Incentive-Based Co-Utility: Co-Utile Reputation Management

Josep Domingo-Ferrer, Oriol Farràs, and David Sánchez

Abstract Well-designed protocols should be self-enforcing, that is, be such that rational participating agents have no motivation to deviate from them. In addition, protocols can have other interesting properties, such as promoting collaboration between agents in a search for a better outcome. In [7, 8], we proposed the notion of *co-utility*, which characterizes a situation in which mutual help is the best rational option to take even for purely selfish agents; in particular, if a protocol is co-utile, it is self-enforcing. However, guaranteeing self-enforcement, let alone co-utility, for any type of agent behavior is not possible. To tackle this issue, in this chapter we detail how reputation mechanisms can be incorporated into existing protocols in order to make them self-enforcing (and optionally co-utile). Moreover, we show how to adapt and extend the well-known EigenTrust reputation calculation mechanism so that: i) it can be applied to a variety of scenarios and heterogeneous reputation needs and, ii) it is itself co-utile, and hence selfish agents are interested in following it. Obtaining a co-utile reputation mechanism creates a "virtuous circle" because: i) the reputation management is self-enforcing and, ii) as a result, it can be used to turn protocols that were not self-enforcing (resp. co-utile) *per se* into self-enforcing (resp. co-utile) ones.

Josep Domingo-Ferrer
Universitat Rovira i Virgili, Dept. of Computer Science and Mathematics, UNESCO Chair in Data Privacy, Av. Països Catalans 26, E-43007 Tarragona, Catalonia, e-mail: josep.domingo@urv.cat

Oriol Farràs
Universitat Rovira i Virgili, Dept. of Computer Science and Mathematics, UNESCO Chair in Data Privacy, Av. Països Catalans 26, E-43007 Tarragona, Catalonia, e-mail: oriol.farras@urv.cat

David Sánchez
Universitat Rovira i Virgili, Dept. of Computer Science and Mathematics, UNESCO Chair in Data Privacy, Av. Països Catalans 26, E-43007 Tarragona, Catalonia, e-mail: david.sanchez@urv.cat

1 Introduction

A protocol can be defined as a sequence of actions prescribed for an interaction between agents, both in the real world (e.g. road traffic rules) and the virtual world (e.g. peer-to-peer computing). A well-designed protocol is such that the participating agents are willing to follow it, that is, they find no motivation to deviate from the protocol prescriptions. In this manner, the protocol can be completed in a self-enforcing way. Ensuring this in the general case (i.e., for any type of agent behavior) is not possible. This is why protocols are usually designed for the most usual type of agents: *rational agents*. A rational –self-interested– agent acts in order to maximize her profit/utility (i.e., the outcome she obtains, such as money, functionality, etc.). Thus, she will only deviate from a given protocol if by doing so she can increase her utility.

While being self-enforcing is essential for a protocol to be adhered to, other desiderata can be conceived. In particular, a protocol might promote mutually beneficial collaboration between agents, in the sense that following it entails a collaboration between agents that improves their utilities with respect to the non-collaborative scenario. This self-enforcing collaboration is captured by the notion of *co-utility* [7, 8] (see also the previous chapter in this book). We say that a protocol is *co-utile* if helping the other agents that participate in the protocol to increase their utilities is also the way to increase one's own utility. That is, the protocol is built on the notion of mutual help as the best rational option.

We can find situations in which interactions are naturally self-enforcing and even co-utile. However, in many real-life cases, "negative" incentives (e.g., costs, lack of privacy, fear of strangers, etc.) may override positive incentives for the agents to follow the rules, let alone help each other. To tackle this problem, artificial positive incentives may need to be added to the agents' utility functions in order to compensate negative incentives. Specifically, we detail a general distributed mechanism that provides reputation as an artificial positive incentive (e.g. a stranger may be less feared if she has a good reputation). It turns out, though, that managing the reputation of the agents in a distributed way constitutes in itself a protocol that requires collaboration (e.g. to calculate, update and disseminate reputations); hence, *it is crucial that reputation management be designed to be co-utile*, so that the collaboration it implies be rationally sustainable. This is a significant aspect that differentiates our reputation protocol from other reputation management solutions [12, 16], which rely on a central trusted authority [11], are application-oriented (e.g., file sharing, commercial transactions, social networks content generation) [25, 1, 26, 3] and/or just offer robustness against some tampering attacks [13, 12], but do not necessarily ensure the protocol to be rationally followed by the agents involved in the reputation calculation.

In this research, we first characterize the obstacles to designing self-enforcing and/or co-utile protocols that could be solved by appropriate reputation mechanisms; at the same time, we identify what a reputation mechanism should offer in order to qualify as appropriate. We then describe our general-purpose distributed reputation model inspired in the well-known EigenTrust[13] mechanism. However,

unlike EigenTrust, which is meant to filter inauthentic content in P2P file sharing scenarios, *our reputation protocol can be applied to a variety of scenarios and heterogeneous reputation needs, and it is itself co-utile, so that self-interested agents wish to follow it*. As a result of the latter, we create a "virtuous circle": our co-utile reputation management is the result of self-enforcing, mutually beneficial collaboration and provides reputations that are used as artificial positive incentives to turn protocols that were not originally self-enforcing (resp. co-utile) into self-enforcing (resp. co-utile) ones.

2 Towards co-utile protocols via reputation management

Co-utility or even strict co-utility [7, 8] (see also the previous chapter in this book) can be reached in many practical situations. In the real world, for example, sharing one's car with one or several passengers can result in mutual benefits: passengers are able to reach their destination faster and the car driver needs to support less expense for the trip (because toll or gas fees are split among the passengers or because the driver can use facilities reserved for high occupancy vehicles or because of both reasons) [2]. In the information society, many peer-to-peer protocols follow the same pattern. File sharing protocols are based on the premise that, by sharing your own files and your upload bandwidth, you will also have access to a larger catalogue of files (shared by other users) with faster download speeds and greater availability than in centralized data storage systems [17]. Peer-to-peer protocols can also provide privacy protection to Internet users. For example, a user of a web search engine such as Google can hide her queries and search patterns from the search engine by requesting other peers to submit her queries (and return the results to her); this may be not only good for the requester's privacy, but also for the submitters' privacy, because the requester's query can be viewed as noise concealing the actual search interests of the submitters to the search engine [6, 9].

In [7, 18, 4, 21, 22] we showed how some of these protocols can be made co-utile. In all cases, this was possible because of the availability of agents with the appropriate types, that is, an appropriate combination of preferences/utilities (e.g., cost/time savings, privacy, functionality, etc.) that made their collaboration mutually beneficial. However, in practice, agent types can be very heterogeneous and "negative" utilities may dominate the benefits of agent collaboration, thereby precluding co-utility. For example, in a ride-sharing scenario, agents may be reluctant to share a car if the passengers and/or the driver are complete strangers [18]; in a query submission scenario, agents may be reluctant to submit queries from other users if that costs them bandwidth or money [5]; in a P2P lending or crowdfunding scenario, lenders and funders may be reluctant to given money to borrowers or projects with unknown reliability records [23, 24]. Another possible obstacle to co-utility occurs when the self-enforcing behavior is *not* to collaborate rather than collaborate. For example, in an uncontrolled file sharing system, purely selfish agents would prefer to download files from others but not to share their own files (to avoid spending up-

load bandwidth), thus becoming "free riders"; eventually, if all agents become free riders, the file sharing system collapses.

Obstacles like the above ones should be tackled if one wants to design protocols that are co-utile for as many agent types as possible. A way to neutralize the negative utilities that prevent co-utile collaboration is to incorporate artificial utilities in the form of rewards (that compensate the negative payoffs) or penalties (that discourage free riding). Note that rewards and penalties can also be used if, rather than obtaining a co-utile protocol, one merely wants to turn into self-enforcing a protocol that is not so.

Reputation is a very versatile artificial utility that can be used both as a reward and a penalty. In general, the reputation of an agent is the opinion of the community on the agent. On the one hand, reputation allows peers to build *trust*, which can neutralize the negative utilities related to mistrust (e.g., fear or reluctance in front of strangers). On the other hand, reputation also makes the agents accountable for their behavior: if an agent misbehaves, her reputation worsens and the other agents mistrust her more and more and are less and less interested to interact with her. In this manner, free riders and also malicious peers (who may try to subvert the system, even irrationally) may be identified and penalized (e.g., through limitation or denial of the service they try to abuse).

Many reputation mechanisms have been proposed in the literature for peer-to-peer communities, either centralized or distributed (see the surveys [12, 16]). However, since in our work reputation is used as a means to turn a protocol into co-utile or, at least, self-enforcing, the reputation calculation/management protocol itself should be distributed and strictly co-utile; otherwise, computing reputations would not be rationally sustainable and would not serve its purpose of inducing co-utility or self-enforcement in other decentralized P2P protocols.

Specifically, we want a reputation protocol with the following features, which should make it amenable to strict co-utility:

- *Decentralization*. Reputations should be computed and enforced by the peers themselves rather than a central authority. In this manner, we avoid depending on a trusted third party [11], which may be compromised and constitute a central point of failure (e.g., rational peers may be interested to attack this central authority).
- *Anonymity*. Reputation management should not rely on identifiers (e.g. IP addresses) that reveal the real identity of agents who help computing the reputation of other agents. Otherwise, the privacy loss may negatively affect the payoffs of collaborating in reputation management. Moreover, the possibility of creating coalitions between agents that know each other may facilitate collusion attacks to the reputation system. Surprisingly, most related works either support very limited anonymity or completely neglect it (see [19]).
- *Low overhead*. Reputation management should not imply a large expenditure of resources (e.g., bandwidth, storage, calculation); otherwise, these negative payoffs may dominate the benefits brought by reputation. To be more specific, a linear [13] or quasi-linear [20] calculation cost would be desirable.

- *Proper management of new agents*. Newcomers should not enjoy any reputation advantage; otherwise, malicious peers may be motivated to create new anonymous identifiers after abusing the system in order to regain the advantages of a good reputation.
- *Attack tolerance*. A number of attacks may be orchestrated in order to subvert the reputation system. Since we focus on rational selfish agents, we are interested in systems that are robust against "rational attacks". In particular, a rational selfish agent may try to subvert the protocol or manipulate the reputation calculation if, by doing so, she can increase her benefit; that is, if via subversion she can obtain a higher reputation at a lower cost than via "good behavior" in the underlying P2P scenario. Thus, the reputation system should implement measures to make the cost of such an attack unattractively high. According to the classification of attacks identified in [12], we want to avoid the following attacks:
  - *Self-promotion*, where agents are able to falsely increase their own reputations at a small or zero cost.
  - *Whitewashing*, in which agents circumvent the consequences of abusing the system to obtain an unfair benefit, for example by creating new "clean" identities or performing Sybil attacks. Analyses of whitewashing and Sybil attacks can be found in [10] and in [15], respectively.
  - *Slandering*, where agents may falsely lower the reputation of other agents if, by doing so, their own reputation becomes comparatively higher.
  - *Denial of service*, in which agents block the calculation and dissemination of reputation values. This may happen, for example, if the reputation calculation has a cost that agents deem higher than the benefits the calculation of their own reputation (by other agents) would bring to them.

After examining a number of decentralized reputation mechanisms [12], in [5], we selected, adapted and extended EigenTrust [13] in order to obtain our strictly co-utile reputation protocol. Eigentrust offers most of the desirable features identified above: distributed reputation calculation, low overhead, anonymity and robustness to attacks. This reputation scheme is designed to filter out inauthentic content in peer-to-peer file sharing networks. Its basic idea is to calculate a global reputation for each agent based on aggregating the local opinions of the peers that have interacted with the agent. If we represent the local opinions by a matrix whose component (i, j) contains the opinion of agent i on agent j, the distributed calculation mechanism computes global reputation values that approximate the left principal eigenvector of this matrix. Anonymity is achieved by means of a distributed hash table (DHT) that randomly links agents and reputation calculation duties. For unstructured networks, other solutions can be used, such as gossip-based reputation distribution protocols [27].

3 Reputation calculation model

The aim of the reputation mechanism is to compute global reputation values in a distributed way. These values are calculated by means of a protocol whereby the agents share their local reputation values. In this section, we extend EigenTrust to compute reputations based on non-binary opinions. The original EigenTrust system was designed for P2P file sharing, and the receiver's opinion on a file download is just binary: either the download has been satisfactory or not. We want to be able to compute reputations based on opinions that have several categories, or are even continuous. The continuous case may also accommodate a situation in which each agent quantifies the benefit/cost resulting from interacting with another agent.

3.1 Calculation of local reputations

The opinion of $\mathscr{P}_i$ on another agent $\mathscr{P}_j$ with whom $\mathscr{P}_i$ has directly interacted is the reputation s_{ij} of $\mathscr{P}_j$ local to $\mathscr{P}_i$. We define this value as the aggregation of payoffs (either positive or negative) that $\mathscr{P}_i$ has obtained from the set of transactions (Y_{ij}) performed with $\mathscr{P}_j$:

$$s_{ij} = \sum_{y_{ij} \in Y_{ij}} payoff_i(y_{ij}).$$

Payoffs may depend on the particular scenario, but the above calculation is general enough to accommodate them. Specifically, we can cope with the following scenarios that cover all the cases described in [12]:

- *Binary payoffs*. These are just binary values, either positive (+1) or negative (-1), according to whether the transaction with $\mathscr{P}_j$ has fulfilled or not $\mathscr{P}_i$'s goals. Binary payoffs make sense in P2P file sharing networks (i.e., the desired file has been obtained or not) or commercial transactions (e.g., in eBay, buyers rate transactions either as positive or negative).
- *Discrete or continuous opinions*. Payoffs may also measure the opinion on a transaction as a discrete or continuous magnitude. In this case, a certain transaction may generate reciprocal reputation updates by the involved agents (i.e., $\mathscr{P}_i$ on $\mathscr{P}_j$ and $\mathscr{P}_j$ on $\mathscr{P}_i$). Most transactions with a subjective outcome (other than the objective outcome of fulfilling or not the task/goal) should be measured in this way. For example, in a car-sharing network, passengers and drivers can rate each other to measure how pleasant the trip was or whether there were any unnecessary delays or overcosts; all of these dimensions can influence the utility of the involved peers and, thus, affect their willingness to collaborate in the future. Commercial transactions can also be rated in this way (e.g., Amazon provides discrete positive ratings), even though the rating is one-sided (not reciprocal). In some cases, the opinion can be the aggregation of several transaction components (e.g., reviews, comments, etc.) [14].

- *Costs*. In the previous cases, reputation values are potentially unbounded (i.e., there is not a limited reputation budget to be distributed among the peers of the network) and independent between the agents. Alternatively, payoffs could reciprocally measure the cost incurred/caused by the agents when helping others/being helped in fulfilling a task/goal. For example, in a P2P file sharing network, if $\mathscr{P}_i$ provides a file to $\mathscr{P}_j$, the payoff of the former with respect to the latter would be a negative value measuring the bandwidth spent by $\mathscr{P}_i$ when uploading the file; likewise, the payoff of $\mathscr{P}_j$ (who has received the desired file) with respect to $\mathscr{P}_i$ would be the inverse positive value. In the privacy-preserving query submission scenario discussed in Section 2 a similar criterion can be applied: if performing a query to the database has a cost (e.g. money) for the submitter, this should be reflected as a positive reputation payoff for the submitter and as a negative reputation payoff for the query originator. In these scenarios, reputation values can measure the difference between the cost incurred by an agent $\mathscr{P}_i$ when helping others and the cost incurred by others when helping $\mathscr{P}_i$. A reputation greater than 0 shows that the agent has helped others more than what others have helped her. Unlike in the previous cases in which reputation values were unbounded, here reputations are positively/negatively balanced across the members of the network; thus, a reputation value around 0 indicates a fair allocation of costs among peers.

It is important for local reputation values computed by the different peers to measure the payoff of similar transaction outcomes in a similar way. For objective outcomes (such as the fulfillment of a task or the incurred cost), this is not problematic; however, for subjective opinions the designer of the reputation system may provide some rules to control the ratings. For example, eBay recently implemented a "subjective" rating[1] which buyers can use to rate sellers from 5 stars down to 1 star regarding several aspects of the transaction (e.g., delivery speed of the goods, shipping charges, etc.); however, eBay associates each value to an objective criterion (e.g., a 5-star shipping charge means free shipping, whereas less than 3-star means a charge higher than the real shipping cost).

3.2 Computing global reputations

We review here how EigenTrust computes global reputations from local reputations; recalling this is needed for the reader to understand what we propose in Sections 4 and 5. In order to properly aggregate the local reputation values computed by each peer, a normalized version c_{ij} is first calculated as follows:

$$c_{ij} = \frac{\max(s_{ij}, 0)}{\sum_j \max(s_{ij}, 0)}.$$

[1] `http://pages.ebay.com/help/feedback/detailed-seller-ratings.html`

In this manner, the normalized local reputation values lie between 0 and 1 and their sum is 1. In other words, each agent has a reputation budget of only 1 that she has to split among her peers proportionally to her positive experiences (negative experiences are truncated to 0). This makes all agents equal contributors to the global reputation and avoids dominance by agents with a larger number of experiences. Moreover, this normalized calculation deters peers from colluding by assigning arbitrarily high values to each other. Finally, the fact that negative reputation values are truncated to 0 prevents selfish agents from assigning arbitrarily low values to good peers. We will later discuss how the protocol can also withstand tampering with reputation values.

A side effect of the truncation of negative values is that reputation values do not distinguish between agents with whom $\mathscr{P}_i$ had a bad experience (negative local reputation) from those with whom $\mathscr{P}_i$ has not interacted so far. Even though this may be seen as a drawback, it also can be viewed as a strength: newcomers (i.e., agents who have not yet interacted with anyone) do not have any reputation advantage, because their reputation is indistinguishable from the one of misbehaving agents. As a result, a selfish agent has no incentive to take a new virtual identity in order to "clean" her reputation after abusing the system (e.g., refusing to help others, in order to minimize her own costs). Likewise, newcomers will become instantly motivated to positively contribute to the system in order to earn the minimum reputation that other agents would require from them.

We can thus see that normalizing local reputation values biases the system towards positive reputation; that is, agents need a minimum *positive* reputation value in order to be trusted by peers, which requires them to help others/contribute to the system first on order to be able to get help later.

In this setting, local reputation values are disseminated and aggregated through the network peers following a transitive reputation algorithm, in order to obtain the global reputation value of each agent. This is the main idea of EigenTrust. Any agent $\mathscr{P}_i$ can compute $\hat{t}_{ik}^{(0)}$, an approximation of the reputation of a potentially unknown peer $\mathscr{P}_k$, by asking the peers with whom $\mathscr{P}_i$ has interacted ($\mathscr{P}_j$) for their local reputation w.r.t. $\mathscr{P}_k$, that is c_{jk}. Since $\mathscr{P}_i$ has already computed the local normalized reputation w.r.t. $\mathscr{P}_j$, that is c_{ij}, $\mathscr{P}_i$ can compute a local estimate of the reputation t_{ik} of $\mathscr{P}_k$ by using c_{ij} to weight $\mathscr{P}_j$'s local reputation; specifically, $\hat{t}_{ik}^{(0)} = \sum_j c_{ij}c_{jk}$. Thanks to the local normalization, $\hat{t}_{ik}^{(0)}$ takes values between 0 and 1. Observe that if we call $\mathbf{c}_i = (c_{i1}, \ldots, c_{in})^T$ and $\mathbf{C} = [c_{ij}]$, then $\hat{\mathbf{t}}_i^{(0)} = \mathbf{C}^T\mathbf{c}_i$, where $\hat{\mathbf{t}}_i^{(0)} = (\hat{t}_{i1}^{(0)}, \ldots, \hat{t}_{in}^{(0)})^T$. If every agent $\mathscr{P}_i$ computes $\hat{\mathbf{t}}_i^{(0)}$, in the next iteration $\mathscr{P}_i$ can compute $\hat{\mathbf{t}}_i^{(1)} = \mathbf{C}^T(\mathbf{C}^T\mathbf{c}_i)$. After m iterations, $\mathscr{P}_i$ will compute $\hat{\mathbf{t}}_i^{(m-1)} = (\mathbf{C}^T)^m\mathbf{c}_i$. Under the assumptions that $\mathbf{C}$ is irreducible and aperiodic [13], in an ideal and static setting the succession of reputation vectors computed by any peer will converge to the same vector for every peer, which we call $\mathbf{t} = (t_1, \ldots, t_n)^T$ and is the left principal eigenvector of $\mathbf{C}$. The j-th component of $\mathbf{t}$ represents the global reputation of the system on each agent $\mathscr{P}_j$.

Computing the global reputation values by the above method is not efficient because the communication complexity is very high. In the following section we

present a protocol to compute global reputation values that emulates the above method in a distributed and more efficient way.

4 Co-utile distributed reputation calculation protocol

In [5], we adapt and extend the EigenTrust secure protocol [13] in order to obtain a co-utile distributed protocol for computing global reputation values. The core idea of the EigenTrust secure protocol is that the reputation value of an agent $\mathscr{P}_i$ is computed by other agents in order to prevent manipulation and minimize the action of malicious peers. This computation is based only on the local reputation reported by those agents with whom the agent $\mathscr{P}_i$ interacted directly.

In our proposal, the computation of t_i (the global reputation of agent P_i) is based on the experience of the agents in A_i, which is the set of agents that provided help to or received help from $\mathscr{P}_i$. Hence, the global reputation values are computed in a distributed way according to the experiences of the agents in the network.

We assume that each agent $\mathscr{P}_i$ has an initial reputation value $t_i^{(0)}$ that is based on previous experiences or is given by default. Each agent has a number M of score managers that will compute her reputation value. These score managers are defined according to a distributed hash table (DHT), which maps each agent to a set of several agents determined by hash functions $h_0, h_1, \ldots, h_{M-1}$. In this way, $h_0(ID_i), \ldots, h_{M-1}(ID_i)$ are the coordinates of the agents computing the reputation value of $\mathscr{P}_i$.

With the above hash mapping, on average every agent is the score manager of M agents, so the work of the agents in the reputation mechanism is balanced. Let D_i be the set of agents for whom $\mathscr{P}_i$ is a score manager; we will also call D_i the set of daughters of $\mathscr{P}_i$. During the computation of the reputation values, the score manager of $\mathscr{P}_d$, say $\mathscr{P}_i$, learns A_d, that is, the set of agents that provided help to or received help from $\mathscr{P}_d$. Then $\mathscr{P}_i$ receives the trust assessments of $d \in D_i$ sent by the agents in A_d. The terms c_{ji} for $j \notin A_i$ are zero. Then, $\mathscr{P}_i$ engages in an iterative refinement to compute the reputation of every $\mathscr{P}_d \in D_i$, based on the score c_{jd} on $\mathscr{P}_d$ by each $\mathscr{P}_j \in A_d$ weighted by the score managers of $\mathscr{P}_j$ (the weight is the current reputation $t_j^{(k)}$ of $\mathscr{P}_j$ held by the score managers of $\mathscr{P}_j$). As a result, no agent $\mathscr{P}_j$ can directly influence the reputation of any other $\mathscr{P}_d$; everything is mediated by the score managers of $\mathscr{P}_j$, who together act as a sort of distributed trusted third party.

The above computations are described in Protocol 1. The main differences between this protocol and the one in [13] are: i) we rely for most of the calculation on the score managers (Step 7), thereby increasing the redundancy of the computation of the reputation values and protecting this computation against malicious agents; and ii) our definition of the agents that provide the local reputation values is more general, since we include any agent that has interacted with the daughter agent (Steps 3, 5, 9, 10 and 11).

Protocol 1 CO-UTILE COMPUTATION OF REPUTATION VALUES

for all agent $\mathscr{P}_i$ **do**
  Submit local reputation values $\mathbf{c}_i$ to all score managers at positions $h_m(ID_i)$, for $m = 0,\ldots,M-1$;
  Collect local reputation values $\mathbf{c}_d$ and the set A_d for all daughter agents $\mathscr{P}_d \in D_i$;
  for all $\mathscr{P}_d \in D_i$ **do**
    Query all the score managers of $\mathscr{P}_j \in A_d$ for $c_{jd}t_j^{(0)}$
    $k := -1$
    repeat
      $k := k+1$
      Compute $t_d^{(k+1)} = c_{1d}t_1^{(k)} + c_{2d}t_2^{(k)} + \ldots + c_{nd}t_n^{(k)}$;
      Send $c_{dj}t_d^{(k+1)}$ to all the score managers of $\mathscr{P}_j \in A_d$;
      Wait for all score managers of $\mathscr{P}_j \in A_d$ to return $c_{jd}t_j^{(k+1)}$;
    until $|t_d^{(k+1)} - t_d^{(k)}| < \varepsilon$ // Parameter $\varepsilon > 0$ is a small value
  end for
end for

Once a reputation manager computes a reputation on a daughter agent using Protocol 1, she keeps the reputation value for that agent until the next protocol execution. Protocol 1 is meant to be run periodically, in order for reputations to stay up-to-date. The reputation update period can be set depending on the activity of the agents, in order to obtain faster updates when the frequency of agent interactions increases. Ideally, the protocol should be run in parallel and asynchronously with respect to the agent interactions.

After the computation of the global reputation values, if an agent $\mathscr{P}_i$ needs the reputation value t_j of $\mathscr{P}_j$ in order to decide whether to collaborate with him, she can query the M score managers of $\mathscr{P}_j$ for his reputation. The M values obtained from the M score managers should be the same, because the inputs of the score managers are the same. However, if some values are different (e.g., if some score managers or agents have altered the computation for some reason), $\mathscr{P}_i$ can take as t_j the most common value among the ones sent by the score managers. If a score manager $\mathscr{P}_k$ does not answer to a query by $\mathscr{P}_i$, then $\mathscr{P}_i$ will assume that $\mathscr{P}_k$ is not active and she will set $c_{ik} = 0$.

In addition to the global reputation, the experience of $\mathscr{P}_i$ with respect to $\mathscr{P}_j$ is also reflected in the local value c_{ij}, if available (i.e., if $\mathscr{P}_i$ and $\mathscr{P}_j$ have already interacted). In some cases, local and global reputation values may not be coherent because the latter is the aggregated version of the former. In general, it is better for $\mathscr{P}_i$ to consider both local reputations $\mathbf{c}_i$ and global reputations $\mathbf{t}$ to make decisions about collaboration; a conservative criterion would be to use the lowest among the local and global reputations.

4.1 Co-utility analysis

Within the co-utility framework, we consider that agents are rationally selfish. For the calculation of reputations, this means that: i) agents want their reputation to be computed (this is their main utility); ii) they are interested in maximizing their reputation (so that it is higher than the reputations of other peers) by any means (i.e., either by correctly following the protocol or by deviating from it). Ideally, the reputation calculation protocol should be self-enforcing (hence discouraging deviation); what is more, it should be co-utile, given that all agents obtain their correct reputation as the outcome utility of the protocol execution. In the sequel, we analyze Protocol 1 and, for each step, we describe the options available to the agents and we justify why correctly following the protocol is the rational choice and, thus, why the protocol is self-enforcing and, in particular, co-utile.

Generally speaking, Protocol 1 encourages the agents to collaborate because the impact of their opinions on the computation of the global reputation values increases when they are active (that is, present in as many sets A_* as possible). On the other hand, the protocol is robust against *self-promotion attacks*, because the agents that actually compute the global reputation values of an agent $\mathscr{P}_i$ (her score managers) are chosen randomly, and they use the information provided by the score managers of the agents with whom $\mathscr{P}_i$ has interacted; thus, the agents cannot manipulate their own reputation values because they do not have direct control on their calculation. Also, Protocol 1 is robust against *whitewashing attacks*: since the local reputation values are truncated to 0, the system does not distinguish between agents with whom there was a bad experience (whose negative s_{ij} is truncated to 0) and those with whom no one has interacted so far. In this way, selfish agents have no incentive to create new virtual identities to reset their bad reputation because, in practice, this does not make them better off.

Let us now be more specific and go step by step. At Step 2, agent $\mathscr{P}_i$ acts as a reputation assessor and sends her local reputation values $\mathbf{c}_i$ to the reputation managers. Any peer $\mathscr{P}_i$ has a rational interest in forwarding to the rest of peers the fair local reputations c_{ij} she has awarded to peers $\mathscr{P}_j \in A_i$. We next justify why:

- If $\mathscr{P}_i$ fails to forward c_{ij}, then c_{ij} is taken as zero, and hence the reputation t_j of $\mathscr{P}_j$ is lower than due. Now, since $\mathscr{P}_j \in A_i$ implies $\mathscr{P}_i \in A_j$, by the expression in Step 9, a lower t_j also results in a lower reputation t_i of $\mathscr{P}_i$, *ceteris paribus*.
- If $\mathscr{P}_i$ forwards a reputation c'_{ij} less than the real c_{ij}, the argument is the same as in the previous item. Hence, Protocol 1 is robust against *slandering attacks by assessors*.
- If $\mathscr{P}_i$ forwards a reputation c'_{ij} greater than c_{ij}, this increases t_j and t_k for agents $\mathscr{P}_k \in A_j$. Although $\mathscr{P}_i$ is also in A_j and thus benefits from the increase, *ceteris paribus* t_i will increase less than t_j and similarly as t_k for $k \neq i, j$. Hence, $\mathscr{P}_i$ will be in a worse relative position with respect to $\mathscr{P}_j$ and possibly other peers.

From Step 3 onwards, agent $\mathscr{P}_i$ acts as a reputation manager and collects the local reputations awarded by all his daughter agents $\mathscr{P}_d$, which in turn allows $\mathscr{P}_i$ to learn the set A_d of peers with whom $\mathscr{P}_d$ has interacted. The same justification above

that $\mathscr{P}_i$ is rationally interested to report fair local reputations ensures that $\mathscr{P}_d$ will report all her local reputations fairly. On the other hand, if the score manager $\mathscr{P}_i$ does not collaborate to compute the global reputation of his daughters (*denial-of-service attack*), the requester of t_d will receive no answer from $\mathscr{P}_i$; in this case, the requester will consider that $\mathscr{P}_i$ not active, and will remove $\mathscr{P}_i$ from the list of available agents. Hence, $\mathscr{P}_i$'s own reputation value will be decreased in the following iteration of the reputation calculation protocol. Thus, a rational $\mathscr{P}_i$ is not interested in denying service.

The iteration starting in Step 7 involves the score manager $\mathscr{P}_i$ and the score managers of peers in A_d. For the same reasons given above to justify that $\mathscr{P}_i$ is not interested in denying service, the score managers of peers in A_d are not interested either.

Finally, even if not denying service, a score manager might send a wrong value when queried by an agent (Steps 5, 10 and 11). Assuming the security parameter M has been chosen so that the number of malicious agents can be safely considered to be less than $M/2$, such misbehavior will have no effect because the final value taken by the requesting agent is the majority value reported the score managers. Two remarks are in order here:

- The score managers are chosen by means of a hash function, that is, randomly. This makes it unlikely for them to collaborate in reporting a common manipulated t_d. If all wrong values can be assumed to be different, malicious score managers could be as numerous as $M-2$ and there would still be two identical correct values of t_d, which would make malicious behavior ineffectual.
- If the requester can tell which reported reputations are wrong, he can lower the local reputation she awards to the corresponding score managers.

Hence the distributed nature of the algorithm and the redundancy of the computation of the reputation values protect agents against *slandering attacks by the score managers*.

From the discussion above, we see that Protocol 1 is self-enforcing. Also, all agents are guaranteed to get as high a reputation as they fairly deserve, so the protocol is strictly co-utile.

5 Reputation-based co-utility enforcement

As discussed in Section 2, negative payoffs resulting from helping others (e.g. the bandwidth spent on transferring a file to another peer) may dissuade rational selfish agents from collaborating, thereby preventing a collaborative protocol from being co-utile. Negative payoffs associated to the execution of the s-th protocol step by an agent $\mathscr{P}_i$ can be formally added to the agent's aggregated utility $u_{i,s}$ as a negative cost $o_{i,s}$ weighted by a coefficient α_i^o expressing the importance attached by $\mathscr{P}_i$ to the cost:

$$u'_{i,s} = u_{i,s} - \alpha_i^o \cdot o_{i,s}$$

If, as a result of the weighted negative cost, the aggregated utility $u'_{i,s}$ is negative, the agent will rationally choose not to execute the s-th protocol step.

Reputation can be used as a positive incentive that neutralizes negative payoffs, as long as it is aligned with the nature of the negative payoffs. For example, if costs represent the reluctance of agents to collaborate with each other (because of the fear to interact with strangers or misbehaving agents), reputations can reflect how much a certain agent is trusted based on the outcome of past interactions with her. In this case, positive reputations t_j of other peers $\mathscr{P}_j$ can be used to neutralize the negative costs incurred by a certain agent $\mathscr{P}_i$ when executing a protocol step that would be beneficial for $\mathscr{P}_j$. Thus, $\mathscr{P}_j$'s reputation at the time of running step s, say $t_{j,s}$, is subtracted from the cost, as follows:

$$u'_{i,s} = u_{i,s} - \alpha_i^o \cdot (o_{i,s} - \alpha_i^t \cdot t_{j,s}), \tag{1}$$

where α_i^t is explained next. Putting aside the other atomic utilities, the cost $o_{i,s}$ incurred by agent $\mathscr{P}_i$ can be seen as a threshold that specifies the *minimum reputation* value that $\mathscr{P}_j$ should have in order for $\mathscr{P}_i$ to follow the protocol that will also help $\mathscr{P}_j$. In this context, α_i^t is a coefficient that allows coherently aggregating cost and reputation values (because global reputations are normalized in the range [0..1], whereas the cost can be an unbounded value). In this way, the α_i^o coefficient now weights the difference between the actual cost $o_{i,s}$ and the agent reputation $t_{j,s}$ that compensates the former.

If several different costs (and reputations) can be associated to an action (e.g., in eBay, sellers are rated independently according to delivery speed, shipping costs, communication, etc.), this approach can be generalized in at least two different ways:

- *Aggregation*. For each peer, costs are aggregated into a single cost, and reputations are aggregated into a single reputation, so that the resulting aggregated reputation can compensate the resulting aggregated cost as per Expression (1).
- *Separation*. For each peer, each cost and each reputation are separately managed. In this case, the utility functions for the agents considering to collaborate will incorporate one compensation term for each cost type and corresponding compensating reputation type. That is, instead of subtracting a single term as in Expression (1), one would subtract one term for each cost-reputation pair being considered.

Reputations can also be viewed as atomic utilities that agents wish to increase (because by doing so they also increase the willingness of other agents to collaborate with them). As a result, an agent $\mathscr{P}_i$ may consider as an additional atomic utility her own reputation gain resulting from executing a certain protocol step s:

$$u'_{i,s} = u_{i,s} + \alpha_i^t \cdot (\hat{t}_{i,s+1} - t_{i,s}). \tag{2}$$

In Expression (2), $t_{i,s}$ is the reputation of $\mathscr{P}_i$ before running step s and $\hat{t}_{i,s+1}$ is an estimate of $\mathscr{P}_i$'s reputation $t_{i,s+1}$ after running step s. If agents rate the outcomes of

actions in an objective and/or similar way (e.g. spent bandwidth, binary fulfillment of an action, etc.), then the exact value of $t_{i,s+1}$ can be anticipated; otherwise, if reputation values result from subjective opinions (e.g., how pleasant was a trip in a shared car), only the estimate $\hat{t}_{i,s+1}$ can be computed. This shows the importance of agents measuring reputations in a homogeneous way.

With the utility of Expression (2), agents are motivated to increase their reputations indefinitely. However, if reputation values measure the difference between the cost incurred to help other agents and the cost caused to other agents for the help received from them (as in the file sharing scenario), agents are no longer interested in systematically increasing their reputation (because of the incurred costs). They will rather aim at maintaining a reputation value that is barely sufficient to obtain collaboration from other peers (e.g. slightly greater than 0 in the file sharing scenario, which means that the agent has helped others more than what others have helped him). Let such target reputation value be t_{target}. Then the utility expression that represents the interest of the agent in achieving this value (but not more than it) is:

$$u'_{i,s} = u_{i,s} + \alpha_i^t \cdot \min((\hat{t}_{i,s+1} - t_{i,s}), (t_{target} - t_{i,s})). \tag{3}$$

In plain words, Expression (3) reflects that an agent that accumulates more reputation than the target does not obtain any more utility than if she sticks to the target. Hence, if step s of the protocol takes the agent beyond the reputation target, she will not be interested any more in continuing to help others in the next step, because helping is costly and she does not need to increase her reputation further. If the agent subsequently requests and obtains help from other peers, her reputation will decrease; as soon as it falls below the target, the agent will be again interested to help others in order to reach the target once more. Thus, agents with utility given by Expression (3) tend to stick to the target reputation, rather than systematically increasing their reputation. If reputation is the difference between costs borne and costs caused to others, everyone's target reputation will be just slightly above 0, which results in a fair distribution of costs.

The additional atomic utilities considered in the previous expressions can also be combined. For example, an agent may evaluate both the reputation of the peer she is helping (with regard to the cost this help will cause) and also the reputation increase that this action is likely to bring to her own reputation. Depending on the importance attached by each agent to each atomic utility and the values of these utilities/reputations, agents with low reputations may become motivated to collaborate more (in order to increase their reputations), even with agents with not so high reputations. This feature is useful in the file sharing scenario, for example, in which reputations measure the difference between the spent upload and download bandwidths: agents with a low reputation (i.e., those who have downloaded more data than they have uploaded) will be more motivated to fulfill file requests by other peers; otherwise, agents with good reputation are likely to turn down their file requests (because of the low reputation of the requesters).

The reputation calculation protocol enables all agents in the network to faithfully demonstrate their reputation/collaboration record in an anonymous way. Thus, the global reputation t_i achieved by each agent (which can influence the decision of other peers to collaborate) is not a component of the secret type of an agent (which could be manipulated to cheat other peers into changing their strategies), but rather it is public global information that is both accessible and reliably verifiable by the other peers at all times. This fact, in turn, ensures that the reputation atomic utility is a reliable incentive to turn a non-co-utile protocol into a co-utile one.

6 Conclusions

Co-utility happens when selfish agents rationally help each other. However, in many situations, negative payoffs (cost, reluctance, fear, etc.) may render mutual help not rational. In this chapter, we have demonstrated the use of reputation to provide artificial incentives that can compensate negative payoffs, thereby making co-utility still attainable. Specifically, we have described the adaptation and extension the EigenTrust mechanism to obtain a general distributed reputation management protocol that can be applied to a variety of scenarios and reputation needs. The resulting protocol is itself strictly co-utile, robust against a number of attacks and compatible with peer anonymity. In this way, we have shown how we can achieve a virtuous circle by which the reputation mechanism is self-enforcing and, in turn, enables co-utility in protocols that would not be co-utile without reputation, due to negative payoffs. This provides a key tool to design self-enforcing protocols that favor mutual help and boost social welfare even in case the agents' natural interests are inauspicious to this end.

Acknowledgements Funding by the Templeton World Charity Foundation (grant TWCF0095/AB60 "CO-UTILITY") is gratefully acknowledged. Also, partial support to this work has been received from the Government of Catalonia (ICREA Acadèmia Prize to J. Domingo-Ferrer and grant 2014 SGR 537), the Spanish Government (projects TIN2014-57364-C2-1-R "SmartGlacis", TIN2015-70054-REDC and TIN2016-80250-R "Sec-MCloud") and the European Commission (projects H2020-644024 "CLARUS" and H2020-700540 "CANVAS"). The authors are with the UNESCO Chair in Data Privacy, but the views in this work are the authors' own and are not necessarily shared by UNESCO or any of the funding bodies.

References

1. Adler, B., de Alfaro, L. (2007) A content-driven reputation system for the Wikipedia. In: Proceedings of the 16th International Conference on World Wide Web (WWW), ACM, pp. 261-270.
2. BlaBlaCar, http://www.blablacar.com/. Accessed: 14 Oct 2015.
3. Carverlee, J., Liu, L., Webb. S. (2010) The SocialTrust framework for trusted social information management: architecture and algorithms. Information Sciences 180(1):95–112.

4. Domingo-Ferrer, J., Megías, D. (2016) Co-utility for digital content protection and digital forgetting. In: Proceedings of the 15th Annual Mediterranean Ad Hoc Networking Workshop, MedHocNet 2016, IEEE, pp. 1–7.
5. Domingo-Ferrer, J., Farràs, O., Martínez, S., Sánchez, D., Soria-Comas, J. (2016) Self-enforcing protocols via co-utile reputation management. Information Sciences 367-368:159–175.
6. Domingo-Ferrer, J., González-Nicolás, Ú. (2012) Rational behavior in peer-to-peer profile obfuscation for anonymous keyword search. Information Sciences 185:191–204.
7. Domingo-Ferrer, J., Martínez, S., Sánchez, D., Soria-Comas, J. (2017) Co-utility: self-enforcing protocols for the mutual benefit of participants. Engineering Applications of Artificial Intelligence 59:148-158.
8. Domingo-Ferrer, J., Sánchez, D., Soria-Comas, J. (2016) Co-utility: self-enforcing collaborative protocols with mutual help. Progress in Artificial Intelligence 5(2):105–110.
9. Domingo-Ferrer, J., Soria-Comas, J., Ciobotaru, O. (2016) Co-utility: self-enforcing protocols without coordination mechanisms. In: Proceedings of the 2015 International Conference on Industrial Engineering and Operations Management, IEOM 2015, IEEE, pp. 1-17.
10. Feldman, M., Padimitriou, C., Chuang, J., Stoica, I. (2006) Free-riding and whitewashing in peer-to-peer systems. IEEE Journal on Selected Areas in Communications 24(5):1010–1019.
11. Guha, R., Kumar, R., Raghavan, P., Tomkins, A. (2004) Propagation of trust and distrust. In: Proceedings of the 13th International Conference on the World Wide Web, ACM, pp. 403–412.
12. Hoffman, K., Zage, D., Nita-Rotaru, C. (2009) A survey of attack and defense techniques for reputation systems. ACM Computing Surveys 42(1):art 1.
13. Kamvar, S. D., Schlosser, M. T., Garcia-Molina, H. (2003) The EigenTrust algorithm for reputation management in P2P networks. In: Proceedings of the 12th International Conference on World Wide Web, ACM, pp. 640–651.
14. Kim, Y. A., Phalak, R. (2012) A trust prediction framework in rating-based experience sharing social networks without a Web of Trust. Information Sciences 191:128–145.
15. Levine, B. N., Shields, C., Margolin, N. B. (2006) A survey of solutions to the Sybil attack. In: Tech report 2006-052, University of Massachusetts Amherst, Amherst, MA.
16. Marti, S., Garcia-Molina, H. (2006) Taxonomy of trust: categorizing P2P reputation systems. Computer Networks 50(4):472–484.
17. Pouwelse, J. A., Garbacki, P., Epema, D. H. J., Sips, H. J. (2005) The Bittorrent P2P file-sharing system: measurements and analysis. In: 4th International Workshop on Peer-To-Peer Systems, LNCS 3640, Springer, pp. 205–216.
18. Sánchez, D., Martínez, S., Domingo-Ferrer, J. (2016) Co-utile P2P ridesharing via decentralization and reputation management. Transportation Research Part C 73:147–166.
19. Singh, A., Liu, L. (2003) TrustMe: anonymous management of trust relationships in decentralized P2P systems. In: Proceedings of the 3rd International Conference on Peer-to-Peer Computing, pp. 142–149.
20. Srivatsa, M., Xiong, L., Liu, L. (2005) TrustGuard: countering vulnerabilities in reputation management for decentralized overlay networks. In: Proceedings of the 14th International Conference on the World Wide Web, ACM, pp. 422–431.
21. Soria-Comas, J., Domingo-Ferrer, J. (2015) Co-utile Collaborative Anonymization of Microdata. In Torra, V., Narukawa, Y. (eds) MDAI, Springer, pp. 192–206.
22. Turi, A. N., Domingo-Ferrer, J., Sánchez, D., Osmani D. (2015) Co-utility: conciliating individual freedom on common good in the crowd based business model. In: Proceedings of the 12th International Conference on e-Business Engineering, ICBE 2015, IEEE, pp. 62–67.
23. Turi, A. N., Domingo-Ferrer, J., Sánchez, D., Osmani D. (2016) A co-utility approach to the mesh economy: the crowd-based business model. Review of Managerial Science. In press.
24. Turi, A. N., Domingo-Ferrer, J., Sánchez. (2016) Filtering P2P loans on co-utile reputation. In: Proceedings of the 13th International Conference on Applied Computing-AC 2016, pp. 139–146.
25. Walsh, K., Sirer, E. G. (2006) Experience with and object reputation system for peer-to-peer filesharing. In: Symposium on Networked System Design and Implementation-NSDI'06.

26. Zhou, R., Hwang, K. (2007) PowerTrust: a robust and scalable reputation system for trusted peer-to-peer computing. IEEE Transactions on Parallel and Distributed Systems 18(4):460–473.
27. Zhou, R., Hwang, K., Cai, M. (2008) GossipTrust for fast reputation aggregation in peer-to-peer networks. IEEE Transactions on Knowledge and Data Engineering 20(9):1282–1295.