Emergent Social Rationality in a Peer-to-Peer System

Andrea Marcozzi and David Hales

Department of Computer Science, University of Bologna, Mura Anteo Zamboni 7, 40127 Bologna, Italy {marcozzi, hales}@cs.unibo.it

Summary. Many peer-to-peer (P2P) applications require that nodes behave altruistically in order to perform tasks collectively. Here we examine a class of simple protocols that aim to self-organise P2P networks into clusters of altruistic nodes that help each other to complete jobs requiring diverse skills. We introduce a variant (called ResourceWorld) of an existing model (called SkillWorld) and compare results obtained in extensive (ten billion interactions) simulation experiments. It was found that for both model variants altruistic behavior was selected when certain cost / benefit constraints were met. Specifically, ResourceWorld selects for altruism only when the collective benefit of an action is at least as high as the individual cost. This gives a minimal method for realizing so-called "social rationality" where nodes select behaviors for the good of the collective even though actions are based on individual greedy utility maximization. Interestingly, the SkillWorld model evidences a kind of *super-altruism* in which nodes are prepared to cooperate even when the cost is higher than the benefit.

1 Introduction

For many applications peer-to-peer (P2P) systems require their member nodes (or agents) to behave in a socially beneficial (non-egotistical) way. Kalenka and Jennings [7] termed this requirement as the Principle of Social Rationality: if an agent has a choice of actions it should chose the action that maximizes the social utility (sum of all agent utilities in the system). This principle can be contrasted with classical individual rationality that states agents should select actions that maximize their individual utility. However, developing protocols for realistic P2P systems that adhere to the principle of social rationality is very difficult and potentially so costly as to negate the benefits. This is because P2P systems have no central control, are potentially huge (composed of millions of nodes) and have high node turnover (with users continually entering and leaving the system). In addition, selfish or malicious nodes can get into the system via hacked client programs. These factors mean that individual nodes, even if they wish to follow a socially rational principle, often will not have enough information to gauge the effects of their actions on others. Recently, simple locally adaptive protocols have been proposed that claim to produce socially rational outcomes through a process of self-organisation even though nodes only act on their own utility values. In this approach nodes preferentially copy other nodes (by duplicating their behaviour and links) that have higher utilities. However, in these previous works only specific scenarios are considered in which certain plausible utility values are selected. In this paper we introduce a variant (*ResourceWorld*) of one such existing P2P scenario [3] (*SkillWorld*). For both models we explored a large space of different cost / benefit values to check if the protocols maximized the collective utility or not. In ResourceWorld we found that if the collective cost of an action was less than or equal to the collective benefit the protocol self-organized the network to a state where nodes selected this action. For SkillWorld we found a less socially rational rule. Additionally we performed a large set of experiments exploring different values of Network Degree (d) and Number of Skills involved in the system (sn) with the aim to find combinations of them leading to socially good performances of the system. Finally some experiments testing the robustness of the protocols under churn condition are exposed.

2 The ResourceWorld Scenario

The *ResourceWorld* model, which takes inspiration from a previous model [4], represents the situation in which nodes in a P2P network can store and serve a single resource from a set (R). Each node may have a maximum of 20 links to other nodes (peers). Each link is bidirectional: if node a is connected to node b, even node b is connected to node a. Links are undirected so the entire network can be considered as an undirected graph where each vertex is a node and each edge is a link.

The state variables of each node is shown in table 1.

Table 1. Nodes state

Parameter	**Value**
Altruism flag	$A \in \{0, 1\}$
Resource/Skill type	$R \in \{1, 2, 3, 4, 5\}$
View	$d = 20$ links to other nodes
Utility	$U \in \mathbb{R}$

The Resource, which indicates the ability held by a single node, is the only parameter which does not evolve: it is not copied during the reproduction phase (see section 4), but it may “mutate” (that is change) with a very small probability. Periodically (at each iteration cycle, where a cycle is all nodes firing in a random order) with probability 0.5, nodes receive a *request* or job (J) to be completed. The request is produced by selecting at random a value from a set of 5 elements ($J \in \{1,2,3,4,5\}$) and the receiving nodes, in order to complete the request, must hold the appropriate resource. Suppose node i having a certain R_i and a certain strategy A_i receives a request J. Three situations can take place:

- $J = R_i$: in this case (cf. figure 1a) i can satisfy the request by itself (doesn't matter if A is set to 0 or 1) and node i increases its utility by a benefit payoff b ($U_i := U_i + b$) ;
- $J \neq R_i \wedge A_i = 1$: in this case (cf. figure 1b) i itself can not satisfy the request but since it is a cooperating node ($A_i = 1$), it can pass the request to its neighbors (given in the node view) in turn. If one of them (say j) has the appropriate resource ($R_j = J$), then the request can be satisfied; in this case node j gets the benefit payoff ($U_j := U_j + b$) while node i pays a cost payoff ($U_i := U_i - c$); if none of neighbors has the requested resource, the request J can not be completed and no payoffs are given;
- $J \neq R_i \wedge A_i = 0$: in this case (cf. figure 1c) since node i is a selfish node, it cannot pass the job to its neighbors, even though they have the needed resource.

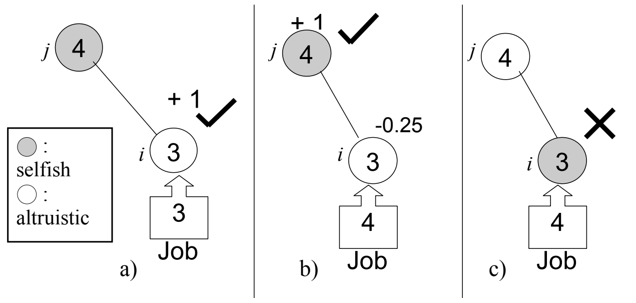

Fig. 1. An illustration of the "ResourceWorld" model. Shading of nodes represents strategy; the number inside the node indicates the *resource*. In (a) node i receives a certain request J: since it has the appropriate resource to satisfy it ($R_i = J$) it gains the benefit payoff ($b = 1$). In (b) $R_i \neq J$: since node i is an altruistic node ($A_i = 1$), it can pass J to its neighbor j which holds the right resource ($R_j = J$); in this case j earns the b payoff and i pays the c payoff ($c = 0.25$). In (c) since node i is a selfish node, it can't pass the request to its neighbor j.

This means that each node receiving a request, if not able to satisfy it with its own resource, it is willing to pay a cost in order to have the request completed.

This minimal scenario represents a situation in which user-level requests of *resources* (disk space, files, programs, etc..) are supplied to a P2P network composed of nodes holding just one of such needed resources. Nodes should be able to self-organize in a way that makes the most of the created request to be satisfied in the shortest possible amount of time. This organization task is performed by the SLAC protocol [5] which is described in section 4.

3 The SkillWorld Scenario

The *SkillWorld* (SW) model [4] is very similar to the ResourceWorld (RW) model we have just described but it represents a different scenario. Here, what we have previously called resource, is called skill (resource and skill are the same thing, we use different names just to distinguish the two models). This skill represent the ability of a node to work on a certain Job (J). The state variables of the nodes is the same reported in table 1. The difference with RW, is in the way by which the nodes interact.

In SW when a node hasn't the right skill to execute a job (J), it looks for an *altruistic* neighbor to exploit, in order to complete the job and receive a benefit. Once selected, the altruistic neighbor will have to pay a cost. We can see this as a "favour" that the selected node does to its friend, but for this it pays a cost in terms of personal resources (disk space, cpu, ecc..). Figure 2 gives an idea of what happens in this model.

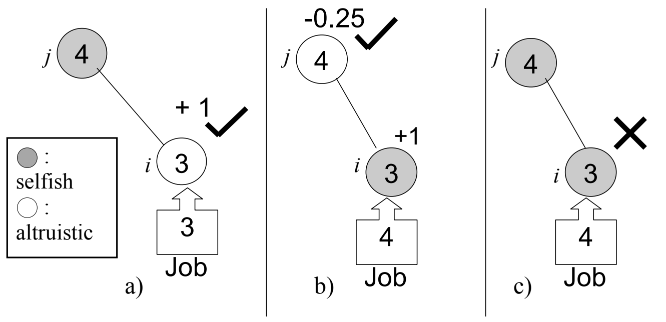

Fig. 2. An illustration of the "SkillWorld" model. Shading of nodes represents strategy; the number inside the node indicates the *skill*. In (a) node i receives a certain job J: since it has the appropriate resource to satisfy it ($S_i = J$) it gains the benefit payoff ($b = 1$). In this case we don't care about the altruism flag. In (b) $S_i \neq J$: node i will look among its neighbor list for an altruistic node; in this case j holds the required skill and is altruistic, so node i earns the b payoff and j pays the c payoff ($c = 0.25$). In (c) since node i has just one selfish neighbor, the job is not executed.

4 SLAC in ResourceWorld and SkillWorld

The SLAC protocol (Selfish Link-based Adaptation for Cooperation) has the ability to produce high levels of cooperation in P2P networks while performing some tasks. It has already been covered in details elsewhere [5, 9] so here we give just a brief overview.

The SLAC algorithm specifies how nodes should update their strategies and links. Figure 3 shows the pseudo-code of the relating algorithm.

At each cycle of the simulation task, with a certain probability, each node selects a random node from the entire network and their utilities are compared. Suppose node i has an average utility greater than j ($U_i > U_j$): in this case j copies node i strategy (not the resource); j drops all its links and moves to i's neighborhood (copies all i's links and adds a link to i itself). After this, "mutation" is applied with different probabilities, separately, to j's links, to j's strategy and to j's resource / skill. This involves changing the links, strategy and resource / skill. The strategy is flipped, the resource is replaced by uniformly randomly selected new resource value and links are wiped and replaced with a single link to a randomly selected node from the population. Mutation of the resource is for introducing some noise in the system.

```
At each cycle with probability $rp$ for node $i$:
    select a random node $j$ from the population:
    if ($U_i \leq U_j$) then
        copy strategy from $j$
        drop each link from $i$
        copy each link from $j$
        link $i$ to $j$
        with prob(M) mutate strategy of $i$
        with prob(MR) mutate links of $i$
    reset utility $U = 0$;
```

Fig. 3. The generic SLAC algorithm. Each node executes this algorithm.

5 Experiments

We tested our protocol over three different settings; we called these *Utility profile, Network Degree and Skill Number profile, Churn profile*:

- **Utility**: we varied the benefit and cost payoffs, keeping all the other parameters fixed, in order to discover when the two protocols performed better;
- **Network Degree and Skill Number**: we varied the number of skills in the system and the network degree, keeping all the other parameters fixed, in order to discover when the two protocols performed better;
- **Churn**: we did several experiment to test the robustness of our protocols under churn conditions

Simulations were carried on *Peersim* [10], an open source P2P systems simulator platform, using the *Newscast* protocol [6] for the management of the overlay topology. The time is divided in cycles and in each cycle each node performs the specific actions described in the previous sections. In each experiment we checked if SLAC maximizes the collective utility or not.

5.1 Utility profile: Experimental Configuration

In this setting we performed a massive number of experiments with both models modifying the utilities with the aim to explore a large space of possible values (varying benefit and cost values). The configuration we adopted in the experiments is shown in table 2.

Table 2. Experiments configuration

Parameter	**Value**
Network size (N)	4000
Network degree (d)	20
Initial topology	*random*
Nodes' Strategy Initialization	$A_i = 0$ (*selfish*)
Skill number	$SN = 5$
Links Mutation *MRT*	0.01
Strategy (*MS*) and Resource Mutation (*MR*)	0.0025
Benefit (b) and Cost (c) payoff	$\{0.1 \ldots 2.0\}$ steps by 0.1

For each configuration we performed 10 different runs and we took the average of the results: this means that we performed $20 \times 20 \times 10 = 4000$ different runs.

5.2 Utility profile: Experimental Results

The performance measure we adopted is the percentage of completed jobs (*Pcj*). All the results that we show are from an average of ten runs per configuration. We found that in **ResourceWorld** to obtain a good Pcj (by good Pcj we mean a value greater than 80%), the benefit payoff must be greater

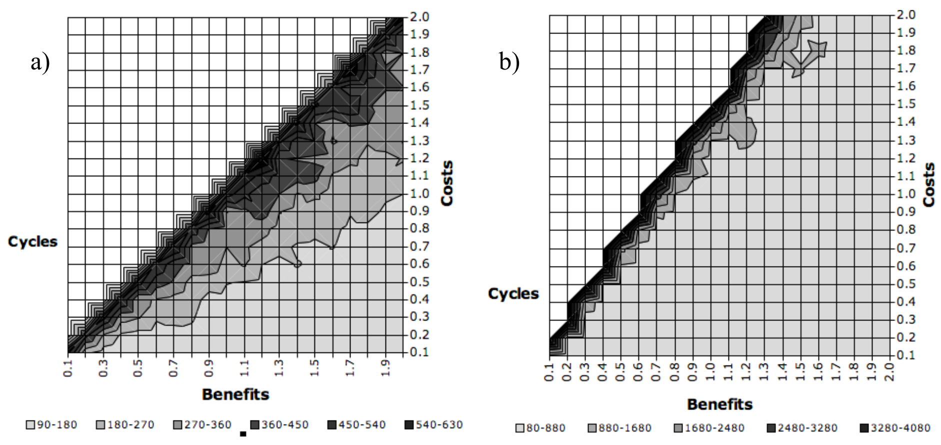

Fig. 4. Map diagram indicating the number of cycles needed to obtain a good level of Pcj ($> 80\%$): (a) relates to the RW model; (b) relates to the SW model. The used parameters for both models are those indicated in table 2. In both diagrams, the top left area indicates that no good results are obtained.

than or equal to the cost payoff ($b \geq c$). When $b < c$ we obtain a very low Pcj (ranging from 25% to 35%); when $b = c$, a good level of Pcj can be obtained, but the system will take longer to achieve this.

From figure 4a we note that in order to obtain a good level of Pcj in a small amount of time (less than 180 cycles) the cost payoff must be smaller than half of the benefit payoff ($c < \frac{1}{2}b$). The larger the b/c ratio the sooner 80% Pcj is reached.

We compared these results from RW with similar experiments using the **SkillWorld** scenario. What we found here is that to obtain a level of Pcj greater than 80%, the cost value can be greater then the benefit. Good levels of Pcj are obtained according to this rule: $c < \frac{3}{2}b$ (see figure 4b). We also found that in order to obtain a good level of Pcj with a small number of cycles, the cost payoff should be smaller than half of the benefit payoff ($c < \frac{1}{2}b$). This also happens in RW.

As we said, the above mentioned results are from an average of 10 simulation runs per configuration. In RW the standard deviation (σ) depends on the ratio between the benefit and the cost payoff: if $c \geq \frac{1}{2}b$ then the standard deviation is approximately half of the average number of cycles ($\sigma \approx \frac{1}{2}n$, where n is the number of cycles); if $c < \frac{1}{2}b$, then $\sigma \approx \frac{1}{6}n$. Figure 5 gives an idea of this.

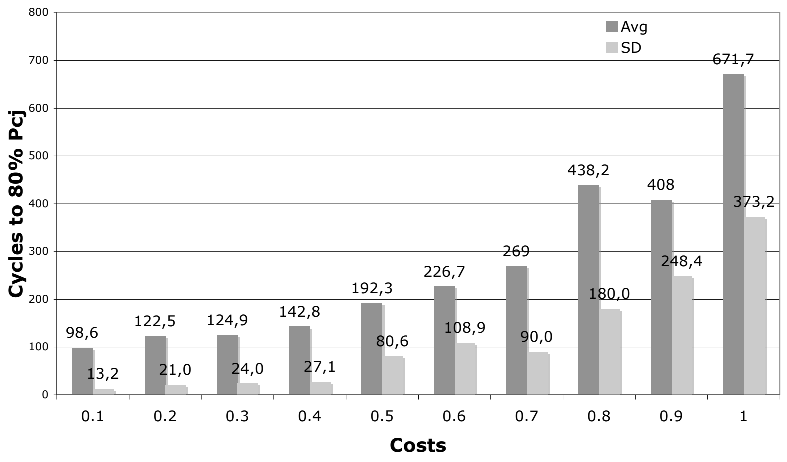

Fig. 5. ResourceWorld: Average number of cycles needed to obtain a good level of Pcj exploring several cost payoffs: $b = 1$, $c = 0.1 \dots 1$. The other parameters are the same used for results in figure 4. We note that when $c \geq \frac{1}{2}b$, both average and standard deviation increase.

5.3 Network Degree and Skill Number profile: Experimental Configuration

We have seen from the previous experiments that both in RW and SW a social behavior is selected when $c < \frac{1}{2}b$. In order to explore further the dependence on the parameters of the model we varied the Network Degree value (d) and the Number of Skills value (sn).

Also here we performed 10 different runs for each configuration and then we took the average of the results. Table 3 shows the configuration we used.

Table 3. Experiment configuration: Network Degree and SkillNumber setting

Parameter	**Value**
Network size (N)	4000
Network degree (d)	$d \in \{10 \dots 21\}$
Initial topology	*random*
Nodes' Strategy Initialization	$A_i = 0$ (*selfish*)
Skill number	$sn \in \{4 \dots 10\}$
Links Mutation *MRT*	0.01
Strategy (*MS*) and Resource Mutation (*MR*)	0.0025
Benefit (b) payoff	1.0
Cost (c) payoff	0.2

5.4 Experimenal results: Network Degree and SkillNumber setting

As we have seen from the previous experiments, good results in a short time and with a small variance can be obtained with $b = 1$ and $c = 0.2$. We adopted these values to perform the new experiments. A first interesting result is shown in figure 6: it gives the percentage of cooperating nodes with different network degrees. For each network degree value, we performed simulations with the number of skill values varying from 4 to 10; then we took the average of these results. The figure shows how with $d < 13$ the percentage of cooperating nodes is quite low and the standard deviation is high. On the other hand with $d \geq 13$ we always obtained a very high number of cooperating nodes indicating a stable behavior of the nodes. For each configuration we performed 180 cycles. For $d < 13$ we also performed additional experiments up to 5000 cycles but the obtained results are very similar to those shown.

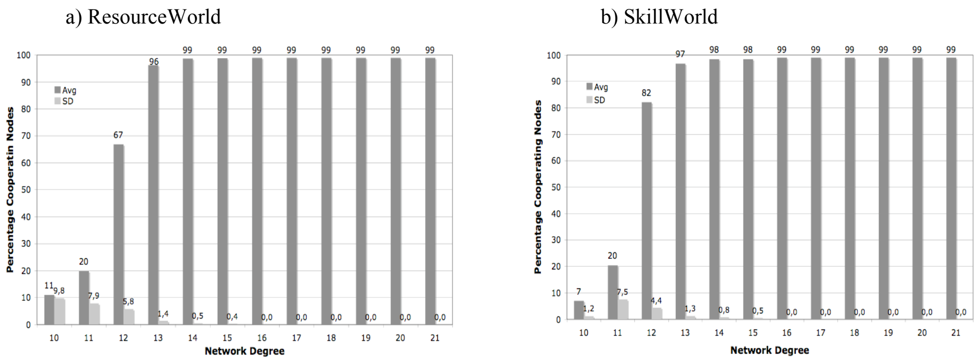

Fig. 6. Percentage of cooperating nodes in RW (a) and SW (b). With both models a high level of cooperating nodes is obtained when $d \geq 13$. The parameters used for these experiments are shown in table 3.

Both in RW and in SW a high number of cooperating nodes is necessary for achieving a good Pcj, so this result helps in the understanding of the next two diagrams (figures 7 and 8). The diagrams indicate the percentage of completed jobs obtained making experiments with different combinations of d and sn. We can note both in RW and SW that when $d \geq 13$ good results can be obtained according to two rules: in RW a good Pcj can be obtained when, for $d \geq 13$, the number of skills involved in the system is smaller than or equal to the half of d minus 1 ($sn \leq \frac{d-1}{2}$); in SW instead, for $d > 13$ it is enough that sn is smaller than or equal to the half of d plus 2 ($sn \leq \frac{d+2}{2}$).

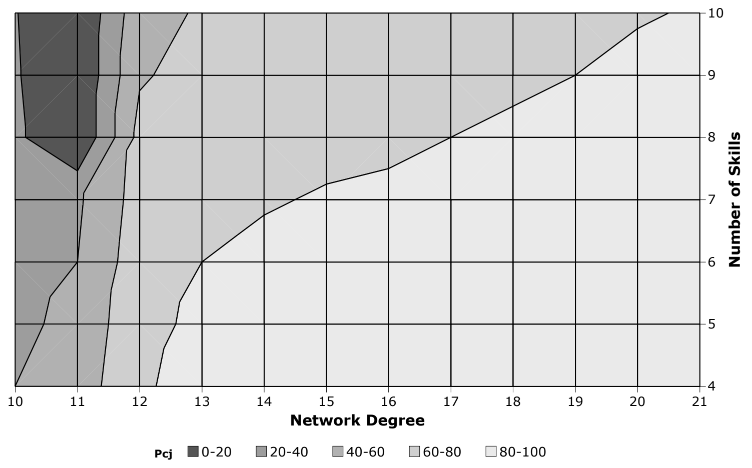

Fig. 7. ResourceWorld: map diagram indicating the Pcj that can be obtained combining different values of d and sn. The used parameters are those indicated in table 3.

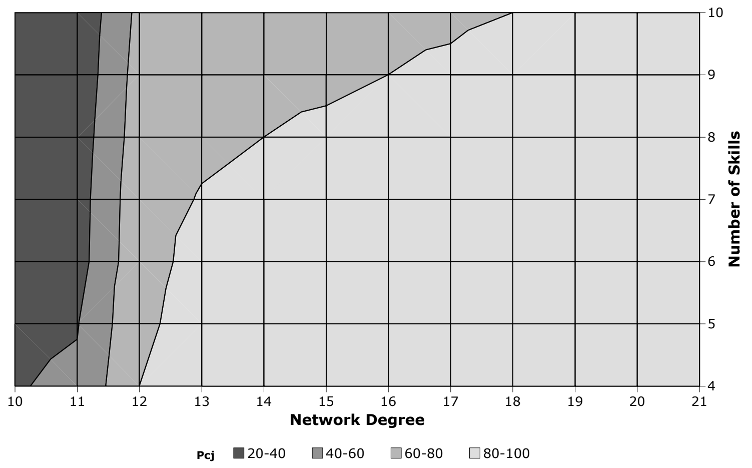

Fig. 8. SkillWorld: map diagram indicating the Pcj that can be obtained combining different values of d and sn. The used parameters are those indicated in table 3.

5.5 Robustness to Churn

A key requirement for robust P2P overlay networks is robustness to "churn" (turnover of nodes), so we did some experiments in order to test the robustness of the two protocols under churning conditions. We introduced various amounts of churn – where old nodes leave and new nodes join the network. In these experiments we reset randomly selected nodes to defect strategy and random skill every 10 cycles. We performed 10 runs for each configuration and we took the average. Table 4 shows the obtained results. We found that RW is more robust to churning than SW even though with 10% of churning results are similar. Figure 9 shows a typical single run of 5000 cycles.

We also performed experiments in which 50% of nodes were replaced at one cycle after high cooperation is reached: we found that in both models high Pcj quickly reforms within few cycles. This may be due to the fact that SLAC already incorporates some noise in the form of mutation to both links and strategies, driving its evolutionary dynamics and, as we have shown, can quickly recover from states of complete defection and link disconnection **??**.

Table 4. Churn: average Pcj obtained with different amounts of churn

Churn %	**ResourceWorld**	**SkillWorld**
3%	0.74	0.40
5%	0.61	0.39
10%	0.45	0.41

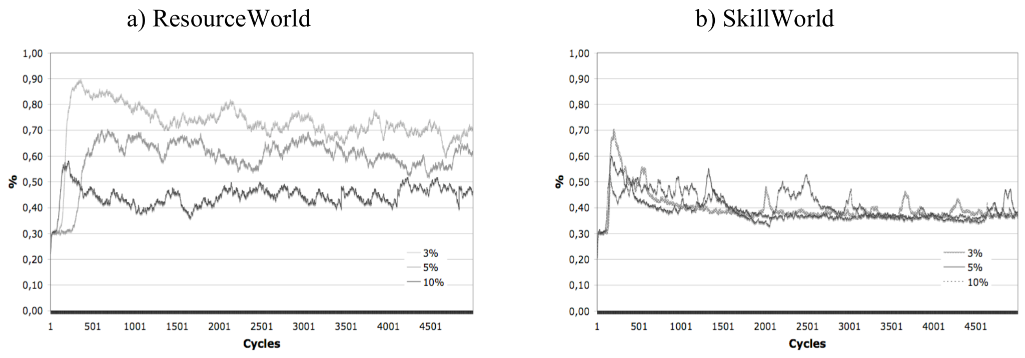

Fig. 9. Churn: Single run experiments. Every 10 cycles a percentage of nodes are reset. We experimented with 3, 5 and 10 percent.

6 Discussion and Future Work

The SLAC algorithm implements nodes that myopically and greedily attempt to maximize their own utility by copying other random nodes if they perform better. In our simulations even though nodes have no knowledge of the underlying game or of the state of the other nodes, a high number of completed jobs is produced in both models. This requires the self-organisation of both altruism and internal specialized structure.

In this work we took the ResourceWorld (RW) and SkillWorld (SW) models, we varied the cost / benefit values and we found two interesting rules (see table 5).

Table 5. Utility profile: Rules leading to high Pcj in RW and SW.

Model	**Rule**
ResourceWorld	$\frac{b}{c} \geq 1$
SkillWorld	$\frac{b}{c} > \frac{2}{3}$

Table 6. Network Degree and Skill Number profile: Rules leading to high Pcj in RW and SW.

Model	**Rule**
ResourceWorld	$sn \leq \frac{d-1}{2}$
SkillWorld	$sn \leq \frac{d+2}{2}$

Varying these utilities, we found that in RW a high number of jobs can be completed when the cost payoff is smaller than or equal to the benefit payoff. In SW, even when the cost payoff is greater than the benefit (for certain values only), cooperation is sustained.

We thought that these results may be influenced by the topology of the network or by the number of skills adopted by nodes, so we did tests exploring different combinations of *Number of Skill* and *Network Degree* values. The obtained results are interesting and show that a high level of Pcj can be reached in both models under the conditions reported in table 6: both with RW and SW we found that with $d > 13$ high cooperation can be obtained. We currently don't know why this happens but we speculate that since links in our model do not incur a cost to the node then the more links a node can maintain then the more likely it is that it will find a cooperative partner with the required skill. Future work will investigate this.

More experiments investigating the topology evolution could also illuminate the results. We have seen from previous works [3] that with SLAC, a network starting from any initial topology quickly self-organizes into a set of disconnected highly clustered components (*tribes*). We think that the same happens here, through some kind of "group-selection" between these tribes leading to the selection of more socially optimal behavior.

However, what is interesting and unique in our results here is that under certain kinds of interaction rules cooperative groups may be selected *even when the cost of cooperative is greater than the benefit*. This indicates a kind of *super-altruism* which is not explicable in terms of overall benefit to the group. To put this another way, although following an evolutionary processes - myopically optimising - nodes sacrifice more of their own utility to help another than the other gains. We believe this is interesting because we believe such behaviour can be seen in human societies but is not easily explainable with current evolutionary theory or rational action models. It would appear that this results from the particular interaction structures and group dynamics for a given scenario. We aim to explore this in more detail in future work.

It is important to note that although selection appears to operate at the group level it is the result of selection (it is an emergent property) operating on the individual node level. What is significant here is that for the RW model this process is sufficient to select a socially rational behavior in the nodes, maximizing collective or social benefit over individual benefit. The SW results however show that such behavior is not always selected and future work will hopefully allow us to give a general theory which can predict what kinds of scenario will and wont produce social benefit following our simple approach.

From the point of view of actually implementing such protocols in P2P systems there are still some potential problems concerning possible malicious behavior. We have assumed that nodes follow the protocols correctly but malicious nodes could attempt to subvert the protocol by lying about their utilities and / or strategies. Although such attacks do damage system performance experiments with similar protocols have indicated that in many cases the damage my be acceptable (i.e. not completely destroy the functioning of the network) [1]. However, perhaps our results can be applied to understanding human social interactions particularly in organisations requiring the co-ordination of jobs and skills to achieve their goals.

References

1. S. Arteconi, D. Hales. Greedy Cheating Liars and the Fools Who Believe Them. *University of Bologna, Dept. of Computer Science, Technical Report UBLCS-2005-21*, December 2005.
2. D. Hales. Cooperation without memory or space: Tags, groups and the prisoner s dilemma. *In MABS 00: Proceedings of the Second International Workshop on Multi-Agent-Based Simulation - Revised and Additional Papers*, pages 157166, London, UK, 2001. Springer-Verlag.

3. D. Hales. Self-Organizing, Open and Cooperative P2P Societies - From Tags to Networks. *Proceedings of the 2nd Workshop on Engineering Self-Organizing Applications (ESOA 2004), LNCS 3464*, pp.123-137. Springer, 2005.
4. D. Hales. Emergent Group-Level Selection in a Peer-to-Peer Network. *Complexus* 2006;3, 2006.
5. D. Hales and S. Arteconi. SLACER: A self-organizing protocol for coordination in peer-to- peer networks. *IEEE Intelligent Systems*, 21(2):29-35, Mar/Apr 2006.
6. M. Jelasity, M. van Steen. Large-Scale Newscast Computing on the Internet. *Internal Report IR-503, Vrije Universiteit Amsterdam, Department of Computer Science*, Amsterdam, The Netherlands, 2002.
7. S. Kalenka and N. R. Jennings. Socially Responsible Decision Making by Autonomous Agents. Cognition, Agency and Rationality. *(eds. Korta, K., Sosa, E., Arrazola, X.)* Kluwer 135-149, 1999.
8. E. Koutsoupias and C. H. Papadimitriou. Worst-case Equilibria. *STACS'99*.
9. A. Marcozzi, D. Hales, G. Jesi, S. Arteconi, O. Babaoglu. Tag-Based Cooperation in Peer-to- Peer Networks with Newscast. *In Self-Organization and Autonomic Informatics (I)* edited by: H. Czap, R. et al. IOS Press, The Netherlands, 2005.
10. *http://peersim.sourceforge.net*.