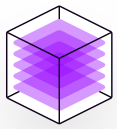


Covee Protocol

Powering the decentralized future of knowledge work with smart contracts, a cryptographic token and a unique mechanism design.

Marcel Dietsch [1] PhD (Oxford)
Jochen Krause [2] PhD (Zurich)
Heinrich H Nax [3] PhD (Oxford)
Joan Omeru [4] PhD (Imperial College London)
Raphael Schoettler [5] PhD (Humboldt University Berlin)
Sven Seuken [6] PhD (Harvard)

We thank our Senior Advisor Professor Alvin Roth (Stanford) for his valuable contributions. We also thank Professor Oliver Hart (Harvard) for his insightful feedback.

August, 2018
www.covee.network

[1] CEO & co-founder, Covee Network
[2] CTO & co-founder, Covee Network
[3] PD, ETH Zurich and Senior Economist, Covee Network
[4] Data scientist & Solidity developer, Covee Network
[5] COO & co-founder, Covee Network
[6] Associate Professor, University of Zurich and Senior Market Designer, Covee Network

Contents

Acknowledgements

We like to express our special thanks to our team for their motivation, their valuable and passioned work, and their constructive critique and suggestions during all phases of the project. Their willingness to dedicate time and resources so generously is very much appreciated.

"Coming together is a beginning. Keeping together is progress. Working together is success." –Henry Ford

1 The future of knowledge work: From corporations to decentralized, self-organizing teams

1.1 Why collaborative knowledge work is stuck in the past

Our societies and economies are becoming more decentralized and flexible as a result of rapid innovation. Information technology enables instant global communication and information sharing at near-zero cost. But why is most of the workforce stuck within the boundaries of centralized intermediaries such as the centuries-old corporation? Let us consider the case of orchestrating successful teamwork of various specialists.

Centralized intermediation and contracting in traditional organizations

We can already collaborate in a decentralized way with our friends and use tools like Skype, Slack, Github, Google Docs. Such collaboration is often based on trust. We can already also collaborate with colleagues who work at a distant office of the same organization using similar tools. Such collaboration is normally enabled by company structures and processes as well as legal agreements including employment contracts. A corporation essentially consists of a set of contracting relationships with clients, competitors, employees, financial institutions and suppliers.

Commons problems in large-scale, decentralized networks

Both trust and traditional organizations are limited in terms of scale. Open network collaboration by contrast is hugely scalable but suffers from "tragedy of the commons" problems, in particular free-riding and low contributions. On Wikipedia, empirical data shows a 90:9:1 contribution split, meaning 1% create content, 9% make occasional edits and 90% only consume information. On Reddit, the distribution appears to be more extreme: only 0.1% create content, 1.9% comment while 98% only read. For Wikipedia this is not a problem – it is a great product as it is because users can add information in small increments over time.

But what about collaboration in open networks when there is an exchange of value among participants and a target outcome that can only be achieved together? Examples include knowledge work, e.g. business analytics problems, medical research projects, complex engineering tasks or algorithms to trade in financial markets. Almost all of these use-cases have been stuck within the boundaries of traditional organizations such as firms, because they enable collaboration through intermediation and contracting, which protect exchanges of value.

Coordination function needed for interdependent collaboration

However, firms also have another function in orchestrating teamwork: coordination. The knowledge work use-cases we mentioned above often involve division of labor and complementary skills of various specialists, i.e. an interdependency between participants. Furthermore, coordination includes matching the right specialists to form teams, resource and task allocation, review processes and payments – all of this is difficult to achieve if the "value exchange" between participants is not secured.

1.2 The problem of centralization

Information technology has given us internet protocols that radically transformed the exchange of information and communication across borders. It makes information sharing permissionless and therefore, has the potential to be highly decentralized. However, in the context of collaboration

on productive work, most of these exchanges and communication still take place intra-firm. We argued that as long as open networks cannot protect the exchanges of value and perform coordination tasks, firms will remain necessary and hence collaboration mostly intra-firm.

Network effects and zero marginal cost

Moreover, there are deeper reasons why information technology has led to centralization and resulted in the near-monopolies of Amazon, Facebook, Google and others. First, network effects combined with zero marginal cost of digital information have led to concentration and market power. These allow incumbents to extract enormous economic rents. As a result, the benefits of network effects mostly go to their centralized operators instead of the network's participants.

Stateless internet protocols

Another important reason is the specification of key internet protocols, especially HTTP. The Hypertext Transfer Protocol enables communication between browsers and web servers, but it is a stateless protocol. That means is does not remember preceding events or user interactions, i.e. the state of the system. Essentially, it defines how data is delivered, but not how it is stored. That is why today's internet giants run and control databases that maintain state. Hence, the statelessness of the protocol is a major cause for the degree of centralization of the internet. The Facebook data scandal of early 2018 is one of many cases which have raised concerns that centrally owned and operated networks may have become be too powerful. Furthermore, centralized organizations and their databases represent a single point of failure as the 2017 Equifax hack showed.

Of course, centralized network operators and owners would prefer to keep their market power and continue to extract economic rents. Similarly, traditional intermediaries would prefer to continue to take a cut of the transaction value they facilitate. And as long as centralized firms are necessary to intermediate, coordinate and secure exchanges of value they will be able to extract rent.

1.3 Blockchain technology is the game changer

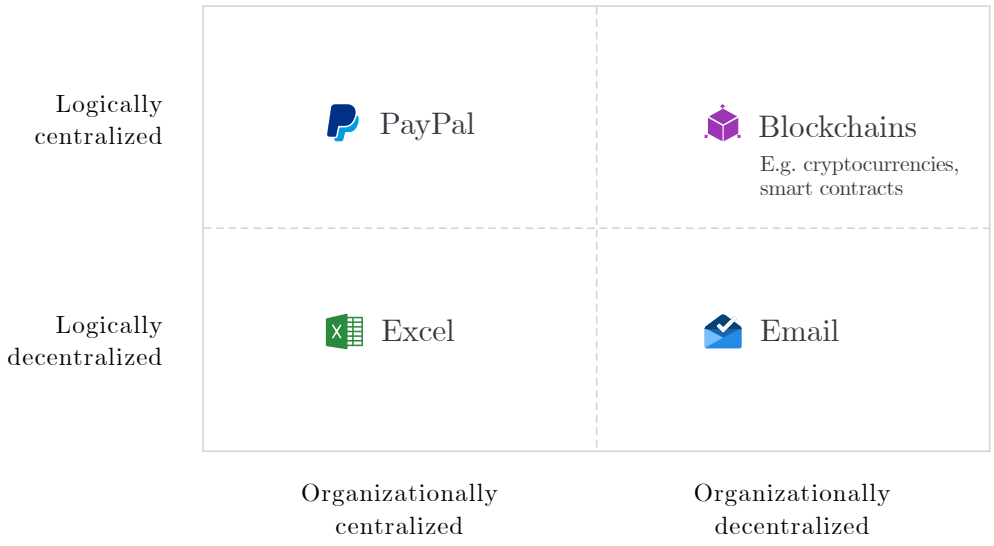


Figure 1: Blockchains create a new category of systems. Source: adapted from Albert Wenger (USV)

Blockchain is a foundational technology whose potential to change existing centralized networks and organizations cannot be overstated. It is a global, distributed, append-only, database hosted by many computers, not one central server. It is a system that, for the first time ever, provides a way of maintaining state without a single authority needing to own the database. (This fixes the statelessness problem of HTTP.) Instead the database can be owned by everyone who participates in the Blockchain's protocol.

The matrix in Figure 1 shows systems in all four quadrants that consist of code (logic) and data stores on which the logic operates. The foundational innovation is that the top right quadrant was not possible before the creation of Blockchains. There was no logically centralized system that was also organizationally decentralized. Logically centralized means there is one commonly agreed state and the system behaves like a single computer, i.e. anyone in the world gets the same answer when querying the system. Organizationally decentralized means that the system is not controlled by a single authority.

The implications of this new technology and the possibilities it creates are fundamental and far-reaching. It challenges every centralized incumbent – from digital networks all the way to traditional organizations. Blockchains have the potential to decentralize the internet's data monopolies and to change not only the boundaries of the firm but its very basis.

- **Blockchains instead of intermediaries**
  By facilitating human cooperation globally, Blockchains make the traditional intermediation function of firms redundant, creating efficiencies by removing the middleman.
- **Cryptographic tokens solve commons (and other incentive) problems**
  By incentivizing participants to create and share value globally, tokens solve important free-riding and contribution problems in open networks.
- **Smart contracts solve coordination problems**
  By replacing legal agreements and coordinating participants in global networks as effectively as firms do in their own closed contexts, smart contracts solve coordination problems at massive scale.

In combination these three innovations secure the exchange of value among distributed participants who do not know and trust each other. The internet enables the permissionless exchange of information at scale. Blockchains now enable the permissionless exchange of value at scale.

Consequently, if (1) intermediation can be made obsolete, (2) commons and incentive problems can be solved at scale, (3) coordination can take place globally and (4) exchanges of value are secured without a centralized authority, then traditional organizations such as firms will become increasingly redundant. A decentralized future in which we create and share value in global, self-organized team collaboration is now possible.

We identified knowledge work as the first and most obvious use-case for decentralized collaboration. Knowledge is location independent, intangible and can be shared at zero marginal cost. Knowledge work does require coordination among participants but in most cases not extreme levels of coordination and integration that is often needed to build sophisticated physical products.

Decentralized governance and consensus mechanisms are key factors in the success of a network for self-organizing knowledge work teams. Similar projects have proposed models that enable everyone to create their own DAOs including customized governance and incentive structures.

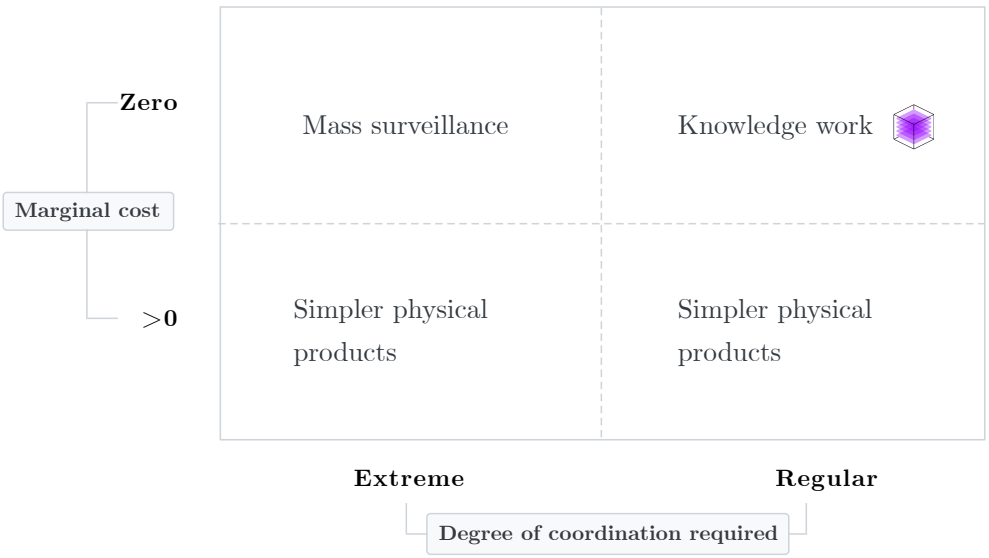


Figure 2: Knowledge work: a clear use-case for decentralization. Source: Covee Network.

Coming up with good rules for organizations and collaboration is hard. There are many decades worth of academic research on game theory, especially designing mechanisms, markets, and incentive designs. Our protocol proposes a set of fair, efficient, strategyproof and transparent mechanisms that stand on the shoulders of this research.

2 Team collaboration: a voluntary contributions game

Before specifying our mechanisms, we need to spell out the key incentive problem in collaborative team efforts, namely that *it may be in the collective interest that everyone contributes, but not in the individual interest to contribute (independently of whether the others contribute or not).*

Mechanisms are needed to solve that problem and to align collective and individual interests, hence the reason for the existence of Covee: to facilitate decentralized, efficient and sustainable teamwork amongst strangers as well as amongst friends.

We consider a team $Q = \{1, 2, \ldots, n\}$ of collaborators that gets together to complete a project. We model this as a non-cooperative game akin to the voluntary contributions game of Marwell and Ames (1979). Each game is described by requiring a total contribution target of E contributions (corresponding to full completion of the project), and by defining individual contribution targets E_i, for all $i \in Q$ (which play a role similar to that of endowments in the standard voluntary contributions game). Contribution targets are set such that $\sum_{i \in Q} E_i = E$, i.e., if every team member contributes fully, then the project will also be completed fully.

Each individual i chooses to contribute a percentage $e_i = [0, 1]$ of his total contribution target E_i resulting in an effective contribution of $x_i = e_i E_i$. We let x denote the vector of this effective contribution decision. Working together on a project has a similar payoff structure for the team members as the payoff structure of a voluntary contributions game where individual payoffs are determined by

$$\phi(x) = E_i(1 - e_i) + \alpha \sum_{j \in Q} e_j E_j,$$

where $\alpha \in (1/n, 1)$ is the marginal per capita rate of return of the game.

This game features a social dilemma because the "social optimum", i.e., the collectively most beneficial situation (where all players contribute everything, i.e., $e_i = 1$, and $\phi_i(x) = \alpha E$) is not a Nash equilibrium. In fact, the Nash equilibrium is characterized by complete non-contribution (all players contribute nothing, i.e., $e_i = 0$, and $\phi_i(x) = E_i$), which is the "social pessimum".

Example

Consider a team with three team members: a domain expert A, a data scientist B, and a machine learning expert C, and for sake of simplicity assume $E = 300$ and $E_i = 100$ for all $i \in Q$. Consider a situation with a marginal per capita rate of return of $\alpha = 2/3$. Then the focal outcomes of the game are as described in Table 1.

		Payoff			
Focal outcome	**Strategies**	**to A**	**to B**	**to C**	**Total payoff**
Social pessimum	All 0	100	100	100	300
Social optimum	All 100	200	200	200	600
One free-rider	B, C: 100 A: 0	233.33	133.33	133.33	500
One contributor	A: 100 B,C: 0	66.66	166.66	166.66	400
All contribute half	A, B, C: 50	150	150	150	450

Table 1: Outcomes in the 3-player Voluntary Contributions Game ($\alpha = 2/3$)

From the viewpoint of each individual team member, the best possible outcome is the one where all others contribute everything, and he contributes nothing, resulting in $\phi_i(x) = E_i + \alpha(E - E_i)$. Similarly, the worst outcome from his viewpoint is the one where he contributes everything, and all others contribute nothing, resulting in $\phi_i(x) = \alpha E_i$. However, from the viewpoint of the collective, the outcome where everybody contributes everything is optimal, and the outcome where everybody contributes nothing is the worst. The relative gains of the social optimum over the social pessimum depend on the rate of return α. We illustrate this, for our example, in Table 2.

We define three terms:

- *Collective gain*: The ratio of total payoffs generated in the social optimum versus the total payoffs generated in the social pessimum. It captures the collective gains of contribution over non-contribution.
- *Temptation*: The ratio of individual payoffs obtained from deviating from the social optimum versus the individual payoff in the social optimum. It captures the relative profitability to deviate from the social optimum.
- *Resilience*: The ratio of individual payoffs obtained from contributing when all others do not versus the individual payoff in the social pessimum. It captures the relative profitability to start contributing, starting from the social pessimum.

Rate of return α	Collective gain $\frac{\varphi \text{ (all contribute)}}{\varphi \text{ (no contributions)}}$	Temptation $\frac{\varphi \text{ (individual free–riding)}}{\varphi \text{ (all contribute)}}$	Resilience $\frac{\varphi \text{ (individual contribution)}}{\varphi \text{ (no contributions)}}$
1/3	1.00	1.66	1.66
1/2	1.50	1.33	1.33
2/3	2.00	1.16	1.16
1/1	3.00	1.00	1.00

Table 2: Incentive Problems in the Voluntary Contributions Game

From inspection of Table 2, it becomes clear that, all else equal, the following three assumptions holds:

- A higher rate of return α increases the collective gain of collaboration over free-riding.
- A higher rate of return α reduces the temptation to free-ride on others as one gains relatively less by doing so.
- A higher rate of return α increases the resilience of collaboration as one loses less by contributing, even if others free-ride.

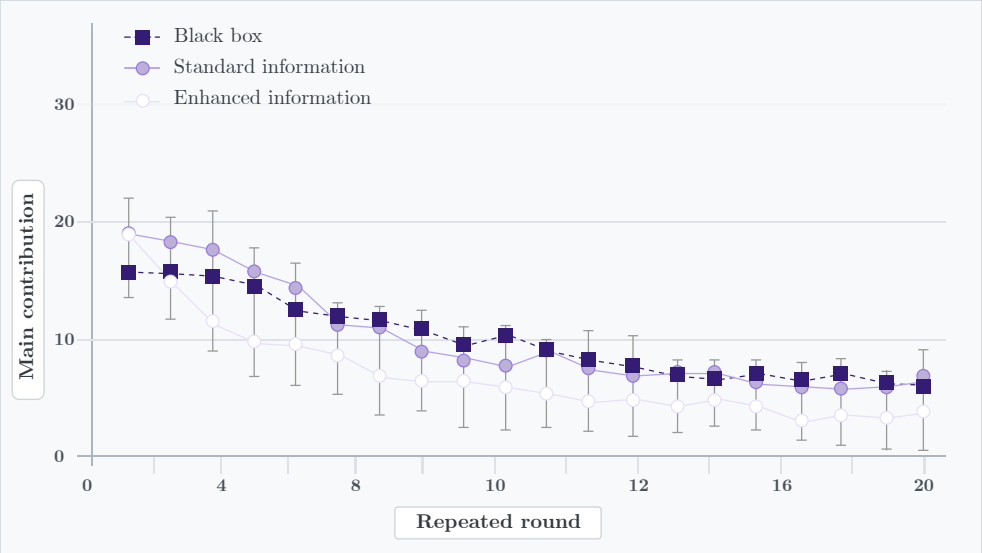


Figure 3: The decline of contributions in a repeated voluntary contributions game. Source: Burton-Chellew et al. (2015).

We know from rich experimental evidence that contributions in voluntary contributions games without a suitable mechanism in place typically decline by roughly half every ten to twenty rounds, especially if new teams are formed each period, see Figure 3.

3 Overview of our unique mechanism design

We have designed multiple mechanisms to mitigate the social dilemma inherent in the voluntary contributions game as illustrated in the previous section. Figure 4 shows a schematic illustration

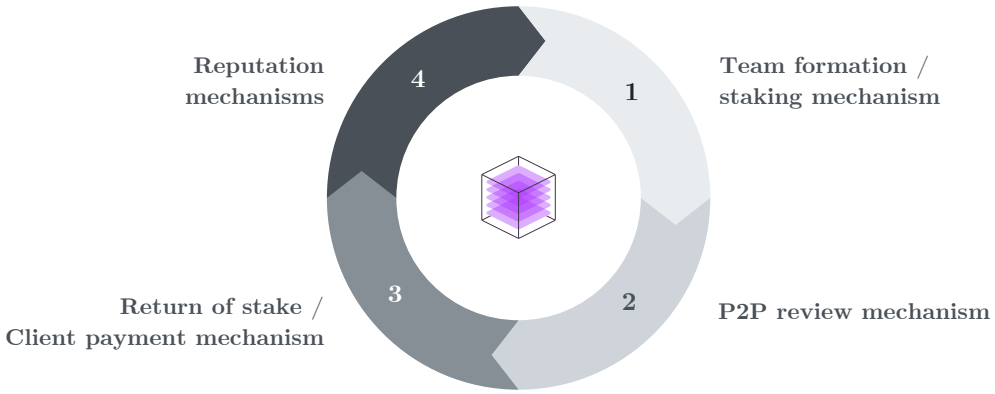


Figure 4: An illustration of the interplay of our mechanisms. Source: Covee Network (2018).

of the overall system from a mechanism design point of view.

We now provide a brief overview of all mechanisms before we describe each mechanism in detail in the following sections.

1. A team formation and staking mechanism (Section 4)
   a) The team initiator decides how many tokens a user must stake to apply to a team.
   b) Users stake tokens to apply for a particular role in a project.
   c) Team initiator reviews applications and selects one user per role.
2. A peer-to-peer (P2P) review mechanism (Section 5)
   a) The team members review each others' contributions.
   b) Based on this, the P2P review mechanism calculates a fair share of the staked tokens and the client-fee that each user will be able to claim.
   c) Those users who contribute more to the project will get a larger share.
   d) The mechanism is *strategyproof*, i.e., no individual team member can improve his payoff by lying about the relative contributions of the other team members.
3. A return of stake and client-fee payout mechanism (Section 6)
   a) At each milestone of the project:
      i. The team initiator reviews the project progress.
      ii. The team members review each other using the P2P review mechanism.
      iii. Part of the client-fee is paid out based on the outputs from (i) and (ii).
   b) At the end of the project, staked tokens are returned to the project's team members based on their P2P reviews.
4. Three complementary reputation mechanisms (Section 7)
   a) A user-rates-user based peer reputation mechanism (Section 7.1)
      i. At the end of each project, team member rate each other in a reputation mechanism.

ii. These long-lived reputation scores can be used in future projects for team initiators to select among applications, and for users to select good team initiators.

b) A client-rates-user based client reputation mechanism (Section 7.1)

i. At the end of each project, the client rates the team members in a reputation mechanism.

ii. This information is stored and displayed separately from the user-rates-user reputation scores.

iii. Future clients can use this information to identify high-performing users.

c) A user-rates-client based market reputation mechanism (Section 7.2)

i. At the end of a client project, users rate a client's performance.

ii. In the future, potential team members can see a client's reputation score before deciding whether they want to apply to a project of the client or not.

Jointly, these mechanisms promise to achieve the following. Within a project, the team formation mechanism and the P2P review mechanism together with the return of stake and client-fee payout mechanism disincentivize free-riding, because:

- Users must stake tokens to apply for and get into projects / teams.
- The P2P review mechanism asks each team member about the relative contributions of the other team members and no user can improve his payoff by lying about the relative contributions of the other team members.
- Users only get their staked tokens back if they contribute well to the project.
- Users only get paid a share of the client-fee if they contribute as expected to the project.
- In summary, a user does not benefit from free-riding. Instead, once he has joined a team, he wants to exert effort to get his tokens (back) and receive his expected share of the client-fee.

Across projects, the reputation mechanisms help clients and users identify high-performing and trustworthy users, and they help users identify trustworthy clients:

- Users need a good reputation to get into good teams/projects during team formation.
- Thus, users have an incentive to behave well within a team such that their long-term reputation is high.
- Clients need a good reputation such that good users apply to their projects.
- Thus, clients have an incentive to treat their teams well/fairly, such that their long-term reputation is high.

We now detail our mechanisms one by one.

4 Team formation and staking mechanism

Goals

"Team formation" is the process of initiating a new project and of assembling a team of users to complete a project. We aim to achieve the following high-level goals: (1) Create synergistic

teams matching the needs (skills) of the project, (2) select users who are motivated to work on the particular project and who are not free-riding on the contributions of their team members, (3) select the best user for each role in the project, and (4) select team members that can work well together.

Main idea

The main idea to achieve these goals is to use a staking mechanism to select the team members for the project. The team initiator defines how many tokens must be staked to be eligible to join the team, and users apply to individual roles by staking the required tokens. In Sections 5 and 6, we design two additional mechanisms that control how the staked tokens are returned to team members, depending on their performance during the project – this discourages users from free-riding in the project. The idea is that only users who are truly motivated and committed to the project and believe that they are most suited for a particular role stake the required amount of tokens and are thus eligible to be selected for that role. In spirit, this idea relates to entry fees in classical labor markets, e.g., see Ishikawa (2002). Amongst the applicants who stake the required amount, the team initiator selects his preferred team (e.g., by checking a user's reputation, by looking at a user's introductory video, track record, etc.), thus making sure that each team member has the required skill set and that the team members are able to work together well. The staked tokens of those users who are not chosen to join the team are returned to the corresponding users.

Staking mechanism

Formally, we use a staking mechanism to realize the team formation process. We let N denote the total set of users, indexed $i \in N$. We Let J denote the set of roles, indexed $j \in J$. To keep the presentation simple, we assume the team only needs one person per role. Furthermore, we assume that each user applies to at most one role (see the extension section for how to relax these assumptions). We assume that each user has a sufficiently high budget of tokens to participate in the mechanism (i.e., no binding budget constraint).

We now describe a basic version of the team formation process (various extensions are described in the extension section). The mechanism has three parts: (1) the project creation / definition phase, (2) the application / staking phase, and (3) the team member selection phase.

Project creation / definition phase

1. *Team initiator role*: Every project needs a *team initiator*. This is a Covee user who is responsible for creating the project, assembling the team, and managing the project.

2. The team initiator selects one of two *project types*:

    a) Client project: In a client project, a company (external to Covee) wants to use Covee to solve a problem. Accordingly, the client pays a fee to the team members for completing the project. For a client project, we assume that the team initiator is an employee of the client.

    b) Non-client project: These projects only involve Covee users, i.e., there is no external principal/client involved and no "client-fee" is paid. Thus, the team initiator is a regular Covee user, but he still has all of the team initiator responsibilities.

3. *Define roles*:

a) The team initiator defines the set of roles needed in this project (e.g., $J = \{T, D, M, C\}$, where T refers to the team initiator, D refers to a data scientist, M refers to a machine learning expert, and C refers to a computer scientist). Importantly, one of the roles must include the team initiator. Going forward, we assume that the first role (i.e., indexed by 1, or 0 if zero-based indexing is used) is always the team initiator.

b) The team initiator defines how much (percentage-wise) each role is expected to contribute to the success of the overall project (not necessarily in terms of hours). Formally, for each role j, the team initiator defines the expected contribution level $c_j \in [0, 1]$; we let c denote the vector of contribution levels. The team initiator's role is also part of this vector. The team initiator's contribution could be set to 0 (e.g., this might make sense if the team initiator only creates the project and oversees the team). For client projects, the team initiator's contribution level is *always* set equal to 0. The contribution levels of all other team members must be positive.

c) For each role, the team initiator briefly describes the tasks expected in the project. This description can but does not have to contain an expected number of hours (e.g., T: provide domain expertise and manages project; D: clean data set with $10,000$ rows and 300 columns; M: build predictive model; C: develop user front-end, e.g., 30-60 coding hours).

4. *Set deadlines, project start date, and project duration*:

   a) The team initiator defines an *application deadline* (date + time) until when users can apply to this project. Additionally, the team initiator sets a *team formation deadline*, i.e., a date + time (e.g., 2 days after the application deadline) by which the team formation process will either be finalized or by which all tokens will be returned to all applicants.

   b) The team initiator defines a *project start date* and the *expected duration of the overall project* (e.g., 2 weeks). Note that the project start date must be after the team formation deadline.

5. *Define required staking amounts*:
   The team initiator defines how many tokens a user must stake to apply for each role in the team. Formally, the team initiator defines $\bar{b^r}$ (could be zero, if nobody has to stake any tokens), which denotes the amount of tokens for a **1% contribution level** in the team. From this we can calculate the *required staking amounts for each role* as follows: $b_j^r = \bar{b^r} c_j$.

6. Information for client projects:

   a) *Client-fee*: The team initiator (client) specifies the "client-fee" F for this project (which is paid out to the users if the project is completed to the client's satisfaction). For now, we assume that the client-fee is specified and paid out in Covee tokens. During the application / staking phase, the team initiator submits the full fee F in tokens to a smart contract.

   b) *Payout contract*: The team initiator (client) specifies all details of the "client-fee payout contract". For now, we assume that some part of the fee is paid out to the team members at each project milestone. We discuss alternative contracts in the extension section.

7. All details of the project (including all deadlines, the required staking amounts, and the client-fee) become part of the project description and become publicly visible once the team initiator starts the application period.

Application / staking phase

1. *Deploy the project on Covee platform.* The following three steps happen in one step on the smart contract level:
   a) *Team initiator submits client-fee (client project only)*: The team initiator submits the fee in Covee tokens to the smart contract (which makes it publicly visible).
   b) *Team initiator stakes tokens (non-client project only)*: The team initiator stakes the number of tokens according to his contribution level, i.e., b_1^r. These tokens are submitted to a smart contract and thereby publicly visible.
   c) The team initiator opens the project to receive applications. At this point in time, the project becomes publicly visible for the first time. At that point in time, all users on the platform can see the project with all of its details and can start applying to it.
2. *Users apply to the project*: A user who wants to apply to role $j \in J$ (except for the team initiator role) stakes the number of tokens that is required for the specific role, i.e., b_j^r. These tokens are submitted to a smart contract and thus become publicly visible. Each user can apply to any role; however, a user can apply to at most one role per project (this is checked by the team initiator, outside the smart contract, e.g., via a video call). Thus, once a user has applied to one role for a particular project, no additional applications are accepted from that user for that project (can be relaxed later).

Team member selection phase

1. *At application deadline → start selection of team members*:
   a) If one or more roles have not received any applications, then declare the team formation process as FAILED and go to step 2. Else, the team initiator proceeds to review all applications.
   b) The team initiator gets access to the full list of applicants, i.e., those users that have staked the required number of tokens. For each of those users, the team initiator can review the reputation scores, introduction videos, previous projects, etc.
   c) For each role j, the team initiator selects at most one person, i.e., the user who shall join the team for role j.
   d) Once the team initiator has selected one winner for each role she can explicitly select "confirm team formation", which *locks in* the winning team members. Making this explicit selection is mandatory – otherwise, the team formation process fails.
   e) As long as the team initiator has not selected "confirm team formation", she can decide that no team shall be formed by selecting "cancel team formation". In that case, the team formation process is declared as FAILED.
2. *At team formation deadline → finalize team formation and reimburse tokens*:
   a) If the team formation process has been declared as FAILED at any point during the process or if the team initiator has not explicitly "confirmed the team formation" by the team formation deadline then the project is terminated, and all staked tokens are reimbursed (i.e., the users are notified in the user interface to claim their tokens).
   b) Else (i.e., if the team initiator has previously confirmed the team formation):

- All winners are informed that they have been selected to join the team.
- All losers (i.e., applicants who did not win a role) are informed that they have not been selected to join the team and their tokens are now reimbursed (i.e., the users are notified in the user interface to claim their tokens).

Smart contract

Program listing 1 shows a brief example of Covee's smart contract source code, which is based on the mechanisms described in this white paper. This specific function allows users to apply for a role in certain project by sending tokens to the corresponding smart contract which is in charge of managing the selection process for a single role in the project.

```
function apply(bytes32 _projectId, uint256 _roleIndex, uint256 _amount, address
    _sender)
    guardStages(_projectId)
    atStage(Stages.ProjectApplicationTime, _projectId)
public
{
    require(msg.sender == address(coveeToken));

    // require that only the initiator can claim the initiator role
    if (INITIATOR_ROLE == _roleIndex)
         require(project[_projectId].initiator == _sender);

    require(projectStakingAmounts[_projectId][_roleIndex] == _amount);
    bytes32 projectRoleId = composeBytes32Uint256Key(_projectId, _roleIndex);
    bytes32 applicantForRoleId = composeBytes32AddressKey(projectRoleId, _sender);
    require(!applicant[applicantForRoleId]);
    if (!projectRoleHasApplication[_projectId][_roleIndex]) {
       projectRoleNumberOfApplication[_projectId] = projectRoleNumberOfApplication[
           _projectId].add(1);
       projectRoleHasApplication[_projectId][_roleIndex] = true;
    }
    applicant[applicantForRoleId] = true;
}
```

Listing 1: Smart contract function that allows users to apply for a certain role in a certain project.

Example

Consider a project with three roles, A, B, C, and corresponding contribution levels of 50%, 30% and 20%, respectively. Assume that the team initiator has defined a required staking amount of "10 tokens per percentage point" for this project. The required number of tokens per role is then simply the product of 10 and the percentages in the role, as shown in Table 3.

A (50%)	B (30%)	C (20%)
500	300	200

Table 3: Required token amounts for each of the three roles in the example project.

Analysis

We now analyze the properties of the team formation process and show that it achieves the goals we have laid out at the beginning of this section.

- *Staking tokens to select motivated users and deter free-riding:*
  One of the main goals of using the staking mechanism is to provide users with an incentive not to free-ride. We guarantee this via the P2P review mechanism and the return of stake mechanism, which we design in Section 5 and 6. Informally, this works as follows. Consider two types of players. The first type of player is the "high effort" type who contributes as much to the project as is expected from him. Thus, in expectation, he expects to get as many tokens back as he staked (see Sections 5 and 6). The second type of player is the "low effort" type, who tries to free-ride, i.e., contribute as little as possible. This player will get less tokens back than he submitted (possibly none); however, he may still have a positive utility if the intrinsic payoff from the project (e.g., from gaining the experience or gaining the final product) is larger than the loss in tokens. Obviously, larger staked amounts disincentivize this behavior. Thus, by defining the required staking amounts, the team initiator can control this effect and deter free-riding.

- *The team initiator is in charge of forming a successful synergistic team:*
  The team formation process is designed with both sides of the market (users and projects/team initiators) in mind. On the one side, users can carefully look at the published projects, read the project descriptions, and apply then to those projects they find most interesting and to those roles for which they feel most qualified for. On the other side, the team initiator is in control of forming the team:
  - This starts with the careful definition of the project (e.g., specifying the roles, the contribution levels, and the milestones).
  - Importantly, this includes the definition of the required amount of tokens. Here, the team initiator must think about how many tokens he wants team members to stake to be allowed to participate in his team. The right amount depends on many factors, e.g., how valuable he expects the final product to be (e.g., a trading algorithm may more valuable than some pro bono project), how fun he thinks his project is, or high his reputation as a team leader is, etc. We expect that any experienced team initiator offering a great project is able to set higher staking amounts (and receive applications) than new team initiators and/or team initiators with a less awesome or valuable project. Thus, the "market of projects" partially determines what staking amounts team initiators can ask for. Note that a team initiator might also use a high required staking amount to signal a high quality to the users. At the beginning, the Covee platform may provide some assistance and guidance to new team initiators in finding appropriate staking amounts.
  - Finally, the team initiator is in charge of selecting a "winner" among all the applicants for a role and assembling the team. This step is important, because the team initiator must make sure that each user has the skills necessary for each role and that the team as whole is able to work well together. The team formation process is designed such that the team initiator is in full control of this step.

- *Participating in the staking mechanism is strategically simple for users:*
  For a user who is only interested in one particular project at a time, applying to a project is strategically simple. All aspects of the project are defined and publicly visible by the time the project is published on the platform. This includes all deadlines (e.g., the application and team formation deadline). Thus, at the moment in time when the user applies to a project, he knows exactly:
  - How much he has to stake if he is selected to participate in the team;
  - how much all other team members have to stake; and

– at what point in time he finds out whether he was selected to participate in the team, or not.

Effectively, the user faces a "take-it-or-leave-it-price" (in terms of the required number of tokens to be staked). The user must think about whether he is willing to invest this number of tokens for the advertised project and can then apply.
If a user is potentially interested in applying to multiple projects published on the Covee platform, then his strategic considerations become more complicated. We discuss a possible solution to this problem in the extension section.

Extensions

We now present a number of extensions to the basic team formation process described above, roughly ordered by priority.

- *Separate the roles of the team initiator and the team manager:*
  For the basic version of the team formation mechanism, we have assumed that the team initiator is also responsible for *managing* the team. However, not every team initiator may be a good team manager. Thus, a natural extension is to separate these two roles/responsibilities and thereby elevate the importance of the team manager. Concretely, the team initiator would then have the option to add a distinct "team manager" role to a project. This applies to client projects (i.e., the client might not have an employee who can manage a Covee team) as well as non-client projects. If we separate these two responsibilities, then it would also make sense to introduce a separate reputation score for "team managers".

- *Team initiator invites users to join the team:*
  In the basic version of the team formation mechanism, users must apply to the project and the team initiator then selects the team members from the applicants. A natural extension is for the team initiator to contact users on the platform directly (e.g., because she already knows those users or because she has found high-quality users through the platform who have not yet applied to the project) to ask them to take on a particular role in the team. Several variations of this extension are conceivable:

  1. Before the application deadline, regular application: the team initiator contacts other users on the platform before the application deadline and invites them to apply to the project, but the application and team formation process proceed as before, i.e., the invited users are treated like all other users.

  2. Before the application deadline, invite-only: As in variant (a), but this time, the project is only visible to users who are invited by the team initiator. Thus, it is an invite-only project.

  3. Before the application deadline, special application: the team initiator invites a user for a particular role. If the user accepts the invitation before the application deadline (and stakes the corresponding amount of tokens) then the project does not accept any more applications for that role. Furthermore, for any applications already received for that role the corresponding token amounts are reimbursed.

  4. After the application deadline has passed: the team initiator invites a user for a particular role, even after the application deadline has passed. If the user accepts the invitation and stakes the corresponding amount of tokens (before the team formation deadline has passed), then only this user can become a winner for this role.

  5. Project with pre-selected users: The team initiator can start a project with only pre-selected/invited users for each role.

- *Users applying to multiple roles:*
  For the basic version of the team formation mechanism, we require that a user can apply to at most one role per project. A natural extension is to enable users to apply to multiple roles per project (e.g., because they are qualified for multiple roles but do not know what the competition is like, or because they even want to perform multiple roles in the team). Different variations of this extension are conceivable:

  1. User can be selected for at most one role: While the user can apply to multiple roles, the team selection process assures that each user is only selected as winner for at most one role per project.

  2. User can be selected for multiple roles: In addition to applying to multiple roles, the user can also indicate whether he would be willing to perform multiple roles in a team (and if so, which combinations of roles). The team selection process (selecting winners) must then assure that each user's constraints regarding "willingness to perform multiple roles" are obeyed. In this case, we would have to make sure that we do not lose the minimum number of distinct team members (i.e., 3) to properly execute the mechanisms in section 5 and 6).

- *Coordinating applications across projects (two-sided matching):*
  The basic version of the team formation mechanism is focused on one particular project, i.e., it only describes the team formation and staking process for one project at a time. Initially, with a small number of projects on the platform, we expect this process to work well. However, once the Covee platform becomes large, many projects may simultaneously be "competing" for users, and users may be interested in applying for multiple different projects at the same time. The current design would make this challenging because (a) users would have to stake tokens for multiple projects in parallel and (b) a user might want to apply to project A before knowing if he is selected for project B and C, and he might not be able to work on all three projects in parallel. To address this challenge, we propose to extend the current design by employing a two-sided matching mechanism. Such clearinghouse mechanisms have successfully been used in practice, for example, to match thousands of medical doctors with positions at hospitals or to match thousands of students with places at public high schools Roth and Peranson (1999). For our domain, such a two-sided matching mechanism would require the following elements:

  1. The application deadlines for projects would need to be coordinated. For example, there could be an application deadline once per day, or once per week.

  2. A user could apply to multiple projects with the same application deadline, indicating his preferences over projects by submitting a "preference order" (i.e., a ranking of his most-preferred projects). The user would only have to stake the maximum amount of tokens required by any of the projects he is applying to.

  3. On the project side, during the team member selection phase, each team initiator would not select "one winner per role", but would submit a "ranking over applicants per role".

  4. Given this input, we could then run a variant of the Deferred-Acceptance Mechanism (see Abdulkadiroglu and Sonmez, 2003) or some other two-sided matching algorithm to compute a matching of users to projects. We leave the detailed specification of this mechanism/algorithm and its analysis to future work.

- *Multiple users per role:*
  For now, we have assumed that the project needs one user per role. This could be extended

to multiple users per role (e.g., for larger projects). This design question dovetails with one of the previous questions regarding how to enable users that are interested in applying to more than one role per project.

- *Different (relative) staking amounts for different roles:*
  For now, we have assumed that the team initiator defines a required staking amount "per percent of contribution to the project" which is the same for all roles. In particular, two roles with the same contribution percentages require the exact same number of staked tokens. While this may seem fair at first sight, this ignores the fact that some users may be in higher demand (or lower supply) than others on the Covee platform. For example, it may be difficult to find highly qualified machine learning experts. A possible extension thus involves letting the team initiator define different required staking amounts for different roles. The "relative prices" could then express the relative demand and supply of certain skills on the platform. Note that this extension would also require a similar extension in the client-fee payout mechanism. In particular, the two extensions would have to be carefully coordinated with each other to make sure that the goal of balancing supply and demand of skills on the platform is indeed achieved. We leave the detailed design of this extension to future work.

- *Pre-filtering of users:*
  In the basic version of the team formation process, all users can apply to a project as long as they are willing to stake the required number of tokens. To reduce the number of unqualified applications, a natural extension is to let the team initiator define certain criteria (which can be automatically checked) that the users must satisfy to be eligible to apply to the project. For example, this can include (a) a minimum reputation level (e.g., 9/10), (b) a minimum number of prior projects, (c) the existence of an intro video, (d) certain hashtags and keywords), etc.

- *Alternative client-fee payout contracts:*
  For client projects, we have assumed that some part of the client-fee (depending on the team initiator/client review rating) is paid out to the team members at each milestone. This "default contract type" may be attractive for may data science projects because the team members (who may rely on this income) do not have to wait until the end of the project before they get their first payment. However, other contract types are also possible. In general, the client/team initiator determines the contract type and publishes this as part of the project description. Alternatives include:

  1. The whole fee is only paid at the end of the project (if a minimum number of success criteria has been reached).

  2. Some part of the fee (e.g., 50%) is paid throughout the project (at each milestone), and the rest of the fee (e.g., 50%) is paid at the very end.

- *Client-fee in a stablecoin:*
  For now, we have assumed that the client-fee is also specified and paid out in Covee tokens. This is desirable (in contrast to using USD, for example) because this easily allows the submission of the client-fee to the smart contract. However, the unknown volatility of the price of Covee tokens may present a potential drawback (i.e., users may prefer a more stable currency). Thus, an alternative solution might be to use a stablecoin.

- *Enable more user coordination during team formation process:*
  For now, we have put the team initiator in charge of assembling the team. However, the individual users do not know who else will be on the team when they apply, even

though they may care with whom they are going to collaborate. A possible extension is to allow users to "back out" of a team if they do not like the team that has been formed. Alternatively, we may design a more coordinated team formation mechanism where users first signal interest, make tentative commitments, etc.

5 Peer-to-peer review mechanism

In this section, we introduce the peer-to-peer (P2P) review mechanism, which is a key building block in our overall design. All team members working on a project together review each others' work at pre-defined moments (milestones) during the project, indicating how much each other team member has contributed to the progress of the project. The P2P review mechanism takes as input these reviews and based on this, calculates how to split some asset (e.g., the total amount of staked tokens, or the client-fee) among the team members. In the next section, we use the P2P review mechanism as a subroutine in the "return of stake and client-fee payout mechanism."

Goals

Our goal is to design a P2P review mechanism that has the following four properties: Specifically, it shall be:

1. **objective**, i.e., a team member's report about his own contribution does not affect how much of the asset another team member obtains
2. **strategyproof**, i.e., no team member can increase how much of the asset he obtains by lying about the relative contributions of the other team members
3. **consensual**, i.e., if there is a division of the asset that is consistent with all team members' (subjective) reviews, then this division is chosen
4. **exact**, i.e., the whole asset is paid out to the team members.

Main idea

Our mechanism is based on the work by de Clippel et al. (2008) who performed a detailed game-theoretic analysis of this problem (i.e., how to split up an asset among a group of people based on subjective reviews). The main idea of the mechanism is to only let each user provide a review of the relative contributions of all other team members, excluding himself. This immediately makes the mechanism *objective.* To also obtain the other three properties, some more work is necessary.

Mechanism (P2P reviews)

We now describe the P2P review mechanism in detail. Let Q denote the set of team members (including the team initiator) working on the specific project under consideration, with $|Q| = n$. In a client project, the team initiator is excluded from the P2P review mechanism; thus, we define the set $Q' = Q \backslash \{T\}$. In a non-client project, the team initiator participates in the P2P review mechanism, and thus $Q' = Q$. We let $|Q'| = n'$ in both cases to unify the notation going forward. We assume that there are at least four team members reviewing each other (i.e., $n' \geqslant 4$); see the extension section where we discuss a solution for $n' = 3$. The P2P review mechanism can be called as a subroutine at any milestone m. The mechanism proceeds as follows:

1. Each user $i \in Q'$ is asked to provide a rating score $s_{ij}^m \in [0,1]$ for each other team member $j \in Q' \backslash \{i\}$ at the current milestone m. The score s_{ij}^m should indicate i's view of the relative "share" of j's contribution to the project's progress between the previous milestone and the current milestone m, not taking i's own contribution into account. The scores must add up to 100%, i.e., $\sum_{j \in Q' \backslash \{i\}} s_{ij}^m = 1$. If a user i does not provide a score for any user j by the P2P review deadline of the current milestone, then we use the user-milestone contribution levels C_{jm} for all users j as specified during the project refinement phase to compute a "default" input. Specifically, we then set $s_{ij}^m = C_{jm}/(1 - C_{im})$ for all $j \in Q' \backslash \{i\}$. The individual scores s_{ij}^m are never revealed to anyone, including to any of the other team members (initially, this means that the individual scores are stored and processed off-chain). For now, we require that all scores are larger than 0, i.e., $s_{ij}^m > 0$. In the extension section we discuss how to allow for scores equal to 0.

2. Based on the individual contribution scores s_{ij}^m we calculate the *relative contribution ratios* from user i's perspective as follows: $S_{jk}^{im} = s_{ij}^m / s_{ik}^m$, i.e., this is i's view of the contribution of j relative to k for milestone m. An example of a peer-to-peer review scoring matrix is given in Table 4.

3. Given $|Q'|$ many team members, we have $|Q'| - 2$ relative contribution ratios for each pair (j,k) (which does not contain any report from j or k). We can thus define the *average relative contribution ratio* between j and k as follows:

$$S_{jk}^m = \frac{1}{|Q'| - 2} \sum_{i \in Q' \backslash \{j,k\}} S_{jk}^{im}$$

4. Next, we define the *average relative contribution ratios without the input of a specific agent* i (needed in the next step). For this purpose, we use notation $-i$ to denote that i's reports are excluded from the calculation. The resulting ratios are:

$$S_{jk}^m[-i] = \frac{1}{|Q'| - 3} \sum_{\ell \in Q' \backslash \{j,k,i\}} S_{jk}^{\ell m}$$

5. We now define an auxiliary function which computes a share for each user i when user j's reports are excluded from the calculation:

$$g_i^m[-j] = \left(1 + S_{ji}^m + \sum_{k \in Q' \backslash \{i,j\}} S_{ki}^m[-j]\right)^{-1}$$

6. We compute the relative contribution f_i^m that user i obtains at milestone m:

$$f_i^m = \frac{1}{|Q'|}\left(1 - \sum_{j \in Q' \backslash \{i\}} g_j^m[-i]\right) + \frac{1}{|Q'|} \sum_{j \in Q' \backslash \{i\}} g_i^m[-j]$$

7. We reveal the final relative contributions to team members:
   - For non-client projects (where $|Q'| = |Q|$), we let $f^m = (f_1^m, f_2^m, \ldots, f_{|Q|}^m)$ denote the final vector of relative contributions.
   - For client projects, $Q' = Q \backslash \{T\}$ applies, we shift all of the users' indices one to the right, i.e., $\hat{f}_{i+1}^m = f_i^m$. We set the relative contribution of the team initiator equal to zero, i.e., $\hat{f}_1^m = 0$, and let $f^m = (\hat{f}_1^m, \hat{f}_2^m, \ldots, \hat{f}_{|Q|}^m)$ denote the final vector of relative contributions.

Analysis

We now analyze the properties of the P2P review mechanism and show that it achieves the goals we have laid out, i.e., that it has the four key properties we have defined at the beginning of the section. The mechanism we have designed is an instantiation of the class of mechanisms proposed by de Clippel et al. (2008). They have proven that, for teams with four or more members (see our extension section for a discussion of teams with two or three members), all mechanisms in this class are (1) *objective*, (2) *strategyproof*, (3) *consensual*, and (4) *exact*. Given this, our mechanism is also guaranteed to have all four of these properties. We will now provide some intuition (proof sketches) for this.

1. Recall that, for the mechanism to be **objective**, we require that a team member's report about his own contribution does not affect how much of the asset another team member obtains. Note that our mechanism only asks each team member i to judge the *relative* contributions of all *other* team members, and does not even ask i about his own contributions. Thus, our mechanism is objective by construction.

2. Recall that, for the mechanism to be **strategyproof**, we require that no team member can increase how much of the asset he obtains by lying about the relative contributions of the other team members. More formally, strategyproofness requires that it is a weakly dominant strategy for each individual team member to truthfully report their subjective beliefs about the other team members' relative contributions. To see why our mechanism satisfies this, consider Step 6 of the mechanism, where we compute f_i^m, i.e., the relative contribution (i.e., the fraction) of user i. Inside this formula, we use two functions. The first function $g_j^m[-i]$ is defined in Step 5 of the mechanism and computes a share for user j while explicitly excluding all of user i's reports. The second function we use in Step 6 is $g_i^m[-j]$ which is the share of user i, while excluding j's reports. Looking into Step 5 again, we see that, to compute this function, we use the terms S_{ji}^m (which do not contain a report from user i), and the sum $\sum_{k \in Q' \setminus \{i,j\}} S_{ki}^m[-j]$ (which also does not contain a report from user i). To summarize, when we compute user i's share in Step 6 of the mechanism, the function f_i^m does not make use of any reports by user i. Thus, user i cannot affect his fraction f_i^m. In other words, by lying about his beliefs regarding the other team members' contributions, user i can neither increase (nor decrease) how much of the asset he obtains. Concretely, he obtains the same share of the asset, independent of what he reports about the relative contributions of the other team members. This implies that it is a *weakly dominant strategy* for the user to *truthfully* report his subjective beliefs about the other team members' relative contributions.

3. For the mechanism to be **consensual** requires that, if there is a division of the asset that is consistent with all team members' (subjective) reviews, then this division is chosen. In de Clippel et al. (2008), the authors have shown that for teams with three team members, there is a unique mechanism that is objective, strategyproof and consensual. Our mechanism is a natural generalization of the 3-person mechanism: when computing S_{jk}^m, we simply take the average of all $|Q'| - 2$ opinions of the ratio of the relative contributions of j and k. This guarantees that our mechanism is also consensual.

4. For the mechanism to be **exact** requires that the whole asset is paid out to the team members. To obtain this property, we follow de Clippel et al. (2008): our mechanism effectively splits the asset into n equally-sized pieces, runs a non-exact mechanism to divide up piece i among all team members while ignoring i's reports, and gives the remainder (due to non-exactness) from piece i to agent i. This guarantees that the whole asset is distributed among the team members, while strategyproofness is maintained. For a

detailed argument regarding how exactness is obtained, please see the formal proof in de Clippel et al. (2008). Furthermore, Tideman and Plassmann (2008) provide a more accessible exposition of the same mechanism.

	Score (A)	Score (B)	Score (C)	Score (D)
A	*	20/90	30/90	40/90
B	10/80	*	30/80	40/80
C	10/70	20/70	*	40/70
D	10/60	20/60	30/60	*

Table 4: Example P2P review scoring matrix with entries s_{ij}^m for a scenario where all team members agree that the relative contributions to the last milestone are $A = 10\%$, $B = 20\%$, $C = 30\%$, $D = 40\%$.

Extensions

- *Three-person Teams:*
  In the definition of our P2P review mechanism above, we have assumed that all teams have at least four or more team members that participate in the P2P review mechanism (in a client project, the team initiator does not participate in the P2P review mechanism; thus, formally we then need at least five members). For teams where only three team members participate in the P2P review mechanism, we need a slightly different mechanism. We adopt an idea from Tideman and Plassmann (2008) to design an exact mechanism for this case:

  1. We define the following auxiliary function:

  $$g_i^m = \left(1 + \sum_{k \in Q' \setminus \{i\}} S_{ki}^m\right)^{-1}$$

  2. We can now compute the relative contributions f_i^m that each user i obtains:

  $$f_i^m = g_i^m \left(\sum_{j \in Q} g_j^m\right)^{-1}$$

  3. For groups of size three, this mechanism is (1) objective, (2) exact, and (3) consensual, but not strategyproof. However, the opportunity for any strategic manipulation is generally small, such that we do not expect this to be a problem in practice (see also the discussion by Tideman and Plassmann, 2008). Furthermore, de Clippel et al. (2008) have shown that for groups of size three, no mechanism exists that satisfies all four properties. Thus, this is probably the best we can achieve for this setting.

- *Zero scores in the P2P review mechanism:*
  In the description of the P2P review mechanism, to simplify the presentation, we have assumed that all scores are larger than 0, i.e., $s_{ij}^m > 0$. However, we can relax this assumption. We only need to take special care in the definition of the contribution ratios

$S^i_{jk} = s_{ij}/s_{ik}$, such that no division by zero occurs. This can be done by explicitly setting these ratios either to infinity or to 0, depending on the inputs, and then handling this appropriately throughout the mechanism. For the details, see de Clippel et al. (2008).

- *Let users only review some of their team members:*
  In the basic version of the P2P review mechanism, we ask the users to review all of their team members. If they do not provide a score for one of them by the P2P review deadline, then we use the default scores instead. However, some users might not be able to evaluate the work of some of their team members (e.g., because they do not have the necessary skills, or because they were not very involved in this part of the project for the last milestone). A natural extension is to modify the P2P review mechanism is such a way that a user can also only review *some* of his team members. The idea is to only elicit information from the users that is truly informative. Note that such a change would have an impact on incentives. If a user only reviews one or two other users, then the mechanism would not be strategyproof anymore, even if the team has four or more users in total. This would have to be addressed.

- *Weight reviews by deviation from initial contribution levels:*
  The basic version of the P2P review mechanism puts the same weight on all team members' reviews when computing the final relative contributions (i.e., everybody gets "one vote" for each other team member). A possible extension would be to consider how much the ratings have deviated from the initial contribution levels (as specified for this milestone). The idea is that a larger deviation is a stronger signal (either that a team member contributed a lot or very little) compared to a rating that is equivalent to the initial contribution levels. However, while this extension is plausible, it is not straightforward, because multiple incentive problems might arise.

6 Return of stake and client-fee payout mechanism

Goals

Once a team has formed, we want to create a good work environment and encourage the team members to collaborate. Our primary goal is to design a system such that free-riding on the contributions of others is disincentivized; instead, we want users to trust each other that everyone will contribute their share to the project. Additionally, our goal is to make sure the client's project is completed with high quality, such that the client has a good experience. Lastly, our goal is to make sure that the client behaves professionally towards the users, i.e., that the users get paid the client-fee if they perform good work.

Main idea

To achieve these goals, we design a system that relies heavily on milestones (managed via the Covee platform) and on the P2P review mechanism described in the previous section. At each milestone, two review processes are kicked off: first, the team initiator reviews the team's progress towards the project, and second, the team members are asked to review each others' contributions. We use the team initiator's review to decide what percentage of the client-fee is paid out to the team at each milestone, and we use the outcome calculated by the P2P review mechanism to decide how to split up the client-fee as well as the staked tokens among the team members. Thus, those team members who contribute most towards the progress of the project obtain the largest share of the staked tokens and of the client-fee.

Mechanism

We now describe the mechanism that governs the return of staked tokens and the client-fee payout (in a client project) in detail. This mechanism is only run if the team formation process was successful (i.e., a team has been assembled and the project can now start). Formally, we let Q denote the set of team members (including the team initiator) working on this specific project with $|Q| = n$. We assume that the team has at least three team members (see the extension section for a discussion on two-person teams). Recall that, during the project creation phase, the team initiator defined the expected contribution levels (for the whole project) $c_i \in [0, 1]$ for each user $i \in Q$. Recall that we require that in a client project, the overall contribution level of the team initiator is set to 0, i.e., $c_0 = 0$.

We now define all elements of the mechanism: (1) the project refinement phase, (2) processes that run at every milestone, and (3) processes that run at the end of the project.

- *Project Refinement:*
  At the *project start date* (as defined in the project description) the team initiator holds a kick-off meeting together with all team members where he discusses the project with all team members, including the sequencing of the work (i.e., when each team member is needed). Based on this discussion, the team initiator then defines a set of M milestones for the whole team, indexed $m \in M$. Those milestones should be spaced out approximately evenly in terms of time (e.g., once a week) and in terms of workload. For each milestone m, the team initiator specifies:
  - *Deadlines*:
    1. A "milestone deadline", by which the milestone has to be achieved.
    2. A "team initiator review deadline", by which the team initiator has to review the project progress and submit a review rating for the milestone, e.g., 3 business days after the milestone deadline.
    3. A "P2P review deadline", i.e., a deadline by which all team members are expected to submit their P2P reviews for all other team members (e.g., 3 business days after the team initiator review deadline).
  - A "milestone contribution level" $C^m \in [0, 1]$, i.e., how much (percentage-wise) this milestone contributes to the overall project. The sum over the contribution percentages of all milestones must add up to 100% (enforce via UI).
  - For each team member $i \in Q$:
    * What exactly needs to be achieved by this team member by milestone (goals, deliverables, etc.; this should be as measurable / objective as possible).
    * A "user-milestone contribution level" $C_{im} \in [0, 1]$, showing how much (percentage-wise) this user is expected to contribute to the success of milestone m, giving rise to the user-milestone contribution matrix C, e.g., see Table 5. We require:
      1. $\sum_i C_{im} = 1$, i.e., the sum over all users' user-milestone contribution levels must add up to 100% (enforce via UI).
      2. $\sum_m C_{im} C^m = c_i$, i.e., across the whole duration of the project, the milestone-weighted contribution levels of each user add up to the contribution levels specified for the user at the project definition stage.

All of this detailed project definition information is saved on the Covee platform in such a way that all team members can see all the information about all other team members. We foresee the Covee project management / workflow product to play a major role in entering this information (specifying milestones, etc.), managing the workflow, and providing access to this information.

- *Milestone procedure:*
  At the end of every milestone, when the milestone deadline has been reached, the team initiator carries out the following tasks.
  - The team initiator reviews the team's progress towards the overall project:
    * Between the milestone deadline and the team initiator review deadline, a review rating $u^m \in [0, 1]$ is submitted describing how close to completion the project is (0% to 100%) at the current milestone m. We require monotonicity in ratings, i.e., $u^m \geqslant u^{m-1}$. Ideally, the team initiator's rating matches the target that was specified during the project definition phase, i.e.,

      $$u^m = \sum_{x=1,\ldots,m} C^x \tag{1}$$

      but it can be lower if certain milestone goals have not been met. We require that the team initiator's rating cannot exceed this target level. Both requirements can be enforced via the UI (i.e., off-chain). If the team initiator does not submit a rating by the team initiator review deadline, then we set it equal to the target level as shown in (1).

    * Note that the project team can "catch up" from one milestone to another, i.e., even if u^m was below target, u^{m+1} can be on target again.

    * We also allow the team initiator to provide textual feedback to the team (e.g., "model fit to data is sub-optimal" or "visualization of results are beautiful"). When the rating is below the target level then the team initiator is required to provide textual feedback, explaining why his rating is so low.

    * The team initiator's review rating and his textual feedback is shared with the project team members.

  - P2P review, user feedback, and transmission to smart contract:
    1. *Collect P2P reviews*: Between the team initiator's review deadline and the P2P review deadline, the users must submit their reviews (via the Covee platform) about all other team members' contributions to the current milestone m. In a client project, the team initiator does not provide input to the P2P review mechanism; in a non-client project, the team initiator also provides input.

    2. *Run the P2P review mechanism*: When the P2P review deadline has passed, we run the P2P review mechanism on the Covee platform, as described in Section 5, to process those reviews. If a user does not submit his reviews by this deadline, the P2P review mechanism uses a "default review" based on the "user milestone contribution levels" as specified during the project refinement phase. The result of the mechanism is each user's relative contribution f_i for all $i \in Q$.

    3. *Provide feedback to users and commit relative contributions to smart contract*: We inform all users via the Covee platform about the vector of relative contributions (shares) for this milestone as computed by the fair share algorithm, i.e.: $f^m =$

$(f_1^m, f_2^m, \ldots, f_{|Q|}^m)$. Additionally, this vector is then also committed to the smart contract (and thereby becomes publicly visible). Note: In a client project, the team initiator's share f_1^m is always equal to 0, which is guaranteed by the P2P review mechanism. This guarantees that, in a client project, the team initiator (who is likely an employee of the client) is not receiving a share of the client-fee or any staked tokens.

- Partial payout of client-fee to all users $i \in Q$ (minus commission fee):
  1. Let d denote the *Covee network fee* (a commission) charged by Covee on this project (in terms of a percentage on the staked tokens).
  2. From the previously committed client-fee F, subtract the commission, i.e., pay $d \cdot F \cdot (u^m - u^{m-1})$ tokens to Covee.
  3. Pay user i the following: $(1-d) \cdot F \cdot (u^m - u^{m-1}) \cdot f_i^m$
     a) Here, $(u^m - u^{m-1})$ denotes the client's view regarding how much progress the team has made towards the project's completion between the last two milestones. We initialize this as follows: $u^0 = 0$.
     b) Correspondingly, f_i^m denotes the team's view (as computed by the P2P review mechanism), how much i has contributed (relatively) to this progress.

- *Project end:*
  When the last milestone of the project has been reached, the project ends with another run of the milestone procedure and the following additional steps.
  - *Return of staked tokens accordingly:*
    From the P tokens originally staked in this project, subtract the transaction fee. That is, send $d \cdot P$ tokens to Covee. Each user $i \in Q$ receives

$$(1-d) \cdot P \cdot \frac{1}{u^m} \cdot \sum_{x=1}^{m} \left(u^x - u^{x-1}\right) \cdot f_i^x,$$

    where $(u^x - u^{x-1})$ denotes the team initiator's view regarding how much progress the team has made towards the project's completion between milestones x and $x-1$. As before, we let $u^0 = 0$. Correspondingly, f_i^m denotes the team's view (as computed by the P2P review mechanism) of how much i has contributed (relatively) to this progress. The fraction $1/u^m$ is a normalization factor (percentage) to ensure that the full amount of staked tokens is returned in the end. If $u^m < 1$, i.e, the team initiator decides at project end that the goal has not been completed to 100%, then the remaining amount of staked tokens $(1-d) \cdot P \cdot (1-u^m)$ is distributed among the users by equally increasing token amounts by the positive percentage factor $1/u^m$.
  - *Potentially reimbursement of the client:*
    Any amount of the client-fee not paid out to the team members is reimbursed to the client. Thus, the client is reimbursed with $F \cdot (1-u^m)$ tokens. Note that Covee does not subtract the Covee network fee from this amount.

Smart contract

Program listing 2 shows an example of Covee's smart contract source code. This specific function, written in Solidity, allows users to claim back their returned, released, or paid out tokens.

	M1 (25%)	M1 (25%)	M1 (25%)	M1 (25%)	Weighted Average
A	10%	10%	10%	10%	10%
B	80%	80%	0%	0%	40%
C	0%	0%	80%	0%	20%
D	10%	10%	10%	90%	30%

A - team initiator; **B** - data scientist; **C** - machine learning expert; **D** - computer scientist

Table 5: Example user-milestone matrix, showing how contributions may differ over the course of a project and how this can be specified via a matrix at the beginning of the project.

```
    function claimMoneyBack(bytes32 _projectId, uint256 _roleIndex)
        guardStages(_projectId)
        external
    {
        require(
            stage[_projectId] == Stages.ProjectInProgress ||
            stage[_projectId] == Stages.ProjectInvalid ||
            stage[_projectId] == Stages.ProjectDefinition ||
            stage[_projectId] == Stages.ProjectEnded
        );

        bytes32 applicantForRoleId = composeBytes32Uint256AddressKey(_projectId,
            _roleIndex, msg.sender);

        if (stage[_projectId] != Stages.ProjectInvalid)
            require(!isTeamMember(_projectId, _roleIndex, msg.sender));

        require(applicant[applicantForRoleId]);
        applicant[applicantForRoleId] = false;
        uint256 reimbursement = projectStakingAmounts[_projectId][_roleIndex];
        coveeToken.increaseApproval(msg.sender, reimbursement);
        emit Reimbursed(_projectId, msg.sender, reimbursement);
    }
    event Reimbursed(bytes32 indexed projectId, address beneficiary, uint256 amount);
```

Listing 2: Smart contract function that allows users to claim back tokens.

Example outcome of the return of stake calculation

Consider a situation where the contribution shares, specified by the team initiator in the description of the project, are 10%, 10%, 50% and 30%, see the left panel of Figure 5. At the end of the project, the P2P review mechanism (based on the users' reviews) may show a different share vector, such that the staked tokens are returned according to the following percentages: 15%, 20%, 40%, 25%.

Analysis

We now analyze the properties of the return of stake and client-fee payout mechanism and show that it achieves the goals we have laid out in the beginning of this section.

First, we disincentivize free-riding via the usage of the P2P review mechanism. If a user tries to free-ride, i.e., if he does not contribute sufficiently to the project, the other team members would see this and then have the opportunity to review him accordingly in the P2P review

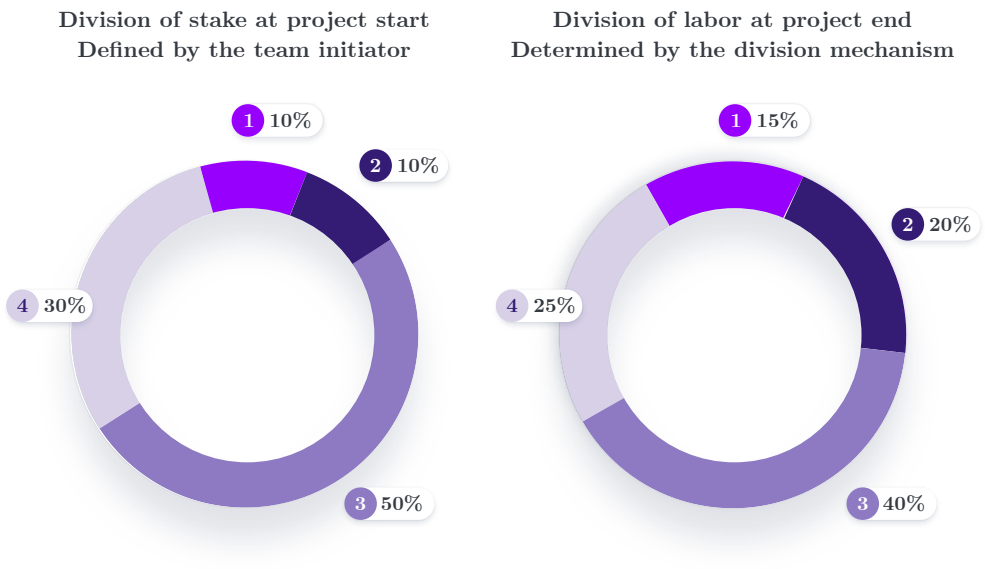


Figure 5: The left panel shows the contribution shares as initially specified by the team initiator in the description of the project. The right panel shows the contribution shares, i.e., the return ratios for the total amount of staked tokens, as calculated by the P2P review mechanism. Source: Covee Network (2018).

mechanism at each milestone. If a user always performs low effort, he will then end up receiving a very small fraction of the staked tokens (most likely getting back less tokens than he has initially staked) and of the client-fee. In contrast, users who exert high effort will be rewarded via the P2P review mechanism, obtain a high fraction of the staked tokens and the client-fee.

Second, we make extensive use of milestones, instead of just letting the client and/or the users provide reviews at the end of the project. This has multiple advantages: (a) Users who underperform (in the view of their teammates) can still increase their performance and/or talk to their teammates about the different expectations and/or start a dispute resolution case (in rare exceptions). (b) Milestones solve the "sniping problem" where every user waits until the last minute of the project, observing whether the other users are exerting effort or not (in which case the project would fail anyway), and then only exerting effort if everybody else does. With the milestones, the users are incentivized to complete their tasks at the latest at the end of each milestone. (c) The milestones also help getting the client involved, which is important for the payout of the client-fee. At each milestone, the client/team initiator reviews the progress, and the client only pays out the fraction of the client-fee corresponding to his view of the completion of the project. This reduces the risk for the client, compared to paying everything at the beginning and then hoping the project will be completed. (d) Furthermore, the client reviews also allow the team and the client to identify problems/different expectations early on, and rectify them. Getting early feedback from the client provides the team with predictability whether the team will be able to complete the project at all and get a sufficiently large fraction of the client-fee (thus reducing risk for the team members). If at one of the early milestones, the team members see that this will likely not be the case (e.g., because the client gives them a low review even though the work has been completed, or because they do not have the necessary skills to complete this project), they have the option to abort the project early on, or start a dispute resolution case.

Extensions

- *Two-Person teams:*
  For teams of size two, the P2P review mechanism does not work anymore, because we

cannot obtain an objective review regarding the relative contributions of the two team members. Given this, we propose that two-person teams are handled more akin to regular employment contracts:

1. The team initiator creates a project and invites applications with a fixed percentage and a fixed staking amount for the open role, and potentially a fixed client-fee.

2. Applicants apply to the role by submitting the fixed staking amount. The team initiator then reviews the applicants and picks his most-preferred candidate.

3. At the end, staked tokens are returned to the team members. If a client-fee was offered, then the client-fee is paid out to all team members according to the percentages (i.e., turning this contract into a conventional take-it-or-leave-it contract). There shall be (a) no reviews, and (b) no mechanisms.

- *Timing of milestone definition:*
  We assume the following timing: First, during the project definition / creation phase, the team initiator defines the project timeline in broad strokes (e.g., project start date and project duration). Second, during the project refinement phase, at the beginning of the project, the team initiator together (after discussing the project with the other team members), define the set of milestones, which includes (for each milestone), in particular, its deadline and how much each team member is expected to contribute to the milestone. Two natural variations of this process are conceivable:

  1. The team initiator defines all milestones (including deadlines and contribution levels) by herself, before the team is assembled, during the project creation / definition phase. This has the advantage that team members already know exactly when their role is needed during the project at the point in time when they apply for the project. Additionally, this avoids complicated discussions about the milestones among all team members. On the other hand, the team initiator may need the input (expertise) of the other team members to define the milestones. Thus, it is not clear whether this variation is preferable or not.

  2. Instead of defining the milestones all at one point in time (either during the project definition phase, or during the project refinement phase), the milestones could be defined "on the fly" – milestone by milestone. This may be closer to today's practice of software engineering projects and may be more suitable in particular for larger/-complex projects. A possible sequencing of activities could look as follows: (i) at the beginning of the project, work out details of the project plan; (ii) kick-off meeting with all team members, including finalization of project plan and definition of first milestone; (iii) sprint until first milestone deadline + evaluation of milestone + definition of next milestone (iv) repeat step (iii) until end of project; (v) final evaluation of the project.

- *Timing of the return of staked tokens:*
  In the current description of the mechanism, the client-fee is partially paid out at each milestone, while staked tokens are only returned at the end of the project. A natural variation of the mechanism is where staked tokens are also returned (partially) at each milestone. This is a straightforward modification.

7 Reputation mechanisms

Overview

There are three sets of reputations:

1. *Peer reputation*: A peer-to-peer reputation system for users that results from Covee users' mutual ratings.

2. *Client reputation*: A client-to-peer reputation system for users that results from clients' ratings of their Covee users.

3. *Market reputation*: A peer-to-client reputation system for clients that results from Covee users' ratings of the clients.

Reputations systems on Covee are intended to foster good behavior amongst users and between users and clients. They help to wash out the bad apples and breed interactions amongst good ones.

Background

Reputation systems are what has been called a "universal currency" Milinski (2016) to overcome free-riding in repeated social dilemma interactions such as the voluntary contributions game. Reputational concerns counteract the incentives to free-ride, because the momentary gains of free-riding result in a reputational loss that reduces expected future gains as others will avoid working in projects involving individuals with a low reputation.

Goals

Reputational concerns create additional incentives to contribute in order to maintain / gain a better reputation by turning a one-shot interaction into a repeated game. A loss of reputation leads to a loss of standing and therefore a lowered chance of entering good teams in the future. A gain in reputation leads to an increased community standing and thus an improved chance of getting into a good team in the future. Thus, reputation turns an anonymous interaction into a non-anonymous one and breeds indirect and direct reciprocity which helps overcome the collective action problem and free-riding.

In essence, a reputation system collects feedback from participants as automated "word-of-mouth". This feedback is aggregated and made available to users, helping them to guide decisions about whether to engage in a transaction. The design of a reputation system involves deciding what information to collect and when, whether to limit who can provide feedback, and how to provide incentives to leave useful feedback. Some market platforms also perform checks before allowing entry into a market (e.g., verified identities on Airbnb, driving experience and criminal history screening on Uber), and we shall build on this idea by using external information to "kick-start" users' and clients' reputations on Covee.

The basic concept underlying reputation systems in online communities is not a new one, as it mirrors traditional word-of-mouth information. What distinguishes reputation systems on online platforms from conventional uses of reputation is that there is a need to design the exact way in which word-of-mouth information is collected and shared. There are many possible ways to collect feedback and many ways to aggregate and report it.

Implementation

We believe a 10-star rating scale is the best choice at this stage (where one design idea is to use the Covee logo instead of a star symbol). Going for 10 levels of rating, we expect there not to be excessive use of the top rating as would perhaps be the case with coarser scales such as the widely used 5-star scale. We know from evidence on eBay, Uber, etc. that ratings on a 5-star scale quickly evolve such that, starting with an initial full usage of the scale, soon predominantly the 1 and 5 stars are given (Nax, 2016). Hence, with many ratings given, the 5-star scale becomes equivalent to a scale where you just vote Up or Down, or give a Like or Dislike. Such a scale is fine (and also used on eBay and other platforms), but requires very large numbers of ratings to be meaningful. A finer-grained scale is therefore preferable for our setting, at least in the initial phases of Covee.

Rating and aggregation

Voting takes place in $\{1, 2, \ldots, 10\}$. An individual i's reputation, denoted by superscripts indicating which reputation system is meant (either $k = C$ for client, P for peer, or M for market), is

$$r_i^k = \frac{1-\delta}{1-\delta^T} \sum_{t=0}^{T-1} \delta^t r_{ij}^t, \tag{2}$$

where T is the total number of projects t (ordered from oldest to most recent) that i participated in. Both T and r_i^k are displayed, the latter rounded to the nearest decimal. $\delta \in (0, 1]$ is a "discount factor" applied to old projects. At the beginning, we set it to 1. Later, it can be set below 1 (once average users/clients have worked on/contracted a sufficient number of projects).

Analysis

In this section, we develop a simple model of a reputation system in the context of a repeated prisoner's dilemma (which we shall abbreviate by PD in some places) and look to provide formal reasoning for the way in which a reputation system can provide sanctioning, and thereby foster cooperative behavior. We consider a setting where each agent is sometimes a consumer and sometimes a provider, and where each party to a transaction has a choice between cooperative and uncooperative behavior.

Figure 6 depicts the familiar Prisoners' Dilemma game. Rather than a repeated game between two agents we consider a repeated game between n agents, with n "large" (and even for simplicity). In each round, the agents are paired up randomly and each pair of agents plays a Prisoners' Dilemma game and each agent can cooperate (C) or defect (D). The only Nash equilibrium of the one-shot game is (D, D), but things are different in the repeated game. Let $h = (\alpha^{(0)}, \alpha^{(1)}, \ldots)$ denote the history of action profiles, where $\alpha^{(k)}$ inA denotes the action profile in period k. Let $\delta \in [0, 1)$ denote the discount factor (note that this discount factor is in the same spirit but not the same as applied to discounting reputation ratings in the previous section). Agent i's "discounted utility" for a particular history h is

$$u_i(h) = \sum_{k=0}^{\infty} \delta^{(k)} \tilde{u}_i(\alpha^{(k)}),$$

where $\tilde{u}_i(\alpha)$ is the utility of the stage game.[1] Depending on δ, the following statement can be made.

[1]Recall the identity $1 + \delta + \delta^2 + \delta^3 + ... = \sum_{k=0}^{\infty} \delta^k = 1/(1-\delta)$.

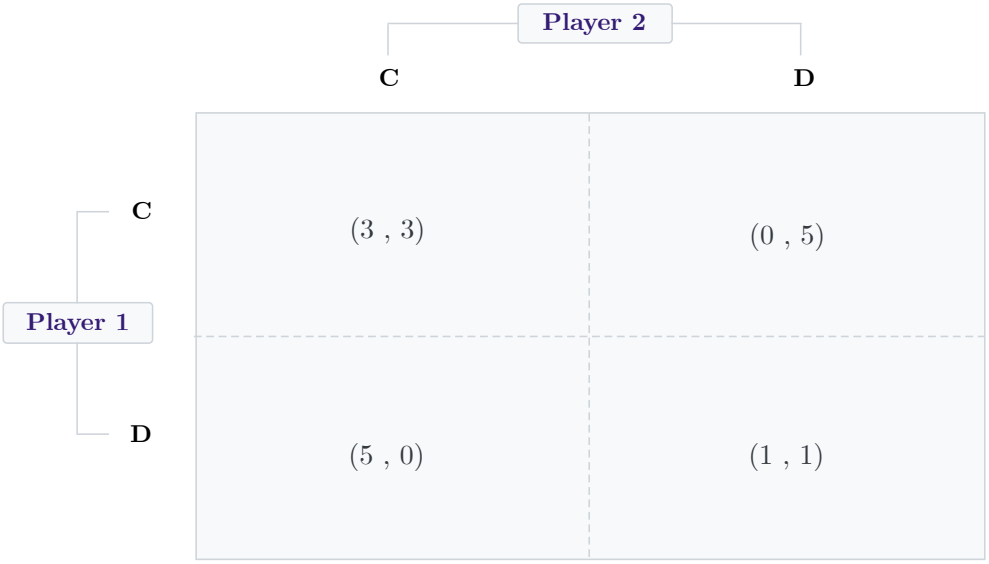


Figure 6: The prisoner's dilemma is a simple version of the voluntary contributions game for two players and binary actions (contribute or not.) Source: Covee Network (2018).

Theorem: Without a reputation system, players need to be very patient (discount factors close to one) for cooperation to be sustained by "grim trigger" in a subgame-perfect equilibrium. With a reputation system, for any discount factor $\delta \geqslant 1/2$, cooperation can be sustained by a "reputational grim-trigger" strategy.

Sketch of proof: In terms of the key strategic incentives, reputation turns the many-person game into a two-person game, where (in)directly reciprocal behavior is able to sustain cooperation using "grim trigger" (i.e. if you deviate from cooperation once to defect, you will not meet a player who cooperates again). See Friedman et al. (2007) for a formal proof of this result.

Extensions

A concern that we have not modeled here is that, in the wider game, strategies involve the possibility of lying about others when reporting feedback. For example, reputational grim-trigger is vulnerable to adversarial behavior where one agent falsely claims that others have defected. This opens up the concern of threats, where an agent threatens to retaliate with a bad report in response to receiving a bad report. We do not expect this to be a major issue, as the success of reputation systems online is a widespread phenomenon. However, we shall return to concerns about strategic behavior and strategic lying about others, including problems of retaliation, in our section on "Dispute Resolution".

7.1 For users: peer reputation and client reputation

Goals

Our main goal is to incentivize good behavior over time, across projects, by users, both toward their peers as well as toward the clients. We want to identify good/medium/bad users, and allow users to exploit this information to discriminate, before they work on high-profile projects. Reputation thresholds, that can be used to identify the right team, allow clients/team initiators to experience and price-discriminate (at least implicitly, by setting a higher reputation threshold).

Mechanism

At the end of the project, the users and the client are both asked to provide a reputation rating (on a 10-point scale). For the client, this request to provide a reputation rating for the team's performance follows immediately the client's project review at the last milestone. For the users, the request to provide a reputation rating for every other team member follows immediately the last P2P review (also at the last milestone). After the client has finished his last project review such that the client-fee payout could be determined, the users are asked one more time to also provide a reputation rating for the client. Note that clients may or may not have sufficiently detailed information to judge/rate users individually. Hence, instead, clients may opt to rate the overall project instead of the individual users, in which case this rating is applied in lieu of an individual rating per user.

Ratings are used to calculate two reputation scores. Let r denote the vector of reputations r_i (i's reputation). For every user, we calculate two scores,

- the "client reputation" $r_i^C \in [1, 10]$ and
- the "peer reputation" $r_i^P \in [1, 10]$,

where r_{ij}^t denotes the reputation rating (or vote) given by i to j at the end of project t. Note that i can either be a client or a peer.

In fact, instead of the above reputations, we use external information instead of peer reputation and client reputation when the respective information necessary to calculate those is not yet available. Once at least 5 reputation ratings are submitted about an individual, use Equation (2) and update in multiples of 5 new ratings being submitted.[2]

External information may include links and information from other accounts such as Kaggle's metrics for data scientists, including "rank", "tier", "medals" and "points". LinkedIn profile information could be used for domain experts and others to verify educational and professional credentials. External information about computer scientists could include Github metrics, including the number of "repositories", "stars", "followers" and "contributions in the last year". Further information could be gathered from ResearchGate and Google Scholar profiles.

In summary:

- At the final milestone, at the end of a project, all team members rate each other. To avoid retaliation, we only publish results after all team members have submitted (locked in) their reputation ratings. We only publish aggregate scores, and only after a sufficient number of ratings (not projects) (e.g., 5) have been submitted. We only update all reputation scores in batches of 5. We may start discounting old information. To avoid manipulation among friends, we could additionally discount ratings that are reciprocal, i.e. of the same users rating each other multiple times.
- Additionally, the client rates the team. Per default, the client rates each team member. If the client cannot rate individual team members, he rates the whole team, that is, the same rating is applied to all team members as if it was a rating of that team member individually.

[2]Multiples of 5 guarantee that you cannot deduce directly from whom you got which rating.

Extensions

It may be interesting to also allow a voluntary detailed reputation feedback form for the peer reputation. Similarly, the client could also provide a voluntary detailed report including text feedback on the team's performance.

However, it is a known fact (Filippas et al., 2017) that people do not like to rate each other badly. Addressing this with a more sophisticated design is future work. For now, we address this with a 10-point rating scale to get decent amount of differentiation. The use of a richer reputation scale could become more useful over time. Our proposal is that, with the 100% milestone, after user has filled out the mandatory "reputation rating", we may want to ask the user if he is willing to provide "more detailed feedback on his team members". Make this voluntary. Here, you can ask individual dimensions (e.g., diligent, fast to respond, expertise, etc.).

Another natural extension is to additionally capture a user's reputation as a "team manager". Not every user may be equally good at managing a team. Over time the role of a team manager may become an import role on the Covee platform. The reputation of the team manager would capture her ability to put together a successful team, define the milestones well, organize/manage the team well, and bring the project to successful completion. Ideally, over time, users would only want to join teams that are managed by users with a high team manager reputation (at least for big projects).

Options for future mechanism upgrades

One option to extend and improve the reputation system is to aggregate all scores into one "top-level" reputation score. For now, we always display both the client and the peer reputation. One possible bottleneck is that, at the beginning, users have no internal (in particular no client) reputation. Hence, Covee includes external information from the beginning on instead.

7.2 For clients: market reputation

Goals

There is another reputation rating associated with clients in order to help users choose the best clients to work for, and to disincentivize clients from hiring a team and not paying them at the end, or other forms of misdemeanor. In addition, the reputation is meant to incentivize clients to work well with their teams (i.e., be good employers). A good client is one who is responsive, pays well and on-time, whose project is well-structured and well-documented; i.e. one for whom it is enjoyable to work.

Mechanism

Once the project has terminated, and payments including client-fees are paid out, users are also asked to rate clients (non-mandatory). We use the same principles as for the user reputation, that is, a 10-star scale.

For i being a client, we calculate "market reputation" $r_i^C \in [1, 10]$ based on r_{ij}^t denoting the reputation rating (or vote) given by i to j at the end of project t, where i is now a user and j a client. Again, as for the peer reputation case, external information is used for the client when user-generated reputation ratings for the client are not yet available. Examples could include a company's LinkedIn company profile or their Upwork employer profile. Once at least

5 reputation ratings are submitted about a client, use Equation (2) and update in multiples of 5 new ratings being submitted.

8 Dispute resolution

In future versions, Covee will design and deploy dispute and conflict resolution mechanisms. Given the breadth and depth of incentives and mechanisms described above, we expect conflicts and disputes requiring external intervention to be rare. However, there may be cases where users and clients have complaints regarding:

- unfair review,
- collusion,
- unfair reputation rating, and
- other forms of misconducts.

The idea is to create a pool of users who can be called upon to mediate and resolve conflicts and in exchange be rewarded with tokens for their service. Candidates would be volunteering users with high reputation scores. As an ad-hoc example, rewards to a mediator could come out of the team's staking pool thereby creating an incentive for the team to resolve conflicts internally. If the dispute cannot be solved within the team, team members have to contribute a share of their staked tokens to an external mediator. Such a system would also have the desirable property of decentralizing decision-making in this matter to network users. The concrete specification and analysis of a dispute resolution mechanism is left for future work.

Endnotes and references

Abdulkadiroglu, A. and Sonmez, T. (2003). School choice: a mechanism design approach. *American Economic Review*, 93(3):729–747.

Ausubel, L. M. and Baranov, O. V. (2014a). The combinatorial clock auction, revealed preference and iterative pricing.

Ausubel, L. M. and Baranov, O. V. (2014b). Market design and the evolution of the combinatorial clock auction. *American Economic Review: Papers & Proceedings*, 104(5):446–451.

Benkler, Y. (2016). *Handbook on the economics of the internet*, chapter Peer production and cooperation, pages 91–119. Edward Elgar.

Burton-Chellew, M. N., Nax, H. H., and West, S. A. (2015). Payoff-based learning explains the decline in cooperation in public goods games. volume 282: 20142678.

Catalini, C. and Gans, J. S. (2017). Some simple economics of the blockchain. Technical report, Rotman School of Management Working Paper No. 2874598; MIT Sloan Research Paper No. 5191-16.

Chaudhuri, A. (2011). Sustaining cooperation in laboratory public goods experiments: a selective survey of the literature. *Experimental Economics*, 14:47–83.

Coase, R. (1937). The nature of the firm. *Economica*, 4(16):386–405.

Cramton, P. (2006). *Combinatorial auctions*, chapter Simultaneous ascending auctions, pages 99–114. MIT Press.

Cramton, P. (2013). Spectrum auction design. *Review of Industrial Organization*, 42(2):161–190.

de Clippel, G., Moulin, H., and Tideman, N. (2008). Impartial division of a dollar. *Journal of Economic Theory*, 139:176–191.

Dixon, C. (2018). Why decentralization matters. `https://medium.com/@cdixon/why-decentralization-matters-5e3f79f7638e`.

Filippas, A., Horton, J., and Golden, J. M. (2017). Reputation in the long-run. Technical report, CESifo Working Paper Series 6750, CESifo Group Munich.

Friedman, E., Resnick, P., and Sami, R. (2007). *Algorithmic game theory*, chapter Manipulation-resistant reputation systems, pages 677–697. Cambridge University Press.

Garbers, Y. and Konradt, U. (2014). The effect of financial incentives on performance: a quantitative review of individual and team-based financial incentives. *Journal of Occupational and Organizational Psychology*, 87:102–137.

Harris, M. (2016). How to beat more than half of world's best data scientists without doing any data science. `http://www.priceactionlab.com/Blog/2016/09/data-science`.

Hart, O. and Holmström, B. (1987). The theory of contracts. In Bewley, T. F., editor, *Advances in Economic Theory, Fifth World Congress*. Cambridge University Press.

Hashim, M. J. and Bockstedt, J. (2017). Overcoming free-riding in user-generated content platforms: punishments and rewards for individuals and groups.

Holmström, B. (1982). Moral hazard in teams. *The Bell Journal of Economics*, 13(2):324–40.

Holmström, B. and Roberts, J. (1998). The boundaries of the firm revisited. *The Journal of Economic Perspectives*, 12(4):73–94.

Ishikawa, T. (2002). The labour market and the distribution of income: the dual labour market approach. *Income and Wealth*, 1(8):176–241.

Jackson, M. O., Rogers, B., and Zenou, Y. (2016). Networks: an economic perspective. Technical report, CEPR Discussion Paper No. DP11452.

Kominers, S., Teytelboym, A., and Crawford, V. (2017). An invitation to market design. Technical report, Working Papers 2017-069, Human Capital and Economic Opportunity Working Group.

Ledyard, J. (1995). *Handbook of experimental economics*, chapter Public goods: a survey of experimental research, pages 253–279. Princeton University Press.

Lindberg, V. (2008). *Intellectual property and open source: a practical guide to protecting code*. O'Reilly Media.

Marwell, G. and Ames, R. E. (1979). Experiments on the provision of public goods. I: Resources, Interest, Group Size, and the Free-Rider Problem. *American Jounal of Sociology*, 84(6):1335–1360.

McKinsey Global Institute (2017). Jobs lost, jobs gained: workforce transitions in a time of automation. `https://www.mckinsey.com/mgi/overview/2017-in-review/automation-and-the-future-of-work/jobs-lost-jobs-gained-workforce-transitions-in-a-time-of-automation`.

Milinski, M. (2016). Reputation, a universal currency for human social interactions. *Philosophical Transactions of the Royal Society B*, 20150100.

Nakamoto, S. (2008). Bitcoin: a peer-to-peer electronic cash system. `https://bitcoin.org/bitcoin.pdf`.

Nax, H. H. (2016). The logic of collective rating. *Frontiers Physics*, 4(15).

Romano, J. P. and Wolf, M. (2005). Stepwise multiple testing as formalized data snooping. *Econometrica*, 73(4):1237–1282.

Roth, A. E. and Peranson, E. (1999). The redesign of the matching market for american physicians: some engineering aspects of economic design. *American Economic Review*, 89:748–780.

Royal Swedish Academy of Sciences (2016). Oliver Hart and Bengt Holmström: Contract Theory. `https://www.nobelprize.org/nobel_prizes/economic-sciences/laureates/2016/advanced-economicsciences2016.pdf`. Scientific Background on the Sveriges Riksbank Prize in Economic Sciences in Memory of Alfred Nobel.

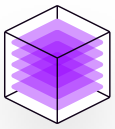
Tideman, T. and Plassmann, F. (2008). Paying the partners. *Public Choice*, 136(1-2):19–37.

Varian, H. and MacKie-Mason, J. (1994). Generalized vickrey auctions. Technical report, University of Michigan.

Vickrey, W. (1961). Counterspeculation, auctions, and competitive sealed tenders. *The Journal of Finance*, 16(1):8–37.

Wenger, A. (1999). *Three essays on the influence of information technology on the organization of firms*. https://dspace.mit.edu/bitstream/handle/1721.1/9683/42588125-MIT.pdf, Massachusetts Institute of Technology.

Wenger, A. (2015). Networks, firms, markets. http://continuations.com/post/126909987225.



Covee Protocol

Powering the decentralized future of knowledge work with smart contracts, a cryptographic token and a unique mechanism design.

www.covee.network

Bahnhofstrasse 3,
8001 Zürich, Switzerland