Managing Obligation Delegation: Incentive Model and Experiment

Managing Obligation Delegation: Incentive Model and Experiment

Liang Chen[1], Cheng Zeng[2], Stilianos Vidalis[1], and Wei Jie[3]

[1] *Department of Computer Science, University of Hertfordshire, College Lane, Hatfield, AL10 9AB, United Kingdom*

[2] *Central Southern China Electric Power Design Institute, No. 668 Minzhu Road, Wuchang District, Wuhan, 430071, China*

[3] *School of Computing and Engineering, University of West London, St Mary's Road, Ealing, W5 5RF, United Kingdom*

Abstract

Obligations are essential part of security policies, which specify what actions a user is obliged to perform in the future. One interesting feature of obligations is unenforceable, that is, the system cannot guarantee that each obligation will be fulfilled. Indeed, obligations go unfulfilled for a variety of reasons. For example, a user may have a family emergency that leads her having little time to discharge assigned obligations. We argue that delegation of obligations can be regarded as a means of providing opportunity for obligations to be discharged. However, this opportunity will be wasted if users who received delegation do not fulfil the obligations eventually. In this paper we propose a mechanism that incentivises users to accept and fulfil obligations for others by rewarding users credits. The amount of credits that can be earned depends on their trust rating, which reflects precisely how diligent of individuals in fulfilling obligations in the past. Users are motivated to raise up their trust ratings by fulfilling obligations for others, in order to earn more credits in the future. To evaluate our approach, we develop a multiple-agent system that simulates a number of different profiles for agents and run experiments for one-hop delegation and cascaded delegation with those agents. The experiments show a rich set of results, one of which confirms that delegation with incentives achieves the best outcome in terms of the number of obligations being fulfilled. Also, we implemented the modified ϵ-greedy algorithm, one of the closely related existing works, in our experimental framework and compared its performance to our approach. The results show that our approach offers greater flexibility and efficiency, as well as a higher obligation fulfilment rate.

Keywords: Delegation of Obligations, Trust, Incentive, Multi-agent systems

1 Introduction

Obligation and authorisation are essential building blocks to form security policies [14, 18, 19, 21, 27], which define the correct behaviour of an information system. A system adopt-

ing this type of policy-based approach for governing agents' behaviour is called normative multiagent systems [4, 11, 12]. Obligations in such systems define what activities have to be performed by whom and when, and these obligatory actions are integral part of the control procedures in many organisations. For example, a course leader is obliged to submit a course assessment report to an external examiner at least three days before the exam board meeting. As the example illustrates, the system can determine whether and when the obligation is fulfilled, but cannot force the course leader to submit the report on time. In other words, the system cannot ensure that obligations can always be fulfilled, but instead it should give every *opportunity* for obligations to be discharged, in order to ensure the correct operation of a system.

One of such opportunities is to allow a user to delegate her assigned obligations to others. We refer to the user who performs a delegation as a "delegator", and the user who receives a delegation as a "delegatee". Indeed, there are a number of organisational motives behind the delegation of an obligation [6, 24]. For example, a user may have been assigned a few obligations which need to be fulfilled at similar sort of deadline. However, due to other work commitments, it would be desirable for the user to delegate some of the obligations to others who have similar competence but have less constraints. There has been some work on delegation of obligations, most of which focuses on operational semantics for delegation and mechanisms for monitoring the fulfilment of delegated obligations [24], as well as identifying responsibilities among users who involved in the delegation of obligations [3]. In multi-agent systems, the study of delegation mechanisms has been receiving a great deal of attention [10, 25], particularly trust-based approaches that are closely related to our discussion here. For example, Chris and Nir [5] explored various strategies for increasing or decreasing trust values for agents positioned differently within a delegation chain. Afanador *et al.* [1] examined the effectiveness of several multi-armed bandit algorithms as a trust system to determine which agents should be delegated tasks.

As the literature suggested, allowing the occurrence of delegation provides an opportunity for the delegator's assigned obligations to be discharged, however, it is not clear that why the delegatee would be willing to fulfil obligations for others unless there is an incentive for them to do so. If delegatees eventually leave the delegated obligations to go unfulfilled, then all the complexities resulting from managing the delegation become unnecessary. The question of how to incentivise delegatees to discharge delegated obligations, to the best of our knowledge, has not yet been adequately investigated. Such considerations are the focus of this paper. More specifically, the contributions of this paper are as follows:

- We introduce a simple trust computation method on the basis of the Subjective Logic model to compute trust rating of users, reflecting their performance on fulfilling obligations. We discuss a number of possible ways of updating the trust ratings for the delegator and delegatee when a delegated obligation is fulfilled or violated. We also discuss the notion of responsibility with respect to fulfilling an obligation; in particular, who is held responsible for a delegated obligation being violated.

- We further propose a credit rewarding scheme that rewards delegatee credits for fulfill-

ing obligations, but the amount of credits that can be rewarded is determined by her trust rating. From the delegator's perspective, he needs to pay credits to the delegatee on her efforts to fulfil the delegated obligation, thus the scheme avoids the situation where users are always seeking to delegate their assigned obligations to others.

- When a delegated obligation arises in the system, we define a set of eligibility criteria for delegatees to bid for the obligation. We also define a number of possible risk preferences a delegator may take in terms of choosing a delegatee, since the chosen delegatee has uncertainty of whether to fulfil the delegated obligation.
- We explore the incentive mechanism in the face of cascaded delegation of obligations with which users on a delegation chain may have different perception on fulfilling the obligation.
- We develop a sophisticated multi-agent system and run experiments to evaluate our models. The results reveal that allowing delegation with the proposed incentive mechanisms results in much less obligations being violated in the system for the one-hop delegation case, as well as the cascaded delegation case.
- We implement the modified ϵ-greedy algorithm [1], one of the closely related works for cascaded delegation, in our experimental framework and compared its performance with our models. The results suggest that our approach offers greater flexibility and efficiency, yielding better performance in terms of the fulfilment rate for delegated obligations.
- We consider the practical application of our proposed model by discussing two real-world scenarios: one for one-hop delegation and the other for cascaded delegation.

In the next section, we summarise relevant background materials on the trust model we employ, and present a simple way of defining obligation properties. We introduce a protocol for obligation delegation and explore incentive schemes for one-hop delegation as well as cascaded delegation in Section 3. In Section 4, we develop a multi-agent system as our experimental framework and conduct sophisticated experiments to evaluate our models. We also compare our models to existing work with respect to its experimental performance. In Section 5, we discuss how our contributions improve and extend related work, and consider the practical application of our models. We conclude the article with a summary of our contributions and some ideas for future work in Section 6.

2 Background

In this section we introduce a simple trust model based on the Bayesian principles and define essential properties of obligation.

2.1 Trust Model

While many more complex trust models exist, we adapt Josang's widely used Subjective Logic based approach [17], which is a relatively straightforward model grounded on Bayesian principles. The use of this model simplifies our experiments and allows us to highlight our approach.

An opinion held by an agent x about agent y regarding issue i is represented by a tuple:

$$\omega^x_{y:i} = \langle \alpha^x_{y:i}, \beta^x_{y:i}, \gamma^x_{y:i}, \delta^x_{y:i} \rangle$$
$$\text{where } \alpha^x_{y:i} + \beta^x_{y:i} + \gamma^x_{y:i} = 1, \quad \text{and } \delta^x_{y:i} \in [0, 1]. \quad (1)$$

The values of $\alpha^x_{y:i}$, $\beta^x_{y:i}$ and $\gamma^x_{y:i}$ respectively represent the degrees of *belief*, *disbelief*, and *uncertainty* regarding the proposition that the agent y will behave as agent x expects with respect to issue i. The base rate parameter $\delta^x_{y:i}$ represents the *a priori* degree of trust that the agent x has about y, before any direct evidence has been acquired.

Opinions are formed and updated based on observations of past performance using two parameters $r^x_{y:i}$ and $s^x_{y:i}$, capturing, respectively, the number of positive and negative experiences observed by x about y with respect to i. With these two parameters, x' opinion about y is computed as follows:

$$\alpha^x_{y:i} = \frac{r^x_{y:i}}{(r^x_{y:i} + s^x_{y:i} + 2)}$$
$$\beta^x_{y:i} = \frac{s^x_{y:i}}{(r^x_{y:i} + s^x_{y:i} + 2)}$$
$$\gamma^x_{y:i} = \frac{2}{(r^x_{y:i} + s^x_{y:i} + 2)} \quad (2)$$

For an initial opinion with no evidence, therefore, $\alpha^x_{y:i} = 0$, $\beta^x_{y:i} = 0$, $\gamma^x_{y:i} = 1$, and $\delta^x_{y:i}$ is typically set to 0.5. Given an opinion computed through Equation 2, a single-valued *trust rating*, which can be used to rank and compare individuals, can be obtained as follows.

$$T(\omega^x_{y:i}) = \alpha^x_{y:i} + \delta^x_{y:i} \cdot \gamma^x_{y:i} \quad (3)$$

2.2 Obligation Trust

There exists a number of policy languages being designed for specifying obligation policies. The most common approach is to use the temporal logic to capture time constraints associated with obligations [13, 28]. However, we take an approach that is similar to [7, 16], which defines a simple data structure capturing the essential components of an obligation.

We assume the existence of a clock, whose ticks are indexed by the natural number $\mathbb{N}$. A *time interval* $i = [t_1, t_2]$, where $t_1, t_2 \in \mathbb{N}$ and $t_1 < t_2$, is the set $\{t_1 \leqslant t \leqslant t_2\}$. We define an obligation *obl* as a tuple $\langle u, a, i \rangle$, where u is a user, a is an action, i is a time interval during which u is obliged to take action a. At a particular point of time t, an obligation

$obl = \langle u, a, [t_s, t_e] \rangle$ may be in one of three states: active, satisfied, or violated. We say obl is *satisfied* if u has fulfilled the obligation (performed action a) at t and $t \in [t_s, t_e]$. We say obl is *violated* if the obligation has not been fulfilled, but t_e has passed ($t > t_e$). We say obl is *active* at t if it is neither satisfied nor violated.

It is important to keep the system always in a *desirable* state where no obligations go violated. However, since the system cannot enforce users to fulfil obligations, some obligations may go violated. What the system can do is to monitor the status of the obligation (e.g., whether it has been violated at some point of time) and to use reward and blame mechanisms to incentivise users to fulfil obligations.

We introduce the concept of *obligation trust* for every user, which represents an user' trustworthiness to fulfilling obligations. Our approach to computing obligation trust for a user is based on the *evidence* the system has observed regarding the user's performance on fulfilling obligations in the past. The evidence we have is a sequence of good (satisfaction of an obligation) and bad (violation) experiences with that user. These experiences can be used to estimate the probability that the user will make obligations being satisfied in the future, and such a probability distribution is called the user's obligation trust in this paper.

Formally, let H be a history of obligation stratification events, whose members are the form of $\langle u, obl, sts \rangle$, where $sts \in \{\mathsf{satisfied}, \mathsf{violated}\}$, representing that the status of obligation obl is caused by user u. Let $r^{sys}_{u:obl}$ represent the observed number of obligations that are satisfied by u and let $s^{sys}_{u:obl}$ represent the observed number of obligations that are violated by u, where the superscript sys represents the system who is an opinion owner. Then we can compute the obligation trust, $T(\omega^{sys}_{u:obl})$, of an user u by following the Equations 2 and 3. In other words, $T(\omega^{sys}_{u:obl})$ represents the system's opinion on the obligation trust rating of u. It can be seen that the initial value for $T(\omega^{sys}_{u:obl}) = 0.5$ when $r^{sys}_{u:obl} = s^{sys}_{u:obl} = 0$, but it will be updated as new evidence appears in the history H. When $r^{sys}_{u:obl}$ and $s^{sys}_{u:obl}$ are obvious from context, we will simply write r_u for $r^{sys}_{u:obl}$, s_u for $s^{sys}_{u:obl}$, and $T(u)$ for $T(\omega^{sys}_{u:obl})$.

3 An Incentive Scheme for Obligation Delegation

In this section, we first introduce the protocol for how the delegation of obligations operates and examine how to update trust ratings as a means of assigning blame or rewards to individuals involved in the delegation process. We then explore the concept of earning reward credits as the key incentivising mechanism and define risk preferences regarding how delegators choose delegatees to maximise their reward credits. Finally, we examine how the incentivising scheme works in cascaded delegation settings.

3.1 Delegating Obligations

In many situations, a user needs to delegate her assigned obligations to someone else, so that the obligations can be discharged by others rather than left to be violated. For example, a user may have too many obligations that need to be discharged within the same period of time, or a user may need to deal with other urgent tasks whose completion are in conflict

with the deadline of assigned obligations. Let us first introduce a delegation protocol, and its resulting data structures. We write $\mathsf{deleg}(obl, W)$ to denote a *delegation request*, meaning that the responsible user appearing in obligation obl requests to delegate it to some user w in the group W. The delegation request $\mathsf{deleg}(obl, W)$ will be evaluated by running a protocol, resulting in a delegatee w being chosen. We informally describe how the protocol works as follows:

1. A user u starts off broadcasting a delegation request $\mathsf{delg}(obl, W)$ to a group of users W in the system. This group of users, for example, has a similar competence level or job responsibilities within an organisation.

2. On receipt of the delegation announcement, each user $w \in W$ who wishes to put themselves forward will evaluate it with respect to their own schedule. If w is eligible to bid, w will submit her current obligation trust $T(w)$ to u;

3. Based upon several such bids being received in response to the announced delegation, u selects the most appropriate user to be assigned with the obligation obl.

4. Finally, u sends an award message to the successful bidder w, and also informs others whose bids were not successful.

In Sects. 3.3 and 3.4, we elaborate on how each user w assesses the eligibility against her own schedule, and how u makes a choice when multiple bids are received.

Following a successful running of the delegation protocol, we assume that the delegatee w has agreed to discharge the delegated obligation, while the delegator still remains as a responsible user for the obligation. We call delegatee w as an *obligated user* for the delegated obligation. We introduce a data structure, namely *delegated obligation*, that records an user-obligation assignment that arises from a delegation request being evaluated. A delegated obligation obl_d extends obl tuple with one additional element w, that is $obl_d = \langle u, a, i, w\rangle$, where w is a user obligated to take action a during the time period i, whereas u is the original user taking responsibility for the fulfilment of obl_d. Specifically, suppose that a delegation request $\mathsf{deleg}(obl, W)$ is evaluated at time t, then this results in updating obl as obl_d, where $obl_d = \langle u, a, (t+1, t_e), w\rangle$. We assume that u is no longer able to fulfil obl_d once it is generated, and only w can, that is because our incentives schemes introduced in Sect. 3.3 would make u fulfil the obligation herself if she is capable to do so. However, user u may take some penalty if obl_d goes violated, thus keeping u in the data structure provides convenience for linking it to H^{obl}.

3.2 Updating Obligation Trust

We now explore how to incentivise users to discharge delegated obligations when delegations described above occur in the system. In such cases, it is not straightforward to give an appropriate assignment of blames or rewards to individuals who are involved in the delegation process.

Let us first clarify the notion of *responsibility* with respect to an obligation. Given an obligation $obl = \langle u, a, i \rangle$, we say that u is held responsible for fulling the obligation. If obl goes violated, the responsible user u should be blamed for the failure. However, when a delegated obligation $obl_d = \langle u, a, i', w \rangle$ arises, we must determine to what extent u is held responsible for obl_d. Broadly speaking, there are three possible cases: *retain*, *share* and *transfer*.

Retain The first retain case means, following a successful delegation of obligation, user u is still solely responsible for obl_d. If w does not complete the obligation within the time interval i', w is not going to be blamed but u will. It is obvious that, in this case, there is no incentive for w to fulfil obl_d, and thus we will not further study it in this paper.

Share The share of responsibility means, after successful delegation, responsibility for fulfilling the obligation concerned is shared between u and w. In other words, both u and w are held responsible for making obl_d being fulfilled.

Transfer In the transfer case, when the delegation succeeds, the responsibility for fulfilling obl_d is transferred to the delegatee w; in particular, the delegator u is no longer held responsible for obl_d.

For the share and transfer cases, it is interesting to explore a number of ways for assigning rewards and blames to both u and w, in order to incentivise obl_d to be fulfilled. This is achieved by proposing different ways of updating obligation trust for u and w as follows.

Given a delegated obligation obl_d, there are two users involved: delegator u and delegatee w. We introduce two *weighting functions* $f_r : \{u, w\} \to [0, 1]$ and $f_s : \{u, w\} \to [0, 1]$, where $f_r(u) + f_r(w) = 1$ and $f_s(u) + f_s(w) = 1$. With the weighting function f_r, the full positive update value of 1 is distributed among users u and w. The function f_s serves the negative update among u and w in an analogous fashion. More specifically, given a delegated obligation obl_d and weighting functions f_r and f_s, we update r and s for users u and w as follows:

$$(r_u, s_u) = \begin{cases} (r_u + f_r(u), s_u) & \text{if } obl_d \text{ is satisfied} \\ (r_u, s_u + f_s(u)) & \text{otherwise} \end{cases} \tag{4}$$

$$(r_w, s_w) = \begin{cases} (r_w + f_r(w), s_w) & \text{if } obl_d \text{ is satisfied} \\ (r_w, s_w + f_s(w)) & \text{otherwise} \end{cases} \tag{5}$$

The two weight functions provide great flexibility on designing a number of possible ways of updating r and s for both users u and w to different degrees. We discuss the flexibility with the following four approaches: the first approach responds to the case of *transfer* of responsibility whereas the rest of the three approaches address the *sharing* of responsibility case.

1. The first approach reflects that w is only user taking responsible for the fulfilment outcome of obl_d. More specifically, if w satisfies obl_d, the system counts it as one

positive evidence added to r_w ($f_r(w) = 1$), and there is no update on r_u ($f_r(u) = 0$). On the other hand, if w violates obl_d, then the negative evidence is only added to s_w ($f_s(w) = 1$), not to s_u ($f_s(u) = 0$). It means that u transfers the responsibility of fulfilling obl_d to w when the delegation succeeds. This may be acceptable to w, as w takes advantage of this to "repair" her positive evidence. However, as the system get evolved, when w sees herself not be able to fulfil obl_d, she may choose to further delegate it to someone else, as she does not want to be penalised with the negative evidence being incremented. We will explore the situation of cascaded delegation in Sect. 3.5.

2. When considering sharing of responsibility between u and w, it is not necessary to hold u and w equally responsible for the fulfilment outcome of obl_d. This is the approach we consider here. Unlike the first approach, if w violates obl_d, then the system counts it as one negative evidence added to s_u ($f_s(u) = 1$), while s_w ($f_s(w) = 0$) remains unchanged. Like the first approach, if obl_d is satisfied, one positive evidence is added to r_w only ($f_r(w) = 1$ and $f_r(u) = 0$). This way of updating obligation trust possesses two appealing characteristics:

   - User w is incentivised to satisfy obligations for others, because this helps to repair its obligation trust rating by increasing its r_w value;
   - If u decides to delegate an obligation to others, then u needs to choose an *appropriate* user so as to ensure that its own s_u would not increment.

   Note that this approach is not without its problems. It may lead u to only delegate obl_d to someone w who she knew, because, for any stranger w, there is no negative impact on w's trust rating if obl_d goes violated. In other words, the incentive for w to fulfil obligations for others is weak in this setting.

3. We now look at a situation in which the weight is evenly divided between u and w, holding them equally responsible for the fulfilment of obl_d, that is $f_r(u) = f_r(w) = f_s(w) = f_s(u) = 0.5$. If w satisfies obl_d, $r_u = r_u + 0.5$ and $r_w = r_w + 0.5$. Likewise, if w violates obl_d, $s_u = s_u + 0.5$ and $s_w = s_w + 0.5$. However, someone may argue that this approach leads u being unfairly rewarded: u may never fulfil obligations for herself or others, but her obligation trust is still increasing due to all obligations for which she is originally responsible are being fulfilled by others.

4. We can rectify the unfairness of the third approach by adjusting the weighting function that gives one positive evidence to w ($f_r(w) = 1$) when obl_d is discharged by w. If obl_d goes violated, u and w are equally penalised by adding 0.5 to their s parameters ($f_s(w) = f_s(u) = 0.5$). Of course, we can slide a bit more negative weights on either direction - towards w or u, for example, setting $f_s(u) = 0.6$ and $f_s(w) = 0.4$. However, the principle is that both u and w should be blamed if obl_d is violated.

Following the discussion of the four approaches to updating obligation trust, we believe that the weights setting at the *fourth* approach is the most appropriate one in terms of

incentivising w to fulfil obl_d with respect to the sharing of responsibility, whereas the *first* approach is the straightforward one corresponding to the transfer of responsibility. We report our experimental results on the effects of these two approaches in Sect 4.

3.3 Earning Reward Credits

In essence, the four approaches being discussed focus on the ways of updating *obligation trust* for users involved in the fulfilment of delegated obligations. However, it is not clear why a user who has high or full obligation trust is willing to accept a delegated obligation and discharge it within a deadline. This in fact leads to a more fundamental question: What is user's obligation trust used for?

Before addressing this question, let us extend the structure of obligation to include an element, called *reward credit*. That is, $obl = \langle u, a, i, c\rangle$, where $c \in \mathbb{R}$ is *reward credit* associated with obl. It means, when obl is fulfilled by u, u is rewarded with the credit c. Of course, u would not receive c if obl goes unfulfilled. For ease of exposition we consider c to be a *constant* for every obligation. Similarly, obl_d is extended to be $\langle u, a, i', c, w\rangle$.

We then take some ideas from the Principal-Agent model [15] to propose an incentive mechanism that utilises each user's obligation trust. If a user decides to delegate one of her assigned obligations to someone else, she has to pay the delegatee some of the reward credit associated with the obligation. Likewise, if a user is willing to take some effort to fulfil an obligation for others, her effort should be paid off by some credit from the obligation. Our basic idea is that a user whose obligation trust is high can charge relevantly more credits, while a user whose trust rating is low receives relevantly less credits.

Formally, suppose that a delegated obligation obl_d is discharged by user w and c is a reward credit associated with obl_d, w's payoff (or "utility") is $\pi = c \times T(w)$, where $T(w)$ is w's obligation trust. This serves as an incentive for users to fulfil obligations for others not only to earn credits but also to bring up their trust rating in order to earn more credits in the future. From delegator u's point of view, u's payoff (or "profit") is the difference between the credit c associated with obl_d and the credit paid to delegatee w, that is $\phi = c - \pi$. A user w, for example, has a full trust rating ($T(w) = 1$) can earn the whole credit c, thus there is no payoff left for u.

For simplicity, we assume that u is rational with an objective of choosing a delegatee w to maximise her payout ϕ. Choosing a user w who has the least trust rating indeed maximises ϕ, but has a high risk that obl_d will go violated, leading to losing credit c completely and her obligation trust being reduced, for the case of share responsibility. Let us look at a simple example to examine the possible choices of u in more details. Given a delegated obligation obl_d whose reward credit is $c = 100$, there are three users w_1,w_2, w_3 eligible to accept obl_d, where $T(w_1) = 0.2$, $T(w_2) = 0.6$, $T(w_3) = 0.3$. Hence u can earn a payoff $\phi_{w_1} = 80$, $\phi_{w_2} = 40$ and $\phi_{w_3} = 70$ by choosing w_1, w_2 and w_3 respectively. In other words, u has probability of 0.2 of earning 80 credits by delegating obl_d to w_1; 40 credits with probability of 0.6 by delegating obl_d to w_2; 70 credits with probability of 0.3 by delegating obl_d to w_3.

With this example, which user w_1, w_2 or w_3 should u delegate obl_d to? We define three types of delegator u with respect to their *risk preference* when choosing w:

- *Risk-averse delegators*: This type of user u prefers to take a low uncertainty on whether delegated obligations will be satisfied, thus she always choose a delegatee w whose trust rating is high but with a low return of payoff (credits). This type of u would choose w_2 as a delegatee for the example above.

- *Risk-seeking delegators*: This type of user u is willing to take a greater uncertainty on the obligation fulfilment in exchange for the potential of higher return of credits. In other words, u would choose w with least trust rating in order to receive the highest payoff. A risk-seeking u would choose w_1 as a delegatee.

- *Risk-average delegators*: This type of user u is seeking a choice that achieves a great balance between return of credits and likelihood of obtaining the return. Taking the example above, u first calculates $\frac{1}{3}\sum_{i=1}^{3}(\phi_{w_i} \times T(w_i)) = 20.33$ and then chooses w_3 whose value $\phi_{w_3} \times T(w_3) = 21$ is closest to 20.33.

In Sect. 4, we run experiments to evaluate the performance of earning credits by the three types of delegators.

3.4 Eligibility of Delegatees

With the credit reward scheme introduced above, delegators would only wish to delegate obligations to others when they are legitimate to do so, because, compared with discharging obligations themselves, they are worse off in terms of earning credits by letting others fulfil their obligations. However, from the delegatee's perspective, they may wish to earn as much credits as they can by fulfilling obligations for others. When a delegation request for an obligation arises in the system, the system may want to restrict who are the eligible users to bid for accepting the obligation. We certainly want to exclude a user who already has a large number of active obligations whose deadlines clashed with the one at request.

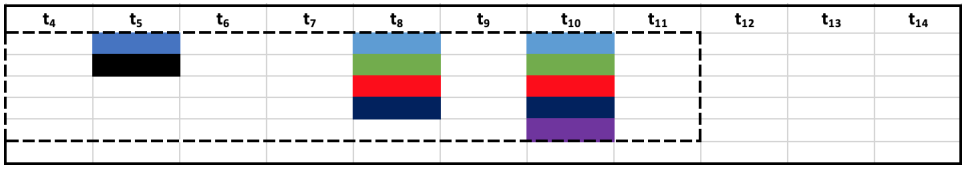


Figure 1: A schedule for w at t_4

We introduce a mechanism that establishes a schedule for each user in the system in terms of their assigned obligations. A *schedule* of a user w is a kind of look-up table that is indexed by time clocks. When an obligation is assigned to w, its fulfilment deadline t_e is marked in the schedule. Fig. 1 shows an example of schedule for w where there are two obligations whose deadline is at t_4, four obligations at t_8, and five obligations at t_{10}.

We also implement a *restricted window* of size n on the schedule, where n indicates the number of time clocks the system would look at, in order to determine how busy w is within the window. Then we can define a threshold h: The system may say that w is not eligible

to accept more obligations if the number of marks within the window n exceeds h. This tends to encode rules such that "you cannot work more than h hours in the next n days". Take the example in Fig. 1, suppose that $h = 11$ and $n = 7$, when an obligation obl arises in the system at t_4, user w is not allowed to bid obl, because her current active obligations within the current window $n = 7$ (shown in dotted boarder in Fig. 1) reaches the threshed $h = 11$. Of course, we can restrict that $h \leqslant n$ in case that each time clock is only allowed to complete one obligation. For example, if we set $h = 5$, then it means that within the next time window $n = 7$, the number of active obligations should not exceed 5.

Having defined criteria for assessing whether or not a user is eligible to accept a delegated obligation, we are now in a position to confirm the computation for Step 2 of the delegation protocol introduced at Sect. 3.1. We can also see that how u chooses a delegatee w at Step 3 of the protocol depends upon their risk preference on earning credits.

3.5 Cascaded Delegation of Obligations

There are some situations where the obligated user w may wish to further delegate obl_d to a third user who is better qualified to fulfil obl_d. Take the example of Fig. 1, user w, at t_6, may realise that she is no longer able to discharge one of her previously awarded obligations, whose deadline is at t_{10}, and then she sends out a delegation request for the obligation in the system at t_6. Such a *delegation chain* is formed when an obligation is delegated from one user to another, who in turn delegates the obligation to a third user, and so on. We model the delegation chain by generalising the data structure of delegated obligations as $obl_d = \langle u, a, (t_s, t_e), c, \langle w \rangle_{i=1}^{k} \rangle$, where $\langle w \rangle_{i=1}^{k}$ is a sequence of users with the index running from $i = 1$ to $i = k$. For example, $obl_d = \langle u, a, (t_s, t_e), c, \langle w \rangle_{i=1}^{3} \rangle$, where $\langle w \rangle_{i=1}^{3} = \langle w_1, w_2, w_3 \rangle$ and w_3 is the *fourth* user in the delegation chain, since the first user in the chain would always be the originator u. Note that, given $obl_d = \langle u, a, (t_s, t_e), c, \langle w \rangle_{i=1}^{3} \rangle$, t_s is the time obl_d awarded to w_3, which should *not* be close to t_e yet, otherwise w_3 would not have sufficient time to fulfil it. When it is clear from the context, we also write $dc = \langle u, w_1, \ldots, w_k \rangle$ to denote a delegation chain.

Let us now examine whether we need to adjust the incentive mechanism in face of delegation chain. The first one is about how to update obligation trust for users appearing in a delegation chain. For the case of transfer of responsibility, we take the straightforward approach that gives one positive update to the last user w_k who fulfils the obligation obl_d, whereas giving one negative update to w_k if obl_d is violated. However, for the share responsibility case, we believe that it is sensible to update our trust in users in the delegation chain to various degrees, in order to reflect that a particular outcome of obligation fulfilment (satisfied or violated) should not reflect equally on all the users' responsibilities for this outcome, and therefore on their trustworthiness. More specifically, we apply the full weight 1 to the last user who discharged the obligation. If the obligation goes violated, we apply an increasing proportion of weight along the delegation chain. For example, the last user in the chain receives the most negative weight added to her s evidence parameter, while the originating user (the first one in the chain) is given least negative weight. Given a delegation chain $dc = \langle u, w_1, \ldots, w_k \rangle$, and $v_j \in dc$ where j denotes a position in the

sequence dc, we compute the negative weight added to s_j as:

$$\frac{2 \times j}{|dc|^2 + |dc|}$$

where $|dc|$ is *size* of the delegation chain dc. Given $obl_d = \langle u, a, (t_s, t_e), c, (w)_{i=1}^{3} \rangle$, it may not be possible for the originating user u to know in advance that w_1 will sub-delegate obl_d to w_2, and in turn w_1 would not know w_2 will sub-delegate obl_d to w_3. Thus it may not be fair to penalise u equally as the last user w_3 who failed to fulfil obl_d. In other words, users at the back of chain who are willing to bid the obligation should bear more responsibilities for fulfilling it.

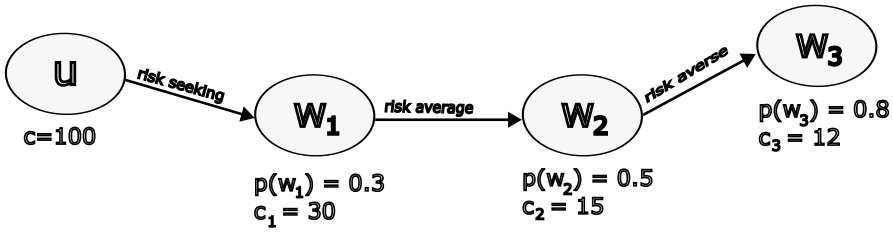


Figure 2: An example of a delegation chain

With respect to earning reward credits by users in the delegation chain, we take our previous approach that a delegator needs to pay the delegatee's effort in fulfilling obligations. However, the amount of reward credits available to earn decreases along the delegation chain. Take an example of a delegation chain shown in Fig. 2, where w_3 can only earn 12 credits ($T(w_3) \times c_2$) despite her trust rating being high ($T(w_3) = 0.8$), that is because w_2 herself only has 15 credits to give away, and this 15 credits is what she earned from c_1 ($T(w_2) \times c_1$). The best position here is u who can earn 70 credits if the obligation is eventually fulfilled by w_3. We can see that c_2 is relevantly low, which means it does not have much credits to earn in c_2, and thus is may be difficult to attract anyone to bid for fulfilling the obligation. This is reasonable because it indirectly forces w_2 to fulfil the obligation as a priority, rather than sub-delegating it. Unless there exists someone w_3 who only cares about increasing her obligation trust, not on how much credits she can earn from fulfilling the obligation.

When comes to risk perception of users in the delegation chain, our scheme provides great flexibility. That is, each user in the chain is free to take their own decision on how to choose a delagatee w. The decision can be made on the basis of their perception of which risk preference (risk-averse, risk-seeking, or risk-average) is the most appropriate one at a particular position of the chain. This results in a delegation chain where various risk preferences are adopted by users along the chain. Let us revisit the example in Fig. 2, where u took a risk-seeking approach for choosing w_1 in order to maximise her credit return 70, while w_1 took a risk-average approach when choosing w_2. User w_2 tends to choose someone w_3 who is most likely to fulfil the obligation by taking the risk-averse approach. We will be running experiments to confirm the performance of obligation fulfilment when allowing such a flexible cascaded delegation versus one-hop delegation only.

4 Evaluation

We developed a simulated multi-agent system to run experiments for evaluating our approach as described above. To provide an accurate assessment of our incentivising mechanisms, we simulate agent behaviour to closely resemble real-life user behaviour. Note that, for consistency with the agent-based environment, we refer to users in our model as agents in this section. In the following, we first outline our experimental framework and then present and analyse our experimental results. Finally, we implement the modified ϵ-greedy algorithm within our framework and compare its performance with our models.

4.1 Experiments

4.1.1 Experimental Setup

In our experiments, we create a fixed number of agents of 100, which is split into three groups with roughly equal in size: 33 risk-seeking agents, 33 risk-averse agents and 34 risk-average agents. We make the system generate 3000 obligations every 50 time ticks. Each obligation is randomly assigned to one of the 100 agents, and its fulfilment window $t_e - t_s$ is different, ranging from minimum 30 time ticks to maximum 80 time ticks.

At every time tick, there are a number of possible actions an agent can choose to take: *acting* on active obligations, *bidding* for delegated obligations or taking a *rest* (do nothing). We believe that it is reasonable to take into account the following assumptions on the setting of agents' behaviour:

- When an agent exhibits a "busy" status at a particular point of time, reflected by a large number of active obligations, the system should, at that time tick, make the agent highly likely act on these obligations and less likely participate in bidding other obligations.
- On the other hand, when an agent has fewer active obligations, the system should assign the agent a high probability of bidding obligations while relatively low probability of acting on the outstanding obligations.

Given an agent u, we write $P_t^u(act)$ to denote the probability that agent u acts on the active obligations at time tick t. We write $P_t^u(bid)$ and $P_t^u(rest)$ in an analogous fashion. We require that $P_t^u(act) + P_t^u(bid) + P_t^u(rest) = 1$. We set a fixed value for $P_t^u(rest)$ as 0.2, which represents that every agent always has an equal chance 0.2 of taking a rest at every time tick. In other words, we always have $P_t^u(act) + P_t^u(bid) = 0.8$.

We then define the eligibility criteria for bidding by setting the restricting window n as 50 time ticks and the threshold h as 40, which means an agent is not allowed to bid if her active obligations exceed 40 within the next 50 time ticks. Let m_t^u be a number of active obligations for agent u at time t. We compute $P_t^u(act) = \frac{m_t^u}{h} \times 0.8$. Suppose that an agent u' active obligations reaches the threshold at time t, that is $m_t^u = h$, then we have $P_t^u(act) = 0.8$, while $P_t^u(bid) = 0.8 - P_t^u(act) = 0$. It means u will not participating

in bidding at time t. In short, as the system evolves, $P_t^u(act)$ and $P_t^u(bid)$ have dynamic updates between each other, which automatically enforces the eligibility criteria for agents to bid.

4.1.2 Interpreting Acting on Obligations

We are now in a position to interpret the action of acting on active obligations as *two* separate concrete actions for agents: *fulfilling* obligations and *delegating* obligations. In other words, we split $P_t^u(act)$ into $P_t^u(ful)$ and $P_t^u(del)$, that is $P_t^u(act) = P_t^u(ful) + P_t^u(del)$, where $P_t^u(ful)$ represents the probability that agent u fulfils an active obligation at time t, while $P_t^u(del)$ represents the probability that u sends a delegation request at time t to the system for other agents to bid. In fact, $P_t^u(ful)$ has exactly the same semantics as u's obligation trust rating, and thus we can compute $P_t^u(ful)$ as $T_t(u) \times P_t^u(act)$. We then have $P_t^u(del) = P_t^u(act) - P_t^u(ful)$.

Having set up the probability of taking each possible action at every time tick, we classify the 100 users into another three groups with respect to their *profile* of managing obligations: *diligent*, *potential* and *unmotivated*. The three profiles are reflected by the different assignment of prior trust values θ with 0.6, 0.4, 0.2 respectively. Table 1 provides an interpretation of the three profiles when the system boots up. It can be seen that, given an agent u and the number of active obligations m_1 being assigned to u at time t_1, she would take $\frac{m_1}{h} \times 0.8$ chance of acting on the active obligations, $0.8 \times (1 - \frac{m_1}{h})$ chance of participating in bidding an obligation, and 0.2 chance of not taking any action. When breaking the probability of acting on obligations into the likelihood of fulfilling obligations and the likelihood of delegating obligations, we need to consider the agent's profile. For example, a user u belonging to the potential profile has initial trust rating of $T(u) = 0.4$. Hence, at time t_1, she would take $0.4 \times (\frac{m_1}{h} \times 0.8)$ chance of executing an obligation and $0.6 \times (\frac{m_1}{h} \times 0.8)$ chance of delegating an obligation. Of course, when the system evolves, all of these probability values get dynamically change, relying on their performance of fulfilling obligations.

With our setting above, the system now has in total *nine* types of users representing different behaviour in terms of fulfilling obligations and choosing delegatees. For example, a diligent agent has three possible ways of choosing delegatees: risk-seeking, risk-averse and risk-average. It is the same for potential and unmotivated agents.

Profile ID	Acting on obligations		Bidding obligations	Doing nothing
	Fulfilling obligations	Delegating obligations		
Diligent	$0.6 \times (\frac{m_1}{h} \times 0.8)$	$0.4 \times (\frac{m_1}{h} \times 0.8)$	$0.8 \times (1 - \frac{m_1}{h})$	0.2
Potential	$0.4 \times (\frac{m_1}{h} \times 0.8)$	$0.6 \times (\frac{m_1}{h} \times 0.8)$	$0.8 \times (1 - \frac{m_1}{h})$	0.2
Unmotivated	$0.2 \times (\frac{m_1}{h} \times 0.8)$	$0.8 \times (\frac{m_1}{h} \times 0.8)$	$0.8 \times (1 - \frac{m_1}{h})$	0.2

Table 1: Agent's behaviour at time t_1

4.1.3 Implementing the Delegation Protocol

We implement a communication protocol between agents for managing the delegation protocol defined in Sect. 3.1. This communication protocol runs in parallel to each possible action taken by an agent defined in Table 1. It takes 1 time tick for announcing a delegation request. We implement a time window $[t_i, t_j]$ allowing multiple agents to bid, which means it will not accept any bid after t_j. The length of the bidding window $[t_i, t_j]$ is defined as 10% of the remaining time ticks of a delegated obligation. For example, if an obligation has 100 time ticks left before it becomes violated, then we make the bidding window having 10 time ticks. Note that we set 1 time tick $(t_j + 1)$ for confirming a successful bidder.

We also implement the transfer and share responsibility cases described in Sect. 3.2 on updating obligation trust and computing credits for one-hop delegation as well as cascaded delegation.

4.1.4 Adding Incentives

Unlike humans, agents themselves would not proactively earn credits by fulfilling obligations unless there exists an incentive motivating them to do so. For example, given an unmotivated agent u at time t_1 whose trust rating is $T_1(u) = 0.2$, and assume that u has outstanding obligations $m^u_{t_1} = 20$ which makes $P^u_{t_1}(act) = 0.4$, we can compute $P^u_{t_1}(ful) = 0.2{*}0.4 = 0.08$ meaning that the agent only takes 8% chance of fulfilling obligations at time t_1. As the system evolves, some of u's outstanding obligations would become violated due to low $P^u_{t_1}(act)$ starting off, which results in $P^u_t(act)$ going even further down whereas $P^u_t(del)$ is increased. In other words, agent u ends up delegating obligations to others at most of the time and barely fulfilling any obligation herself. This suggests that agents need an incentive to make their $P^u_t(ful)$ go up gradually as the system evolves.

Our approach to providing an incentive is to shift a value from $P^u_t(del)$ to $P^u_t(ful)$ at every 50 time ticks, and the amount being shifted is determined by the *credit growth rate* of agent u during this period. In other words, we compute the credit growth rate by comparing the agent's credit, for example, at t_{50} with her credit at t_1, that is $(c^u_{t_{50}} - c^u_{t_1})/c^u_{t_1}$. We introduce an *exponential decay* function $f(x) = e^{-2x}$, where e is Euler's constant, $x \in [-1, 1]$ represents the credit growth rate, and $f(x)$ indicates the corresponding value that needs to be shifted from $P^u_t(del)$ to $P^u_t(ful)$. We can see from Fig 3 that, if the agent's credit growth rate is *negative*, a proportionally *large* $f(x)$ value is shifted from $P^u_t(del)$ to $P^u_t(ful)$. If the growth rate is positive, then a proportionally *small* value is shifted to $P^u_t(ful)$. Of course, we normalise the $f(x)$ value as $s \in [0, 0.8]$ for shifting, that is computing $P^u_t(del) - s$ and $P^u_t(ful) + s$. In summary, our incentivising approach here is to keep $P^u_t(ful)$ with a proportional increase at every 50 time ticks.

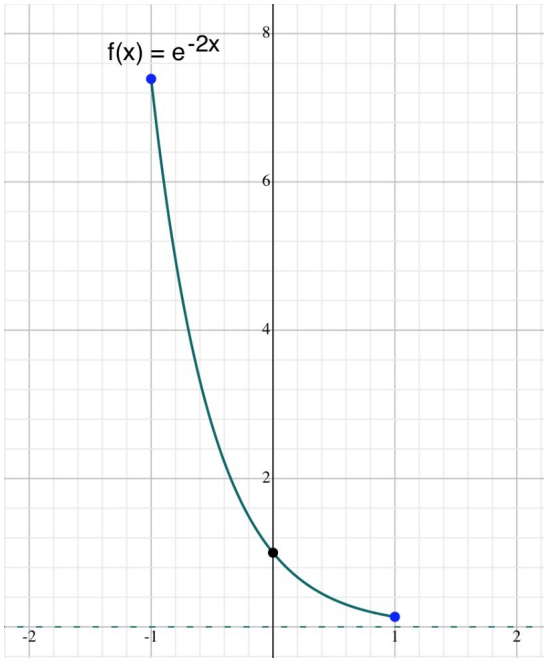


Figure 3: An exponential decay function for changing trust rating

4.2 Results

4.2.1 The Effect of Incentives

We run the above settings for 3000 time ticks, which generates 180,000 obligations in total. We now compare the results with respect to the following cases:

- *Delegation with no incentive*: This case considers the opportunity of allowing delegation to occur, which leads to agents' behaviour defined in Table 1 getting dynamically changed over the time. In order to make a comparison, we check the obligation fulfilment rate for the transfer of responsibility and the share of responsibility cases. As we expected, Fig. 4 shows that the fulfilment rate for both cases is similarly low, about 35%. That is because, with no incentive imposed, agents' trust rating goes down over the time, resulting in more and more obligations being violated. The change of agents' trust rating is illustrated in Fig. 5a and 5b. We can see that, for the transfer case, diligent agents' trust rating drops significantly at initial 1000 time ticks and then stabilises around 0.35 during the remaining times. However, for the share case, the trust rating of diligent agents goes down gently over the time from 0.6 to 0.38. Note that the change pattern of trust rating for both potential and unmotivated agents is consistent for the share and transfer cases.

- *Delegation with incentives*: This case takes into account the incentive mechanism we introduced within our experimental settings. Basically, the likelihood of each user

to fulfil obligations gets increase over the time because of the incentive to increasing their trusting rating with the exponential decay function. Fig. 5c and 5d show that, for both transfer and share cases, the trust rating of diligent users does not fall but instead levels off around 0.6 over the time. This is because their credit growth rate has been keeping as positive and most likely that the increase of their trust rating at every 50 time ticks has gradually reached the limit (as seen little change of the $f(x)$ value when $x \in [0.5, 1]$ in Fig. 3). In comparison, the trust rating of both potential and unmotivated agents has went up (with a significant grow for unmotivated agents) to around 0.52 for both transfer and share cases. This demonstrates that our incentivising mechanism takes a great deal of effect, resulting in the number of obligations being fulfilled (reaching 68%) as shown in Fig. 4

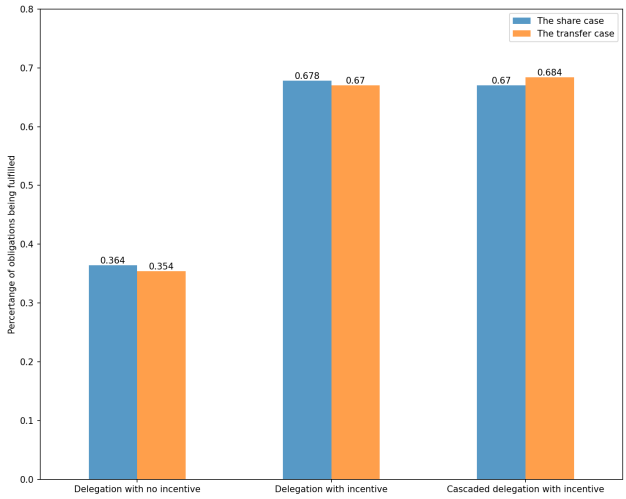
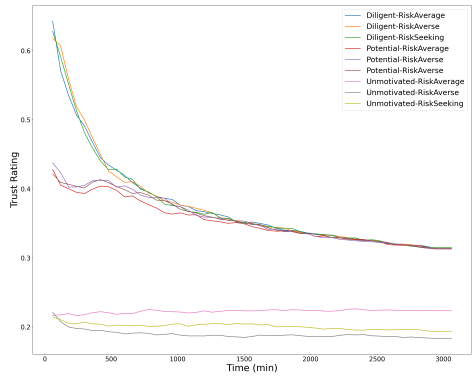


Figure 4: A comparison for obligation fulfilment

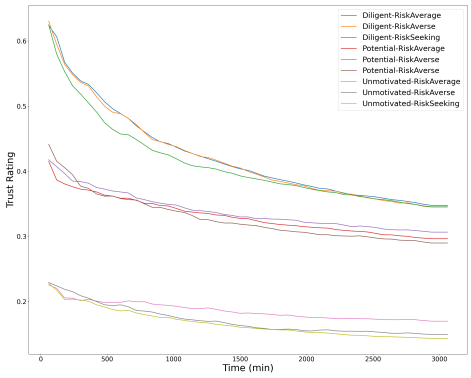
- *Cascaded delegation with incentives*: This case relaxes the restriction on the one-hop delegation to support all the features of cascaded delegation introduced in Sect. 3.5. An agent, at every time tick, has a $P_t^u(del)$ chance of choosing to further delegate an obligation that has been previously delegated to her, provided that the obligation has reasonable fulfilment window remaining (we set at least 40% of the original window as a condition). With the incentive mechanism employed, Fig. 4 shows another great result (67%) on the percentage of obligations being fulfilled. Looking at the change of trust rating for the different types of agents, Fig. 5e and 5f show little difference on the increase tread in comparison with the ones in Fig. 5c and 5d.

4.2.2 Further Analysis

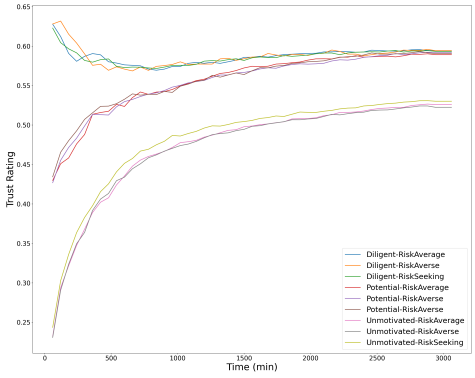
Our experiments produce a rich set of data capturing the behaviour of nine different types of agents with respect to fulfilling and delegating obligations. We would like to do further



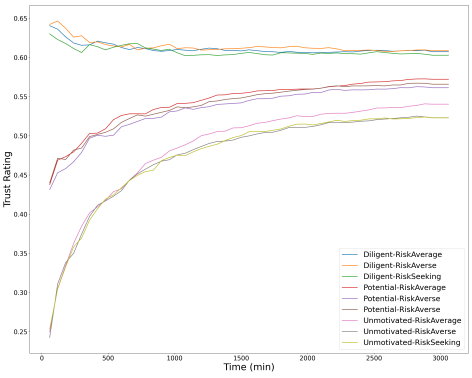
(a) The transfer case - no incentive - one hop



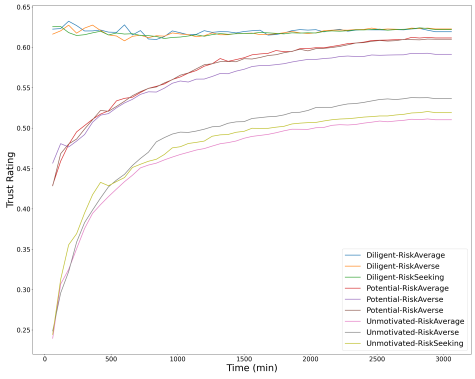
(b) The share case - no incentive - one hop



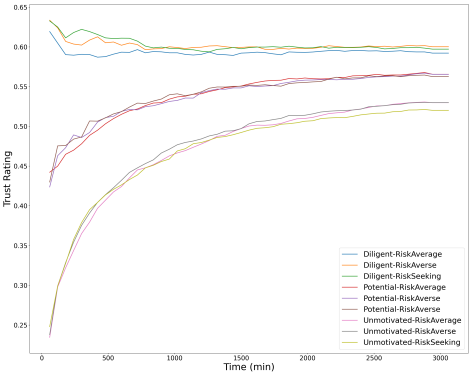
(c) The transfer case - incentive - one hop



(d) The share case - incentive - one hop

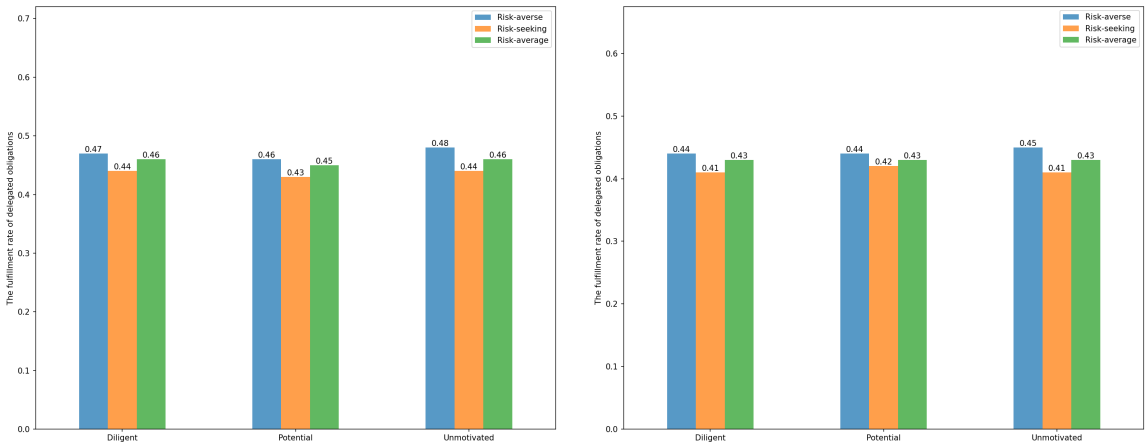


(e) The transfer case - incentive - cascaded



(f) The share case - incentive - cascaded

Figure 5: The change of agents' trust rating over 3000 time ticks



(a) The share case - incentive enabled - one hop (b) The share case - incentive enabled - cascaded

Figure 6: The fulfilment rate of delegated obligations for nine profiles of agents

analysis on the data in order to verify some intuitive features of our model.

Let us take a look at whether or not agents with different risk profiles indeed exhibits different fulfilment ratios for their obligations being delegated. Fig. 6a shows results for the share case with incentive employed for one-hop delegation. We can see that agents who are taking a risk-averse approach always achieve the highest fulfilment rate (about 46%) for their obligations being delegated, no matter how diligent they fulfil obligations themselves. That is because they always choose a delegatee among others, who is most likely to fulfil the delegated obligation. Similarly, agents with risk-seeking profiles are always making the lowest fulfilment rate (about 41%), while agents with risk-average sit in the middle achieving around 44% fulfilment rate for their delegated obligations. Fig. 6b also shows the same pattern for the fulfilment rates of delegated obligations for the cascaded case.

Secondly it would be interesting to observe whether there exist differences between the transfer and share cases. We make a comparison between these two cases against a number metrics including the fulfilment rate of delegated and non-delegated obligations, trust rating and earned credits. However, we cannot see an obvious sign that one case's performance is outweighed by the other. For example, as illustrated in Fig. 4, the fulfilment rate for all obligations in the share case is only slightly different compared to the one for the transfer case.

When making a comparison between cascaded delegation and one-hop delegation, at first glance, there is no obvious difference between them in terms of the number of obligations being fulfilled, as showed in Fig. 4. On the number of delegated obligations being fulfilled, Fig. 6a and 6b show that the one-hop delegation case performs slightly better than the cascaded delegation case. This is because agents who involved in cascaded delegations may have different profiles, causing unstable trend in fulfilling those obligations. Furthermore, when looking into the data about the fulfilment of *awarded obligations*, we observe some

interesting differences for these two types of delegation. Let us take an example of the share case in which the incentive is imposed, Fig. 7 shows that the unmotivated agents make much more awarded obligations being *violated* in the one-hop delegation case in comparison with the corresponding number of violation for the cascaded delegation case. Intuitively, the unmotivated agents, in the cascaded case, often delegate awarded obligations further down the chain to potential or diligent agents, thereby these agents have a better chance to discharge the obligations. In contrast, diligent agents, who have risk-seeking profile in particular, have *less* awarded obligations being violated in the one-hop delegation case. This is due to the fact that the cascaded case give diligent users opportunities to earn more credits by delegating obligations further to less trustworthy (potential and unmotivated) agents who may eventually violate the obligations. The same pattern is available for the potential agents as seen in Fig. 7. We can also clearly see that unmotivated agents lead to much more awarded obligations being violated than the number of potential and diligent agents.

In summary, the analysis of our experimental results reveal the following guidelines, which provide important insights when implementing our models in practice.

- One-hop delegation with incentive is the preferred option in comparison with the cascaded delegation, since the cascaded delegation adds more complexity in forming a delegation chain to fulfill a delegated obligation but it does not offer a better fulfilment rate of obligations overall.
- A system incorporating our incentive model is free to choose either the transfer of responsibility for delegation or the share of responsibility for delegation, because both cases exhibit a very similar performance in all metrics.
- It is recommended to implement all the three risk profiles for choosing delegatees. While agents with different risk profile have slight different performances in terms of fulfilling delegated obligations, that difference can be neglected in comparison to versatility that is available for agents to earn credits.

4.3 In Comparison to the ϵ-Greedy Algorithm

To the best of our knowledge, the only comparable work to what we have developed is by Afanador *et al.* [1] who considers the problem of cascaded delegation as a recursive Multi-Armed Bandit (MAB) problem. They evaluated the effectiveness of several modified MAB algorithms, including the ϵ-greedy, Thompson sampling, and the Upper Confidence Bound (UCB) algorithm, in the cascaded delegation settings. In this section, we choose to implement their ϵ-greedy algorithm in our simulated multiple-agent systems and compare its performance to ours, because the ϵ-greedy algorithm typically represents the features of the MAB approach.

The ϵ-greedy algorithm [26] is a simple method for balancing exploration and exploitation by selecting a random agent to delegate to with a probability of ϵ (exploration), while

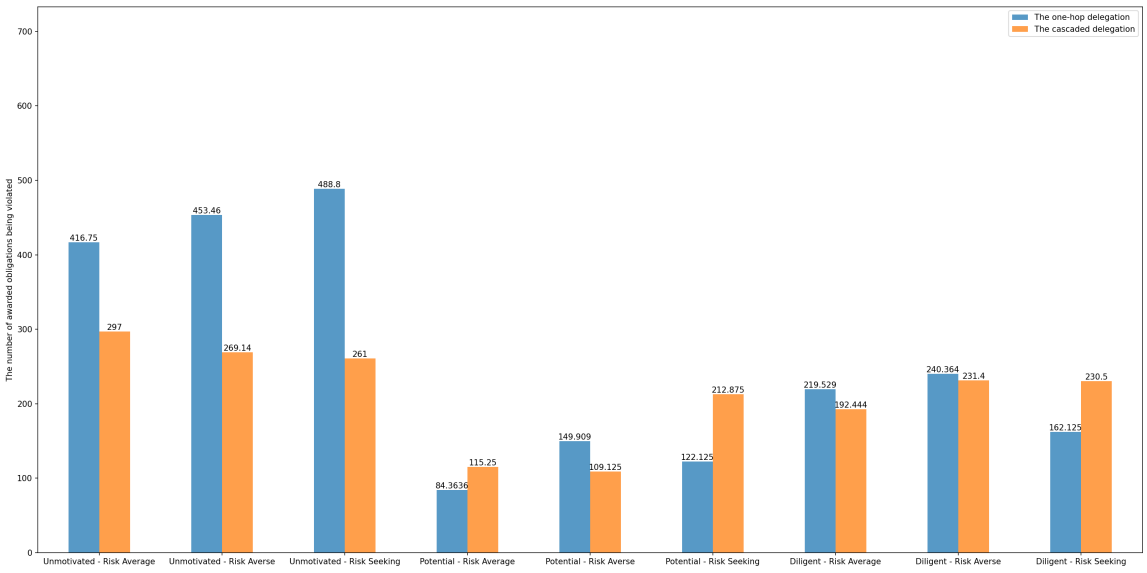
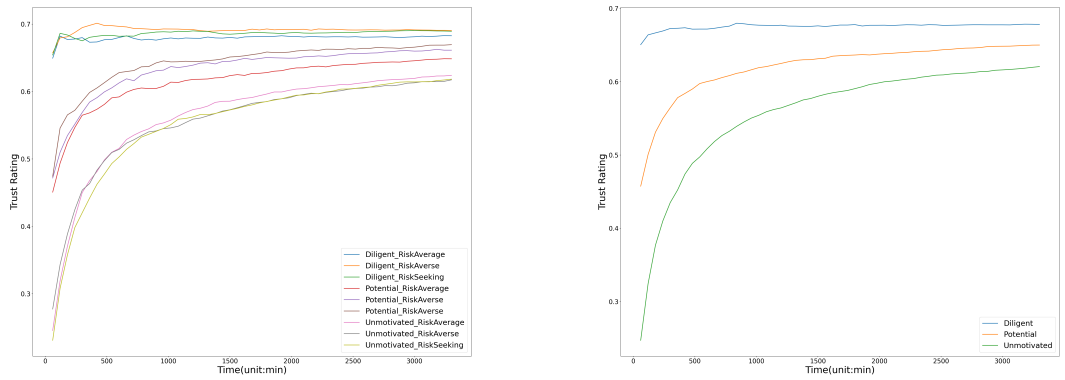


Figure 7: The number of awarded obligations being violated for the one-hop delegation and cascaded delegation

choosing the agent with the highest historical success rate with a probability of $1 - \epsilon$ (exploitation). With the algorithm, a delegating agent makes a decision on choosing which agent to delegate to on the basis of the expected *utility* of such a delegation. For example, agent A needs to delegate an obligation, and has 3 neighboring agents: B, C, and D. Each agent has a certain expected utility for fulfilling or onward delegating the obligation. Using the ϵ-greedy algorithm, agent A will take a probability of ϵ to explore by choosing one of B, C, or D at random. With probability of $1 - \epsilon$, agent A will exploit by choosing the agent with the highest estimated utility (based on previous successful fulfilment of obligations). If agent A chooses to delegate to agent B, then agent B may either execute the obligation or delegate to one of its own neighbors, again using the ϵ-greedy algorithm. After the obligation is fulfilled, the actual utility is recorded, and the estimates for the agents involved are updated, refining future delegation decisions. The algorithm uses a recursive approach to calculate the utility of every possible delegation throughout multiple agents in a network. The goal is to create an optimal chain of delegations from the delegating agent to other agent in the network.

To ensure fair and accurate comparative results, our approach and the ϵ-greedy algorithm must be implemented in the same simulated multi-agent environment. Therefore, we made several adjustments to our experimental framework. First, we arranged the 100 agents in a 10×10 grid to establish interconnections among them. In our approach, whenever a delegating agent (delegator) broadcasts an obligation request, only the directly connected agents receive the announcement, and they will see whether they are eligible to bid based on the bidding criteria. The delegator will consider only those who submitted the bid. This



(a) Trust rating - our approach

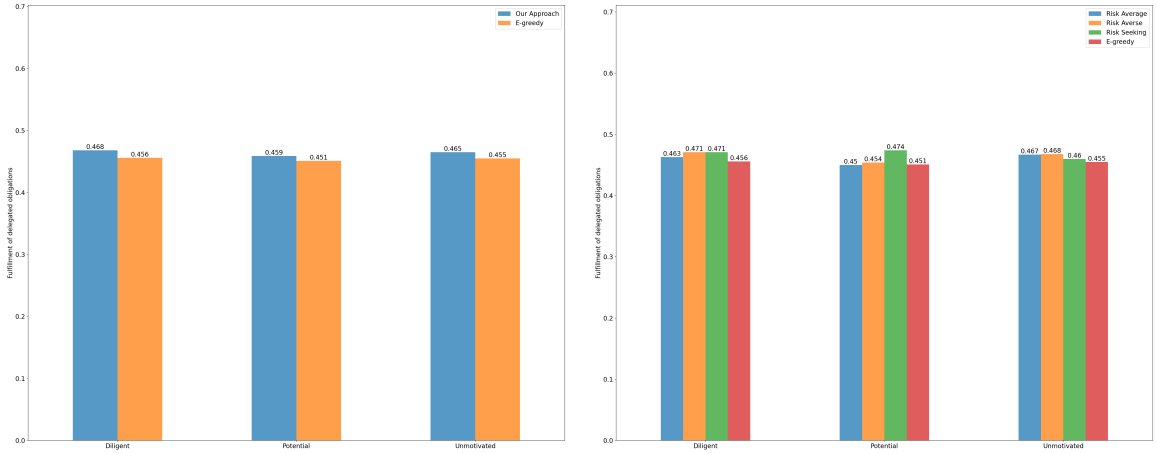
(b) Trust rating - ϵ-greedy

Figure 8: The change of agents' trust rating

adjustment is reasonable for our approach, as we can simply treat the interconnected agents as a group that is formed with similar competence in fulfilling obligations. In contrast, the ϵ-greedy algorithm requires to build such a network of connected agents as part of its input. This is, in fact, one of the limitations of the ϵ-greedy approach, as it requires complex recursive calculations to obtain an utility for each possible delegation across the network of agents. To mitigate these complex recursive calculations, we made a second adjustment by limiting the depth of the delegation chain to five. In other words, in the ϵ-greedy algorithm, once the visited chain reaches a depth of five, the node is treated as a leaf node, and its neighbouring nodes are no longer visited. For our approach, a delegation depth of five is sufficient to capture all of its design features without compromising functionality.

Note that our approach and the ϵ-greedy algorithm differ in how delegatees are chosen in the cascaded delegation setting: In our approach, a delegator makes a *local* decision on which agent to delegate an obligation to, based on the delegator's risk profile (perception), while the ϵ-greedy algorithm determines onward delegations by maximising expected utility across a *global* network of agents. This means we can retain all the experimental setup elements that are *irrelevant* to the delegation decision-making process, such as the agent performance profiles (diligent, potential, and unmotivated), trust rating, reward credits, as well as incentives (the exponential decay function $f(x)$).

When looking at the trust ratings of different types of agents in Fig. 8a and Fig. 8b, the increasing trend for diligent, potential, and unmotivated agents is consistent between our approach and the ϵ-greedy algorithm, which is due to the effect of the incentivising mechanism $f(x)$. However, there are a few subtle differences that we would like to highlight: for diligent agents, the ϵ-greedy algorithm stabilises the trust rate at 0.6799, which is lower than the results of our approach for all three risk profiles: risk-averse (0.6904), risk-seeking (0.6870), and risk-average (0.6828). For potential agents, the trust rating stabilises at 0.6635



(a) Our approach vs ϵ-greedy

(b) Our approach (risk profiles) vs ϵ-greedy

Figure 9: The fulfilment rate of cascaded delegated obligations

using the ϵ-greedy algorithm, which is lower than the risk-averse (0.6729) and risk-seeking (0.6784) profiles in our approach, but higher than the risk-average profile (0.6624). These results indicates that the ϵ-greedy algorithm operates effectively within our experimental framework.

The most anticipated result from comparing our approach with the ϵ-greedy algorithm is shown in Fig. 9a, where our approach consistently achieves a higher fulfilment rate of delegated obligations compared to the ϵ-greedy algorithm across all three performance profiles (diligent, potential and unmotivated). Fig. 9b presents additional results (fulfilment rates) for agents with different risk profiles. For diligent and unmotivated agents, the ϵ-greedy algorithm performs worse than any type of risk taker in our approach. However, for potential agents, the ϵ-greedy algorithm only slightly outperforms the risk-average profile by a margin of 0.001 but performs significantly worse than both the risk-averse and risk-seeking agents. Fig. 9b also shows that there is no clear pattern indicating whether diligent, potential, or unmotivated agents perform better than one another. This is expected, as the incentives ($f(x)$) have a significant impact on increasing agents' trust rating, and the majority of obligations delegated from unmotivated or potential agents are fulfilled by diligent or potential agents.

In summary, our experimental comparison results suggest that our approach outperforms the ϵ-greedy algorithm in terms of the fulfilment rate of cascaded delegated obligations. This demonstrates that adopting our approach leads to fewer obligation violations in the system, which is the key goal of implementing obligation delegations. We would also like to highlight one point regarding computational efficiency, as observed through our experiment: Unlike the ϵ-greedy algorithm, which involves complex recursive computations over a network structure, our approach enables a delegator to broadcast a delegation request to any set of agents deemed appropriate, and choose an appropriate delegatee based on their own

risk perception, demonstrating both efficiency and flexibility.

5 Discussion

While we have referred to the work of Afanador *et al.* [1] in our experimental study, it is worth highlighting here some related approaches which attempt to address the issues of obligation fulfilment. Also, of the significant experimental results we presented for our models, we explore their practical applications by illustrating real-world scenarios where these models can be applied.

5.1 Related Work

There has been considerable research on the frameworks for modelling and managing obligations. However, to the best of our knowledge, no previous work has studied incentive mechanisms for fulfilling obligations in the presence of delegation. In this section we are going to review the existing work that are most closely related to ours and highlight the novelty of our contributions.

Firstly, there exists a sizeable body of work on exploring interactions between authorisation and obligations. Irwin *et al.* [9, 16, 22, 23] formally study the problem of maintaining *accountability* for system states when there exist dependencies between obligations, that is, one or more obligations provide necessary privileges or resources to enable the fulfilment of other obligations. A state is said to be accountable if the only reason for obligations going unfulfilled is due to user's negligence rather than a lack of necessary authorisation or resources. In order to focus on incentive schemes for discharging obligations for others, we assume that actions in the obligations are not subject to access control, and thus they can always be fulfilled. Of course, one of interesting future work is to model and analyse the accountability problem in our framework. Also there has been some work to structure incentives that motivate users to fulfil obligations in access control systems. For example, Chen *et al.* [7] look at a number of ways of applying obligations to account for the risk incurred by granting access requests. Like ours, they proposed incentives for users to fulfil obligations but the incentives are about granting users with risky access or restricting users from future access. Baracaldo and Joshi [2] use obligations as a means to deter insider attacks. Basically, one of conditions determining whether to grant an access request is to evaluate how trustworthy of the requester in fulfilling an obligation resulting from granting the access. Similar to ours, their approach to computing a trust value for each user is based on user's historical performance of fulfilling obligations, but their computation model is more complex to account for strategic malicious users who adapt its behavioural pattern to earn a high trust value.

Another strand of work close to ours is delegation of obligations or tasks. Schaad and Moffett [24] identify the delegation of obligations as a recurring phenomenon in an organisation context and propose policy constructs in Alloy to deal with delegation operations and reviews. Specifically, when a subject delegates an obligation to another subject, the

delegating subject loses its assignment to the obligation but a new review obligation is created and assigned to her as a means to account for the delegated obligation. Unlike ours, their work does not address how to incentivise and monitor subjects to fulfil their assigned obligations. Ben-Ghorbel-Talbi *et al.* [3] use a logical method to define different kinds of responsibilities when delegation of obligations is occurred, including functional responsibility, causal responsibility, liability and sanctions. However, there is no study of complexity in terms of managing and reasoning with these responsibilities when a large number of delegation operations are granted and some of which may result in conflicting responsibilities. Norman and Reed [20] present the use of the Hamblin logic capturing a responsibility-based semantics of delegation, which provides a rich account of how responsibility is transferred, acquired and discharged during and after delegation. In particular, their theoretical model is able to capture the case of a task being delegated to a group and to analyse the consequent responsibilities of each of the parties involved in the group. This inspires us to consider how to incentivise users in a group to work in a collaborative manner to discharge a delegated group obligation. Burnett and Oren [5] evaluate a number of different weighting strategies which are used to update trust for individuals involved in a delegation chain. Unlike our work, none of their strategies tends to incentivise users to executing tasks with the exception of Chen *et al.* [8]. This work explores some simple incentive mechanisms for users to fulfil obligations for others but does not consider delegating the responsibility of fulfilling the obligations. Also their experimental work is preliminary without thorough examination of the proposed incentive mechanisms.

In short, we believe that our research is complimentary to some of the above mentioned work by providing a fair rewarding scheme that encourages users to discharge obligations for others, and conducting a series of controlled experiments with a simulated system and a rigorous statistical analysis of the results.

5.2 Practical Considerations

Delegation of obligations is commonplace and naturally arises in the real world. In many situations, an individual's workload becomes overwhelming. To avoid failure, obligations must be delegated to other team members. In the delegation process, it is essential to consider why team members would accept the delegated obligation and how the delegator selects the most suitable individual to ensure the obligation's fulfilment. Our models are designed with these factors in mind, and the results of our experiments confirm that our models effectively tackle the issue of obligation fulfilment. In the following sections, we introduce two scenarios to illustrate the practicality of our proposed models: one involving one-hop delegation and the other focusing on cascaded delegation.

5.2.1 One-Hop Delegation in Team Collaboration

Context Sarah and John are both team members on a project with no seniority between them. They are jointly responsible for ensuring that the project is completed on time and

within budget. Sarah, feeling the pressure of a heavy workload, seeks to delegate some of her responsibilities to balance tasks more evenly across the team.

Delegation Sarah and John discuss the team's workload, and Sarah proposes that John take full responsibility for conducting the weekly project status meetings. After clearly communicating the objectives and providing necessary documentation, Sarah entrusts John with complete control over this aspect of the project. From that point on, Sarah no longer needs to be involved in preparing for or running the meetings.

Obligation Once John accepts the task, he is fully accountable for leading the meetings and ensuring they are effective. Sarah no longer needs to manage or oversee this specific duty. However, both Sarah and John will share in the credit for the successful outcome of the meetings, as they are vital to the project's overall progress. The completion of this obligation is now entirely in John's hands.

Incentive for John To motivate John to fulfill the obligation with excellence, both Sarah and John agree that the success of the meetings will be recognized as a joint achievement, factoring into their overall team performance evaluations. Additionally, if John manages the meetings well, it can bolster his reputation as a strong contributor, opening the door to future leadership roles or opportunities within the company. This shared recognition acts as an incentive for John to fully engage in the task and ensures that both will benefit from the successful management of the project.

Outcome With John handling the weekly meetings independently, Sarah is free to focus on other key responsibilities. John, motivated by the shared credit and the potential for professional growth, diligently fulfills his role. The project progresses smoothly, with both team members benefiting from their complementary efforts. John's ownership of the task empowers him to demonstrate leadership, while Sarah can devote more time to high-level project concerns.

5.2.2 Cascaded Delegation in Construction Project Management

Context In a large construction project, such as building a skyscraper, multiple team members are responsible for different aspects of the project, but there is no formal hierarchy or seniority between them. The project has multiple phases, from design to construction, and involves collaborative decision-making and responsibility delegation across the team. Each person is accountable for their tasks, and once a responsibility is delegated, the original delegator is no longer involved in completing that specific obligation.

Primary Delegation In the initial stage, a team member (Alex) who is focused on the overall project timeline and budget, takes on the responsibility of ensuring the foundation is completed on schedule. However, due to other priorities, Alex decides to delegate this

responsibility to another team member, Jordan, who has expertise in foundation work. Before delegating, Alex evaluates the risk of whether Jordan will be able to complete the task successfully. Alex takes into account Jordan's track record, skill set, and workload, as Alex knows that failure to complete the task will affect both of them.

Cascaded Delegation Jordan, now fully responsible for overseeing the foundation work, realises that managing all aspects of it would be too time-consuming and decides to further delegate specific duties. Jordan asks Morgan to oversee the day-to-day on-site work for the foundation, including managing the schedule and quality of the construction teams. Jordan provides Morgan with all the necessary information and documentation and hands over full responsibility for this task. Jordan is aware that Morgan's performance directly impacts whether the foundation is completed successfully. If Morgan fails, Jordan will face significant penalties, but Morgan will receive the harshest consequences as the person who accepted the final delegation. Therefore, Jordan chooses Morgan carefully, assessing their ability to deliver under pressure.

Further Delegation Morgan takes on the responsibility for the day-to-day oversight but realizes that the project is too large to handle alone. Morgan decides to delegate specific tasks related to different parts of the foundation work, such as concrete pouring, rebar installation, and excavation, to Casey. Casey is now responsible for managing these specific tasks, ensuring they are completed correctly and on time. Morgan knows that the successful execution of these tasks is critical for completing the foundation. If Casey fails, Morgan will face penalties for delegating the task poorly, but the greatest penalty will fall on Casey. Morgan takes this risk into account before making the delegation, considering Casey's attention to detail and past performance.

Accountability and Penalties If the foundation work is completed successfully, everyone in the delegation chain (Alex, Jordan, Morgan, and Casey) receives their proportion of credits. However, if the foundation is not completed on time or fails to meet the required standards:

- Casey, the last person in the delegation chain, will receive the most significant penalty, as they were responsible for directly managing the critical tasks.
- Morgan, who delegated to Casey, will also face penalties, but they are less severe because Morgan delegated responsibly based on an assessment of Casey's capability.
- Jordan, who delegated to Morgan, will face a smaller penalty, as the failure occurred further down the chain, outside their direct control.
- Alex, the original delegator, receives the least penalty because they delegated early on, and the ultimate failure occurred after multiple layers of delegation and responsibility.

Outcome In this model, each person carefully evaluates the risk before delegating tasks, knowing that once the responsibility is passed on, they are no longer required to fulfil the obligation but will still bear some responsibility if the task is not completed. The individual at the end of the chain (Casey) carries the greatest risk and responsibility but also has the most control over the task.

This approach ensures that while delegation can help distribute workload, everyone remains incentivised to choose their delegate wisely and avoid passing the responsibility to someone incapable of fulfilling the obligation.

6 Concluding Remarks

In this paper we argue that one effective means of managing obligation fulfilment is via delegation of obligations. We propose a protocol for managing the delegation process that involves with announcement, bidding and awarding steps. We further develop a model that combines trust update with a credit rewarding scheme to incentivise users to fulfil delegated obligations. We also explore how the incentive mechanism can be extended to the case where cascaded delegation of obligations occurs. We develop a multi-agent system for all the features of our model and run experiments on the system to evaluate the model's performance. In particular, we implement the incentive mechanism using an exponential decay function, which enables agents to increase their probability of fulling obligations proportionality in line with their past performance. We conduct a comprehensive analysis of our experimental results to measure agents's performances in terms of change of trust rating, fulfilment rate of obligations, etc. We also compare agents behaviour and performance in the transfer responsibility versus the share responsibility, as well as one-hop delegation versus cascaded delegation. To highlight the novelty of our models, we implement the modified ϵ-greedy algorithm from the work of Afanador *et al.* [1] within our experimental framework and compare its performance with our models. The results suggest that our models outperform the ϵ-greedy algorithm in terms of efficiency, flexibility, and the fulfillment rate of cascaded delegations. Since our experiments have confirmed the advantages of our models, we explore their practical applications by presenting two real-world scenarios.

One immediate future work is to investigate the interaction with authorisation when delegation of obligation occurs. For example, when John delegates an obligation of preparing the quarterly sales report to Charlie, Charlie may not have permission to view the sales database. Who should grant the authorisation to Charlie: the system or John? What are the security implications for doing so? We would like to take a formal approach to study the balance between authorisation and obligation fulfilment.

References

[1] Juan Afanador, Murilo S. Baptista, and Nir Oren. Algorithms for recursive delegation. *AI Communications*, 32(4):303–317, 2019.

[2] Nathalie Baracaldo and James Joshi. Beyond accountability: using obligations to reduce risk exposure and deter insider attacks. In *Proceedings of the 18th ACM Symposium on Access Control Models and Technologies*, pages 213–224, 2013.

[3] Meriam Ben-Ghorbel-Talbi, Frédéric Cuppens, Nora Cuppens-Boulahia, Daniel Le Métayer, and Guillaume Piolle. Delegation of obligations and responsibility. In *Proceedings of the 26th IFIP TC 11 International Information Security Conference*, pages 197–209, 2011.

[4] Guido Boella, Leendert W. N. van der Torre, and Harko Verhagen. Introduction to normative multiagent systems. *Computational & Mathematical Organization Theory*, 12(2-3):71–79, 2006.

[5] Chris Burnett and Nir Oren. Sub-delegation and trust. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*, pages 1359–1360, 2012.

[6] Cindy Candrian and Anne Scherer. Rise of the machines: Delegating decisions to autonomous ai. *Computers in Human Behavior*, 134:107308, 2022.

[7] Liang Chen, Jason Crampton, Martin J. Kollingbaum, and Timothy J. Norman. Obligations in risk-aware access control. In *Proceedings of the 10th Annual International Conference on Privacy, Security and Trust*, pages 145–152, 2012.

[8] Liang Chen, Cheng Zeng, and Stilianos Vidalis. An incentive mechanism for managing obligation delegation. In *Proceedings of the 17th International Conference on Risks and Security of Internet and Systems*, volume 13857, pages 191–206, 2022.

[9] Omar Chowdhury, Murillo Pontual, William H. Winsborough, Ting Yu, Keith Irwin, and Jianwei Niu. Ensuring authorization privileges for cascading user obligations. In *Proceedings of the 17th ACM Symposium on Access Control Models and Technologies*, pages 33–44, 2012.

[10] Rachael Colley, Umberto Grandi, and Arianna Novaro. Unravelling multi-agent ranked delegations. *Autonomous Agents and Multi-Agent Systems*, 36(1):9, 2022.

[11] Natalia Criado, Estefania Argente, and Vicente J. Botti. Open issues for normative multi-agent systems. *AI Communication*, 24(3):233–264, 2011.

[12] Davide Dell'Anna, Mehdi Dastani, and Fabiano Dalpiaz. Runtime revision of sanctions in normative multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 34(2):43, 2020.

[13] Daniel J. Dougherty, Kathi Fisler, and Shriram Krishnamurthi. Obligations and their interaction with programs. In *Proceedings of the 12th European Symposium On Research In Computer Security*, pages 375–389, 2007.

[14] Danielle Ferguson, Yan Albright, Daniel Lomsak, Tyler Hanks, Kevin Orr, and Jay Ligatti. PoCo: A language for specifying obligation-based policy compositions. In *Proceedings of the 9th International Conference on Software and Computer Applications*, page 331–338, 2020.

[15] Bengt Holmstrom and Paul Milgrom. Multitask principal-agent analyses: Incentive contracts, asset ownership, and job design. *Journal of Law, Economics, & Organization*, 7:24–52, 1991.

[16] Keith Irwin, Ting Yu, and William H. Winsborough. On the modeling and analysis of obligations. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, pages 134–143, 2006.

[17] Audun Jøsang, Ross Hayward, and Simon Pope. Trust network analysis with subjective logic. In *Proceedings of the 29th Australasian Computer Science Conference*, pages 85–94, 2006.

[18] Hussein Joumaa, Ali Hariri, Ana Petrovska, Oleksii Osliak, Theo Dimitrakos, and Bruno Crispo. Obligation management framework for usage control. In *Proceedings of the 29th ACM Symposium on Access Control Models and Technologies*, page 149–157, 2024.

[19] Ninghui Li, Haining Chen, and Elisa Bertino. On practical specification and enforcement of obligations. In *Proceedings of the Second ACM Conference on Data and Application Security and Privacy*, pages 71–82, 2012.

[20] Timothy J. Norman and Chris Reed. A logic of delegation. *Artificial Intelligence*, 174(1):51–71, 2010.

[21] OASIS. *eXtensible Access Control Markup Language (XACML) Version 3.0*, 22 January 2013. OASIS Standard (E. Rissanen, editor).

[22] Murillo Pontual, Omar Chowdhury, William H. Winsborough, Ting Yu, and Keith Irwin. Toward practical authorization-dependent user obligation systems. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, pages 180–191, 2010.

[23] Murillo Pontual, Omar Chowdhury, William H. Winsborough, Ting Yu, and Keith Irwin. On the management of user obligations. In *Proceedings of the 16th ACM Symposium on Access Control Models and Technologies*, pages 175–184. ACM, 2011.

[24] Andreas Schaad and Jonathan D. Moffett. Delegation of obligations. In *Proceedings of the 3rd International Workshop on Policies for Distributed Systems and Networks*, pages 25–35, 2002.

[25] Suho Shin, Keivan Rezaei, and Mohammadtaghi Hajiaghayi. Delegating to multiple agents. In *Proceedings of the 24th ACM Conference on Economics and Computation*, page 1081–1126, 2023.

[26] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, 2nd edition, 2018.

[27] Kevin P. Twidle, Naranker Dulay, Emil Lupu, and Morris Sloman. Ponder2: A policy system for autonomous pervasive environments. In *Proceedings of the 5th International Conference on Autonomic and Autonomous Systems*, pages 330–335, 2009.

[28] Cheng Xu and Philip W. L. Fong. The specification and compilation of obligation policies for program monitoring. In *Proceedings of the 7th ACM Symposium on Information, Compuer and Communications Security*, pages 77–78, 2012.