Emergent socially rational Utilitarianism in a Peer-to-Peer System

DELIS

Dynamically Evolving, Large-Scale Information Systems

Overview

University of Bologna - Dept. of Computer Science - Andrea Marcozzi and David Hales

Abstract

For many applications peer-to-peer (P2P) systems require their member nodes (or agents) to behave in a socially beneficial way. This requirement is known as the "Principle of Social Rationality" (Kalenca 1999): if an agent has a choice of actions it should chose the action that maximizes the social utility. This principle can be contrasted with classical individual rationality that states agents should select actions that maximize their individual utility.

Recently, simple locally adaptive protocols have been proposed (Marcozzi 2005, Hales 2005) that claim to produce socially rational outcomes through a process of self-organization even though nodes only act on their own utility values. We took the SkillWorld model (Hales 2005) and modify the utilities to explore a large space of possible values. In each case we checked if the protocol maximized the collective utility or not. This new model is called *"ResourceWorld"*.

The ResourceWorld Model

The model represents the situation in which nodes in a peer-to-peer network can store and serve some *resources* (***R***). Each node may have a maximum of 20 links to other nodes (peers). Each link is bidirectional.

Parameter	Value
Altruism flag	$A \in \{0,1\}$
Resource/Skill type	$R \in \{1,2,3,4,5\}$
Maximum view size	$d = 20$
Utility	$U \in \mathbb{R}$

Table1. Nodes state

The resource (or *skill*) is the only parameter which does not evolve: it is not copied during the reproduction phase (SLAC), but it just mutate with a very small probability. Periodically (at each iteration cycle) with probability *0.5*, nodes receive a request for a *job* (***J***) to be completed. The request is produced selecting at random a value from a set of 5 elements (*J* is a number from 1 to 5); the receiving nodes, in order to complete the request, must hold the appropriate resource. Figure 1 shows what happens with this model.

The SLAC algorithm

The SLAC algorithm (Hales 2006) assumes that peer nodes have the freedom to change behaviour and drop and make links to nodes they know about. In addition, it is assumed that nodes have the ability to discover other nodes randomly from the network, compare their performance against other nodes and copy the links plus (some of) the behaviours of other nodes. We have seen from previous works (Marcozzi 2005) that it has the ability to produce high levels of cooperations in P2P networks.

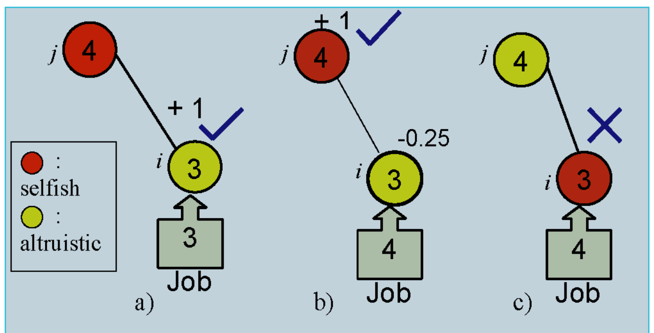

Fig. 1. An illustration of the "ResourceWorld" model. Shading of nodes represents strategy; the number inside the node indicates the *resource*. In (a) node *i* recives a certain request *J*: since it has the appropriate resource to satisfy *J*, it gains the benefit payoff (*b=1*). In (b), *i* doesn't hold the required resource: since it is an altruistic node, it can pass *J* to its neighbor *j* which has the righ resource; in this case, *j* gets the *b* payoffs and *i* pays the *c* payoff (*c=0.25*). In (c), since node *i* is a selfish node, it can not pass the job to its neighbor *j*.

Experiment configuration and results

We performed a massive number of experiments with this model; we played with the utilities (see figure 3) with the aim to explore a large space of possible values. Both benefit (***b***) and cost (***c***) payoffs were ranging from *0.1* to *2.0* by steps of *0.1*. For each configuration we performed 10 different runs and we took the average of the results. Hence we performed

20 X 20 X 10 = 4000

different runs.

The utility measure we adopted is the percentage of Completed Jobs (Pcj), which is the average between the number of request submitted and the number of request completed at each cycle. Figure 2 shows the number of cycles needed to obtain a good level of Pcj (by good Pcj we mean a value greater than 80%) fixing the benefit payoff (b=1) and varying the cost (from 0.1 to 1). From figure 3 we found that to obtain a good Pcj, the benefit payoff must be greater or equal than the cost payoff (***b ≥ c***). When b < c we obtain a very low Pcj (ranging from 25% to 35%); when b = c, the system will take longer to achieve this. To obtain a good level of Pcj in a small time, c must be smaller than the half of b (***c<1/2b***). The bigger is the difference between b and c, the sooner 80% Pcj is reached.

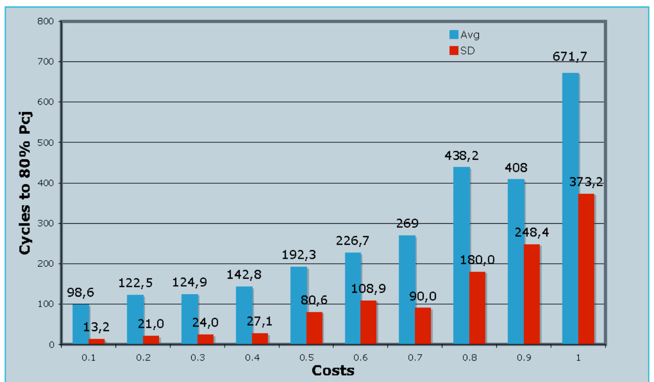

Fig. 2. Average number of cycles (and standard deviation) needed to obtain a good level of Pcj exploring several cost payoffs: b=1; c=0.1 ... 1. When c≥1/2b, both average and standard devistion increse.

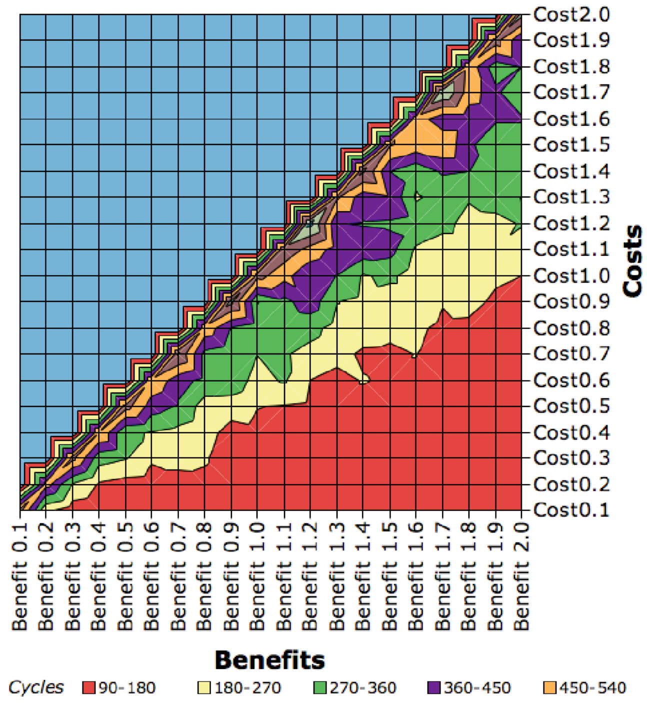

Fig. 3. Map diagram indicating the number of cycles needed to obtain a good level of Pcj (> 80%). The top left half of the map indicates that *no good results* can be obtained (very low *Pcj*). The right down half of the map indicates that good results can be obtained according to the rule ***c ≤ b***.

Time to 80% Pcj	Rule
$181 \ldots 740$ cycles	$c \leq b$
$90 \ldots 180$ cycles	$c < \frac{1}{2}b$

Table 2. Rules discovered from the experiments

Future Works

Varying the benefit and cost payoff we obtained two interesting rules (see table 2). It seems that under these rules the system gives good outcomes.

We think that these results may be influenced by the topology of the network and by the number of skills. What happens if we perform the same experiments with networks having a small degree? Or what happens, if the number of skill involved is smaller or bigger then *20*? This might be the subject of future research.

A. Marcozzi, D. Hales, G. Jesi, S. Arteconi, O. Babaoglu (2005) 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, Netherlands.

D. Hales (2005) Emergent Group-Level Selection in a Peer-to-Peer Network. Proceedings of the European Conference on Complex Systems (ECCS'05), Nov. 14th, Paris, France. Publishers: i6doc, Belgium.

D. Hales and S. Arteconi (2006) SLACER: A self-organizing protocol for coordination in peer-to-peer networks. IEEE Intelligent Systems, 21(2):29-35.

E. Koutsoupias and C. H. Papadimitriou (1999) Worst-case Equilibria. STACS'99.

S. Kalenka and N. R. Jennings (1999) Socially Responsible Decision Making by Autonomous Agents. Cognition, Agency and Rationality. (eds. Korta, K., Sosa, E., Arrazola, X.) Kluwer 135-149.

Contact
Andrea Marcozzi
Dept. of Computer Science
Mura Anteo Zamboni 7
40127 Bologna, Italy

Fax: +39 051 2094510
Phone: +39 051 2094515
E-Mail: marcozzi@cs.unibo.it
URL: www.cs.unibo.it/~marcozzi

Contact
David Hales
Dept. of Computer Science
Mura Anteo Zamboni 7
40127 Bologna, Italy.

Fax: +39 051 2094510
Phone: +39 051 19980461
E-Mail: dave@davidhales.com
URL: www.davidhales.com

DELIS: Integrated European Project founded by the "Complex Systems" Proactive Initiative within the Sixth Framework Programme