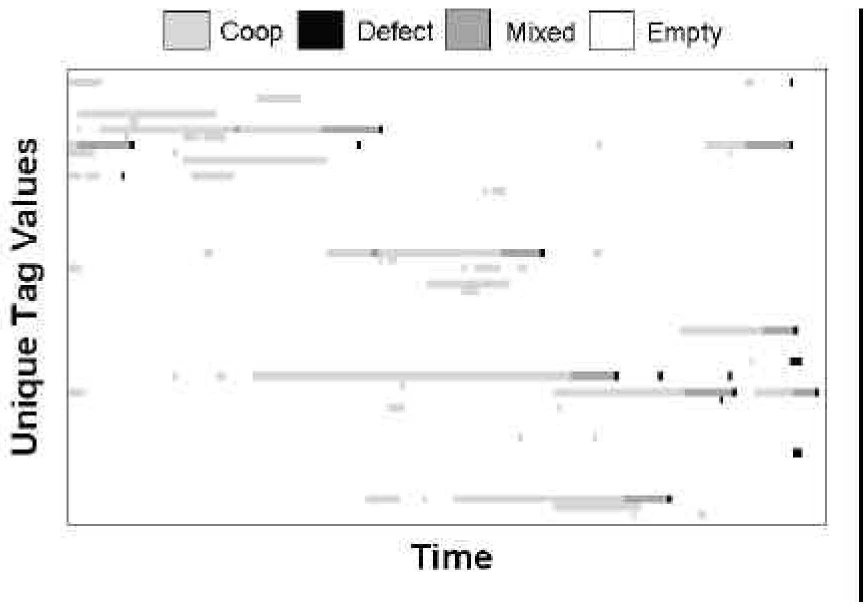


Figure 2. Visualization of 200 cycles (generations) from a single simulation run showing cooperative groups coming into and going out of existence. See the text for an explanation

2.6 TagWorld Discussion

The results indicate that by partitioning the population into distinct "interaction groups" tags facilitate a kind of "group selective" process in which defection is kept in check. However, from this model it is unclear how important the binary string form of the tags is – does the large string (remember L = 32 for high cooperation) impose some kind of necessary topological structure? Previous experiments with the model in which the tag was replaced by a single value (a real or integer number) produced low cooperation so it appeared that the bit string structure was important. However, recently, we discovered that the significant factor was differential mutation between tag and strategy. An artifact of a long tag string is that (given mutation is applied with equal probability to each bit) the mutation rate is effectively increased significantly. Mutation applied to the tag as a whole is $1-(1-m)^{L} \sim 0.0315$ (where m = 0.001 and L = 32). This means that the effective mutation rate on the tag is well over one order of magnitude higher than on the strategy.

In order to test if differential mutation alone produced the high cooperation we re-implemented the model but replaced the binary tags with a single real number and increased mutation on the tag by a factor (f) relative to the strategy bit [13]. We experimented with different values for f and found that indeed a high f produced high cooperation whereas a low f leads to low cooperation. Results are shown in Figure 3. Notice that cooperation comes in three phases – a low phase with $f < 4$, a high phase with high $f > 6$, and an uncertain phase between the two, where the populations can go into either high or low cooperative regimes depending on initial conditions and on-going stocasticists (i.e. different numbers from the pseudo random number generator in each run).

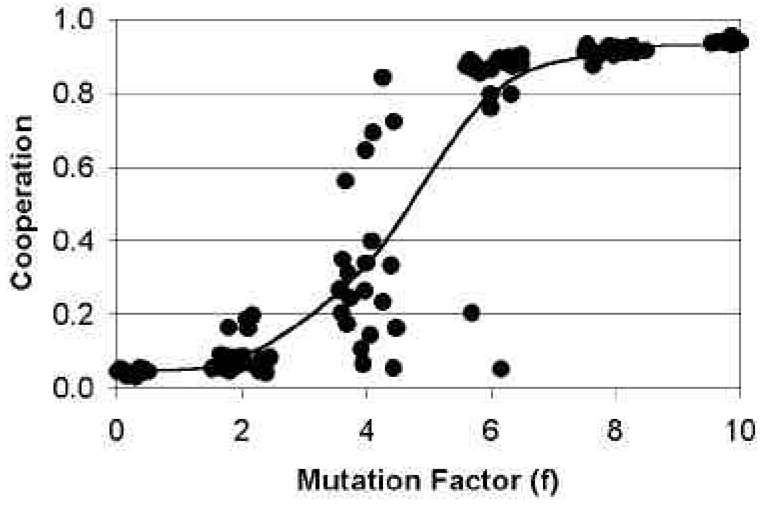


Figure 3. Results from each simulation run plotting mutation factor (f) against cooperation. There are 20 runs per f value (slight noise has been added to the x-axis for readability). The solid line represents a smoothed average curve through the runs.

Minimally these results indicate that there is nothing special about binary strings of tags – rather it is differential mutation that is important. The tag needs to change faster than the strategy. Intuitively this makes some sense, the mechanism of cooperation is driven by cooperative groups forming more quickly than defectors can invade and destroy them. With high mutation on the tag, a cooperative group, as it grows, will tend to spawn more new cooperative groups. A group containing defectors does not grow and quickly deflates and dies (as discussed above).

We wish to move these desirable cooperative properties into a scenario that is closer to our target P2P file-sharing application. We do this by implementing a new model.

3. The NetWorld Model - from tags to networks

We now consider how to translate the cooperation producing tag mechanism into a network (as a first step towards our target file sharing P2P application). We represent the network as an undirected graph in which each vertex represents a node and each edge represents a link between nodes. We assume each node has a fixed capacity (a maximum degree) of links defining its neighbors (a neighbor list).

The underlying mechanism driving cooperation within the TagWorld simulation is the formation and dissolution of sharply delineated groups of agents (identified by sharing the same tag). Each agent could locate group members from the entire population. Each member of the group had an equiprobable chance of interacting with any agent in the population sharing the same tag. In this sense each agent could determine which agents were in their group.

If we assume a sparse P2P network in which each node (peer) knows of some small number of other nodes (neighbors) and those neighborhoods are highly interconnected (clustered) such that most neighbors share a large proportion of other neighbors then we have something similar to our tag-like groupings. Instead of a tag (a marker) we have an explicit list of neighbors. In a highly clustered network the same list will be shared by most of the neighborhood. In this sense you can visualize the table of known peers (the neighbor list) stored in each agent as the tag.   It is shared by the group and is the key by which the group can directly interact with each other. To this extent it defines a group boundary. A nice feature of this way of defining group boundaries is it is a kind of watertight method of isolating nodes into their neighborhoods since a node cannot directly interact with another node that it does not know of.

Initially we will consider only direct interactions between neighbors in our network model. In a sense this is all that can ever happen in P2P systems. Indirect interactions between nodes that do not share neighborhoods have to mediate by direct interactions between intermediate nodes. Essentially, one can view all interaction as with neighbors (even if those neighbors are actually proxies for other more remote nodes). If cooperation can be established between the majority of neighborhoods in a network then it follows that any pair of nodes in the network that are connected will have a good chance of being able to find a *path of cooperation* through the network[10].

In order to capture this kind of neighborhood interaction in the simplest possible way we have each node in the network play a single round of PD (see above) with a randomly chosen neighbor. No information is stored or communicated about past interactions and the topology is not fixed (see below).

3.1 The Evolutionary Rewiring Algorithm (ERA)

In the TagWorld model change was produced over time by mutation and differential reproduction based on average payoff. How can these be translated into the network? In our network model we do not view nodes

[10] We did some limited experiments in task domains requiring long paths of cooperators and found this was a difficult task for the network – see later discussion.

as “reproducing” in a biological or cultural sense[11]. However, it is consistent with or initial assumptions that nodes may relocate to a new neighborhood in which a node is performing better than itself. That is, we assume that periodically nodes make a comparison of their performance against another node randomly chosen from the network[12]. Suppose node (i) compares itself to (j). If (j) has a higher average payoff than (i) then (i) disregards its neighbor list and copies the strategy and neighbor list of (j) also adding (j) into the list. This process of copying can be visualized as movement of the node into the new neighborhood that appears more desirable.

Mutation in the TagWord model was applied after reproduction. Each bit of the tag and the strategy was mutated (flipped) with low probability. Since we are using the same one bit strategy we can apply mutation to the strategy in the same way. We therefore flip the strategy bit of a node with low probability immediately after reproduction (the movement to a new neighborhood as described above). Since we treat the list of neighbors in each node as the tag a mutation operation implies changing the list in some way. But we can’t simply randomly change the list; we need to change the list in such a way as to produce an effect with closely follows what happens when mutation is applied in the tag model. In that model, tag mutation tended to give agents unique tags – i.e. tags not shared by other agents at that time. However, in the model agents could interact with a randomly chosen agent with non-matching tags if none existed with identical tags. In this way tag mutation lead to the founding of new tag groups. In the network model we don’t want to isolate the node completely from the network otherwise it will not be able to interact at all. However, we don’t want to move into an existing neighborhood (as with reproduction) but rather to do something that may initiate the founding of a new neighborhood. So we pragmatically express tag mutation as the replacement of the existing neighbor list with a single neighbor drawn at random from the network.

We now have our analogues of reproduction and mutation for the network model. Reproduction involves the nodes copying the neighbor lists and strategies of others obtaining higher average scores. Mutation involves flipping the strategy with low probability and replacing the neighbor list with a single randomly chosen node with a low probability Figure 4.

Perhaps the biggest shift we have made here is in the group-level topology. In the TagWorld model groups were distinct (those sharing the same tag) and interactions was strictly within groups. Here, groups may overlap since neighboring peers will not necessarily share the same neighbors. In this sense the group

[11] The interpretations placed on most of the previous tag models have been cultural or biological.
[12] Currently we do not model the process of finding this “out-group” node. We assume that the network could provide the service but this might be a problem – see later discussion.

boundaries are no longer absolute. This change could be significant and may destroy the cooperation process. For example, although the model discussed in [24, 26] has potentially overlapping and changeable boundaries and appears to encourage altruism – it in fact is not able to solve a social dilemma [7]. In the next section we outline results from a network model - NetWorld[13] - we find that although significant properties of the cooperation process are changed, scalable, robust and high cooperation is produced.

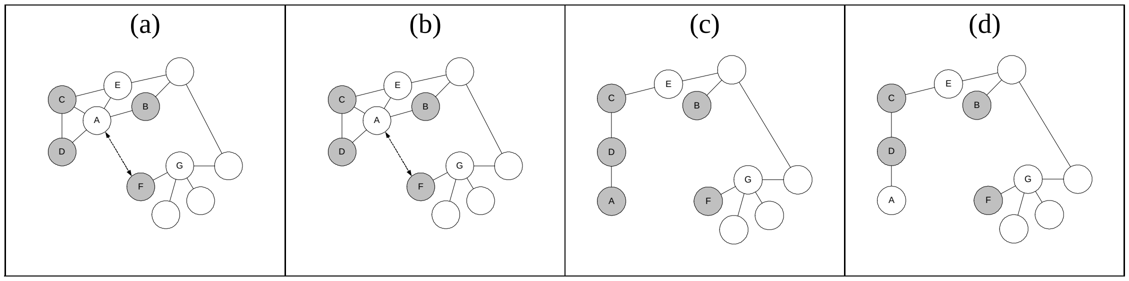


Figure 4. An illustration of "replication" and "mutation" as applied in the Evolutionary Rewiring Algorithm (ERA). Shading of nodes represents strategy. In (a) the arrowed link represents a comparison of utility between A and F. Assuming F has higher utility then (b) shows the state of the network after A copies F's links and strategy and links to F. A possible result of applying mutation to A's links is shown in (c) and the strategy is mutated in (d).

3.2 Description of NetWord

The NetWorld model is composed of a set N of nodes (or peers). Each node stores a list of other nodes it knows about (we term this the neighbor list). In addition to the neighbor list each node stores a single strategy bit indicating if it is to cooperate or defect in a single round game of the PD. Neither the strategy bit nor the list is normally visible to other nodes. Initially nodes are allocated a small number of neighbors randomly from the population. Periodically each node selects a neighbor at random from its list and plays a game of PD with it. Each node plays the strategy indicated by its strategy bit. After a game the relevant payoffs are distributed to each agent. Periodically pairs of randomly chosen nodes (i, j) compare average payoffs. If one node has a lower payoff then the strategy and neighbor list from the other node is copied (effectively moving the lower scoring node to a new neighborhood). Mutation is applied to the strategy with probability m and to the neighbor table with probability mf[14]. Mutation of the strategy involves flipping the

[13] There are many other ways tags could be translated into networks. For example, agents could move around the network between nodes carrying tags or agents sharing a node could be seen as sharing a tag. We hope to explore some of these variations in the future.
[14] In all cases given here f = 10. So the mutation on the neighbor table is one order of magnitude higher than on the strategy. We take this from our previous model – see discussion later concerning the effect of *f* in this model

bit. Mutation of the neighbor list involves clearing the list and replacing it with a single randomly chosen node from the population. Figure 5 gives an outline pseudo-code algorithm of the NetWorld simulation main loop.

The neighbor lists are limited in size to a small number of entries. The entries are symmetrical between neighbors (i.e. if node (i) has an entry for node (j) in its list then node (j) will have node (i) in its list). If a link is made to a node that has a full neighbor list then it discards a randomly chosen neighbor link in order to make space for the new link. Also if a node is found to have no neighbors when attempting to play a game of PD (this can happen if neighbors have moved away) then a randomly chosen node is made a neighbor by being added to the neighbor list. The neighbor lists are limited in size to a small number of entries. The entries are symmetrical between neighbors (i.e. if node (i) has an entry for node (j) in its list then node (j) will have node (i) in its list). If a link is made to a node that has a full neighbor list then it discards a randomly chosen neighbor link in order to make space for the new link. Also if a node is found to have no neighbors when attempting to play a game of PD (this can happen if neighbors have moved away) then a randomly chosen node is made a neighbor.

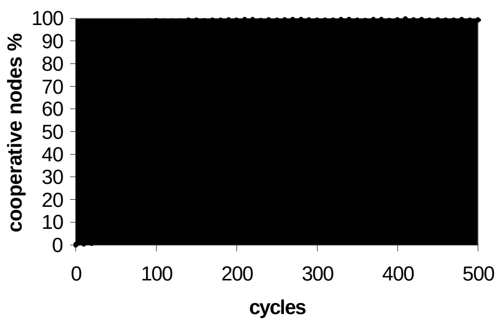
```
LOOP some number of generations
    LOOP for each node (i) in the population
        Select a game partner node (j) randomly from neighbor list (nl)
        Node i and j invoke their strategies and get appropriate payoff
    END LOOP
    Apply the Evolutionary Rewiring Algorithm:
    Select N/2 random pairs of nodes (i, j) from the population
    Copy neighbor list (nl) and strategy (s) of higher scoring node into lower scoring node
    Apply mutation to nl and s of each reproduced node with probability mf and m
END LOOP
```

Figure 5. Outline pseudo-code algorithm of the NetWorld simulation main loop

3.3 Results from NetWord

Figure 6 gives some typical results (up to $N = 2 \times 10^4$). In these experiments the mutation rate was $m = 0.001$ (increased by a factor of $f = 10$ when applied to mutation of the links) and the PD payoffs were as per the previously described TagWorld model (see above). Similar results were obtained up to $N = 10^6$.

(a)



(b)

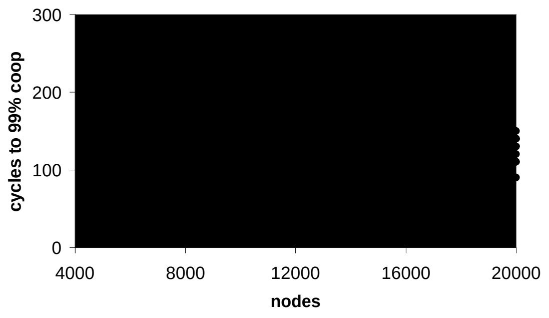


Figure 6. Charts show results from NetWorld the model. The (a) shows a typical time series (N = 10,000 nodes) giving the % of cooperative nodes. Chart (b) shows the number of cycles before 99% of nodes are cooperative for various values of N. Each dot is an individual run.

Notice that although there appears to be *no scaling cost* (with convergence to high cooperation taking approximately the same number of runs for different N) *we have lost the reverse scaling cost property* observed in the TagWorld model. Interestingly, when the mutation factor (f) was decreased to 1 (making mutation rates equal for links and strategy) high cooperation *still occurred* but the time to reach it did not scale – there was what appeared to be an exponentially increasing non-linear upper bound scaling cost (in time) meaning that some runs were not converging after thousands of cycles for large N (>10,000)[15]. We still therefore need the high mutation rate on the links to produce a well scaling mechanism.

3.4 NetWorld discussion

The PD task domain although useful is a rather impoverished. As an initial proof of concept it shows that at least some kinds of social dilemma can be solved. But the behaviors (PD strategies) and coordination required is trivial - although the dilemma itself is *not* trivial. Following our “method” of making only piecemeal changes, while preserving the desirable emergent properties, we now apply the same basic NetWorld algorithm in a new model which implements a simplified P2P file sharing scenario. In the next section we present results from the FileWorld model – we find that we can indeed use the basic NetWorld technique to produce non-selfish behavior from nodes acting as bounded optimizers in a file sharing scenario but we also note a number of important issues that moving from the PD game to something more realistic raises.

[15] The way that these scaling costs have changed over the TagWorld model appear to offer, via further analysis, a deeper understanding of the mechanisms effect scaling costs, however we leave further discussion of this to later work.

4. The FileWorld Model – tags in a p2p structure

In this section we outline the results of applying the evolutionary node-rewiring algorithm used in NetWorld in a simulated P2P file-sharing scenario. The scenario is a simplified form of that given in [29]. It models a flood-fill query process where nodes periodically generate and forward queries to their neighbors. Neighbors either process the queries (by producing a hit or forwarding it to all their neighbors) or ignore them depending on their capacities and internal state variables.

Each node (*i*) is defined by three state variables: an answering power *Ai*, a questioning power *Pi* and a capacity *Ci*. Both *Ai* and *Pi* are in the range [0..1] and *Ci* takes some cardinal value greater than zero. The values of each variable quantifies the behavior of a node over some unit of time *t*. *Ci* indicates the capacity of the node given as total number of queries. When a node generates or answers a query this takes one unit of capacity. *Pi* givens the proportion of the capacity *Ci* that will be allocated to *generating* new queries. Conversely, *1 – Pi* of the capacity will be allocated to *answering* queries from other nodes. The answering capacity *Ai* gives a probability that a node can directly match a query (producing a hit). It represents indirectly the amount and quality of files served by the node. For the experiments given here all nodes have fixed values of *Ai* = 0.4 and *Ci* = 100 but we allow *Pi* to be adapted by the node (see below).

Over a single time period each node i may process a total of *Ci* queries. This capacity is divided between generating *Pi* • *Ci* new queries (passed to neighbor nodes) and reserving enough capacity to process (1-*Pi*) • *Ci* queries from neighbors. *Pi* therefore represents a measure of selfishness. If *Pi* = 1 then node *i* uses all its capacity to generate new queries - ignoring queries from neighbors. If *Pi* = 0 then *i* uses all its capacity processing queries from neighbors - generating none itself.

4.1 Outline of a simulated time period

In a simulated time period, *C* • *N* nodes (where *N* is the number of nodes in the population) are selected randomly from the population, with replacement, and "fired". If a fired node still has capacity to generate queries it generates one query and passes this to its neighbors otherwise the node takes no action. When a node (*i*) receives a query, if it has spare capacity, it processes the query. With probability *Ai* a "hit" is produced for the query. If no hit is produced the query is passed to the neighbor nodes of *i*. If a node has no capacity left to process a query it is ignored – no action is taken and the query is not passed on. Queries are not passed on indefinitely but have a preset "time-to-live" (*TTL*) after which they are ignored by all nodes. In all experiments presented here *TTL* = 3 - meaning that queries never get more than a maximum of 3 nodes depth from the originating node. The process of firing nodes in random order with replacement introduces noise in the form of some nodes firing more often than others and some nodes not being able to generate their full quota of queries. We view this as reasonable since it introduces realistic kinds of noise such as non-synchronized nodes with differential processing speeds etc.

4.2 Application of the evolutionary rewiring algorithm (ERA)

After each time period (that is after $N \cdot C$ node firings) the evolutionary rewiring algorithm (ERA), as used in NetWorld above, is applied. N / 2 pairs of nodes (i, j) are selected from the population at random with replacement. If $Ui > Uj$ then node j drops all existing links and copies node i's links and additionally linking to node i itself. Also Pj is set to Pi (copying the query handling behavior of i). If $Ui < Uj$ then the mirror process is performed (i copying j). In the case $Ui = Uj$ then a randomly selected node (i or j) is designated "winner" and the process proceeds as if that node had a higher U value.

For the experiments present here we used a utility value equal to the *total number of hits* obtained by a node in the time period. Obviously this would tend to be higher if P was higher (generating more queries). We used the utility value of total hits (per node) since this gives an apparent incentive for freeloading. If *average hits* per query is used there is no commons tragedy – because nodes wont generally increase their utility by performing more queries.

After any node i copies another node j it applies mutation, with low probability, to the links and the Pi value. With probability m, Pi is changed to a random value selected uniformly from the range [0..1]. With probability $10m$ the links from i are removed and replaced with a single link to a randomly chosen node from the population (see above).

4.3 Description of FileWorld

For the purposes of simulation we represent the network as an undirected graph in which the degree of any node is fixed at a maximum value (20 in all cases). When any operation requires a further link from a node, the node simply deletes a randomly chosen existing link and continues with the new link operation. We experimented with various initial topologies for the graph, including randomly connected, lattice, "small world" and completely disconnected. All produced similar results to those presented here. We also experimented with different initial P values. Again we found we obtained similar results (even when all P values are initially set to zero – see later). The results given (unless otherwise stated) start with initially random graphs and randomly selected P values.

4.4 Results from FileWorld

In order to gain a baseline benchmark, that measures how the network behaves without the application of the evolutionary re-wiring algorithm, we ran 10 trials for 10 cycles on static networks with randomly initialized topologies and *P* values. We did this for a number of network sizes *N* = [200..51200]. All other values were kept as previously described. Since in the static case nothing changes, the averaging over 10 cycles is done simply to smooth out the stocasticities of the model. Averaging over 10 different trials (with unique pseudo-random number seeds) averages over the different initial random network topologies and *P* values.

We considered the following two measures for benchmarking: the average number of queries generated per node in a cycle (*nq*) and the average number of hits per node generated per cycle (*nh*). We found that the baseline level for these measures was with low variance *nq* = 49.45 and *nh* = 20.13 in all cases. Calculating *nh / nq* gives an average hit rate per query generated = 0.41. We might expect *nq* = 0.5 since the *P* values are selected uniformly randomly but this slightly lower value is a result of the (random selection with replacement) method of firing nodes as described earlier.

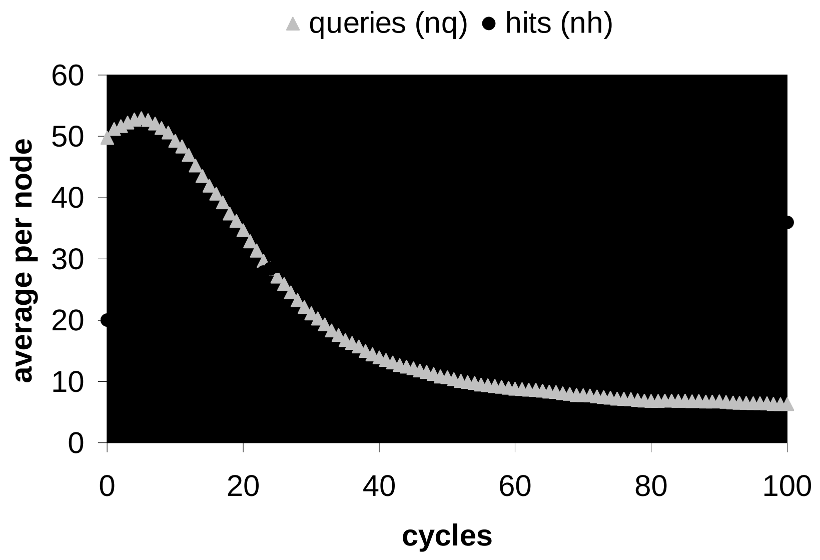
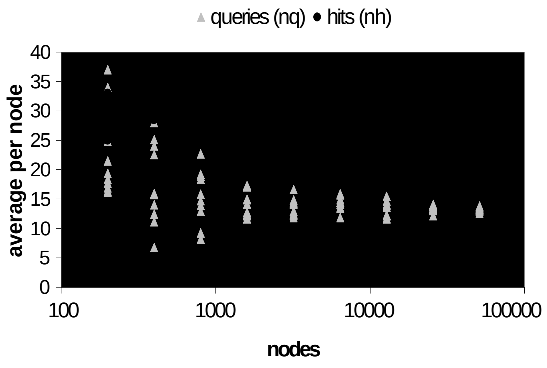


Figure 7. A typical run for a 10^4 node network. Notice that *nq* and *nh* values initially get worse than the baseline values (*nq* ˜ 50, *nh* ˜ 20) but then quickly improve.

Given these baseline values for *nq* and *nh* we can investigate the effect of applying the evolutionary rewiring algorithm (ERA). If results give a consistently higher number of hits (*nh*) by keeping the number of queries generated (*nq*) low then ERA is suppressing the self-interest of the nodes and thus benefiting of the network as a whole. Figure 7 shows a time series for a typical run for a network of size N=10^4 with ERA enabled. As can be seen, over time, *nq* decreases and *nh* increases. Notice also that initially these values move in the opposite direction, indicating an initial favoring of selfish behavior, but this is soon corrected. This shows that the evolutionary process (forming cooperative groups within the network) takes a few cycles to get started from the initially randomly initialized network.

Figure 8a shows *nq* and *nh* measures averaged over cycles (40-50) for different network sizes (with 10 independent runs for each network size). Notice that as size of the network increases the variance of the individual runs decreases – indicating that larger networks are less sensitive to on-going stocasticities and initial conditions, since we are using average values this appears intuitive. Figure 8b shows the numbers of cycles before high hit values are attained (when *nh* > 30). Again 10 independent runs are shown for various network sizes. As before the variance of results decreases as network size increases and there is no significant increase in the number of cycles required for larger networks - suggesting that ERA scales well in this scenario also (i.e. there is no cost for larger networks).

(a)



(b)

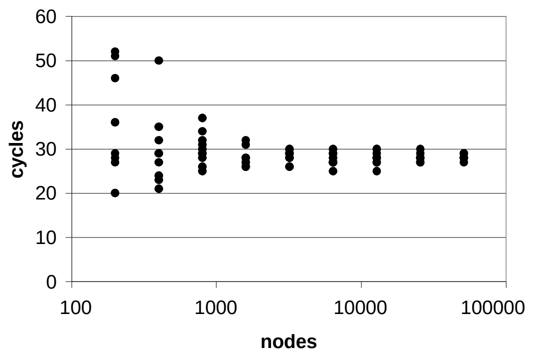


Figure 8. Chart (a) shows the values of *nq* and *nh* (averaged over cycles 40..50) for different network sizes. Chart (b) shows cycles required before high hit values (*nh* > 30) are attained. In both charts there are 10 independent runs for each for each network size.

4.5 FileWorld discussion

Our initial experiments with the FileWorld model suggest that ERA *does indeed control the self-interest of the nodes* by keeping down the number of queries generated and hence increasing total hits. We still do not fined a reverse scaling property (as we did in TagWorld) but we appear to have preserved the zero scaling cost properties from the NetWorld model. This implies that the ERA has at least some general applicability – *without changing the basic algorithm for the domain* it performed well.

Although the ERA performs well in the FileWorld domain, this is too easy a problem for developing a functional file sharing system. By fixing the answering power of each node to a relatively high level (40% of queries can be answered) we create a rather beneficial "file rich" environment. It would be of interest to add a cost to nodes having high answering powers (diminishing utility in some way) and allow them to adapt it in the same way as their links and questioning power. Under those conditions would the system still produce high hit rates? Also we have not explored the kinds of topologies produced in the NetWorld or FileWorld domain. However, in both these scenarios a network disconnected into a number of components will still produce respectable functionality (since long range routing is not a requirement of these task domains)[16]. We need to refine the task domain to include long range routing between distant nodes.

5. Concluding Discussion

5.1 Related work

The mechanism we apply is distributed; each node only has to concern itself with its own interactions with those it is connected with. In this way it contrasts to centralized systems of trust such as [19].

ERA bears similarities to the more complex "SLIC" algorithm [29]. However, in that work (from which we adapted our FileWorld task scenario) "incentive structures" are explicitly programmed, with each node monitoring the service it receives from its neighbors and updating weights which moderate the future service it offers to others. There is therefore explicit retaliation programmed into the model (as applied in [4] - an actual deployed and working application). For the model presented in this paper the incentives effectively *emerge* from the dynamic behavior of the nodes (moving in the network) rather than being explicitly programmed in. Nodes therefore do not need to monitor or store the performance of others – reducing overheads. Also in [29] simulations are only applied to scenarios in which single "probe nodes" behave selfishly – nodes do not adapt their behavior to increase their utilities network wide. Consequently it is not clear how the model would react when *all nodes* are acting selfishly rather than just a small number (however, this is mentioned as future work).

[16] Our initial investigation of NetWorld topologies shows that indeed networks tend to form into disconnected componants that constantly grow and reform (rather like the tag groups in TagWorld). Interestingly, since the network is in constant flux it would appear that forms of "temporal routing" could be applied for long-range task domains [21].

Previous models inspired by game theoretical approaches offer some similar insights [22] but in this work network topology is not explicitly modeled and the strategies relay on repeated game strategies (i.e. tit-for-tat in the Iterated PD [30]). Such iterated strategies require on-going interactions with recognizable individuals – our model does not rely on this since it is based on mechanisms that work well in the single-round game.

Our model bears comparison to the social simulation of leadership dynamics presented in [31]. However, this model is deterministic and relies on agents knowing the strategies of others when moving. Never-the-less, agents in the model move around a social network, making and breaking links based on self-interest and play the single round PD – insights from this model may be applicable to SLAC (the comparison process is on-going work).

5.2 Limitations and future work

It seems that the desirable properties from the previously discussed tag models [12, 14, 24] have been usefully carried over into a dynamic network scenario. The ERA algorithm potentially offers a very generally applicable mechanism for controlling selfish behavior in many possible P2P task domains without the need to program and test explicit incentive mechanisms for each domain. In the work presented here *the ERA algorithm effectively emerges an incentive mechanism from the selfish moving behavior of the nodes* - by ostracizing selfish nodes over time. This happens because although a selfish node may do well for a while it will tend to lose its exploited neighbors as they find other nodes that are members of more cooperative groupings and hence have higher utilities. Additionally selfish nodes doing well (exploiting neighbors) are a signal for copycats to latch onto them and their neighborhood and exploit it also (speeding up the dissolution of the cluster).

NetWorld is the final model we present in this article. We claim that we have moved closer to our target application by a progressive succession of models. However we have not yet reached a stage where we could implement a target application. Certainly several more iterations are required before we can begin to consider deploying our technique in an application. However we are optimistic – unfinished work on a more detailed and realistic model indicates that the sort of results we report here will survive further steps towards implementation in real P2P applications.

There are some general issues that we need to address to move our model towards engineering applicability. We consider these kinds of issue would apply generally to models that start life as simulations of social or biological evolutionary processes: firstly, how to capture conceptions of utility and utility comparison; Secondly how to interpret and implement reproduction; thirdly, how to deal with non-adaptive agents; fourthly, implementing random selection of nodes.

Utility is a fundamental concept in much evolutionary modeling. It is assumed that some value or measure can be determined for each agent that defines its fitness. In the models described in this article the relative simplicity of the scenarios means that selecting a utility measure is easy – this is not the case for more complex and realistic domains [??]. This is especially true when rewards are high-delayed. It would be desirable therefore to eliminate the simplistic conception of utility and move towards a "performance targets" kind of arrangement (this might be similar to satisficing – see below). We have assumed in our model that agents can freely compare utility but this would be problematical in many application domains. Two possible workarounds are possible; we could abandon maximization of utility and apply a "satisficing" metric. When agents "satisfice" they do not change behavior after they reach some "aspiration" level of performance. This has some intuitive appeal, and would appear technically more feasible, it raises new questions: how does one determine the satisfaction level?[17], how and when (if at all) should it change? The other potential solution is for nodes to monitor the performance they receive from at least two subsets of their neighbors and then drop and replace the links to the poorer performing neighbor subset.

Within biological and even cultural models the idea of reproduction makes sense – successful behaviors are copied and increase. Within an application domain such as a P2P system nodes don't reproduce directly. It has been claimed that we can model nodes (or node behaviors – i.e. peer clients) as reproducing because this just captures the human users of systems installing and using clients that appear to offer better results - fast file downloading say. This interpretation is advanced by [23]. It is important to realize the dramatic implications of this interpretation however; it is that we are no longer modeling potential deployable mechanisms alone but rather the behavior of a system with *humans within the loop of the mechanisms*. There are two problems with this: currently there appears insufficient data (although there is some [1]) to know how humans behave in these contexts. Also, this limits the applicability of any mechanisms to those involving constant human intervention which is currently slow and expensive and hard to deploy – there are only so many peer clients the average user is going to install and monitor[18]! We

[17] Interestingly, because humans are considered to often "satisfice", rather than optimize, there is a body of work including simulations which could be drawn on [27].
[18] If someone invented an easily configurable client with an accessible scripting language, this would change.

sidestepped the issue in our models by keeping the number of nodes fixed and interpreting reproduction as *node movement* by rewiring – this begs the question of how you stop agents ignoring such rewiring rules.

This latter secondary problem leads onto our next major issue – how do you ensure agents (or nodes in our case) do in fact follow an adaptive process. In our models we have assumed all the agents follow the same adaptive process but in a deployed system – why would they? A robust system cannot assume this. Our conception of robustness has been that the system is resistant to mutation on selected components under the assumption of utility maximization but what if a subset of the population simply stopped adapting and acted selfishly, permanently, ignoring the fact that others were doing better. How robust would our models be then? This kind of "whitewashing" behavior, can, at least initially, be answered in each domain by running various attack scenarios. However, for a more general solution (if one exists) we might need a different translation process as we move between evolutionary and application models.

5.3 Methodological Observations

We have presented a sequence of three models, going from the abstract to the more concrete. Two questions naturally arise: *firstly*, why bother with the more abstract models – why not go straight to the most concrete; and *secondly*, couldn't the most concrete simulation have been designed from first principles? The answers to these questions are strongly related – the answer to the first lies in the answer to the second.

It is quite possible (although somewhat unlikely) that the ERA algorithm could have been designed by considering mechanisms for the prevention of 'commons tragedies' in P2P systems, from first principles. However, in that case one would not have any *theory* to indicate: the scope or generality of the technique; the fundamental reasons why it works; how it can be adjusted and how not; key characteristics to look for to check it is working right; and ways of fixing it if it does not work. The more abstract models facilitate the establishment of such a theory by providing simpler simulations about which to look for and test such theory. The more abstract models embody the theory in the more concrete models. For example experiments upon the TagWorld model (and closely related cousins) suggested the hypothesis that the rate of tag mutation needs to be higher than that of strategy mutation [13]. This hypothesis carries over to the more concrete models and provides a condition which can be checked for there. Again, although it is theoretically possible that a designer might have guessed such a condition purely by examining the FileWorld model, this is extremely unlikely.

Thus by providing testable theory about the more concrete applications, the more abstract models give more confidence as to the reliability of the applications (as well as how to adjust it etc.). This is important because with such distributed and stochastic algorithms such as ERA it is unlikely that one is going to be able to verify their properties by inspection and extremely unlikely that this will be possible using formal methods [6]. So the theory about the applications provides some of the confidence in the system that would otherwise be missing. Confidence in the theory comes from its repeated survival in simulations experiments. In this way one can see the close parallels with the classic scientific experimental method [5].

Acknowledgements

Thanks go to Mark Jelasity from the University of Bologna, Dept of Computer Science, for perceptive discussions, observations and pointers concerning P2P systems.

References

1. Adar, E. and Huberman, B. A. (2000) Free Riding on Gnutella. First Monday Volume 5, No. 10.
2. Axelrod, R. (1984) *The Evolution of Cooperation*, Basic Books, New York.
3. Channon, A.D. and Damper, R.I. (1998) Evolving novel behaviors via natural selection. In Adami, C. et al. (eds.) *Proceedings of Artificial Life VI*.Cambridge, MA: MIT Press, 384-388.
4. Cohen, B. (2003). Incentives build robustness in bittorrent. In Workshop on Economics in Peer-to-Peer Systems, 2003.
5. Edmonds, B. (2004) Using the Experimental Method to Produce Reliable Self-Organised Systems. To be presented at ESOA 2004, AAMAS, New York, July 2004. (http://cfpm.org/cpmrep131.html)
6. Edmonds, B. and Bryson, J. (2004) The Insufficiency of Formal Design Methods - the necessity of an experimental approach for the understanding and control of complex MAS. To be presented at AAMAS 2004, July, New York. (Version at http://cfpm.org/cpmrep128.html).
7. Edmonds, B. and Hales, D. (2003) Replication, Replication and Replication - Some Hard Lessons from Model Alignment. *Journal of Artificial Societies and Social Simulation* 6(4). http://jasss.soc.surrey.ac.uk/6/4/11
8. Epstein, J. and Axtell, R. (1996). *Growing Artificial Societies: Social Science from the Bottom Up*. Cambridge MA: The MIT Press.
9. Gilbert, N. and Conte, R. eds. (1995). *Artificial Societies: the Computer Simulation of Social Life*. London: UCL Press.

10. Gilbert, N. and Doran, J. eds. (1994). *Simulating Societies: the Computer Simulation of Social Phenomena*. London: UCL Press.
11. Gnutella. http://www.gnutella.com/
12. Hales, D. (2000) Cooperation without Space or Memory: Tags, Groups and the Prisoner's Dilemma. In Moss & Davidsson, (eds.) *Multi-Agent-Based Simulation*. LNAI 1979:157-166. Springer. Berlin.
13. Hales, D. (2004) Change Your Tags Fast! - a necessary condition for cooperation? To be presented at the MAMABS, co-located with AAMAS, New York, July 2004.
14. Hales, D. and Edmonds, B. (2003) Evolving Social Rationality for MAS using "Tags", In Rosenschein et al. (eds.) *Proceedings of the 2nd International Conference on Autonomous Agents and Multi-agent Systems*, (AAMAS03), ACM Press, 497-503.
15. Hales, D., Edmonds, B., Norling, E. and Rouchier, J. eds. (2003) *Multi-Agent Based Simulation III*, Proceedings of the 4th International Workshop, MABS 2003, Melbourne, Australia, July 2003. Berlin: Springer, Lecture Notes in Artificial Intelligence, 2927.
16. Hardin, G. (1968) The Tragedy of the Commons, *Science*, 162:1243-1248.
17. Holland, J. (1993). The Effect of Labels (Tags) on Social Interactions. Santa Fe Institute Working Paper 93-10-064. Santa Fe, NM.
18. JASSS, The Journal of Artificial Societies and Social Simulation (http://jasss.soc.surrey.ac.uk/JASSS.html)
19. Kamvar, S. D., Scholosser, M. T. and Garcia-Molina, H.. (2003) The eigentrust algorithm for reputation management in p2p networks. In the 12th International WWW Conference, 2003.
20. Kazaa. http://www.kazaa.com
21. Kempe, D., Kleinberg, J. and Kumar, A. (2000) Connectivity and inference problems for temporal networks. *Proc. 32nd ACM Symposium on Theory of Computing*, 2000.
22. Lai, K., Feldman, M., Stoica, I. and Chuang, J. (2003). Incentives for cooperation in peer-to-peer networks. In Workshop on Economics of Peer-to-Peer Systems.
23. MojoNation. http://www.mojonation.com
24. Riolo, R. L. , Cohen, M. D. & Axelrod, R.. (2001) Evolution of cooperation without reciprocity. *Nature* 414, 441-443.
25. Sichman, J. S., Bousquet, F. and Davidsson, P. eds. (2002) *Multi-Agent-Based Simulation II*. Lecture Notes in Artificial Intelligence 2581. Berlin: Springer-Verlag.

26. Sigmund, K. and Nowak, M. A. (2001). Tides of Tolerance. *Nature* 414, 403-405.

27. Simon, H.A. (1976) *Administrative Behavior*, 3rd edition. New York: The Free Press.

28. Smith, J. M. (1982) *Evolution and the Theory of Games*. Cambridge University Press. Cambridge.

29. Sun, Q. and Garcia-Molina, H. (2004) SLIC: A Selfish Link-based Incentive Mechanism for Unstructured Peer-to-Peer Networks. In *Proceedings of the 24th IEEE international Conference on Distributed Systems*. IEEE computer Society. 2004.

30. Trivers, R. (1971) The evolution of reciprocal altruism. *Q. Rev. Biol.* 46, 35-57.

31. Zimmermann, M. G. , Egufluz, V. M. and San Miguel, M.. (2001) Cooperation, adaptation and the emergence of leadership. In Kirman, A. and Zimmermann, J.B. (eds.), *Economics with Heterogeneous Interacting Agents*. Berlin: Springer, 73-86.