Towards the Evolution of Social Structure

Bruce Edmonds, Emma Norling
Centre for Policy Modelling
Manchester Metropolitan University

Motivation

To what extent can social structure result from evolutionary processes, as opposed to deliberately organised by a collection of intelligent individuals? That is, could social organisations (which are usefully identifiable as an entity in their own right) result from a mechanism which only relies on the processes of random variation and selective reproduction. Or, on the other hand, is it necessary for the individuals to have acquired (through evolution or design) sufficient cognitive abilities to deliberately organise themselves into such entities (using planning, reasoning, deliberate experiment, anticipatory learning or the like).

The question is important for the answer (w.r.t. any particular system) makes a difference. For example, in the former (evolutionary) case the abilities of the individuals can co-evolve with the social structures that they can (unthinkingly) sustain so that some of their features (e.g. a propensity towards cooperation or a distrust of strangers) could have evolved to reinforce or build upon the existing social structures and so boot-strap the complex society we inhabit. If this is not the case then we must understand the particular social structure as essentially intended institutions built *subsequent* to the abilities developed in other circumstances. This might be important in either understanding observed societies or producing/managing new ones (including artificial ones, such as Multi-agent systems).

In a sense this question is the complementary half of the Machiavellian Intelligence theses that our intelligence evolved (partly) because it provided an ability to manipulate social structures and situations to our (individual) advantage. It can be seen as a part of the, more general, Social Intelligence Hypothesis [2], which is that suggestion that our intelligence gives us evolutionary advantage via the social structures it enables. An example of this is the ability to imitate [5] which may allow groups of people to develop a culture of skills, rules and traditions that equip that group to inhabit specialist ecological niches [20].

Of course evidence has the primary role as far as observed social structures are concerned. For example, the apes can not sustain as sophisticated social structures as humans as so the present abilities of humans that go beyond those of apes must be crucial for their production and/or maintenance. However we do not have strong evidence about how our present abilities developed over biological evolution, there might have been a whole succession of structure then ability then structure etc. which resulted in the current position. Clearly new social structures and institutions have arisen over a historical timescale and thus it seems clear that some of our social structures rely upon our present abilities.

On the other hand, some social structures clearly do not require complex cognition by its individuals. Slime molds organise themselves (under the right conditions) into what can be meaningfully called a single entity (a fruiting structure). Since they, as individuals, have no (or only simple) information processing abilities, some social structure does not require complex cognition.

Thus the question should be rephrased into: ***which*** *mechanisms might result in* ***what*** *social structures under* ***which*** *conditions?*

Computational simulation/systems can help answer this question by establishing some possibilities in terms of the causal connection between mechanisms, conditions and resulting structures. That, is by starting to map out the possible triples of: (mechanism, conditions, results). This is not an easy task, for simulations can be deceptive in terms of the robustness of their results against "small" changes in their set-up and, given their intrinsic complexity, we can easily be mistaken as to our interpretation of how and why they produce the results they do.

This paper looks at one stream of computational experiments that start to address this project – they examine how a particular class of mechanism (tag-related mechanisms) can produce group-like structures for cooperative behaviour. In this sense this paper is a mini-survey and prospective concerning these models and their results – models and results in which I have played only a part, the main work being done by others including: David Hales, Rick Riolo and Emma Norling.

About Tags

"Tags" are a socially distinguishable mark or signal, for example wearing a smart suit [17]. The important thing is that tags are not "hardwired" to any particular behavioural trait so that even if a particular tag comes to be associated with some behaviour (clerical garb is worn by trustworthy individuals) the connection is fallible – someone new can always adopt the tag regardless of whether they exhibit the associated behaviour (e.g. a con man pretending to be a priest). A biological example is where a wasp effectively advertises its ability to sting via its yellow and black stripes and thus reduces danger to itself by warning off others. The stripes are the tag of the species – although they may have originated as the result of a random mutation it persists because of the evolutionary advantage they confer. However, other species may evolve to use the same tag without the ability to sting (e.g. the hoverfly) to exploit the signal that the tag has come to be associated with. Thus although tags can be initially meaningless, they can allow for the recognition of classes of individual, albeit in a fallible manner. Thus displaying and recognising tags can be seen as a minimal ability that allows for a new set of social structures to arise, namely the appearance of clusters that can be sensibly thought of as "groups". For cooperative behaviour, the basic rule is to preferentially interact with those with the same or similar tags.

Example 1 – integral tags on the one-shot PD game

I start with the simplest model that I know of, which is the one described in [11]. Here there is a population of individuals simulated over a number of discrete generations. Each generation, each individual is paired with a fixed number, *n*, of others with each of whom it plays a one-shot prisoners' dilemma game (PD) according to its strategy flag, which is fixed as cooperate or defect. At the end of each generation each individual's score is totalled and individuals are propagated into the next generation in accordance to this score (higher scoring individuals are propagated more, on average, than lower scoring ones). There is also a small probability of a progeny's strategy "mutating" between cooperation and defection.

If the model were only as so far described the result is that quickly defectors would come to dominate the population. However the ability to display and recognise tags is

now added to the mix and this radically changes the outcomes. With tags: each individual has a tag represented by an integer from a certain range; progeny of individuals have the parent's tag (with a small probability of mutation to a random integer); thus the prevalence of tags as well as the strategies are determined by an evolutionary process. The exploitation of the tags comes from the basic rule that individuals only "pair" with those with the same tag if that is possible (otherwise completely randomly). More information about this model is given in the appendix.

The dynamics observed in this model can be summarised as follows: eventually, by chance, a small group (usually a pair) of only cooperative individuals with the same tag arises; these are only paired with each other and hence score highly compared to the background population and hence these individuals are preferentially propagated into the next generations with the same tag and strategy; eventually an individual arises with the same tag but a defector; this individual does better than anyone and hence is propagated at the highest rate so the set of individuals with this tag becomes flooded with defectors; now the cooperative individuals in the group do very poorly compared to the background population and hence are selected out leaving only defectors with that tag; these defectors also do badly and are selected out. It makes sense to interpret all those with the same tag as a "tag group". This "life cycle" occurs repeatedly and in parallel – there are tag groups arising and dwindling all the time. This rising and falling of tag groups is illustrated in Figure 1.

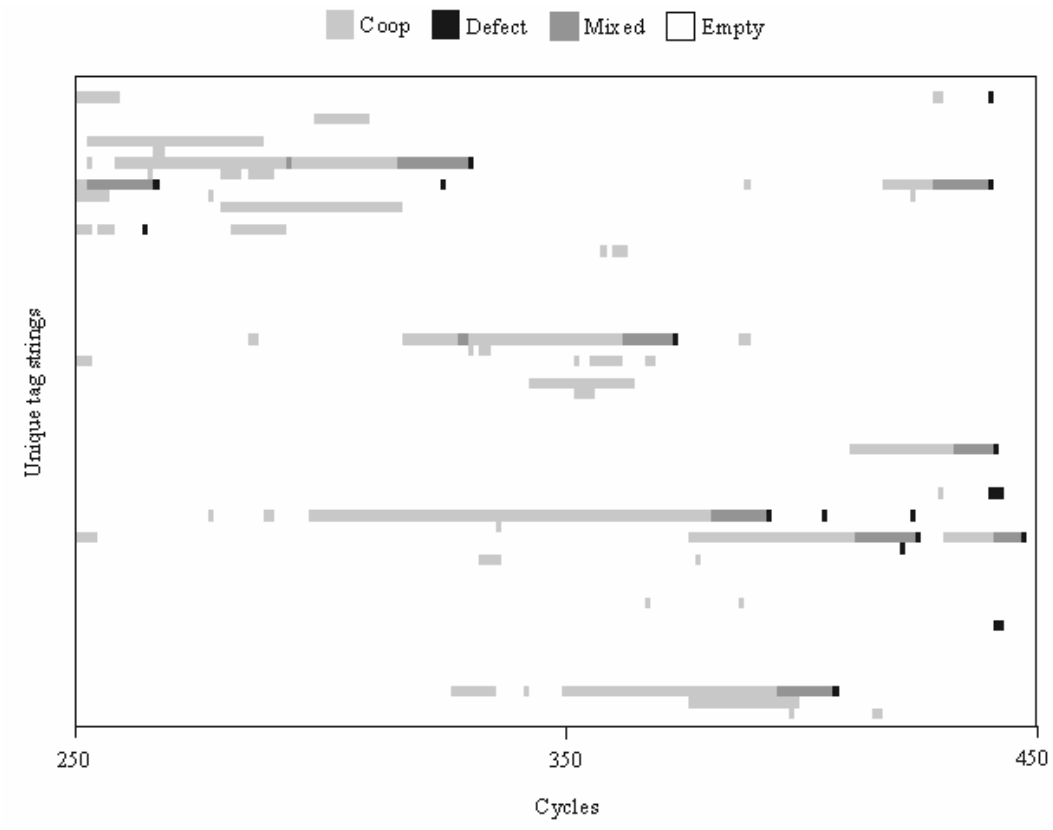

Figure 1. the continual rise and fall of tag groups

If the rate at which cooperative groups arise is sufficiently high, the average time until a defector arises long enough, and the "killing effect" of a defector on a tag group fast enough, then the overall level of cooperation can be high, even given that cooperation is not an "evolutionary stable strategy". The graph is not very informative - a high level of cooperation is quickly established thus it looks almost identical to Figure 1.

There have been many variations of this kind of model explored, including where tags are represented as a real number and there are degrees of tolerance, with probabilistic preferences for interaction based on tags, different selection and mutation regimes etc. The important property of these models is they show a way of maintaining the overall level of cooperation even when it is (a) possible for defectors (who cooperate with no one) to arise and (b) where it is locally and immediately beneficial for an individual to do so. Furthermore the variety of models in which such an effect occurs indicates its robustness. Of course if one builds in further protection against defectors then the tag groups can be extremely stable over long periods of time. However, this is not very surprising and depends upon such an enforcement being there – this is the sort of enforcement that might be possible to implement with more sophisticated cognitive machinery using punishment or forced ejection (tag-change).

However, regardless of whether the tag cooperation is crisply defined (as in the model described above) or fuzzily defined using probabilities for interaction or tolerance ranges and real number tags, the clusters of individuals with similar tags who preferentially interact with each other and display such characteristic "life-cycles" are usefully identifiable as "groups" in the same way that a connected cluster of cells might be identified as a person. However such groups have no structure beyond this clustering – they simply partition the population (possibly fuzzily).

Example 2 – single group formation with floating point tags

Here there is a fixed population of individuals that are randomly paired a fixed number of times. They are each randomly paired a fixed number of times with other individuals to whom they donate 1 (and incur a lesser cost, *c*). The sum of their score determines their "fitness" which affects the rate with which they reproduce into the next generation.

Here, the tag behaviour of individuals is determined by two floating point numbers, both drawn from [0, 1]: the first is its tag value and the second its "tolerance". An individual will cooperate with another it is paired with if the difference between its tag value and that of the individual it is paired with is less than or equal to (=) its tolerance value. Thus the tolerance value represents how cooperative an individual is: a tolerance value of 1 means that it will cooperate with every other it is paired with; a tolerance value of 0 means that it will only cooperate with another whose tag value is exactly the same as its own and no others. This mechanism is illustrate in Figure 2.

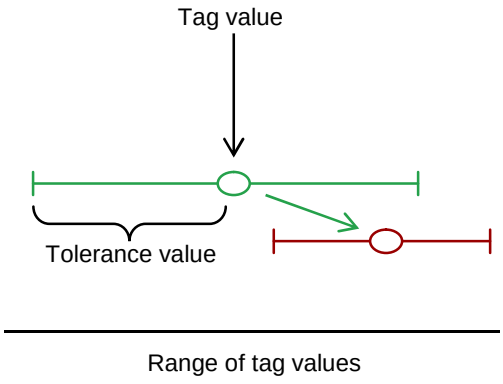

Figure 2. The tag/tolerance mechanism – the individual with the larger tolerance donates to the one with smaller tolerance but (in this case) not vice versa

The dynamics of this model are that cooperation is very quickly established and is extremely stable from there on with only very rare collapses of this group and the establishment of a new group. The cooperation is composed of only one dominant group at a time, and this group is mostly composed of exact tag-clones of each other who are forced to cooperate with each other due to the model set-up. Thus this model implements a sort of kin-selection, whereby individuals give to those directly related to them. If one changes this model so that complete defection is possible, so that cooperation only occurs if the difference between its tag value and that of the pairee is strictly less than (<) its tolerance value then the cooperation effect disappears. Thus the tolerance value here is not really important and only really has a function as a symmetry breaking mechanism (depending upon the selection mechanism implemented). For a complete analysis of this model see [7].

Example 3 – evolution of "symbiosis" with floating point tags

The next model makes a small step towards adding structure with a group, by adding (and encouraging) mixtures of individuals with different skills within a group using tags. This is the biologically-inspired model described in [8]. Here as well as tags, individuals have a single specific skill from a small range. "Nutrition" is provided to the environment in kinds that correspond to those skills – only individuals with the right skill can "harvest" each unit of nutrition. Individuals can store a certain amount of each kind of nutrition. However individuals need all kinds of nutrition to survive (expending some of each every generation) and more of all kinds to reproduce. Here individuals are randomly paired a number of times and may share any excess of any kind of nutrition they are storing with those with sufficiently similar tags to their own. As before the individuals do not directly gain anything for themselves by sharing (indeed they lose potential future survival and reproductive possibilities) so one might expect that, over evolution, more "selfish" individuals (those that share with few or none at all) would dominate.

Here, the tag behaviour of individuals is determined by two floating point numbers drawn from [0, 1]: the first is its tag value and the second its "tolerance". An individual will share excess resources (in any store) with another it is paired with if the difference between its tag value and that of the individual it is paired with is strictly less than its tolerance value. This follows the tag structure of the model in Example 2 above but the strict inequality allows for completely selfish individuals which destroys the cooperation effect reported there. This mechanism is illustrated in Figure 2 above.

The resources in this model (of different kinds) are used up in sharing, living, and reproducing, thus the population size is controlled by the input of resources and the efficiency of the sharing. For individuals to survive and flourish they need to receive gifts of nutrition of the types they can not themselves harvest – thus the more types of nutrition are necessary the more difficult it is to survive via cooperation (in the example below there are 3 kinds). More information about this model is given in the appendix.

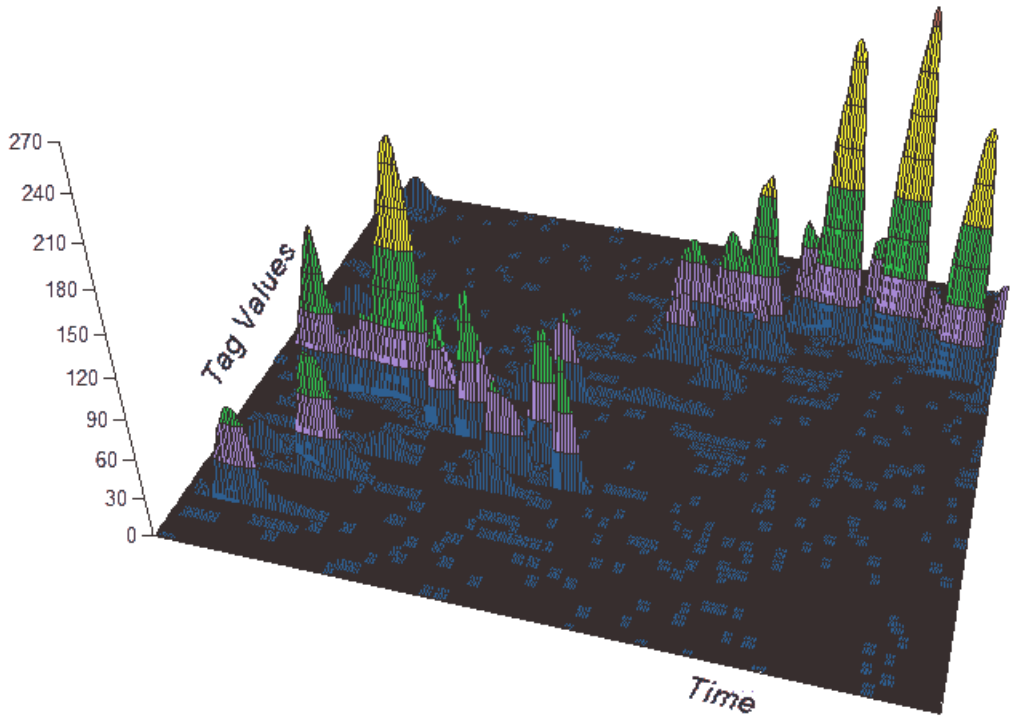

Figure 3. The rise and fall of tags in this example

The dynamics in this case are, in one sense, more complicated that in Example 1, although they follow the same basic outline. Clusters of cooperating tag groups do arise, punctuated by short periods of complete collapse (see Figure 3). Somehow, by chance, a small group of individuals covering the necessary range of skills and with compatible tag and tolerance values arises; these then are much more successful than the background population and quickly reproduce to form the dominant group; soon mutations arise with smaller tolerance values in one of the individuals with one of the skills – this one does better than the others (because it donates less) but if it reproduces too much its progeny starves itself of the necessary other nutrients because there are fewer of the others to donate to them. Thus as individuals mutate to ever more selfish individuals in a downwards arms-race the dynamics become more like those of Lokta-Voltara predator-prey dynamics. Eventually the dynamics, whose oscillations grow in amplitude, kills off one of necessary sub-groups and the population collapses back to the near-zero state until a new group arises again by chance. This is illustrated in the figure immediately below.

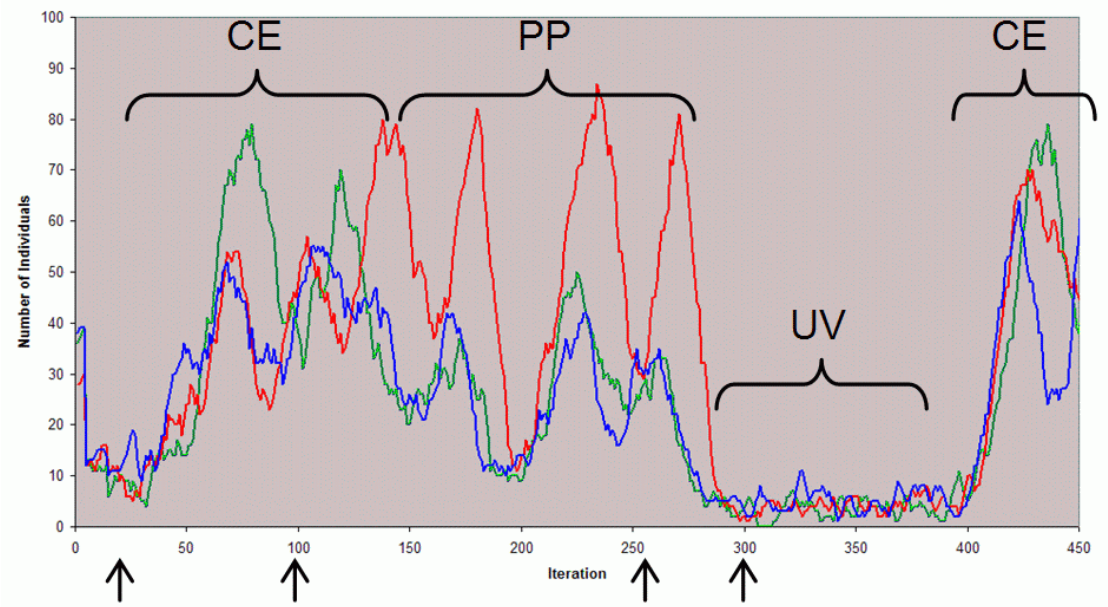

Figure 4. The dynamics of the different skill groups: each line shows the number with each skill; the system starts (at the first arrow) in an unviable state; then a cluster of cooperating tag groups emerge (second arrow, the CE=coexistence phase); but by 250 (the third arrow) a predator-prey type dynamics has been established; which finally breaks down into another unviable state (fourth arrow and UV=unviable phase).

Due to the limited resources which force a kind of "winner takes all" dynamics between types of individual, only one cluster of tag groups occurs at any one time in this model – there is no parallelism between these clusters. Thus this model represents the cooperation that could occur within a single niche only. But it does show how cooperation can occur and be maintained with competing cooperating clusters. Unlike in the previous model the tag groups are not sharply defined but are fuzzy, with "hangers-on" at their edges who survive by receiving nutrition through a chain of intermediaries outwards. Thus although the clusterings are very definite when viewed in the results the "groups" are not crisply delineated in terms of their interactions.

The extension of tags to local structure

The models above use a low information group membership indication (a single integer or floating point number) and simple rules determining whether an interaction occurs. This only supports fairly simple, unstructured groups. It can only encode crisp or fuzzy (respectively) group membership. Richer and more informative can be encoded in a social network structure, where the presence of a link (respectively absence) between two nodes indicates whether the might interact (respectively not interact). Thus, if one visualises the network, one may see the separate (or almost separate) groups with many internal connections but few (if any) connections between the groups occurring. Thus in the model below there are no tags other than the set of connections a node has – it is these connections which determine the structure.

Example 4 – job sharing using the SLAC algorithm

In this model there is a population of nodes of fixed number. Each of these can be connected to others (up to a maximum number) who they can interact with. Each node has a skill number (from a limited number of types indicated by an integer), a strategy bit (cooperate or defect) and a list of its connections. Each generation a

fixed number of "jobs" are allocated to nodes at random, each of these jobs corresponds to one of the skill types. If a node has the right skill itself it does the job and gains a fixed reward. If not, it asks a fixed number of nodes it is connected to to do the job for them, if they are cooperative and have the right skill they do the job for the original node (thus incurring a cost themselves), the original node gets the reward if this happens. Thus a cooperative node does not directly benefit from doing work for others. A nodes score is the total of its rewards minus the costs it incurred over that round of jobs. This task, called "SkillWorld", is introduced in[9].

On top of this structure is an evolutionary algorithm based upon imitation and mutation. That is, nodes pick a random other node and compares their score with the score of that node. If it is doing better it drops its links, links to that node, and imitates its connections and strategy (after each link is made if either exceeds its maximum number of connections is randomly drops one). Every now and then a node "resets" with a small probability, that is drops all of its nodes, and attaches to a random node. Also there is a small probability that a node changes its strategy bit. This is called the SLAC algorithm (Selfish Link-based Adaptation for Cooperation) [13]. There is more information about this model in the Appendix. The result of the this algorithm on the SkillWorld task is the rapid establishment of cooperation in the population which remains pretty stable and high from then on – see Figure 5.

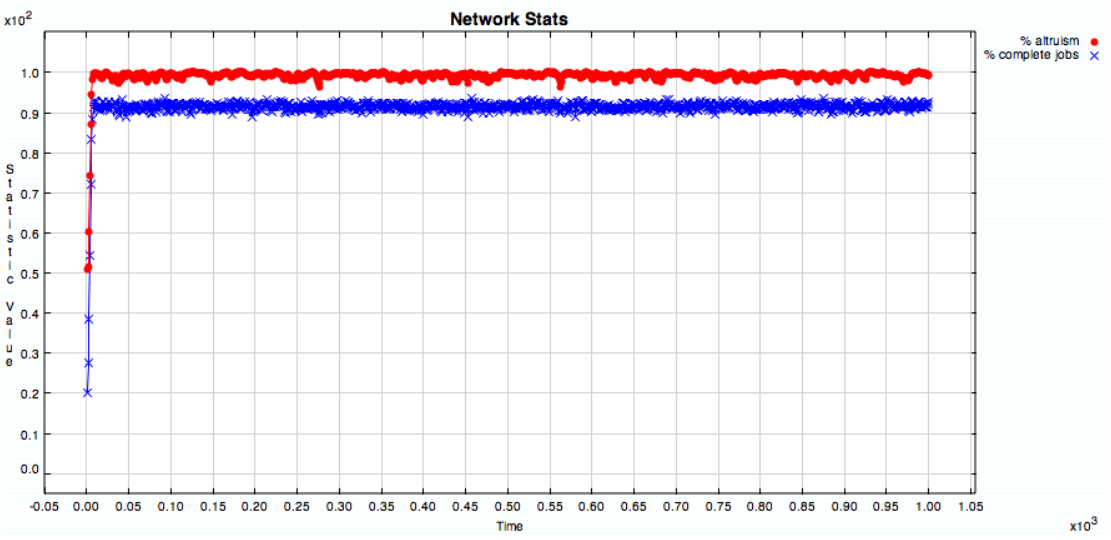

Figure 5. level of cooperation in the SLAC algorithm

The sequence of network visualisations in Figure 6 shows the development of the network structure in this model. It starts with lots of disconnected nodes which quickly form up into clumps after 12 generations; at generation 117 a new defector arises (in the small circle); which then multiples into a bigger clump by generation 120 at the expense of the group co-operators it is connected to; but by generation 122 all the cooperative nodes that were connected to them have imitated ones that were not and the non co-operators have become isolated and disappear completely by the next generation.

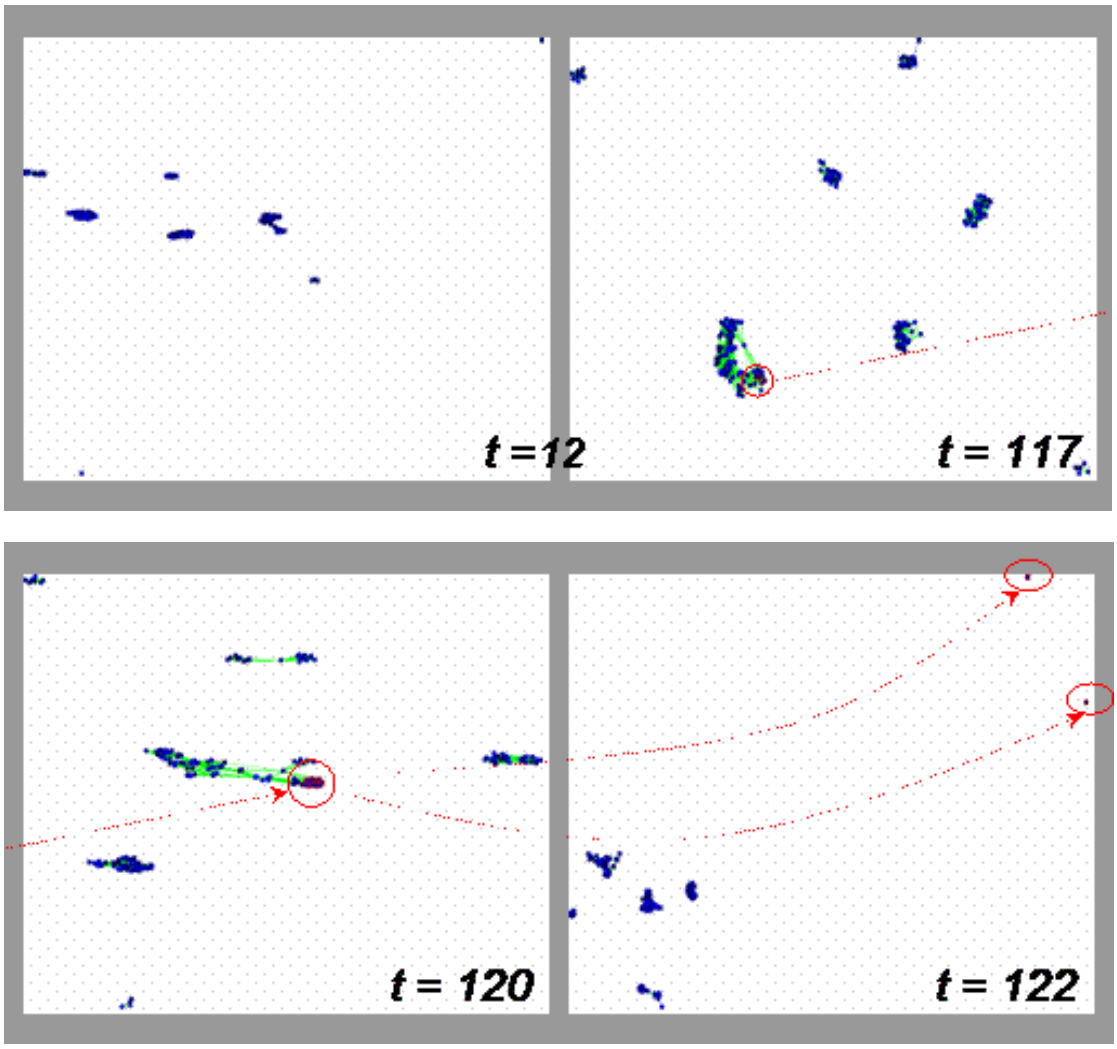

Figure 6. The appearance then isolation of defectors, the defectors are circled in each "snapshot"

Thus here we see more structure resulting from the model as a result of an evolutionary mechanism. It turns out to be robust against many possible attacks against it. However the structures developed are not very sophisticated. It is not good for tasks which require competition between individuals or that need the network to be substantially connected (such as search via flood-fill queries in an P2P network).

In SLAC when a node imitates another it drops all its previous links – this corresponds to a probability of dropping existing nodes of 1. If one implements the model so that this probability can be set to any floating point number within [0, 1], then one can "tune" the resulting network so that it stays connected and yet can maintain the connectedness of the whole [16]. For example setting this probability to 0.95 causes an almost imperceptible drop in the overall level of cooperation (see Figure 7) and yet results in an almost completely connected network. Somehow the co-operators are able to "avoid" the defectors without breaking into lots of separate groups. I have not shown some example structures that this results in since it is very difficult to visually discern the structure.

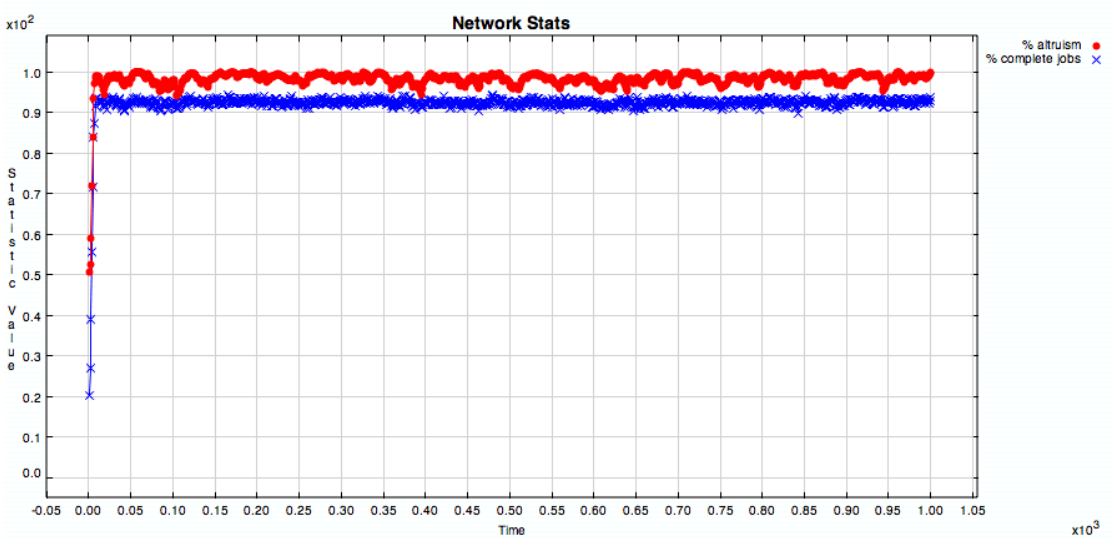

Figure 7. level of cooperation with SLACER p=0.95

Example 5 – SLAC on unidirectional P2P file- sharing task

The final example given in this survey is that of a more realistic model of file-sharing via a P2P network. P2P networks are those without a centralised or client-server structure – every computer is both a client and a server. Each computer in the network "knows" of a limited number of others with which it interacts. To communicate with others further away the messages have to go through intermediary computers. Well known P2P networks that operate on this basis work as a virtual layer over the internet include: e-donkey and Bittorent.

This simulation is quite complex, so I do will not describe it in full here but rather sketch it. In this, nodes seek files by sending queries through the network using a "flood fill" algorithm – contacting other nodes they know – for a set number of "hops" (arc transitions). This is a decentralised network in that each node only "knows" about a limited number of other nodes and is not aware of the whole network (for example via access to a central server). To search the network a node sends its query to the nodes it knows and they pass it on to those they know and so on. This process continues until the queries have been passed on for a certain maximum number of hops at which point the relevant copy of the query "dies".

If a node is currently sharing its files (it does not have to) and it happens to possess a requested file, that file is sent back to the originator of the query. Each node has a "satisfaction" level. When a node gets a file it does not have (and wants) its satisfaction level increases. Satisfaction decays exponentially at 10% per cycle. This means that nodes have to keep succeeding in their quest for files if they are to remain satisfied. If the satisfaction level of a node drops low enough then it will copy the connections and sharing strategy of a neighbouring node which is doing better (or with a low probability drop out of the network altogether and be replaced by a new node with a single random connection). This constitutes a social imitation process based on relative performance.

Unlike the network in the previous example, the network here is a directed one so that if node A sends queries to node B the reverse does not necessarily occur. Each node thus represents a person who is controlling some P2P software on the Internet. If the controller is unsatisfied with the number of files they are getting they may choose to imitate the way that another controller operates their software.

The general structure of the network that develops is illustrated in Figure 8 below. Due to the dynamics of the model, a core partition develops which is totally connected and the rest of the network (the periphery) mostly consists of branches that link into this core either directly or indirectly. There may also be one or more small isolated groups which quickly "die". This is a result of the dynamic node behaviour. If there were any nodes that were on a branch leading away from the core, these would not be viable in terms of file searching success and hence their satisfaction level would fall until they reset their connections to random ones thus "breaking" the branch. Of course, before this structure is established (and during transitory periods afterwards) different patterns may occur but the combination of core partition and periphery seems to be the "attractor" for the system.

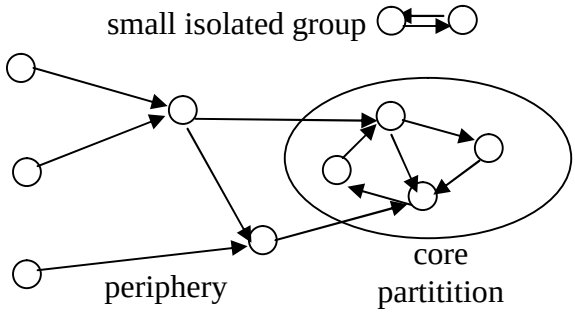

Figure 8. An illustration of the typical network structure that results in the P2P model – there is a core partition that is totally connected and a collection of branches feeding into this. Arrows show the directions in which queries for files might be sent.

Although there is a lot of churn in the network in this model, it does seem to act so that the co-operators are better connected than the defectors. Table 1 shows some statistics for a typical run of this model over 600 simulation iterations of a 1000 iteration run. Uniformly the co-operators have a higher utility, average number of links and average centrality than defectors and those in the core group also get higher measures than the equivalent kinds of non-core individuals.

Table 1. Average network properties by node type: network measures

Type	Average utility	Average number of links	Average centrality
in-coop	0.790694	2.967343	0.405458
out-coop	0.512293	2.500713	0.309861
in-def	0.373527	2.005525	0.269064
out-def	0.324489	1.492763	0.189887

However, when one looks at the patterns over time one does *not* see a good correlation of number of links or centrality score to utility (highest is 17% even over a variety of lags – see [6]). The measures for a single node are illustrated in Figure 9. This is because of the short term effects of defectors copying others with better links or more central contrasted to the inability of defectors to maintain such advantageous positions due to the co-operators they affect, relocating away. Thus the role of

structure in this model is quite complex and certainly not a case of being able to effectively exclude defectors from one's group.

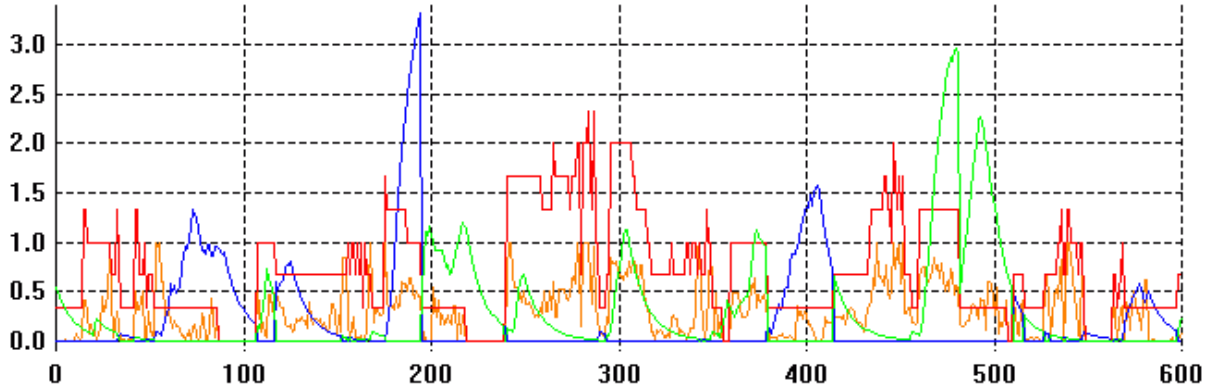

Figure 9. The last 600 cycles of a simulation run for one particular node. The red line shows 1/3 of the number of outward links the node has. The orange line shows the measure of centrality: from 0 to 1 with 1 being the most central node. The blue/green line shows the level of satisfaction for the node. (It is blue when it is not making its files available for sharing and green when it is.)

Discussion

The co-evolution of structure and individuals

In the above we have seen how different "mechanisms" can result in different structures within an evolutionary process. What we do not see is the Darwinian evolution of these mechanisms so as to produce social structures useful to the individuals who inhabit them (or otherwise). This would allow individuals to evolve a social "extended phenotype" along with their other characteristics.

Meaningful evolution of groups

In many of the examples above the structures that arise do undergo a sort of "life-cycle" and selection does indirectly and eventually occur upon individuals depending upon the viability of the structures they are in (and where they are in them). However one is not justified in saying that the structures are evolving in any Darwinian sense. For the evolution of these structures more is necessary – new groups would have to be spawned from old groups and (at least some of) the characteristics of the old group would need to be transmitted to the new group it spawned. The transmission of these group characteristics would need to be very accurate, implying some sort of instruction or blueprint being transmitted, but with some small source of variation too.

However it is possible that the sort of mechanisms described in this paper could provide enough structure for such group evolution mechanisms to be implemented on top. Thus these sorts of systems might be a "stepping-stone" towards true group-level evolution.

The TransWorld Challenge

None of the structures in the above examples are very sophisticated. They do not seem to have much significant internal structure (as far as we can discern anyway). They are certainly not as sophisticated as, say, the food web structures found in ecology or the value-chains observed in developed economies.

For example, the environment might afford certain computations or transformations of some resources and the individuals are specialised in terms of which of these transformations they can do. For example resources A, B and C might be available in the environment, half the individuals might be able to effect the transformation $A,B?\ D$ whilst the others able to do $C,D?\ E$, where E is a valuable/useful resource. The challenge here is to see what evolutionary mechanisms might allow individuals to organise themselves into appropriate networks/structures so as to produce what is needed from what is available without heavy negotiation or planning. This challenge is the subject of ongoing research.

Conclusion

Although work of the above kind can suggest possibilities of how individual abilities and social structure interrelate in observed societies (which may, or may not, turn out to be true), we anticipate that the real value of this kind of work is for informing the "design", formation and management of complex, distributed IT systems. That such mechanisms are useful in such systems is already evident, for example the "reputation system" of Ebay or the "tit-for-tat" systems in BitTorrent **Error! Reference source not found.**. Understanding some of the possibilities concerning what basic facilities/abilities/mechanisms enable the development of "higher-level" social structures and institutions has the potential to multiply the usefulness of such systems.

Of course, just because a system's properties have been deliberately embedded into such systems does not mean that the dynamics that occur in such simulations will also occur in the target system. The continual investigation and re-modelling of such systems is necessary to understand them, given the ever-present possibility of emergence in such systems. However having a good catalogue of some of the possible connections gives the best possible basis from which to do this.

References

[1] Axelrod, R. (1984) *The Evolution of Cooperation*, Basic Books, New York.

[2] Barkow, J. H., Cosmides, L. & Tooby, J. (Eds.), (1992) *The adapted mind: evolutionary psychology and the generation of culture*. Oxford University Press.

[3] Cohen, B. (2003) Incentives Build Robustness in BitTorrent. Presented at the 1st Workshop Economics of Peer-2-Peer Systems, 2003; (http://www.sims.berkeley.edu/research/conferences/p2pecon/papers/s4-cohen.pdf)

[4] Byrne, R. W. & Whiten, A. (eds.) (1989). Machiavellian Intelligence: Social Expertise and the Evolution of Intellect in Monkeys, Apes, and Humans. Oxford Science Publications, OUP.

[5] Dautenhahn, K. and Nehaniv, C. L. (eds.) (2002) *Imitation in Animals and Artifacts*. Cambridge, MA: MIT Press.

[6] Edmonds, B. & Chattoe, E. (2005) When Simple Measures Fail: Characterising Social Networks Using Simulation. CPM Report 05-158, MMU. Presented at the Social Network Analysis: Advances and Empirical Applications Forum, Oxford, July 16-17 2005. http://cfpm.org/cpmrep158.html

[7] Edmonds, B. & 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] Edmonds, B. (2006) The Emergence of Symbiotic Groups Resulting From Skill-Differentiation and Tags. Journal of Artificial Societies and Social Simulation, 9(1). (http://jasss.soc.surrey.ac.uk/9/1/10.html).

[9] Hales, D. & Arteconi, S. (2005) Friends for Free: Self-Organizing Artificial Social Networks for Trust and Cooperation. Sept 2005, http://arxiv.org/abs/cs.MA/0509037.
[10] Hales, D. & Edmonds, B. (2002) Evolving Social Rationality for MAS using "Tags". CPM Working Paper 02-104. The Centre for Policy Modelling, Manchester Metropolitan University, Manchester, UK (available at: http://cfpm.org/cpmreps.html).
[11] Hales, D. (2000), 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, 1979:157-166. Berlin: Springer.
[12] Hales, D. (2002) Wise-Up! - Smart Tag Pairing Evolves and Persists The Centre for Policy Modelling, Manchester Metropolitan University, UK. Discussion Paper CPM-02-90. (http://cfpm.org/cpmrep90.html)
[13] Hales, D. (2004) From Selfish Nodes to Cooperative Networks – Emergent Link-based Incentives in Peer-to-Peer Networks. In proceedings of The Fourth IEEE International Conference on Peer-to-Peer Computing (p2p2004), 25-27 August 2004, Zurich, Switzerland. IEEE Computer Society Press.
[14] Hales, D. (2005) Change Your Tags Fast! - a necessary condition for cooperation? Proceedings of the Workshop on Multi-Agents and Multi-Agent-Based Simulation (MABS 2004), LNAI 3415, Springer, 2005.
[15] Hales, D. and Edmonds, B. (2005) Applying a socially-inspired technique (tags) to improve cooperation in P2P Networks, *IEEE Transactions in Systems, Man and Cybernetics*, 35:385-395.
[16] Hales, D.; Arteconi, S.; Babaoglu, O. (2005) SLACER: randomness to cooperation in peer-to-peer networks. Proceedings of the Workshop on Stochasticity in Distributed Systems (STODIS'05) including in the Proceedings of IEEE CollaborateCom Conference, Dec. 19, 2005. San Jose, CA.
[17] Holland, J. (1993), The Effect of Labels (Tags) on Social Interactions. SFI Working Paper 93-10-064. Santa Fe Institute, Santa Fe, NM. (available at: http://www.santafe.edu/sfi/publications/wplist/1993)
[18] Margulis, L. and Sagan, D. (1986) Microcosmos: Four Billion Years of Evolution from Our Microbial Ancestors. Summit Books, New York.
[19] Norling, E. and Edmonds, B. (2006) Why it is Better to be SLAC than Smart. 1st World Congress on Social Simulation (WCSS'06), Kyoto, Japan, August, 2006. ()
[20] Reader, J. *Man on Earth*. London: Collins, 1988.
[21] Riolo, R. L., Cohen, M. D. & Axelrod, R (2001), Evolution of cooperation without reciprocity. *Nature*, 411:441-443.
[22] Roberts, G. and Sherratt, T. N. (2002) Similarity does not breed cooperation. *Nature* **418**:449-500.
[23] Sigmund, K. and Nowak, M. A. (2001) Evolution - Tides of tolerance. *Nature* **414**:403.
[24] Wilson, D.S. & Sober, E. (1994) Reintroducing group selection to the human behavioural sciences. *Behavioral and Brain Sciences* **17**(4): 585-654.

Appendicies – model specifications

Iterated PD tag model

This is described in [12].

There is a fixed population of agents, each of who have a strategy bit (cooperate or defect) and a tag (represented by an integer from a certain range). Each generation each individual is paired with a given number of others (not themselves) and play a PD game according to their strategy. If there are other individuals with the same tag as them their partners are randomly selected form those, otherwise from the whole population. Their score over these games determines the extent to which they are propagated into the next generation (in this case using Roulette Wheel propagation). There is a small chance of a mutation in the tag and a smaller chance of a mutation in the strategy bit.

The algorithm

```
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
    (with some, low, level of mutation)
END LOOP
```

Key Parameters are:

? The payoffs of the (PD) game, clearly for this algorithm to succeed there must be a general advantage to cooperating
? The number of individuals
? The number of possible tags
? The mutation probability for changing tag
? The mutation probability for changing strategy
? The number of games/pairings per generation

The initial population is seeded with individuals with random tags and all defectors. For high levels of cooperation to be achieved, one needs there to be enough tags, enough games/pairings per round and, generalising a little, the probability for tag mutation must be higher than that for changing strategy [14].

Floating Point tags model of resource donation

This is described in complete detail in [21], and critiqued in [22] and [7].

The model has a fixed population of agents. Each agent has two (inheritable) characteristics: a tag (a floating point in [0,1]) and a tolerance (a floating point in [0,1]). Each simulation has is divided up into a finite number of discrete time periods (called 'generations'). Each generation is divided up into a number of phases:

1. creation/replication of new population from the old;
2. pairing and donation – for each agent in t
3. summing up the fitness scores;
4. reporting and statistics.

The only permanent variables that change are the tag and tolerance values for each of the 100 individuals. Temporary variables (that only have effect within a generation) are the donations and the fitness scores).

The parameters are as follows, (the value or range of values are in brackets).

? The population (100);

? The number of other agents each agent is ‘paired with’ for potential donation, P, (1, 2, 3, 4, 6, 8, 10);
? The cost to an agent of making a donation, c (0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6);
? The benefit gained from receiving a donation (1);
? The probability of tag mutation (0.1);
? The probability of tolerance mutation (0.1);
? The standard deviation of the Gaussian noise added to tolerance during mutation (0.01);
? The standard deviation of the Gaussian noise that is always applied to the tag value during reproduction (0).

In [21] *c*, the donation cost, and *P*, the number of pairings, are varied. In this paper the following are never varied: the population size, the tag range, the probability of tag or tolerance mutation; or the standard deviations of the tolerance mutation Gaussian noise.

The algorithm was as follows:

```
For each generation
    For each agent, A
        Randomly select another different agent, B
        If total score of B is strictly greater than the total score of A (selected bias)
        Then replace A with B
        Else keep A
    Next agent
    For each agent, a
        With probability 0.1 replace its tag with a new random value in [0, 1]
        With probability 0.1
            Add Gaussian noise (mean 0 sd 0.01) to tolerance
            If tolerance less 0 set to 0, if greater 1 set to 1
        Reset A’s score to zero
    Next agent
    For each agent, A
        Randomly select another different agent, B
        If (tag of A) – (the tag of B) < (the tolerance of A)
        Then add 1 to score of B and subtract c from score of a
    Next agent
Next generation
```

At the start of the first generation, the agents were initialised with all tag values and tolerance values being uniformly randomly selected from the interval [0, 1]. All values were independently drawn in each run of the simulation.

Specialist nutrition model

This is described in complete detail in [8]. The model structure of tags and tolerances in a [0,1] range comes from [21], the motivation for improving on that model came from [7]. The nutrition structure that was added on was suggested by reading texts on evolution and symbiosis, including [18].

Structure

There are fixed number of nutrition types and corresponding skills for gathering that type. There is a fixed amount of nutrition units distributed in each iteration, though each is allocated to a random nutrition type.

There is a variable population of individuals, each of which is characterised by the following characteristics: a tag value (a real from [0,1]); a tolerance value (a real from

[0,1]); a skill type (an integer); for each nutrition type: a reservoir holding an amount of that resource (a real), and an age (an integer).

Resource Flow

Resources are broadly conserved within each nutrition type. It enters via distribution (as well as via new non-reproduced entrants) and leaves via the life tax, the excess above the storage limit of individuals and with the death of individuals.

This principally enters the model via the direct distribution of units in the form of the different nutrition types. These are randomly distributed in units of 1 to these different types, then all those individuals who posses the appropriate skill gather that resource kind, equally sharing that resource.

Also new individuals (the initial population, the 2 new individuals that enter the population each time, and the progeny of individuals that reproduce) are given a fixed amount in each reservoir (*intialFood*). In the case of reproduction these amounts are subtracted from the corresponding reservoirs of the parent, so that reproduction does not increase the value of resources.

Each individual is now randomly paired with a fixed number (*numPairings*) of other individuals, without replacement. In each pairing event an amount of resource (maybe of several types) may be transferred from giver to recipient, if some conditions are satisfied. These conditions are: (1) The recipient must be one of those randomly chosen that time; (2) the difference in tag values must be strictly less than the tolerance of the giver; and (3) the giver must have more than a set amount (*foodOfTypeAboveWhichIsExtra*) in the corresponding reservoir. For each unit that is donated, *donationCost* is subtracted from the giver but only *donationBenefit* given to the recipient. The excess in the reservoir is shared equally among all recipients who qualify.

The individuals' reservoirs can only store up to a fixed maximum (*maxResevoir*). Above that resources are simply lost.

Each unit of time, a 'life tax' (of *foodUsageRate*) is subtracted from each reservoir of each individual.

If an individual has accumulated more than a fixed amount (*foodOfTypeNecessaryForReproduction*) in *all* of their reservoirs then they reproduce. The resources in the offspring are subtracted from the parent.

If an individual has less than a fixed amount (*foodOfTypeBelowWhichTagDies*) in *any* reservoir then it dies, also if it has reached its maximum age (*maxTagAge*). Resources of those that die are lost.

Dynamics

Once going, mutation can occur in tag or tolerance values. Apart from such mutation the main changes are in the number of individuals with particular skills and tags that results from their success in getting enough nutrition of all types so as (a) not to prematurely die and (2) to reproduce. The fact that all individuals will eventually die (if only because they reach the maximum age) makes the whole system more dynamic.

Initialisation

For the initial population, and the two entrants each iteration are given a random skill (one of 1, … *numFoodTypes*), a random tag value (from [0, 1]), a random tolerance (from [0, *MaxTol*]), a zero age and *initialFood* in each of its reserves.

Algorithm

```
Generate individuals, giving them all initialFood in each resource and each an independent
randomly chosen skill, tag and tollerance
For each generation
    Add maxNumNew new individuals (with random tags)
    Units of resource are randomly distributed amoung nutrition types
        – individuals with a skill equally share in that type
    For each individual, D
        For each pairing from 1 to numPairings
            Randomly choose another individual without replacement, O
            For each resource type, R
                If D has more of R than foodOfTypeAboveWhichIsExtra
                    and the absolute difference between D's and O's tag
                    is strictly less than D's tollerance
                Then
                    Subtract donationCost in R from D
                    Add donationBenefit in R to O
            Next resource type
        Next pairing
    Next individual
    For each individual
        subtract foodUsageRate from each resource
        If any resource < foodOfTypeBelowWhichTagDies then it dies
        If all resources > foodOfTypeNecessaryForReproduction
        then
            replicate indidivual (with possible mutation),
                subtracting new progeny's resources from parent
    Next individual
Next generation
```

Parameters

Below is an explanation of each of the major parameters, giving its name, its range (if this is varied), its default value and a brief explanation.

- *DonationBenefit:* (default value 0.95)
- *DonationCost*: (default value 0.95) – the proportion of a donation that the donee receives
- *MaxTime*: [1…8] (default value 1000) – the number of iterations the simulation runs for
- *FoodOfTypeAboveWhichIsExtra*: (default value 5.0) – if an individual has more than this of any type of resource it may donate this
- *FoodOfTypeBelowWhichTagDies*: (default value 0.0) – if any resource reaches this level the individual dies
- *FoodOfTypeNecessaryForReproduction*: (default value 4.0) – if all resources of an individual are above this level it will reproduce
- *FoodUsageRate*: (default value 0.25) – the amount subtracted from all resources held by all individuals as the life tax
- *InitialFood*: (default value 1.0) – how much new individuals (either born, initial population or new entrants) have in each of their stores
- *InitialPopSize*: [0,… 100] (default value 100) – the size of the initial population

? *NumNew*: [0,… 2] (default value 2) – how many new entrant enter the population each iteration
? *MaxReservoir*: [5, 20] (default value 7.5) – the most that an individual can store of each type of resource
? *MaxTagAge*: [15, 30] (default value 15) – the age at which individuals die (if they have not died of starvation before)
? *MaxTol*: [0.1, 1] (default value 0.1) – the maximum tolerance value that individuals can have
? *NumFood*: [200, 8] (default value 200) – how much food is distributed into the model each iteration
? *NumFoodTypes*: [1,…,4] (default value 3) – how many different types of resource are available and necessary for survival and reproduction
? *NumPairings*: [1,…,10] (default value 6) – the number of times each individual is randomly paired with others for possible donation
? *ProbMutVal*: [0, 1] (default value 0.05) – the probability that the tag and tolerance values are mutated during reproduction (independently for each)
? *SdMut*: [0, 1] (default value 0.05.) – the standard deviation of the Gaussian noise that is added to a value that is mutated

Skillworld model

This is described in [Norling ?ref?].

There are a fixed number of nodes, *N*, each of which have: a strategy flag (cooperate or defect); a list of those they are linked to (up to a maximum number, *D*); and a skill (an integer from 1 to *S*). Each generation tasks are distributed to nodes at random (an average of *J* per individual); the tasks are of different types (integers from 1 to *S*); if the node has the right skill for a job it does it, if not it asks a node it is connected with at random to do, if the other node has a cooperative strategy it does it (at a cost *C* to itself), if the task get done (by whomever) the original node get a reward, *R*. After all the jobs are processes (either done or not), the total rewards and costs are summed for each node, this is its score; then each node chooses one other node at random (from the whole population, the sequence of choices randomised each time) and compares its score with the chosen one – if its own score is less it copies its links and strategy and then links to that node (every time it makes a link which causes it to exceed its maximum number of links it randomly drops one). There is also a small chance (*M*) that all the links are dropped and a new, random one made, as well as a smaller probability (*MR*) that its strategy is changed.

The system is initialised with a random set of links and random initial strategy (the proportion of co-operators being given by *R*), but the same dynamics eventually appear even if one starts with all cooperators. In the SLACER variant of the above algorithm, when links are dropped they are dropped with a certain (high) probability (*DP*) so, on the whole, a few old links remain. The links in this model are symmetric, which has consequences for nodes being linked to – when a node drops a link to them they drop the link back, and if a node links to them, they link back and may have to drop a random link if they have reached its maximum. Also it is notable that the imitation strategy relies on the strategy being imitated along with the links.

The parameters were:

? *N*, Number of individuals, 2000
? *D*, Max links per individual, 20

? *M*, Link mutation rate, 0.01
? *MR*, Strategy mutation rate, 0.001
? *J*, Average number of tasks per individual per time step, 10
? *R*, Reward for successful completion, 1.0
? *C*, Cost of performing job for other, 0.25
? *R*, Initial proportion of altruists in population, 0.5
? *S*, Number of skills/task types, 5
? *DR*, the probability each old link is dropped in imitation and reset actions

P2P file-sharing model

This is described in **Error! Reference source not found.** it is an adaption of the model in [6] but to be as similar to the SkillWorld model described above whilst still keeping the P2P file-sharing structure.

There are a fixed number of individuals, *N*. Each of these has: a list of nodes to whom it can send/forward queries to (up to a maximum of *D*); a strategy bit which determines whether it will share its "files" with others; a store of files it has; a current target sequence of files and a "satisfaction" score (a floating point number).

The task they are in involved in is the following: "files", represented by integers are (initially) randomly distributed amongst the nodes (*IFSS* each). At any particular time a node will have a target file, represented by a low integer – the node aims to find files indicated by multiples of this integer, so if its base target is 7, it looks for {7, 14, 21, 28, …}. To do this is sends out queries for these files starting at the lowest it does not have – this query is propagated using a randomised "flood-fill" method which works as follows. A selection of its links is randomly selected without replacement, *QF* of them and any "live" queries (including its own) are forwarded to these if the query is not satisfied at their node and if the query has any "hops" left. Queries start with a "life" of *NH* – each time they are propagated along a link this count is decremented by 1. When the query's count reaches 0, it is no longer propagated. If a node receives a query for a file it has and it is a cooperating node then it "sends" the file back to the node that originated the query. Each hop of a query, generation of a new query, or sending back a file takes one unit of time. Lots of queries and files are being processes in parallel, but each node waits for a successful (or otherwise) result from one query before initiating another. Every time a file receives a file it was looking for and does not have it increases its satisfaction measure by 1, every time a node sends a file it has it decreases its satisfaction measure by *SFS*. To keep the dynamics going for this task, there is a small probability (*PAF*) of adding new files from "external" sources each iteration and a small probability (*PNT*) of the node deciding upon a new base target.

The SLACER algorithm is applied so that the above task/process of looking for files is done for a certain number of iterations (*J*), then each node picks a random other: compares its total score for the period with the other's; if the other's is higher it links to that node; (with a high probability *DR* for each link) drops its other links; and copies both links and strategy of the other node. With another small probability (*M*) it drops its existing nodes (with probability the high probability, *DR*, for each link) and randomly links to a node. Finally, there is a small probability of it changing its strategy, *MR*.

? *N*, Number of individuals, 250
? *D*, Max links per individual, 20

? *QF*, Num Links a Query is Forwarded Along, 3
? *M*, Link mutation rate, 0.02
? *MR*, Behaviour mutation rate, 0.002
? *NH*, Maximum number of query hops, 3
? *SFC*, Send file cost, 0.05
? *IFSS*, Initial file store size, 10
? *PAF*, Probability of adding a new file to a node's store, 0.02
? *PNT*, Probability of getting a new target set of files, 0.02
? *NI*, Number of iterations before the SLAC(ER) algorithm is applied
? *DR*, the probability each old link is dropped in imitation and reset actions
? *J*, Number of iterations per generation, 50