"Trusting Agents for trusting Electronic Societies", Rino Falcone, Suzanne Barber, Jordi Sabater and Munindar Singh editors. 2005.

A Protocol for a Distributed Recommender System

José M. Vidal

University of South Carolina, Columbia SC 29208, USA,
vidal@sc.edu,
http://jmvidal.cse.sc.edu

Abstract. We present a domain model and protocol for the exchange of recommendations by selfish agents without the aid of any centralized control. Our model captures a subset of the realities of recommendation exchanges in the Internet. We provide an algorithm that selfish agents can use for deciding whether to exchange recommendations and with whom. We analyze this algorithm and show that, under certain common circumstances, the agents' rational choice is to exchange recommendations. Finally, we have implemented our model and algorithm and tested the performance of various populations. Our results show that both the social welfare and the individual utility of the agents is increased by participating in the exchange of recommendations.

1 Introduction

In the last few years we have seen the proliferation of recommender systems [1]. Almost every web retailer seems to have software that can recommend an item for purchase. These recommendations are made based on the users' purchasing and browsing history, that is, what everyone bought and what everyone looked at. The popularity of recommender systems confirms their effectiveness in recommending items that the users find appealing. However, we find one major drawback with these systems—they are all centralized implementations. This design choice leads to several problems.

- The central server becomes a central point of failure and a central target for attack by those seeking to gather information about all users.
- It assumes that users are willing to give the information to the central server. Users do not receive anything in return for this information.

Our goal is to find ways to enable the emergence of a truly distributed recommender system in an Internet populated by selfish agents. To achieve this goal our system must provide the agents with an incentive to trade recommendations with each other and a protocol for enabling this trade. We also eschew the idea of a system where agents buy and sell recommendations for several reasons.

- The processing of payments adds a large cost to the transaction and requires the use of a trusted third party, which leads us back to a semi-centralized solution.

Table 1. Summary of notation.

D	set of documents, $d \in D$.
A	set of agents, $i \in A$.
$L_i(d)$	a proposition that is true if i likes d.
$R_i(d)$	a proposition that is true if i has read d.
P_r	the payoff for reading a liked document.
C_r	the cost of reading a document.
C_m	the cost of sending a message.

– Users have historically been reticent to allow automated agents to spend real money unless the agents are given very specific rules of behavior.

Instead, we plan to use information itself as a currency. Since agents are already presumed to have access to the user's preferences they can trade these preferences with other agents for more recommendations that might be useful to the user. The problem then becomes that of ascertaining the value of a recommendation. That is, we need to determine the utility an agent can expect to derive from giving one of its recommendations to another agent. Or, more specifically, under which circumstances should an agent give a recommendation to another.

To better understand this problem and identify situations under which this economy of information might arise we have developed a formal model which we describe in Section 2. We use this model to derive some analytical results. The model simulation and test results are shown in Section 4. Section 5 shows some work related to our research. Finally, we present our conclusion in Section 6.

2 The Model

We envision a world populated with a finite set of agents A and a much larger but still finite set of documents D, where $d \in D$ is some document. For each agent $i \in A$ we define $L_i(d)$ to be true if and only if agent i likes document d. An agent does not have direct access to its L set—it can only find out which documents it likes by actually reading them. We use the notation from [2] for representing an agent's knowledge. Specifically, if i knows that it likes d, then we say that $K_i L_i(d)$ is true. Also, the set of documents that i knows that it likes is given by $L_i^i \equiv \{d \in D \mid K_i L_i(d)\}$. Similarly, the set of documents that i knows that $j \in A$ likes is given by $L_i^j \equiv \{d \in D \mid K_i L_j(d)\}$. In our model agents initially do not know which documents they like. An agent has to read a document in order to determine whether it likes it or not.

We also establish some costs and payoffs for our model. P_r is the payoff an agent receives for reading a document it likes, that is, i reads a document d for which $L_i(d)$ is true. C_r is the cost of reading a document. We assume that it takes the same amount of resources to read any document. We also assume that $P_r > C_r$ so the agent will always derive a positive utility from reading a document that it likes. Finally, C_m is the cost of sending a message. These costs

		j	
		Nothing	Send
i	N	$0, 0$	$x_i(j), -C_m$
	S	$-C_m, x_j(i)$	$x_i(j) - C_m, x_j(i) - C_m$

Fig. 1. Payoff matrix that two agents face when they meet.

are all valued in terms of the agent's utility. As such, we can say that the utility i receives for reading d is given by

$$U_i(d) = \begin{cases} P_r - C_r \text{ if } L_i(d) \\ -C_r \qquad \text{otherwise.} \end{cases} \tag{1}$$

The agents' interactions are kept as simple as possible so that we may study the system's dynamics. Specifically, the agents meet in pairs and have a chance to concurrently send each other a recommendation. That is, if i and j meet then i can choose to tell j about some d that i knows is in L_i, similarly j can choose to tell i about some other d' that j knows is in L_j. Of course, each one could also choose to say nothing.

When two agents meet each must decide whether to tell the other about a document it likes. Since each one of them has two choices we can represent this decision with the game matrix shown in Figure 1. The matrix shows that if the agents decide to do nothing they will receive no utility. If one agent decides to send a message but receives no message from the other one then his payoff is simply $-C_m$ because this is the cost of sending a message. We ignore all long-term implications of an agent's actions since these will be considered when exploring the dynamics of the system; at this time we are only interested in the immediate payoffs to the agents.

The utility value represented by $x_i(j)$ captures the utility that agent i accrues when it receives a recommendation from agent j, similarly for $x_j(i)$. While we cannot calculate an exact value for $x_i(j)$, we can calculate its expected value using some probability calculations. Specifically, we can determine that if i receives a message from j stating that it likes d then the payoff i can expect to receive by reading d is given by

$$\begin{aligned} x_i(j) = r_i(j)(&\mathbf{Pr}[L_i(d) \,|\, L_j(d)] \cdot (P_r - C_r) \\ &+ (1 - \mathbf{Pr}[L_i(d) \,|\, L_j(d)]) \cdot (-C_r)), \end{aligned} \tag{2}$$

where $r_i(j)$ is defined as the probability that i will receive a document from j that i has not read. This equation states that the payoff is equal to the probability that j will send a document d which i has not read times the expected payoff. This expected payoff is given by the probability that i likes d given that j likes d times the payoff for reading a liked document plus the probability that i does not like d given that j likes d times the cost of reading a disliked document. Notice that we can consider $x_i(j)$ to be the expected value of $U_i(d)$ given that

d is a document sent from j to i. The value of $x_j(i)$ is calculated in a similar manner to $x_i(j)$. We also note that

$$\mathbf{Pr}[L_i(d) \mid L_j(d)] = \frac{\mathbf{Pr}[L_i(d), L_j(d)]}{\mathbf{Pr}[L_j(d)]}, \tag{3}$$

by using Bayes Theorem. As such, $x_i(j)$ will be equal to $x_j(i)$ if $\mathbf{Pr}[L_i(d)] = \mathbf{Pr}[L_j(d)]$, that is, if the prior probability for the agents' liking a document is the same. This means that, if we can assume that all agents are equally discriminating in their taste then we can also assume that $x_i(j) = x_j(i)$. If these facts are common knowledge in the system then the fact that $x_i(j) = x_j(i)$ should also be common knowledge.

While the value of (3) can only be determined with knowledge of L_i and L_j, i can try to approximate it given its knowledge. That is, i can assume that its sampling of the document space is even and that j's recommendations are also even and, therefore, the likelihood that a new recommendation from j will also be liked by i will reflect the past behavior. Specifically, i can assume that

$$\frac{\mathbf{Pr}[L_i(d), L_j(d)]}{\mathbf{Pr}[L_j(d)]} \approx \frac{|L_i^i \cap L_i^j|}{|L_i^j|}. \tag{4}$$

Using this approximation and assuming that $r_i(j) = 1$, a safe assumption if there are a lot of documents or if we can assume that j knows the documents that i has read, we can determine that

$$x_i(j) \approx \frac{|L_i^i \cap L_i^j|}{|L_i^j|} \cdot (P_r - C_r) + \left(1 - \frac{|L_i^i \cap L_i^j|}{|L_i^j|}\right) \cdot (-C_r). \tag{5}$$

All the values in this equation can be calculated by i. An agent can, therefore, use this equation to determine its expected payoff at runtime, as long as $|L_i^j| > 0$. If $|L_i^j| = 0$ then i has no information about j's likes so the best it can do is assume that a recommendation by j will have the same expected utility as reading a randomly chosen document[1].

If $x_i(j) > 0$ and $x_j(i) > 0$ then the payoff matrix of Figure 1 becomes a Prisoner's Dilemma matrix. As such, we would expect Tit-for-Tat to be the evolutionary stable strategy [3]. In a population of Tit-for-Tat players this means that all players will choose to send. We can also determine that $x_i(j) > 0$ as long as

$$\frac{|L_i^i \cap L_i^j|}{|L_i^j|} \geq \frac{C_r}{P_r}. \tag{6}$$

Once an agent decides that it is going to tell the other one about a document that it likes, it must choose a document. That is, which d from among the $K_i L_i(d)$ should i send to j? There are three possible ways for i to choose a document. It could choose randomly from either one of the following sets:

[1] Our ongoing research explores the possibility of having agents recommend other agents, a technique which can provide the agent with a better estimate of the expected utility from unknown agents.

1. $\{d \in D \mid K_i L_i(d)\}$,
2. $\{d \in D \mid K_i L_i(d) \wedge \neg K_i L_j(d)\}$,
3. $\{d \in D \mid K_i L_i(d) \wedge \neg R_j(r)\}$.

The first choice is the simplest to implement since it only requires choosing randomly from a set of documents. The second choice can be implemented if i keeps track of all the recommendations it has received in each encounter. The third choice can only be implemented if i knows all the documents that j has read. This knowledge could be acquired either via direct communication or by having all agents "post" a list of all the documents they have read, without specifying whether they liked them or not, on a place that all others can access, for example, in a web page. In the tests given on the next section we assume that agents do post these lists. It is also clear that each of these methods will perform better than the next one since each one has a reduced probability of providing already-known information to the other agent. That is, in the third choice agent i is guaranteed to choose a document that the other agent has not read, in the second choice there is a small probability that the chosen document will have already been read by j (but i does not know that j likes it), and in the first choice this probability is even greater.

As an aside, we also note how (2) captures the need for agents to have correlated preferences in order to enable some cooperation. That is, if i and j have completely uncorrelated preferences then

$$\mathbf{Pr}[L_i(d) \mid L_j(d)] = \frac{\mathbf{Pr}[L_i(d)] \cdot \mathbf{Pr}[L_j(d)]}{\mathbf{Pr}[L_j(d)]} = \mathbf{Pr}[L_i(d)] \tag{7}$$

and

$$x_i(j) = \mathbf{Pr}[L_i(d)] \cdot (P_r - C_r) + (1 - \mathbf{Pr}[L_i(d)]) \cdot (-C_r). \tag{8}$$

Therefore, if i and j have uncorrelated preferences then i's payoff does not depend on j's recommendation and simply reflects i's discriminating taste in documents, that is, $\mathbf{Pr}[L_i(d)]$. As such, j's recommendation to i has the same expected value for i as simply choosing a document at random so i will choose not to ask j for a recommendation since that incurs an extra C_r cost.

In summary, our analysis leads us to several conclusions.

- The agents need to explore their domain by reading randomly chosen documents and exchanging recommendations with randomly chosen agents, otherwise they will fail to explore the whole space.
- It is probably safe for the agents to assume that $x_i = x_j$ since it is likely that all agents will be equally selective.
- The value of x_i can be approximated with (5) which can be determined from the agent's observations.
- If $x_i > 0$ and we assume that $x_i = x_j$ then the agents are faced with Prisoner's Dilemma payoffs so we can expect the system to evolve towards a Tit-for-Tat strategy.
- If the two agent's preferences are not correlated the agents will rather choose documents randomly than engage in exchange.

2.1 An Agent's Choice

An agent in our model is presented with a series of choices. It must first decide whether it wants to randomly choose a document that it has not read or ask another agent for a recommendation. If it decides to read a randomly chosen document d it can expect a utility of

$$\mathbf{Pr}[L_i(d)] \cdot (P_r - C_r) = \frac{|\{d \in D \mid L_i(d)\}|}{|D|} \cdot (P_r - C_r). \tag{9}$$

The agent will not know the value of this probability since it does not know L_i. It could, however, try to estimate it based on its past experience by using the ratio of L_i^i to the total number of documents i has read (R_i). That is, use $|L_i^i|/|R_i|$ as an approximation of the $|L_i|/|D|$ ratio. If, on the other hand, the agent chooses to communicate with another agent then it must choose whether to pick an agent at random or try to maximize its expected utility given what it knows about the other agents. By choosing an agent at random the agent's expected utility will be once again given by (9), except that the agent will also possibly add to its L_i^j knowledge which might be useful on future interactions. On the other hand, the agent could choose to pick the agent which in the past has given him the best recommendations. That is, choose the j that maximizes (5). The expected payoff will be the value of $x_i(j)$ for that j. For the cases where i does not know anything about the documents that j likes, i assumes that the expected utility will be given by (9).

Notice that both of the choices that the agent has to make—whether to choose a document at random or ask some agent, and whether to ask an agent at random or pick the one that maximizes the expected utility—are instances of the classic explore versus exploit problem in machine learning (also referred to as the n-armed bandit problem). The consensus solution to this problem consists of having the agent explore with a small probability. If the environment is fixed then this probability can be slowly reduced over time. Our environment is not fixed since the other agents will also be changing their behaviors and new documents are found all the time. Therefore, agents in our environment will likely choose to always explore with a small but non-zero probability.

When i receives a request from another agent j to exchange documents, i must determine which document to send, if any. This decision can also be made by comparing $x_i(j)$ with C_m. If it will cost i more to send the message than it expects to receive from the recommendation then it is better off not sending the message.

All this reasoning is captured by the algorithm shown in Figure 2. The algorithm's use of R_j means that it assumes that agents have access to the list of documents that other agents have. We envision agents that post a list of the documents they have read, without including whether they liked them or not. This list could also include documents about which the agent has no interest in receiving any recommendations. For example, using an indexing mechanism such as the Dewey decimal system or the Digital Object Identifier System (`doi.org`), an agent could state that it is not interested in some subset of documents. All other

agents would simply pretend that the agent had already read those documents and will not recommend any of them to the agent.

when acting:
if random 1.0 $<$ document-explore **then**
  read a randomly chosen document
else if random 1.0 $<$ agent-explore **then**
  exchange recommendations with a randomly chosen agent
else
  for all $j \in A - \{i\}$ **do**
    if $|L_i^j| = 0$ **then**
      $p_i(j) \leftarrow \frac{|L_i^i|}{|R_i|} \cdot (P_r - C_r)$
    else
      $p_i(j) \leftarrow x_i(j)$ // *as defined in* (5)
    end if
  end for
  $j \leftarrow \arg_j \max p_i(j)$
  if $x_i(j) > C_m$ **then**
    send random d from $\{d \mid K_i L_i(d) \wedge \neg K_i R_j(d)\}$
  else
    read a randomly chosen document
  end if
end if

when j **requests exchange:**
if $x_i(j) > C_m$ **then**
  send random d from $\{d \mid K_i L_i(d) \wedge \neg K_i R_j(d)\}$
else
  send nothing
end if

Fig. 2. Decision algorithm for agent i.

3 Modeling User Preferences

We now present our experimental model. It is important that this experimental model not be confused with basic model we introduced in Section 2. The experimental model is meant to be used as a way to simulate the possible behaviors of agents that represent real users. Since we cannot perform the necessary experiments on hundreds of real users, we have instead built an experimental model that hopes to capture the type of preferences and preference relationships between agents that we might see in the real world.

We represent each document with an n-dimensional binary vector $\mathbf{d}$. One can imagine that each of the elements in the vector represents a feature of the

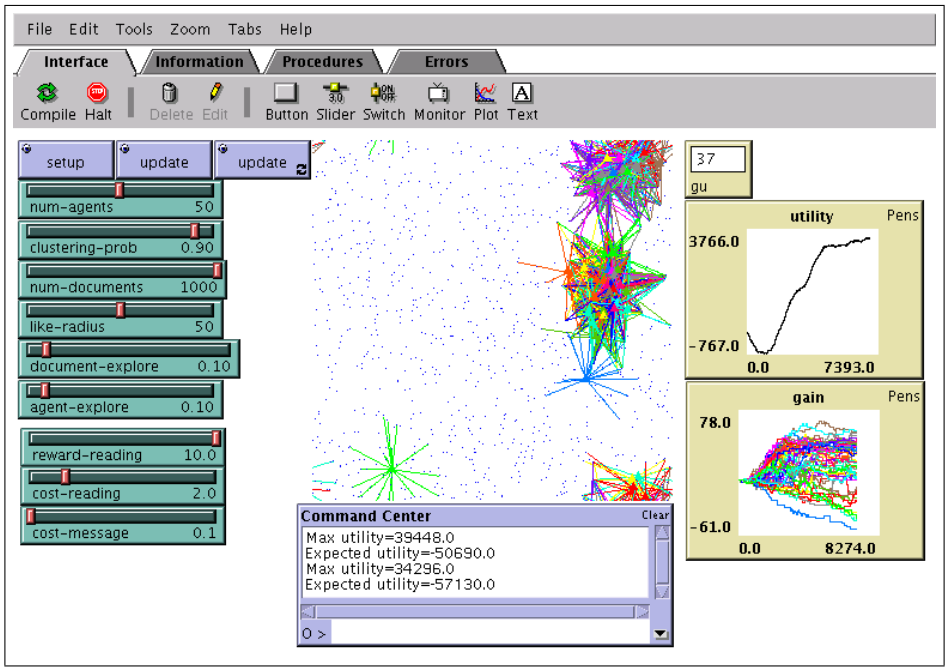

Fig. 3. Graphical user interface for our distributed recommendation simulation. The triangles represent the agent. The 2-dimensional field represents all possible preferences vectors so that an agent's position in the field corresponds to its preference vector. The documents are represented by points whose position represents their location in the preference space. The lines emanating from an agent show the documents it has already read. The picture also shows a couple of graphs which are updated dynamically as the program runs and show the total utility and individual gains from exchange.

document in question. Each agent also has an n-dimensional binary preference vector $\mathbf{p_i}$. We say that $L_i(\mathbf{d})$ if and only if

$$\frac{\mathbf{d} \cdot \mathbf{p_i}}{n} \leq r, \tag{10}$$

where r is some arbitrary but constant number between 0 and 1. We can envision agent i's preferences denoted by a point $\mathbf{p_i}$ in n-dimensional space. All the documents that i likes are within a distance r of this point. If j's $\mathbf{p_j}$ is close to $\mathbf{p_i}$ then the agent will like many of the same documents.

4 Implementation and Test Results

Our theoretical analysis lead us to the conclusion that our simple exchange mechanism will be incentive-compatible for the agents as long as the agent's

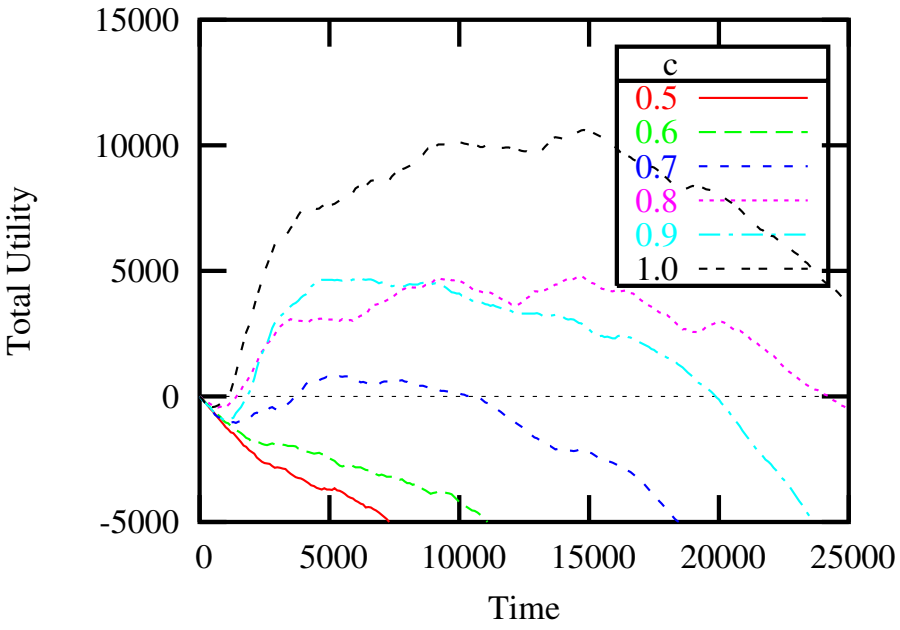

Fig. 4. Total utility over time assuming a fixed number of documents (1000). Each curve corresponds to a difference value of c, the clustering probability.

preferences are similar enough. However, we still do not know how likely it is that we will see situation where agents are similar enough, how the agent's exploration rate affects the system, or what characteristics do the dynamics of the system exhibit. In order to answer these questions we implemented a simulation of our distributed recommender system protocol using NetLogo [4]. You can run our applet and examine the source code for these experiments at `http://jmvidal.cse.sc.edu/netlogomas/distributedrec.html`. NetLogo facilitated the quick prototyping of our model. The facilities it provides for GUI creation also allowed us to explore many different parameter combinations. We only report the most interesting results.

Unless otherwise noted, all the experiments consist of 50 agents with 2-dimensional preference vectors in a toroidal space (i.e., a square space were the top and bottom edges are connected as well as the left and right edges), 1000 documents, $r = 1/6$, $P_r = 10$, $C_r = 2$, $C_m = 0.1$, and both `agent-explore` and `document-explore` (from Figure 2) fixed at 0.1.

4.1 Standard Agents

Our first experiment was designed to explore the relationship between the agents' similarities and the expected utility gain from exchanging recommendations. However, in order to perform such a test we first has to determine how the agents'

interests would be clustered. We developed a simple algorithm for generating clustered preference vectors. In our algorithm, the first vector is chosen randomly with a uniform probability distribution. Each vector after this one is chosen by a simple rule. With a probability of c, which we call the **clustering probability**, the vector is chosen to lie in a randomly chosen location that is somewhere within a small distance of an existing and randomly chosen vector, otherwise it is placed on a random location. The algorithm generates one large grouping of agent preferences when $c = 1$ and a completely random placement of preferences when $c = 0$.

The results of our first experiment can be seen in Figure 4 which shows the total utility over time. The total utility is the sum of all the agents' accrued utilities. Each one of the curves in the graph corresponds to a different value of c. We notice that in all the curves there is an initial dip into negative utility. This dip exists because the agents have no initial knowledge about the other agents so they start by randomly choosing agents and exchanging recommendations with them. It is only after some time that agents learn which other agents provide good recommendations. We see how the learning turns around the total utility for the cases where $c \geq .6$. However, for cases where $c \leq 5$ the total utility for the system spends little time in positive territory, if at all. We can deduce that in these type of scenarios the agents, on average, would not have an incentive to trade recommendations.

For comparison, we can calculate that the probability that an agent will like a document is equal to $\pi/36$ since $r = 1/6$. Therefore, an agent that reads all 1000 documents, say by choosing one randomly each time, will be expected to accumulate a utility of $1000(\pi/36 \cdot (P_r - C_r) + (1 - \pi/36) \cdot (-C_r))$ which in our examples works to be 94, so the total utility of 50 of these agents would be 4719. By contrast, the total utility for $c = .9$ and $c = .8$ climbs to 5000, and 10000 for $c = 1$.

Another interesting feature of the curves in Figure 4 are the upswings and downswings in the total utility. That is, we notice that sometimes the utility seems to be monotonically increasing for a long time and other times it is decreasing for a long time. This emergent behavior is explained by the system's search for new documents. As time passes the agents in a cluster become more and more likely to exchange recommendations with each other but these recommendations end once they have recommended to each other all the documents that they know about. At this time the agents go back to simply reading documents at random which, on average, causes their utility to decrease. But, when one of the agents discovers a new document that it likes this recommendation starts to propagate throughout the cluster, increasing the utility each time.

The fact that the total utility for all agents ends up decreasing for all the curves in Figure 4 might seem to contradict our claims that the agents have an incentive to engage in recommendation exchanges but the utility decline is simply an artifact of the use of a fixed number (1000) of documents. In Figure 5 we show the results for an identical experiment but, in this case, at each time step we add a new randomly generated document to the system with a probability of 1/10.

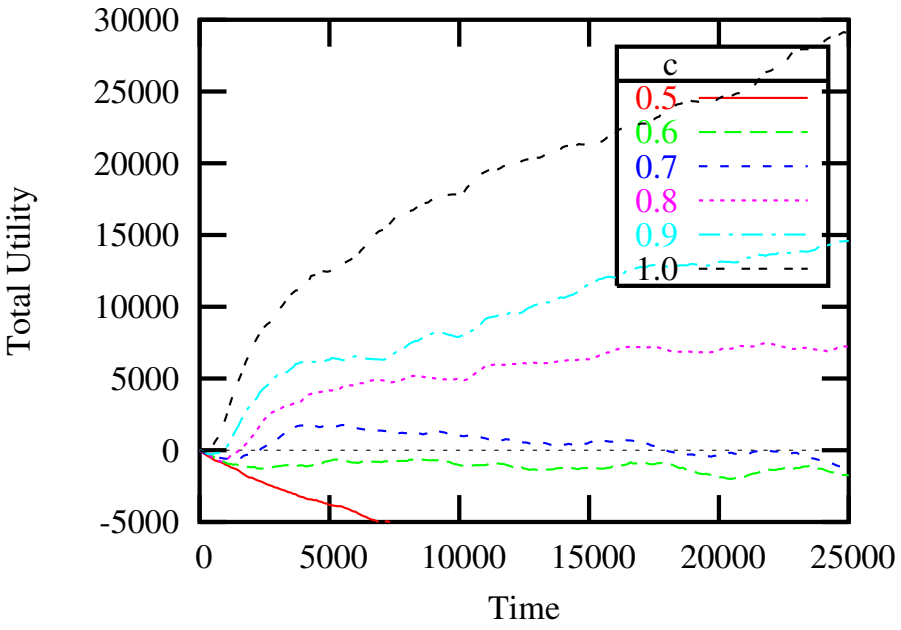

Fig. 5. Total utility over time assuming we start with 1000 documents and then add, on average, one more every 10 time steps.

That is, the system still starts with 1000 documents and a new document are added, on average, every 10 time steps. We can see that in this case the curves do not decline. In fact, the total utility in these cases is expected to keep increasing with a slope roughly proportional to the arrival rate of new documents.

While the total utility measures are a useful way to measure the expected utility for an agent, they do not answer the individual agent's question of whether or not it should bother to exchange recommendations. That is, will an agent in fact receive more utility if it agrees to exchange recommendations? We define the utility gain of an agent to be the utility it received from obeying a recommendation it received, which would be $P_r - C_r$ if the agent liked it and $-C_r$ otherwise, minus the expected utility the agent would get if it chose a document at random and read it. We show the utility gain for every one of the 50 agents in Figure 6, which uses $c = 1$. We can see that 49 of the agent accrued a positive utility gain and only 1 agent had a negative utility gain. Therefore, there is an overwhelming probability that agents who exchange documents will gain extra utility by doing so.

One expects that the probability of gaining utility by exchanging documents will decrease as the clustering probability decreases. Figure 7 confirms this expectation. It shows the utility gain for 50 agents using $c = .9$. We notice that there are now 33 agents with positive utility gain and 17 with negative gain. An

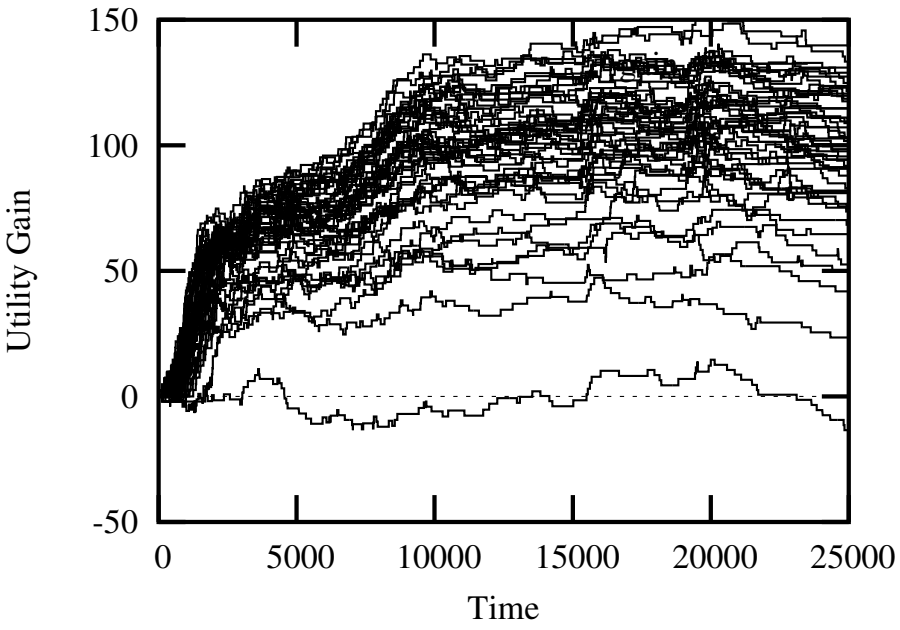

Fig. 6. Total utility gain for every agent over time, using $c = 1$. We continue to add a document each time with probability 1/10

analysis of this system showed that it was those agents that lie within a cluster which overwhelmingly received the utility gain. We can conclude that even in cases where not all agents are in a cluster, those that are in the cluster will benefit from the exchange of recommendations.

4.2 Greedy Agents

Once agent designers have access to the results we presented in the previous sections they should be convinced that, as long as they expect their preferences to be close to those of a sizable subset of agents, that the best strategy is to provide and accept recommendations. However, a designer might wish to further improve on this strategy by creating a "greedy" agent which tries to exploit the knowledge of others. An exceptionally greedy agent would not provide any recommendations to any other agent—simply accepting recommendations from them. This strategy will surely and quickly backfire as the agent becomes ostracized by the others who learn that it does not provide useful recommendations. A more reasonable greedy strategy would be to reduce the `agent-explore` rate to a very low number. Such a low exploration rate would lead an agent to keep exchanging recommendations with the best agents that it has found. It would only search for new agents to exchange recommendations with when it had used up all the recommendations from the agents it knows provide good recommendations.

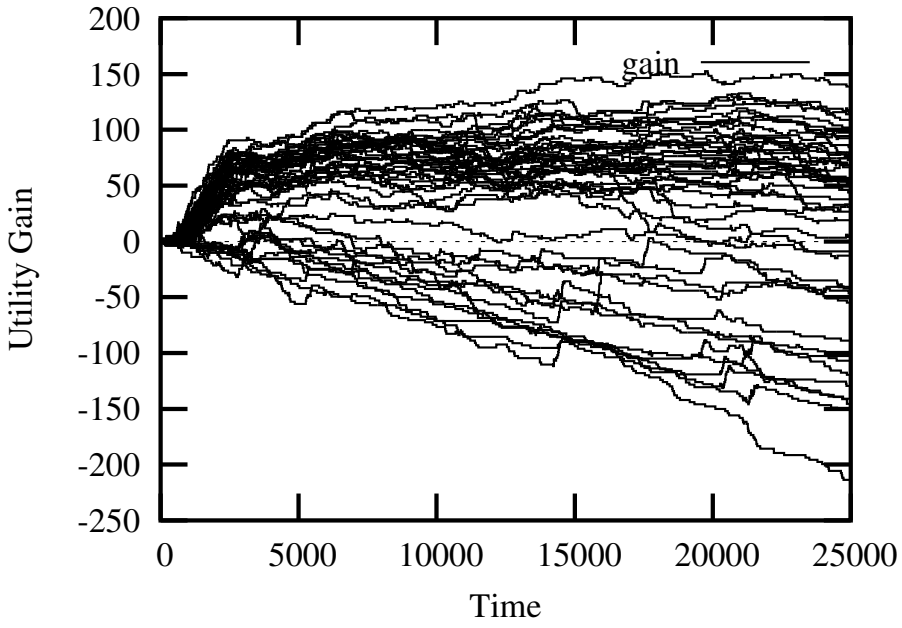

Fig. 7. Total utility gain for every agent over time, using $c = .9$ and 1000 documents.

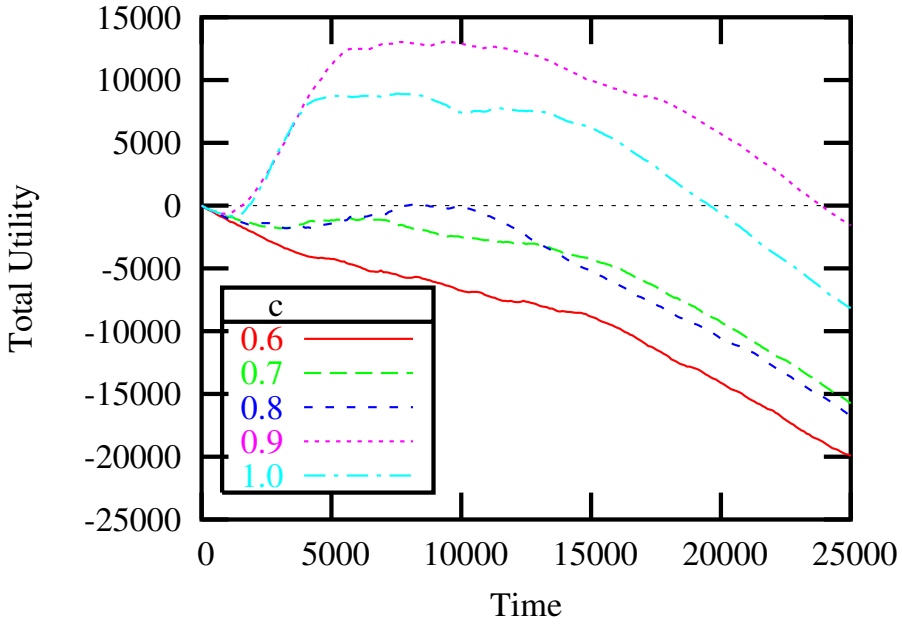

Fig. 8. Total utility over time with `agent-explore` set to .05.

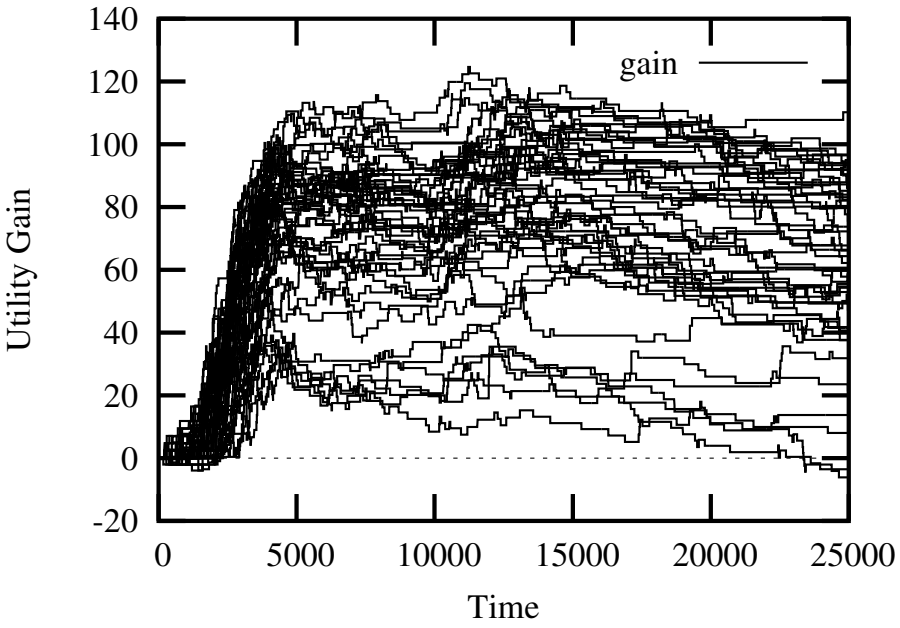

Fig. 9. Gain of each agent for $c = 1$ with `agent-explore` set to .05.

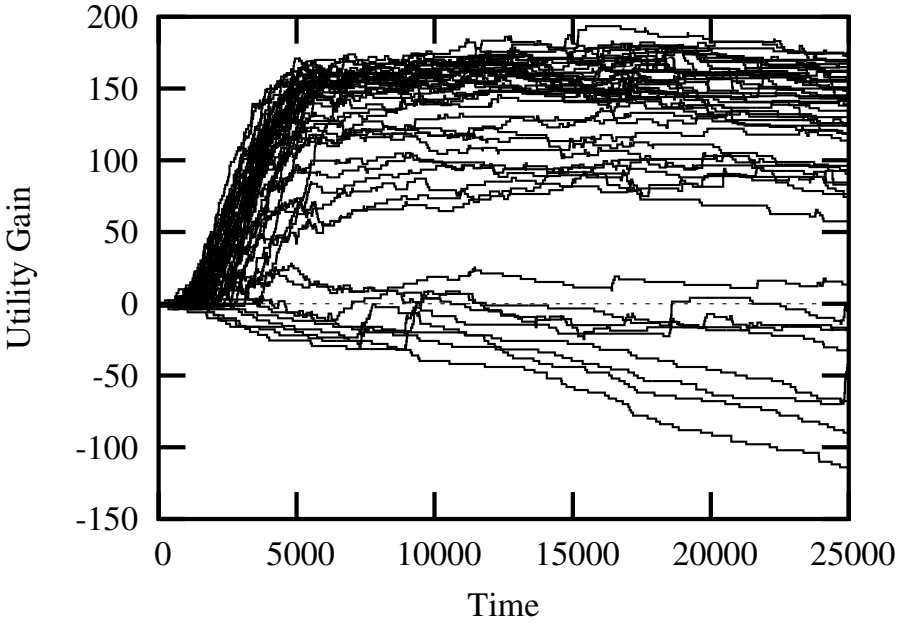

Fig. 10. Gain for each agent for $c = .9$ with `agent-explore` set to .05.

In order to determine what would happen to a system composed of these greedy agents, we repeated the same tests as before but using an `agent-explore` value of .05. The total utility accrued over time for a series of runs can be seen in Figure 8. We can see that the agents still manage to accrue a positive utility over time. In fact, comparing it with Figure 4, we can see that the population of greedy agents accrues more utility for the case where $c = .9$. However, we also note that the utility is almost the same for the case where $c = 1$ and is much lower for all the other cases where $c \leq .8$. These results show that a greedy population could have a greater social welfare than our standard population but only when almost all the agents have similar interests.

We also notice how the utility for the case where $c = .9$ is actually larger than the case where $c = 1$, a surprising result. We can gain some insight into this result by looking at Figure 9 which shows the gain from exchange for all the agents for the case were $c = 1$, and Figure 10 which shows the gain for all the agents for the case when $c = .9$. We notice that for $c = 1$ nearly all agents had something to gain from exchanging recommendations but, on average, these gains were not as high as those of some of the agents for the $c = .9$ case. That is, for $c = .9$ the agents that similar interests to others actually did better, in absolute terms, than in the $c = 1$. Specifically, we can deduce that in heterogeneous populations the individual gain from exchanges is higher because these communications are needed to determine which other agents have interests that are to those of the agent. In a more homogeneous populations simply choosing an agent at random will often have the same expected effect that one achieves with longer periods of modeling.

This result provides further motivation for an agent to engage in recommendation exchanges, especially in those cases where the agent fears that it is not similar to all others. Notice that this result goes against the intuition that one has nothing to gain from exchanging recommendations when the agent's preferences are not highly correlated to those of others. In fact, we have just shown that it is exactly in those situations where the exchange of recommendations, and the modeling of other agents that goes along with it, delivers a higher utility gain to the agent because it is in those situations that the agent needs to be able to differentiate between the agents that have interests similar to it and those that do not.

4.3 Results Summary

In general, our results from all our tests have shown that distributed incentive-compatible recommendations as dictated by our model are viable in that they increase both the average individual agent's utility as well as the social welfare. However, the benefits are not distributed evenly among the agents. Those agents that have interests that are similar to many other agents' interests usually perform better. While this result is probably to be expected, it does present a problem for the widespread adoption of our protocol. That is, an agent who believes that its interests are unique might decide not to join the community and simply read documents at random. Our results raise the question of whether we

can find a (distributed) method whereby an agent can quickly determine how many other agents with similar interests exist. Such a method would enable agents to decide whether or not to join a particular population of recommending agents.

We also found it interesting how "communities" of agents worked together to quickly find all the interesting documents in their area. The moment one of the agents found a new interesting document it would tell others who would tell others and so on, until they had all read the document. Initially, we had planned to estimate the expected utility from reading a random document to include the expected gains from sharing this document, if the agent liked it, with all the other agents it knows have given it good recommendations in the past. However, the dynamics of our model showed that this value decreases very fast once the agent tells just one other agent.

5 Related Work

There is an established body of literature concerned with the construction of recommender systems [1], which includes systems such as PHOAKS [5], the Referral Web [6], and many others. However, almost all them are centralized, with a few exceptions. Yenta [7] implements a decentralized protocols where agents form clusters of like-mindedness. However, they assume cooperative agents so their protocol is not incentive compatible. Economists have also studied the possibility of a market for evaluations [8] and concluded its viability. The proposed market, however, relies on a centralized auctioneer and their model assumes that all agents have very similar preferences.

Our work also finds much affinity with research being done in peer-to-peer systems such as JXTA [9], Gnutella [10, Chapter 8], Freenet [10, Chapter 9], and others. In fact, by analyzing and clarifying the individual incentives to the agents our work hopes to enable the realization of peer-to-peer networks that are immune to the freeloader problem experienced by current peer-to-peer systems [11], although we have not yet achieved this goal. It is our belief that for these type of networks to succeed the proper incentives have to be given and, furthermore, these incentives will not be in the form of monetary currency which brings with it the problems of accountability and liability, but instead the incentives will be either in the form of information itself or in the form of relationships with other agents. That is, multiagent systems will be needed in order to realize the promise of peer-to-peer information exchange on an Internet scale.

In [12,13,14] the authors develop reputation management protocols which are also based on the agents' past experiences. Their approach uses the Dempster-Shafer theory of evidence [15] for determining how to aggregate evidence from various sources each with possibly different trustworthiness. Their model also differs from ours in that they are concerned with determining the trustworthiness or reputation of agents while we are concerned with finding other agents that can provide good recommendations. In [16] they develop a pricing mechanism where agents get paid to provide good recommendations.

An interesting approach is presented in [17] where the authors describe a system where the users of the recommendations provide rewards to the recommenders with the best recommendations, thereby providing them an incentive to continue to give good recommendations. While their design is distributed, it does not tell us how the recommenders choose their recommendations. As such, the system is orthogonal to our proposed protocol. We are currently examining the possibility of merging the two approaches.

6 Conclusion

We have presented a domain model that captures the most important aspects of the distributed recommendations scenario. We analyzed this model and showed that engaging in an exchange is the rational choice as long as the agent believes that the other agent has interests that have proven to be sufficiently similar. We then gave an algorithm that agents can use for deciding when to exchange recommendations and with whom. Finally, we tested our algorithm on a various simulated scenarios. The results confirmed our prediction that trading would ensue and would increase the social welfare of the system. Our tests also provided more details into the system's dynamics. We showed that while the total utility does increase, individual agents who do not have common interests with other agents do not participate in this gain. We also showed how populations of greedy agents (i.e., agents that usually prefer to trade with a well-known partner instead of reading a random document) can outperform our standard agents in populations where most of the agents have largely similar interests.

We believe that the issues of trust and recommendations are tightly related. That is, our agents can be said to gain trust in other agents' recommendations via experience. As such, we view our protocol as a specific instance of the trust acquisition problem.

Our ongoing research continues to expand on these results in order to support the recommendation of agents themselves. That is, we aim to find the value of having one agent recommend another agent to some third agent. The long-term goal of this work is the development of a framework that selfish agents can use do determine exactly how much each one of their opinions/recommendations is worth and how to realize this worth. Such a framework would enable the creation of an Internet-wide distributed recommender system.

Finally, we note that we have ignored possible privacy issues. For example, it is conceivable an agent might not want to reveal it's preferences to other agents or an agent might not want to be seen as belonging to a particular community of interest. Clearly, an agent that does not want to reveal anything will not be able to participate in our protocol. However, it is possible that agents could mask their true preferences by adding noise to their recommendations. We need to study how such noise can be added so that it maintains some privacy for the agents but still allows the protocol to work.

References

1. Resnick, P., Varian, H.R.: Recommender systems. Communications of the ACM **40** (1997) 56–58
2. Fagin, R., Halpern, J.Y., Moses, Y., Vardi, M.Y.: Reasoning About Knowledge. The MIT Press, Cambridge, MA (1995)
3. Axelrod, R.M.: The Evolution of Cooperation. Basic Books (1984)
4. Wilensky, U.: NetLogo: Center for connected learning and computer-based modeling, Northwestern University. Evanston, IL (1999) `http://ccl.northwestern.edu/netlogo/`.
5. Terveen, L., Hill, W., Amento, B., McDonald, D., Creter, J.: Phoaks: a system for sharing recommendations. Communications of the ACM **40** (1997) 59–62
6. Kautz, H., Selman, B., Shah, M.: Referral web: combining social networks and collaborative filtering. Communications of the ACM **40** (1997) 63–65
7. Foner, L.N.: Yenta: A multi-agent, referral based matchmaking system. In: Proceedings of The First International Conference on Autonomous Agents. (1997)
8. Avery, C., Resnick, P., Zeckhauser, R.: The market for evaluations. The American Economic Review **89** (1999) 564–484
9. Gong, L.: JXTA: A network programming environment. IEEE Internet Computing **5** (2001) 88–95
10. Oram, A., ed.: Peer-to-Peer. O'Reilly (2001)
11. Adar, E., Huberman, B.A.: Free riding on gnutella. First Monday (2000)
12. Yu, B., Singh, M.P.: An evidential model of distributed reputation management. In: Proceedings of the 1st International Joint Conference on Autonomous Agents and MultiAgent Systems. (2002) 294–301
13. Yu, B., Singh, M.P.: Distributed reputation management for electronic commerce. Computational Intelligence **18** (2002) 535–549
14. Yu, B., Singh, M.P., Sycara, K.: Developing trust in large-scale peer-to-peer systems. In: Proceedings of First IEEE Symposium on Multi-Agent Security and Survivability. (2004) 1–10
15. Henry E. Kyburg, J.: Bayesian and non-bayesian evidential updating. Artificial Intelligence **31** (1987) 271–293
16. Yu, B., Li, C., Singh, M.P., Sycara, K.: A dynamic pricing mechanism for p2p referral systems. In: Proceedings of Third International Joint Conference on Autonomous Agents and Multi-Agent Systems. (2004) 1426–1427
17. Wei, Y.Z., Moreau, L., Jennings, N.R.: Recommender systems: a market-based design. In: Proceedings of the second international joint conference on Autonomous agents and multiagent systems, ACM Press, New York, NY. (2003) 600–607