Managing Innovation in a Crowd*

Daron Acemoglu† Mohamed Mostagir‡ Asuman Ozdaglar§

January 2014

Abstract

Crowdsourcing is an emerging technology where innovation and production are sourced out to the public through an open call. At the center of crowdsourcing is a resource allocation problem: there is an abundance of workers but a scarcity of high skills, and an easy task assigned to a high-skill worker is a waste of resources. This problem is complicated by the fact that the exact difficulties of innovation tasks may not be known in advance, so tasks that require high-skill labor cannot be identified and allocated ahead of time. We show that the solution to this problem takes the form of a skill hierarchy, where tasks are first attempted by low-skill labor, and high-skill workers only engage with a task if less skilled workers are unable to finish it. This hierarchy can be constructed and implemented in a decentralized manner even though neither the difficulties of the tasks nor the skills of the candidate workers are known. We provide a dynamic pricing mechanism that achieves this implementation by inducing workers to self select into different layers. The mechanism is simple: each time a task is attempted and not finished, its price (reward upon completion) goes up.

Keywords: crowdsourcing, crowd innovation, hierarchies, matching.

JEL classification: D83, D20, L22.

*We thank Glenn Ellison, Luis Garicano, Karim Lakhani, Jonathan Levin, David Miller, and seminar participants at Duke, Johns Hopkins, Michigan, Microsoft Redmond, MIT, University of Texas-Austin, and University of Washington for useful comments and discussion. We gratefully acknowledge financial support from Draper Labs and the Toulouse Network for Information Technology (supported by Microsoft).

†Department of Economics, MIT.

‡ Ross School of Business, University of Michigan.

§ Laboratory for Information and Decision Systems, MIT.

1 Introduction

Goldcorp, a gold mining company based in Vancouver, terminated its mining operations and was on the brink of bankruptcy as a result of increases in production costs and its inability to reliably estimate the value and location of gold on its property. As a final maneuver, the company published its geological data online and asked people to submit their estimates and estimation methods in exchange for prize money that totalled a little over half a million dollars. The crowd identified 110 targets that led to the extraction of a staggering $8 million ounces of gold, transforming the $100 million company into a $9 billion powerhouse. Today, Goldcorp is the second largest gold company in the world.[1] More recently, the website "fold.it" provided the public with a chance to design and fold proteins, a computational biology task that routinely stumps supercomputers. Within three weeks, the crowd of participants were able to determine the structure of an AIDS-related enzyme, a task that the scientific community had unsuccessfully tackled for over a decade, and that represents a significant step forward in finding a cure for the disease.

These are two examples of a growing number of success stories where innovation is sourced out to the public.[2] Historically, improvements in information and communication technology have led to major revisions in the nature of the firm (see, e.g., Chandler (1977) and Garicano and Rossi-Hansberg (2006)), and some argue that crowd innovation is poised to be a natural next step in this chain of evolution. The idea that innovation can happen outside of the confinements of a firm is not new, see for example Von Hippel (1976); Chesbrough et al. (2008); Von Hippel (2009), but it is only now that the technological infrastructure has made it possible to facilitate this phenomenon on an unprecedented scale.

Despite its successes, crowd innovation remains a technology in infancy, using mostly ad hoc methods and mechanisms, and faces a wealth of challenges in resource allocation, pricing, and incentive design. A firm can now have immediate access to an unlimited supply of labor and a wide pool of talent and skills, but extracting the good from the bad and managing this pool of workers is fraught with difficulties.

This paper presents a simple model of crowd innovation in which a firm seeks to assign a range of innovation tasks of unknown difficulty to a set of heterogeneous workers using a virtual marketplace. Unlike organizations, where employers can assign tasks to workers at will, crowd innovation takes place in a large and anonymous virtual market where employers cannot assign workers to specific tasks and do not know the workers' specific skills. In addition, the nature of innovation tasks precludes specific information about how difficult these tasks are and what type of workers will be able to complete them.

[1] "Innovation in the age of mass collaboration", Bloomberg Businessweek 2007-02-01.

[2] The emerging technology of "crowdsourcing" sources what has come to be known as "Human Intelligence Task" or HITs to a virtual marketplace. HITs are simple tasks that can be performed by almost any one (for example, determining whether a photograph contains a certain object or not). Because of their easy nature, these tasks are priced extremely low, paying a fraction of a cent on average. We use the term "crowd innovation" for the application of crowdsourcing techniques to innovation tasks, which are more difficult and require skills or innovative solutions to problems that cannot otherwise be routinely handled within a firm.

Our first contribution is conceptual. We show that the informational and coordination difficulties described above not withstanding, crowdsourcing and innovation can be organized as if they were taking place within an organization, but with some constraints resulting from the market allocation. In particular, we show that by providing appropriate rewards and incentives in the marketplace, the problem of crowd innovation can be related to the problem that an organization faces in assigning workers of different skills to tasks of different difficulties.

Our technical contribution formalizes this idea. We first characterize the optimal matching of tasks to workers in environments in which tasks have unknown difficulties but worker skills are known and workers can be assigned to sets of tasks (and thus their "voluntary participation" decisions can be ignored). Under these assumptions, we show that the optimal matching takes the form of a *skill hierarchy*: tasks are first attempted by the less-skilled workers and high-skilled workers only engage in a task if workers with lesser skills are unable to finish it. In particular, workers are arranged into groups according to their skill and tasks are offered to groups in a sequential fashion. We then derive the structure of the optimal hierarchy by studying a dynamic programming formulation. The most distinctive property of the optimal matching is that it ensures that the heterogeneous skills of the available workers are utilized efficiently.

We then show that this optimal matching can be implemented when worker skills are unknown and workers choose when to participate and which tasks to work on. This implementation relies on a simple pricing scheme in which rewards for completing tasks increase the longer these tasks remain uncompleted (i.e., the more times they are unsuccessfully attempted). What makes this pricing scheme support the optimal matching is that, as we show, there is a natural *comparative advantage* property in crowd innovation: if a worker of a given skill level prefers to participate later rather than earlier, then more skilled workers would also choose to do so.[3] Thus the increasing prices of difficult (uncompleted) tasks and comparative advantage jointly provide incentives for higher-skilled workers to delay their participation.[4]

Most closely related to our paper is the work on the design of knowledge or skill hierarchies within organizations, in particular, in Garicano (2000); Beggs (2001); Garicano and Rossi-Hansberg (2004); Antràs et al. (2006); Garicano and Prat (2013). In these papers, workers belong to the same firm and can cooperate on solving problems in order to increase output, and workers in different levels of the hierarchy specialize in solving only certain kinds of problems that complement other existing skills. The decision of which skills to obtain is a decision that is endogenous to the workers and the organization, and can be based, as in Garicano (2000), on the difficulty distribution that the firm faces. This ensures that the skills in the pool of workers and difficulties in the task pool are aligned. This feature, though it makes these models elegantly tractable, does not provide an adequate description of crowd innovation environments, where a mismatch between task difficulties and skills is commonplace, where skills are often very dispersed and unknown, and where voluntary participa-

[3] See Roy (1951); Sattinger (1975); Tinbergen (1974); Teulings (1995) on the role of comparative advantage in other problems of assignment of workers to tasks.

[4] A small literature discusses practical ways of pricing Human Intelligence Tasks in crowdsourcing. See, e.g., Horton and Chilton (2010) and Faridani et al. (2011).

tion of workers is a central issue. In addition, in this literature, there is an "infinite supply of tasks": each worker in the lowest level of the hierarchy produces a task, and if they are unable to perform these tasks satisfactorily, then can ask for help from others in the organization. This naturally leads to a pyramid-like hierarchy. These differences undergird several disparities in the results of the two lines of work. First, though in our setting the optimal matching of skills to tasks is also hierarchical (meaning that more skilled workers are assigned to tasks that less skilled workers attempt and cannot complete), this hierarchy does not necessarily take the familiar pyramidal shape that it takes in these works (as well as in most of the organizational economics literature) because it may be optimal to preserve middle-skilled workers for higher echelons of the hierarchy and allocate only the lowest skilled ones to the bottom layer. Second, in contrast to Garicano's work, where more (possibly infinite) layers in the hierarchy can increase output, increasing the number of layers in the hierarchy may be detrimental. This result highlights the importance of the potential mismatch between skills and tasks. More layers can lead to strictly lower output because they expose more workers to the most difficult tasks that they would not be able to complete (this contrasts with the case where some subset of workers can choose their skill level so as to be able to complete even the most difficult tasks). Finally, another difference from the work on knowledge hierarchies is that crowd innovation works in the marketplace, without knowledge of the exact skills of workers and without the ability to precisely control the assignment of a worker to a specific task, and thus the optimal hierarchy in crowd innovation is not organized by a firm, but results from a market equilibrium.[5]

Recent work by Fuchs et al. (2013) examines an environment similar to ours where producers encounter tasks of unknown difficulties and try to solve them with the help of outside consultants of unknown skills. A key difference is once again that Fuchs et al. (2013) look for a contractual solution to this matching and informational asymmetry problem. Their optimal assignment is thus implemented through a "firm-like" structure in which the highest skilled workers ("consultants") are residual claimants, and employ lower skilled workers and incentivize them through contracts. In contrast, as noted in the previous paragraph, the optimal hierarchy in our setting emerges as a market equilibrium. More specifically, it is implemented via the marketplace using a simple pricing mechanism.[6]

Also complementary to our work is the literature on innovation tournaments. An innovation tournament encourages competition amongst participants in order to obtain a prize. There is still little understanding of how to select the value and timing of these prizes to ensure that the goal of the tournament is achieved.[7] Our paper takes a different approach by combining elements of both cooperation and competition, since workers compete for tasks but at the same time, under the mechanism described above, act as if they are part of a cooperative, hierarchical organization that is

[5]This point relates our work to the literature on experts Wolinsky (1993), where the primary difficulty is judging whether or not a service that an expert provided is justified. This difficulty arises from informational asymmetries between the expert and the party that receives and pays for the service. Recent work shows how an expert can reveal information in a way that manipulates the receiver even when all parties involved are rational Kamenica and Gentzkow (2011).

[6]Another differences that there is no matching among workers in our model.

[7]A common method in the literature models an innovation tournament as a simple all-pay auction and utilizes existing results in that area to draw conclusions about tournaments Moldovanu and Sela (2006); Chawla et al. (2012).

working in tandem on a project.[8]

Finally, our model can be viewed as a specific instance of stochastic matching problems in computer science, where nodes in a graph are connected by edges and these edges have 'success probabilities' attached to them. A matching of two nodes is successful with a probability equal to the probability of the edge that joins these nodes, and success (or failure) is realized upon the actual match. In our setting, one node is a worker and the other is a task, and the probability of successful match indicates how likely it is that the difficulty of the task is within the worker's skill. There has been some recent interest in this problem because of its applications to online dating and ad allocation problems, for example, as in Feldman et al. (2009); Bansal et al. (2010). The stochastic nature of the problem leads to an exponentially large dynamic programming formulation with no adequate approximate solution (see, e.g., Chen et al. (2009)). Because the supply of workers in a crowd innovation environment is usually very large, we circumvent some of the technical problems encountered in the stochastic matching literature by studying the continuum limit.

The paper is organized as follows. Section 2 introduces the model and defines the assignment and implementation problems. Section 3 derives the necessary results to analyze the dynamic programming formulation of the assignment problem. Section 4 provides a pricing scheme for the implementation problem. Section 5 details the role of various assumptions in our results and discusses how the results apply in more general settings, and Section 6 concludes the paper. Proofs are presented in the Appendix.

2 Model

2.1 Setup

There is a set $\mathcal{W}$ of workers indexed by $i \in [0, A]$ and a set $\mathcal{T}$ of tasks indexed by $j \in [0, 1]$. Both sets are equipped with a Lebesgue measure λ and we assume that $\lambda(\mathcal{T})$ is normalized to one and that $\lambda(\mathcal{W}) > \lambda(\mathcal{T})$, i.e. there are more workers than tasks. The difficulty d of a task in a set $T \subseteq \mathcal{T}$ is a nonnegative scalar distributed according to F_{d_T}. Similarly, the skill s of a worker in a set $W \subseteq \mathcal{W}$ is a nonnegative scalar distributed according to F_{s_W}. We will assume that the workers in $\mathcal{W}$ are ordered in ascending order of skill, so that $s_i \geq s_j$ if $i > j$. Define $G(s) : \mathbb{R} \to [0, \lambda(\mathcal{W})]$ to be the measure of workers whose skill is less than or equal to s and let $G^{-1}(w)$ be its inverse, i.e., $G^{-1}(w)$ is the skill at or below which a measure w of workers reside.[9]

Motivated by the crowdsourcing application which takes place in the marketplace, we assume that once a set of workers have been given access to a set of tasks, each will freely choose which tasks to work on. We also assume that working on a task is time-consuming, so the worker can at most work on a single task. This motivates our definition of a matching, which we give next.

[8] Based on a large data set, Boudreau et al. (2011) find that more unrestricted entry into a tournament is useful the more difficult (in a probabilistic sense) an innovation task is. We find that for some scenarios, even though in our setting workers attempt tasks sequentially, the same recommendation holds true.

[9] The difference between $F_{s_W}(s)$ and $G(s)$ is that the first is a probability distribution, while the second is not since $\lambda(\mathcal{W}) > 1$.

A *matching* $m(W,T)$ is a measurable map satisfying measure consistency: for every open interval $Z \subseteq W, \lambda(Z) = \lambda(m(Z))$. Furthermore, if $\hat{W}$ and $\bar{W}$ are in W and $\hat{W} \cap \bar{W} = \emptyset$, then $m(\hat{W}) \cap m(\bar{W}) = \emptyset$. Therefore, a matching does not specify which of the workers in W will work on which of the tasks in T, but imposes that this assignment will be one-to-one (since a single worker cannot simultaneously work on two tasks).

Define the output of a pairing of a worker of skill s with a task of difficulty d to be equal to 1 if $s \geq d$ and 0 otherwise. The output of a match m, $\theta(m)$ is the total amount (measure) of tasks successfully completed under m, and is thus given by integral of the output over all task-worker pairs in m.

We also assume that the time a worker needs to spend working on a task until he realizes that he cannot complete it is much smaller compared to the time spent by workers who are actually able to finish their tasks.[10] This implies that over time, though a single worker cannot successfully work on two tasks, a single task can be sequentially assigned to two different workers. This then motivates a central aspect of our approach, which allows matching to be sequential. In a sequential matching process, a first match m_1 would assign some tasks to some workers, and then a second match m_2 would assign some tasks (possibly containing uncompleted tasks from the first match) to another set of workers and so on. To capture this idea formally, we denote by W_i and T_i the sets of workers and tasks that constitute the i^{th} match. The set of tasks completed in the i^{th} match is denoted by T_i^c and the set of tasks leftover after the i^{th} match is denoted by T'_{m_i}.

Definition 1. Let $\mathcal{T}$ be a set of tasks. A **sequential** matching μ of tasks to workers is a series of matchings $\{m_i\}$ with the following properties:

1. $W_i \cap W_j = \emptyset$ for all i and j,
2. $T_1 \subseteq \mathcal{T}$ and $T_i \subseteq T'_{m_{i-1}}$ for $i = 2, ...$

The first part of the definition implies that a worker can be part of at most one matching. The second states that the pool of tasks available at any point is equal to the set of tasks we started with minus the tasks that were completed in earlier matchings. Notice that a one-shot matching m of T to W can be written as a degenerate sequential matching μ with $T_1 = T$ and $W_1 = W$. As mentioned, we use $F_{d_{T_i}}$ and $F_{s_{W_i}}$ to refer to the distributions of difficulties and skills in the task and worker groups at stage i respectively. Throughout the paper we denote a sequential matching by μ and a regular,

[10] While an unsuccessful worker could keep a task for longer rather than reveal that he is unable to complete it, we assume that this is not the case, a result that can be derived formally by assuming that the worker has some positive opportunity cost of time somewhere else. There is also a possibility that a worker who fails to complete a task ignores reporting the task's status and simply moves on, leaving the task in limbo. One strategy to deal with uncompleted tasks remaining in limbo is the one employed by crowdsourcing platforms like Amazon's Mechanical Turk, which gives only a fixed amount of time to a worker to communicate the status of the task, after which the task is returned back to the pool of unassigned tasks.

We might also remark that in a hierarchical matching in the form we derive below, because remaining tasks become successively more difficult in expectation, a worker who is unsuccessful at one stage would prefer to exercise his outside option rather than attempt further tasks in later stages. We discuss how our results are modified when workers can participate in further stages after failing to complete their assigned tasks in Section 5.

one-shot matching by m. The output of a sequential matching is the total output of the individual matchings it comprises: $\theta(\mu) = \sum_i \theta(m_i)$.

The next definition introduces a key concept of our analysis.

Definition 2. Consider a sequential matching and for all i, let $\min\{W_i\} = \{\min_j s_j | j \in W_i\}$ and $\max\{W_i\} = \{\max_j s_j | j \in W_i\}$ be the skills of the least and most skilled workers, respectively, in W_i. A sequential matching is **hierarchical** if $\max\{W_i\} \leq \min\{W_l\}$ whenever $i < l$.

A hierarchical matching is a partition of workers into several groups based on skill, such that workers in group i are less skilled than workers in group $j > i$. Tasks are then offered to the least-skilled group, where some tasks may be completed and some not. The remaining tasks are assigned to the next group and the process repeats.[11]

From the discussion above, a sequential matching is likely to take longer to conclude than a one-shot matching. To capture this, we assume that the employer can afford to have at most k sequential rounds of matchings. The objective then is to find a output-maximizing matching μ^* that consists of at most k matchings. Let us denote the number of individual matchings in a sequential matching μ by $|\mu|$, then the employer wants to solve

$$\begin{aligned} &\max_{\mu} \theta(\mu) \\ &\text{s.t. } |\mu| \leq k \end{aligned} \tag{1}$$

In this paper, we consider two versions of the employer's problem: the *assignment problem*, and the *implementation problem*, which we next discuss.

2.2 The Assignment Problem

The assignment problem involves solving (1) assuming that worker skills are known and workers can be assigned to different stages of matching (without respecting any additional incentive compatibility or participation constraints). In imposing that the assignment of heterogeneous workers to tasks be performed without respecting incentive compatibility constraints, this assignment problem is similar to the approach in Garicano (2000) and the subsequent literature on knowledge hierarchies in organizational economics, with the important differences noted earlier of having a finite (vs. infinite) measure of tasks and exogenous worker skills. Also, for our purposes, this problem is just a step in the mathematical characterization of the market allocation rather than an economic problem of interest.

Note also that even in (1), we are imposing the constraints implied by the institutional structure of outsourcing and crowd innovation problems whereby within a particular stage of a sequential matching, the employer has no control over which worker will work on which task.

[11] We use the terms "groups", "levels", and "layers" interchangeably throughout.

2.3 The Implementation Problem

The implementation problem characterizes the optimal matching subject to the additional informational and incentive compatibility constraints of workers — which we ignore in the assignment problem. This problem can in general be quite difficult in view of the myriad of incentive compatibility constraints which may distort the optimal matching away from the "first-best" matching — the solution to the assignment problem. In our case, however, we will show that with an appropriate pricing scheme, the assignment problem can be decentralized, which is a much more tractable problem than a general characterization of "second-best" matching mechanisms.

3 The Assignment Problem: Analysis and Optimal Matching

In this section we analyze the structure of the optimal sequential matching. We restrict our decision variable in stage i to be the choice of W_i. This group is assigned a measure of tasks *uniformly at random* from the pool of available tasks, i.e., choosing a particular group of tasks to match to workers is not part of the decision process. This is a consequence of the fact that, as discussed in the previous section, in a crowd innovation environment workers who decide to participate pick whichever task they choose from the available group of tasks with no control from the employer.

In light of this, we use the notation $m(W, T)$ to mean a random assignment of tasks in T to workers in W. We will sometimes abuse notation slightly by referring to $m(W_i, T)$ as m_i, the i^{th} match.

The structure of the optimal sequential matching may not be obvious at first. One can imagine a scenario where the difficulty distribution makes assigning tasks to more skilled workers first and then moving to less skilled ones optimal.[12] The idea is that if there are few difficult tasks and lots of easy tasks, then the skilled workers would finish the difficult tasks that can later burden the less skilled workers, and hence these latter workers would get a chance to focus on the easy tasks without running into difficult ones. This intuition turns out to be incorrect in our setting. The reason, which will be highlighted in detail in Lemma 1, is that once tasks go through a matching, the difficulty distribution of the remaining tasks is never easier –in a stochastic dominance sense– than the original distribution before the matching. The main result of this section is the following theorem, which characterizes the optimal sequential matching.

Theorem 1. *Consider a set of tasks $\mathcal{T}$ of unknown difficulties and a set of workers $\mathcal{W}$ with $\lambda(\mathcal{W}) > \lambda(\mathcal{T})$ and smooth distributions F_{d_T} and F_{s_W}. There exists a hierarchical matching μ^H of tasks to workers such that $\theta(\mu^H) \geq \theta(\mu)$ for all μ.*

Theorem 1 states that the optimal matching is obtained by arranging workers in a hierarchical fashion. The rest of this section provides a series of lemmas that establish the structure of the optimal matching and ultimately lead to the proof of the theorem. We first illustrate these ideas through a simple example.

[12] In fact, many of the approximation algorithms for the stochastic matching problem discussed in the Introduction rely on the greedy algorithm, which here corresponds to assigning tasks to the most skilled workers first.

Example 1. Consider a set of tasks $\mathcal{T}$ with measure normalized to unity and let the distribution of difficulties $F_{d_\mathcal{T}}$ be $U(0,1)$. Let $\mathcal{W}$ be a set of workers with $\lambda(\mathcal{W}) = 2$ and the distribution of skills $F_{s_\mathcal{W}}$, also $U(0,1)$. Assume that the number of desired groups, k, is equal to 1, then the optimal one-shot matching assigns all tasks to the group of workers whose skills are in $(0.5, 1)$, i.e. the most skilled group of workers with measure equal to 1. The output of this matching is equal to 0.75, since any task with difficulty less than or equal to 0.5 will be finished with probability 1 and a task with difficulty $x > 0.5$ will be finished with probability $2(1-x)$, leading to a total output of

$$\frac{1}{2} + \int_{0.5}^{1} 2(1-x) f_{d\,\mathcal{T}}(x) dx = \frac{1}{2} + \frac{1}{4} = \frac{3}{4}.$$

Assume now that $k = 2$, then it can be verified that the optimal grouping is as follows: W_1 contains all workers with skills in $(0.25, 0.75)$ and W_2 contains all workers whose skills are in $(0.75, 1)$. The matching $m(W_1)$ has $\theta(m_1) = 0.5$ (since any task with difficulty less than 0.25 is finished with probability one, any task whose difficulty is over 0.75 is not finished, and any task whose difficulty is in $(0.25, 0.75)$ is finished with probability 0.5). The leftover tasks from this matching constitute the group $\mathcal{T}'$ which has measure 0.5 and difficulty distribution $F_{d_{\mathcal{T}'}}$ This distribution has two types of tasks, those that are in $(0.5, 0.75)$ and those that are in $(0.75, 1)$, with each type having an equal measure of 0.25. Matching $m(W_2)$ assigns the remaining tasks to W_2 and has $\theta(m_2) = 0.375$, since all tasks with difficulties less than 0.75 will be finished, and tasks with difficulties in $(0.75, 1)$ will be finished with probability 0.5. Thus the total output of this matching is equal to $\theta(m_1) + \theta(m_2) = 0.875$, which is higher than the one shot matching output of 0.75.

Example 1 shows the idea behind our approach. The matching with two groups was able to improve on the matching with a single group because in the latter, some of the skilled workers were assigned tasks that could have been done by other less skilled workers, resulting in under-utilization of their skills. The two-group solution filters out some of these task early in the process, so that there is a smaller chance that the skills of more skilled workers in the second group are wasted on these easier tasks. In particular, as we proceed through groups, tasks that would be too easy (meaning that they can be done by less skilled workers) are completed, leaving more difficult tasks to more skilled workers. It is also worth mentioning that the optimal two group solution does not use any worker whose skill is less than 0.25. This is because the assignment process of tasks to workers within a group is random, and so it is possible to decrease a group's output by adding unskilled labor to a group. In the example, the size of the first group of workers is equal to the size of the set of tasks, and so adding less skilled workers to that group can only reduce output. Finally, note also that the optimal two-group solution is contiguous; once we start the first group at a certain skill, every worker whose skill is above that skill participates. As we will show, all of these properties hold for all optimal sequential matchings.

We now turn our attention to formalizing the above intuition. The following elementary facts will be useful at several distinct points in our analysis.

Fact 1. (Probability of successful completion) Let F_{d_T} and F_{s_W} be the distributions of difficulties and skills in a group of tasks T and a group of workers W, respectively, and consider matching $m(W)$. The probability that a randomly-chosen task is successfully completed in m is given by $\pi(m)$ and can be written as

$$\pi(m) = \int_{d_1}^{d_2} (1 - F_{s_W}(\delta)) f_{d_T}(\delta) d\delta,$$

where $[d_1, d_2]$ is the domain of difficulties in T and f_{d_T} is the density of difficulties.

The probability of successful completion is simply the probability that a worker in W is matched with a task in T such that the difficulty of the task is within the worker's skill. A one-shot matching m of tasks to workers results in a partition of tasks into three regions, defined as follows.

Fact 2. (Task regions) Consider matching $m(W)$ and assume for simplicity that $\lambda(W) = \lambda(T)$. Let $s_l = \min\{s_i | i \in W\}$ and $s_h = \max\{s_i | i \in W\}$. We can then divide T into three regions:

R1 These are the tasks that can be done by every worker in W. The probability that a task is in R1 is equal to $F_{d_T}(s_l)$, and this region has measure $\lambda(T)F_{d_T}(s_l)$.

R2 These are the tasks that cannot be completed by any worker in W. The probability that a task is in R2 is equal to $1 - F_{d_T}(s_h)$, and this region has measure $\lambda(T)(1 - F_{d_T}(s_h))$.

R3 These are the tasks that can be done by some, but not all, workers in W, and their difficulties lie in (s_l, s_h). The probability that a task is in this region is equal to $F_{d_T}(s_h) - F_{d_T}(s_l)$, and this region has measure $\lambda(T)(F_{d_T}(s_h) - F_{d_T}(s_l))$.

For two random variables A and B, we write $A \succ B$ to indicate that A first-order stochastically dominates B, i.e. $A \succ B$ implies that $F_A(x) \leq F_B(x)$ for all x, with strict inequality for some x.

We also define a notion of dominance for worker groups. Recall that the set of workers is arranged in ascending order of skill.

Definition 3. (Group dominance) A group of workers W_1 dominates group W_2 if $\min\{s_i : i \in W_1\} > \max\{s_j : j \in W_2\}$. We write this as $W_1 \succ W_2$.

In order to arrive at the characterization in the beginning of this section, we need to understand how the output of a matching depends on the composition of workers and tasks that this matching comprises. Lemmas A.1, A.2, A.3, and Corollary A.1 in the Appendix collect useful results about the relationship between output and various worker group configurations. These results help us develop the two central lemmas used in the proof of Theorem 1 and for the analysis of the dynamic programming formulation in the next section. The first of these relates the pre- and post- matching difficulty distributions.

Lemma 1. *(Stochastic Dominance) Let $m(W, T)$ be a matching and T^c be the set of tasks completed in m. Let $T' = T \setminus T^c$ and assume that $T' \neq \emptyset$, then $d_{T'} \succ d_T$.*

Lemma 1 states that the distribution of difficulty in T' is skewed more towards difficult tasks than it is in T, and therefore the chances of selecting a difficult task from T' is higher than it is in T. We can gain an understanding for why that is the case by examining the task regions and how they change after a matching. Note that the task regions in Fact 2 are arranged in order of difficulty. Tasks in $R1$ are easier than tasks in $R2$, which in turn are easier than tasks in $R3$. As tasks go through a matching, the sizes of these regions shrink at different rates, with the size of $R1$ shrinking the most, followed by $R2$, and with the size of $R3$ remaining constant. This means that the tasks that come out on the other side of the matching have a higher proportion of $R3$ tasks (as well as the more difficult tasks in $R2$) than before the matching, and therefore the overall distribution becomes more difficult. This remains true regardless of the skills of the workers involved in the matching, with the exception of the degenerate case where any worker can finish any available task and thus the resulting difficulty distribution is the same as the distribution before the matching (if $\lambda(T) > \lambda(W)$) or there are no tasks remaining (if $\lambda(T) \leq \lambda(W)$).

The second lemma examines what happens when we rearrange workers.[13]

Lemma 2. *(Swapping Lemma) Let W_1 and W_2 be two arbitrary groups where neither $W_1 \succ W_2$ nor $W_2 \succ W_1$ and denote by μ the matching $\big(m_1(W_1, T), m(W_2, T'_{m_1})\big)$. There exists a rearrangement of workers in $W_1 \cup W_2$ into two groups $\bar{W}_1$ and $\bar{W}_2$ such that $\bar{W}_2 \succ \bar{W}_1$, $\bar{\mu} = \big(m_1(\bar{W}_1, T), m(\bar{W}_2, T'_{m_1})\big)$, and $\theta(\bar{\mu}) \geq \theta(\mu)$.*

This Swapping Lemma implies that the output of a matching that is obtained from two groups of workers of arbitrary composition can be improved upon by swapping some workers between the two groups to achieve a rearrangement where the workers in the second group all have higher skills than those in the first. The proof of this result uses several intermediate results that are collected in the Appendix and are informally summarized in the following discussion. We first show in Lemma A.4 that for two groups with one group dominating the other, it is always best to arrange these groups in a hierarchical fashion, with the less-skilled group first. The reasoning for this is that since task assignment is random, when the dominating group comes first, it finishes some tasks that the dominated group could have finished. And yet this also implies that some of the tasks that would have been completed by the dominating group but are too difficult for dominated group will remain unfinished. Having the dominated group attempt tasks first would filter out some of the easier tasks that both can finish, ensure a better utilization of the skills of the dominating group, and achieve a greater output overall.

It is important to point out that while the combined output of two groups increases when they are arranged in a hierarchical fashion, the output of the dominant group can only decrease, since by Lemma 1, it now faces a more difficult distribution than the one it would have faced if it came first in the ordering. The output of the dominant group is in fact lower when the skills in the dominated group are higher. This is because more tasks that the dominant group can finish are completed before they reach that group. This fact is formalized in Lemma A.5 and will play a particularly important role in the implementation problem because workers always prefer to be in an earlier group regardless of their skills, as these groups face easier difficulty distributions, and part of the implementation solution

[13] Related results are present and play a similar role in the literature on knowledge hierarchies, e.g., in Garicano (2000).

is to adjust the workers' incentives in order to re-align their goals with those of the employer.

The next step in the proof of Lemma 2 is to show that any improvement over a hierarchical arrangement has to maintain a hierarchical structure. This is shown in Lemma A.6. We then use this result to show, by means of a contradiction argument, that the optimal arrangement has to be hierarchical. The argument is that workers in a non-hierarchical matching can be rearranged into a hierarchy, and we then use Lemma A.6 to show that any improvement over this hierarchical matching must lead to a hierarchy, and not to the original non-hierarchical configuration, and therefore the initial configuration cannot be optimal.

3.1 Dynamic Programming Formulation

Our analysis of the structure of the optimal matching relies on a dynamic programming formulation for the problem. The state and action space, along with the transition function, can be given as:

- **State** The state at stage κ is written as $(\mathbb{W}_\kappa, T_\kappa)$, where $\mathbb{W}_\kappa$ is the set of remaining workers and T_κ are the tasks remaining at the beginning of κ.
- **Action** At stage κ, the decision or action is what group $W_\kappa \subseteq \mathbb{W}_\kappa$ to assign tasks to.
- **Transition** The transition function is derived from the outcome of matching $m_\kappa(W_\kappa, T_\kappa)$. The new state is given by $(\mathbb{W}_{\kappa+1}, T_{\kappa+1})$, where $\mathbb{W}_{\kappa+1} = \mathbb{W}_\kappa \setminus W_\kappa$ and $T_{\kappa+1} = T_\kappa \setminus T_\kappa^c$.

Recall that the number of stages (or equivalently, groups), k, is an input to the problem. The value function at stage κ satisfies

$$V_\kappa = \max_{W_\kappa \subseteq \mathbb{W}_\kappa} \theta\big(m(W_\kappa, T_\kappa)\big) + V_{\kappa+1}(\mathbb{W}_{\kappa+1}, T_{\kappa+1})$$

with $V_{k+1} = 0$. Let us examine the solution to the formulation above when $k = 1$. In this case, the problem is equivalent to searching for the best possible one-shot matching under the informational constraints of the problem. Not surprisingly, in that scenario the employer chooses the top measure of workers to participate in the matching. How would the optimal grouping change for the case $k = 2$?

Before answering this question, it would be useful to point out that having more groups is not always better. We will have more to say about this in the next section, but the intuition for two groups is the following: as the proportion of difficult tasks increases, it becomes more likely that a worker will pick a task that he cannot finish. In a one-shot setting, a difficult task may lead to wasting the efforts of at most one worker, but as more groups are introduced, this task has the potential to repeatedly 'defeat' other workers as well, as it gets carried through from one group to the next. These workers could have been better utilized in performing other tasks, which is what would happen in a one-shot setting. Thus breaking up a group may lead to an overall decrease in output. The next section discusses when having more groups is profitable, and assuming that to be the case, the solution to $k = 2$ follows directly from the Swapping Lemma, giving the following proposition, which we then use as the basis for our inductive proof of Theorem 1.

Proposition 1. *There exists an optimal hierarchical matching for $k = 2$.*

Using the results of this section and with induction on the number of groups, we arrive at a proof of Theorem 1. Lemma 1 establishes that later groups (groups with higher indices) face more difficult tasks regardless of the composition of earlier groups. Therefore, by Lemma A.1, workers with lower skills would struggle if they are placed in latter groups but have a better chance at completing tasks if they are in earlier ones. As an extreme example, consider the first group W_1 and let it contain a unit measure of the most skilled workers available, then any other group that follows W_1 has to have output equal to zero, since the workers in these groups have to tackle tasks at which more skilled workers have failed. Conversely, by placing less skilled workers in earlier groups, the (easy) tasks that they finish will not get in the way of more skilled workers later on. Instead, these workers will be assigned a bigger share of tasks that are commensurate with their skills, leading to an overall increase in output.

The optimal matching, which as already noted is a hierarchy, satisfies the following properties.

a) Necessary condition for optimality: at the beginning of round $i < k$, the number of workers left should be strictly higher than the measure of remaining tasks. At $i = k$, the measure of leftover tasks should be at least equal to the measure of workers remaining.

b) The optimal groups are contiguous. Let $s_1 < s_2$. If workers with skills less than s_1 are assigned tasks and workers with skills higher than s_2 are assigned tasks, then workers in (s_1, s_2) should also be assigned tasks.

c) Depending on the desired number of groups, the optimal solution may include a threshold for skill $\bar{s}$ below which no worker is chosen to participate (in Example 1, $\bar{s} = 0.25$).

3.2 Examples and Comparative Statics

Beside incentive issues, which we discuss in Section 4, the solution to the dynamic programming formulation further highlights the differences between our model and the existing work on assignment of heterogeneous workers to tasks in the models of knowledge hierarchies. In particular, the works by Garicano and Beggs, which were discussed in the Introduction, predict that the optimal hierarchy always has a pyramidal structure, with successive layers getting smaller and smaller in size. Moreover, it is possible that the optimal hierarchy has an infinite number of levels. Both of these are not true in our model of crowd innovation. First, successive groups can fluctuate in size, implying that the optimal hierarchy may not look like a pyramid. Second, not only is the optimal number of levels in our hierarchy always finite, but a hierarchy with more layers may even lead to a *strictly lower* output. Third, the optimal hierarchies in our model can display some counter-intuitive behavior. For example, by increasing the difficulties in the task pool, the optimal number of levels in the hierarchy can go down, instead of up.

Some of these differences are a consequence of the fact that Garicano's framework considers an "infinite pool of tasks" which need to be produced or processed by workers at the bottom layer of

the hierarchy, who can then ask for help from others if they are unable to perform these tasks. As a result, the optimal allocation organizes workers into one (bottom) layer of producers and multiple layers of problem solvers. Each worker in the bottom "producer" layer "generates" a task, and only sends this task up to the problem solvers if they cannot complete it themselves. Consequently, tasks at each level of the hierarchy are those that were not solved at the previous level, naturally implying a pyramid-like structure. The presence of the fixed set of tasks that can be tackled by any worker at any stage implies that optimal matching may not look like a pyramid. Additionally, the assumption that skill acquisition is endogenous in the baseline knowledge hierarchy models ensures that there is no mismatch between task difficulties and worker skills. In contrast, such mismatch is at the heart of our model and is the reason why more layers can lead to strictly lower output — because they expose more workers to tasks that they may be unlikely to complete, thus precluding them from using their skills for other tasks.

We now present a series of examples to illustrate the main ideas. To communicate these in the clearest possible fashion, we focus on a series of examples where both tasks and workers come with three levels of difficulties or skills, **L**ow, **M**edium, and **H**ard. We use the notation of, say, $\{\frac{1}{4}L, \frac{3}{4}M, \frac{1}{4}H\}$ to denote that a quarter measure of tasks/workers are of Low type, three quarter measures of Medium type, and a quarter measure of High type. All examples follow our assumptions about $\lambda(\mathcal{T}) = 1$ and $\lambda(\mathcal{W}) > \lambda(\mathcal{T})$. We start with two examples. The first illustrates a case when the solution follows the standard pyramidal hierarchy, and the second has a solution where the hierarchy is a reverse-pyramid.

Example 2. (Pyramidal Hierarchy) Consider $\mathcal{T} = \{\frac{1}{4}L, \frac{1}{2}M, \frac{1}{4}H\}$ and $\mathcal{W} = \{\frac{1}{4}L, \frac{3}{4}M, \frac{1}{4}H\}$. The optimal hierarchy for $k = 2$ has $W_1 = \{\frac{1}{4}L, \frac{3}{4}M\}$, and $W_2 = \frac{1}{4}H$. It can be verified that under this arrangement $\frac{7}{8}$ of the tasks are completed successfully. Furthermore, the second group of workers is of size smaller than the previous one, giving the familiar pyramidal structure.

Example 3. (Non-pyramidal Hierarchy) Consider the same setting as Example 2 but with a distribution of workers that shifts the mass from high-skilled to low-skilled workers, so that $\mathcal{W} = \{\frac{1}{2}L, \frac{3}{4}M\}$. The optimal hierarchy for $k = 2$ now has $W_1 = \{\frac{1}{2}L\}$ and $W_2 = \{\frac{3}{4}M\}$, with ~ 0.66 of the tasks finished. Notice that the first group is smaller than the second and a pyramidal structure is no longer maintained.

Observe how we go from a pyramidal to a non-pyramidal structure in the examples above, even though the differences between the two examples are minimal. To understand what gives rise to this difference, note that the hierarchical matching process always attempts to balance two effects: (a) by putting skilled workers in early groups, part of their output is wasted on easy tasks, and (b) by putting workers in later groups, their output can be compromised by receiving tasks that they are unequipped to deal with (cf. Lemma 1). In Example 2, the distributions of difficulties and skills were such that it was best to protect the high-skilled workers from as many low- and medium-difficulty tasks as possible, even if this meant compromising some of the medium-skilled workers to low-difficulty tasks, and hence the low- and medium-skilled workers participated in the first group. In Example

3, it was best to protect the medium-skilled workers from as many low-skilled tasks as possible, and this gave rise to the configuration in the example.

It is tempting to extrapolate these examples in a natural way in order to arrive at an optimal policy for how groups are divided, but this process is more subtle than it first appears. The next example illustrates this and provides an entry point to our discussion of the optimal number of groups.

Example 4. Consider Example 2 but assume now that we want to find the optimal assignment for the case $k = 3$. It seems plausible to have three groups as follows: $W_1 = \{\frac{1}{4}L\}, W_2 = \{\frac{3}{4}M\}, W_3 = \{\frac{1}{4}H\}$. The idea, based on the immediately preceding discussion, is that by keeping each of these skills in its separate group, then we can achieve maximum utilization for each skill level. This turns out to not be the case, and in fact, the optimal grouping is still exactly the same as in Example 2. In particular, even though we can use one more group, by doing so we actually obtain a worse outcome than if we had stuck with the $k = 2$ groups solution.

Example 4 can be generalized to give the following result.

Proposition 2. *Let μ^i be the optimal hierarchical matching that uses i groups and let μ^j be a hierarchical matching that uses the exact same set of workers as μ^i but divides them into $j > i$ groups, then $\theta(\mu^i) > \theta(\mu^j)$.*

This result is in contrast to Garicano's work, where it is possible that more layers always increase output and the optimal number of layers can be infinite. Here, without any costs of communication, more layers can strictly reduce output. In fact, one cannot benefit from adding an extra layer of workers to an optimal solution of k groups unless more workers are introduced, i.e., more groups require more entry. This immediately implies that for a fixed-size population of workers the number of layers in an optimal solution is always bounded. The condition that having more groups requires more entry is necessary but not sufficient: the additional (and less-skilled) workers should possess skills that will allow them to reduce the load on more skilled workers later on. The result above is a consequence of Lemma 1 and the fact that tasks cannot be differentiated based on their difficulties.

Following this line of reasoning, we can investigate the optimal number of groups given the supply of workers, where optimal number means *the minimum number of groups above which there is no increase in output*. In general, it is not easy to analytically characterize the optimal number of groups, though a straightforward computational approach to answering this question would utilize Proposition 2 to figure out the optimal solutions to $k = 1, k = 2, ...$, with output monotonically increasing in k until it drops or stays constant for the case $k = k^* + 1$, indicating that k^* is the optimal number of groups in the hierarchy.

The next example shows that the effect of increasing the difficulty of the task distribution on the optimal number of groups is ambiguous. Again, this contrasts with existing models where higher difficulties in the task pool call for more groups.

Example 5. Let $\mathcal{T} = \{1L\}$ and $\mathcal{W} = \{\frac{3}{4}L, \frac{1}{2}M\}$. Obviously, any group $W \subset \mathcal{W}$ with $\lambda(W) = 1$ will finish all the tasks, and $k = 1$ is optimal. Now increase the difficulty in $\mathcal{T}$ so that $\mathcal{T} = \{\frac{3}{4}L, \frac{1}{4}M\}$, then output for the case $k = 1$ is strictly less than 1, and can be improved on by $k = 2$ and setting

$W_1 = \{\frac{2}{3}L, \frac{1}{3}M\}, W_2 = \{\frac{1}{6}M\}$, which is a configuration that again finishes all the available tasks. Now make the tasks in $\mathcal{T}$ even more difficult by letting $\mathcal{T} = \{\epsilon L, \frac{1}{4}M, (1 - \frac{1}{4} - \epsilon)H\}$, where ϵ is the amount of low-difficulty tasks in the task pool. For $0 < \epsilon <\approx 0.245$, the optimal k is again equal to 1, i.e., the optimal number of groups decreases even though the difficulties in the task pool increases, and the optimal group is given by $\{\frac{1}{2}L, \frac{1}{2}M\}$. However, for $0.245 < \epsilon < \frac{3}{4}$, the optimal k goes back to 2, with the optimal groups being $W_1 = \{\frac{3}{4}L\}$ and $W_2 = \{\frac{1}{2}M\}$.

In addition to establishing that as tasks become more difficult, the optimal number of groups need not increase, and can in fact decrease, the previous example provides an important insight into the workings of the assignment process. In particular, how the number of groups is governed by a threshold on the ratio between the low-difficulty and high-difficulty tasks in the task pool. When there were not that many low-difficulty tasks, it was not beneficial to separate the low and medium-skilled workers into two groups, because only a small proportion of the medium-skilled workers were exposed to the easy tasks. When low-difficulty tasks had more substantial presence in the task pool, separation of low- and medium- skilled workers is beneficial because more medium-skilled workers get more medium-difficulty tasks ands therefore are more appropriately utilized.

4 The Implementation Problem

We return to our full model where worker skills are private information and workers themselves decide whether and when to participate. We will show that with a simple pricing mechanism, the employer can nevertheless achieve the first-best allocation characterized in the previous section.

In order to implement the first-best allocation from the assignment problem, we utilize a mechanism that charges a participation or entry fee q for workers and a tiered payment system for tasks. A task finished in group $i, i = 1, ..., k$ pays an amount p_i, with $p_i > p_j$ for $i > j$. Fix a configuration of groups and let the probability that a worker with skill s finishes a task when he is a member of group W_i be given by $\psi(s, W_i)$. The expected payment that this worker receives is equal to $p_i\psi(s, W_i)$. Groups are in equilibrium when no worker wants to change the group he is in. Denote by group W_0 those workers who do not participate, then a formal definition for an equilibrium is the following

Definition 4. (Group Equilibrium) Groups $W_1, ..., W_k$ are in equilibrium if for any skill s, a worker with skill s in $W_i, i = 1, ..., k$ receives expected payment $p_i\psi(s, W_i) \geq p_j\psi(s, W_j)$ for all $j \neq i$.

Thus for a worker to voluntarily choose to belong to group W_i, we must have $p_i\psi(s, W_i) \geq p_j\psi(s, W_j)$ for all $j \neq i$. Lemma A.8 shows that the probability of finishing a task is non-increasing in the index of a group.

The following lemma is critical to the implementation of the optimal assignment.

Lemma 3. *(Comparative Advantage) Define* $\phi(s) = \frac{\psi(s,W_i)}{\psi(s,W_j)}$ *for* $j > i$. *Then* $\phi(s)$ *is decreasing in* s.

Comparative advantage follows from the monotone relationship between skills and difficulties, where a worker with higher skill can finish any task that a worker with lower skill can. Lemma

3 states that more skilled workers perform *relatively* better on more difficult tasks than less skilled workers do. In the context of our model, this means that for two workers in group i with skills s_1 and s_2 and with $s_1 < s_2$, the drop in probability of success for the worker with skill s_1 upon moving to a higher group will be larger compared to the drop for the worker with skill s_2 if he makes the same move.

We now give the main result of this section.

Theorem 2. *Let $W_1^*, ..., W_k^*$ be an optimal k-level hierarchical assignment. There exists a participation fee q and payments $p_1 < p_2 < ...p_{k-1} < p_k$ that induce an equilibrium in which a worker chooses to be in group W_i^* if and only if he is selected for that group in the optimal allocation.*

Let $s_i = \{\min s | s \in W_i\}$. The proof of Theorem 2 shows that the values for the entry fee q and payments $p_1, ...p_k$ can be obtained by solving the following simple system of equations.

$$\begin{aligned} p_1\psi(s_1, W_1) &= q \\ p_i\psi(s_{i+1}, W_i) &= p_{i+1}\psi(s_{i+1}, W_{i+1}) \quad i = 1, 2., , , k-1 \end{aligned}$$

The first equation above indicates that the least skilled worker who participates only breaks even in expectation. Because $\psi(s, W_1)$ is monotonically increasing in s, any worker whose skill is less than s_1 will have negative expected payoff and therefore will not participate. Similarly, by recalling that groups are contiguous, the second equation gives a condition that makes workers at the boundary between successive groups indifferent. Again, from the monotonicity of $\psi(s, W_i)$ (increasing in s and decreasing in i), we find that an increase in payment p_{i+1} over payment p_i by a factor $\frac{\psi(s_{i+1}, W_i)}{\psi(s_{i+1}, W_{i+1})}$ will provide less payoffs to a worker with skill less than s who decides to move to group $i+1$, and a worker whose skill is higher than s and decides to move to group i. This is because, upon a move to a group with higher index, the drop in probability of successful completion for a less skilled worker is not sufficiently compensated by the increase in payment, and the opposite is true for a more skilled worker who contemplates moving in the opposite direction.[14]

Example 6. We revisit Example 1 to show how the optimal two group assignment can be implemented. We want to find q, p_1, and p_2 that will provide the correct incentives to workers to participate in the groups they were allocated to in that example. Let us denote by the dummy group W_0 the

[14] It is also useful to note that Theorem 2 shows that the first-best characterized in the previous section can be decentralized as an equilibrium. It does not establish that this is the unique equilibrium. Multiplicity can arise for the following reason: if the most skilled workers were all to select to participate in one of the earlier groups, say the first group, then the set of tasks that remain uncompleted after the first round may become more difficult than in the optimal matching because instead of the lowest skill active workers, now some of the highest skill workers are failing to complete these tasks. Given this, other high skill workers may no longer find it optimal to wait for later matchings at the prices determined in Theorem 2.

This sort of multiplicity is unlikely to be endemic because if the most skilled workers want to be in the first group, then all workers want to be in the first group, and the excess supply of workers in the first group will reduce the return to each significantly. This also suggests one way of dealing with this problem: if there is sufficient excess supply of workers at a certain stage of matching, then reduce the set of tasks assigned at that stage until the expected return to any worker is very low.

group of workers who choose not to participate. Recall that the optimal groups were (in terms of skill) $W_0 = (0, 0.25)$, $W_1 = (0.25, 0.75)$, and $W_2 = (0.75, 1)$. By setting $q = 0.25, p_1 = 1$, and $p_2 = 1.5$, we obtain the desired hierarchy. To see this, let us examine a deviation of a worker of skill s in W_0. This worker makes 0 in that group, and by deviating to join group W_1, he obtains an expected utility of $p_1\psi(s, W_1) - q$. For this example, $\psi(s, W_1) = s$, and so the expected payoff from deviation is equal to $1s - 0.25$, which is less than zero for $s < 0.25$. Similarly, a worker of skill s in W_1 makes an expected profit of $1s - 0.25$, which is higher than 0 but also higher than what he would make had he decided to join W_2 instead. In that case he will be making $p_2\psi(s, W_2) - q = 1.5(s - 0.25) - 0.25$, which is less than $1s - 0.25$ for $s \in (0.25, 0.75)$. Finally, a worker of skill s in W_2 has $\psi(s, W_2) = \frac{1}{2} + \frac{s-0.75}{0.5}$ for an expected payoff of $1.5(\frac{1}{2} + \frac{s-0.75}{0.5}) - 0.25$. This is higher than what a worker whose skill is in $(0.75, 1)$ stands to gain by moving to W_1.

In summary, a simple mechanism that implements the optimal assignment announces the entry fee q and the prices p_i for each group. A worker chooses either not to participate at all or to participate in exactly one of the groups. A worker participating in group i is not allowed to participate in group $j > i$.[15]

5 Extensions

We now present five extensions that explore the role of various assumptions in our analysis so far. The first extension highlights the fact that our main result on replicating the optimal assignment of tasks to workers through market matching in crowd innovation depends on the exact information available to firms, and in particular, on information about what unsuccessful workers have worked on. If firms can monitor this, then the optimal hierarchy cannot be achieved with a simple pricing mechanism like the one we have utilized so far (though more complicated pricing and information-sharing mechanisms can achieve it). Second, we introduce a type of task, which we refer to as an "unstructured task," that adds some murkiness to the notion of what a skilled worker is. Performing these tasks might require skill, luck, or a combination of both, and so with some probability, these potentially valuable tasks can be performed by less skilled workers. We then show that in the presence of such tasks the optimal structure of the organization need not be a hierarchy any more. This highlights the role of the monotone relationship between worker skills and tasks difficulties in our results. Finally, we discuss a couple of extensions that we left out from our analysis but whose inclusion has no effect on our main results.

5.1 Semi-Observable Skills

The analysis so far has focused on the case where the employer cannot assign specific tasks to specific workers (cf. Section 2.2). We now discuss what happens when this is not the case, and compare this

[15]The solution also makes it clear why such a worker would in fact never wish to participate for later groups given the optimal pricing scheme.

scenario with our earlier results. We refer to this setting as one with "semi-observable skills"— the employer cannot ex ante observe worker skills but after assignment can verify what the skill level of the worker was. The fact that the employer can observe the workers' skills after they fail to complete a task provides extra information about unfinished tasks relative to the case in which the employer only knows the scent of workers who may have attempted the task. This extra information can only help in the (re)assignment process. Nevertheless, the hierarchical structure of the solution is still maintained because its main driver — attempting to best-utilize low-skilled workers and protect skilled labor from easy tasks — is still in effect. In particular, Lemma 1, which states that the difficulty distribution does not become easier still applies, and the previous reasoning can be used to show that it is still optimal to assign tasks to less-skilled workers first. The possibility that the optimal hierarchy takes a non-pyramidal shape still exists, as evidenced by Example 3, which is unaffected by this change.

A halfway case between unobservable and semi-observable tasks is when the employer can partition the set of tasks and workers at a certain stage into subgroups and ensure that matching is within subgroups. This is still gives additional information relative to our benchmark but less information than the semi-observable case where the employer knows the exact skill of the worker who failed to complete a task. (The difficulty is in ensuring that matching is within subgroups, something that employers may not be able to achieve in a virtual marketplace). In this setting, the semi-observable case can be approximated by breaking down a group into several smaller groups and keeping track of which tasks were not completed by which group. Revisiting Example 1 clarifies both how this approximation might work and why semi-observable skills improve output.

Example 7. Consider Example 1 and break group W_1 into two groups, W_{1a} and W_{1b}, such that W_{1a} has workers whose skills are in $(0.25, 0.5)$ and W_{1b} has workers whose skills are in $(0.5, 0.75)$. Break W_2 into two groups W_{2a} with skills in $(0.75, 0.90625)$ and W_{2b} with skills in $(0.90625, 1)$. Let the tasks remaining after matching $m(W_{1i})$ be denoted by T'_{1i}, then the breakdown of tasks in T'_{1a} is as follows: $\frac{1}{16}$ tasks are in $(0.25, 0.5)$, $\frac{1}{8}$ tasks are in $(0.5, 0.75)$, and $\frac{1}{8}$ tasks are in $(0.75, 1)$. Tasks in T'_{1b} are as follows: $\frac{1}{16}$ tasks are in $(0.5, 0.75)$ and $\frac{1}{8}$ tasks are in $(0.75, 1)$. Now let $m(W_{2a}) = T'_{1a}$ and $m(W_{2b}) = T'_{1b}$ (note that all of these sets have the same measure). The total output of these four groups is ≈ 0.92, improving on the output in Example 1 while using the same set of workers.

The reason we were able to improve output in the previous example is the newfound ability to have more targeted matchings, which is facilitated by access to more information about unfinished tasks. Here, we were able to assign the tasks T'_{1a} that had an easier distribution to the less skilled workers in W_2 (those workers in W_{2a}). As the granularity of these groups become finer, we approach the case of semi-observable tasks. This breakdown of groups is still implementable using similar ascending pricing schemes like the one we introduced in Section 4.

5.2 Unstructured Tasks

Our results follow from the type of tasks we have considered. The completion of these tasks requires a certain set of skills, and we assume that these skills are cumulative, i.e., a worker whose skill allows

him to complete task 1 is also able to complete task 2 if $d_1 > d_2$. We can thus think of these as "*structured tasks*". Suppose now that in addition to these tasks there is another type of task whose completion is independent of a worker's skill. Some of the early success stories in crowd innovation have an element where a task that has eluded experts was completed by random workers through chance or educated guess work. Often, these tasks are characterized by the fact that the actual skill set required to finish them is not well-understood. We designate these tasks as *unstructured*, and model them as a kind of task that can be finished by any worker with some small probability. If we now consider a task pool that contains both structured and unstructured tasks, then there are two possible scenarios: (a) either the type of task is an observable attribute, so that we can determine the type of the task by simply examining it, or (b) we cannot distinguish between the two types. The solution to the first scenario is obvious, and simply involves separating the tasks into two separate pools, structured and unstructured, and solving two different problems in parallel. The structured tasks are dealt with in a hierarchical fashion, and the unstructured tasks are simply given to low-skilled workers who were not selected for the hierarchical match. The second scenario is more challenging, and does in fact break the result that the optimal arrangement for maximizing task completion is a hierarchy, as the following example illustrates.

Example 8. Consider $\mathcal{T} = \{\frac{1}{3}M, \frac{1}{3}H, \frac{1}{3}U\}$, where U denotes unstructured tasks. Let $\mathcal{W} = \{\frac{1}{3}L, 1M, \frac{1}{3}H\}$ be a set of workers and let the probability of completing an unstructured task be strictly less than one. Assume that $k = 2$ and that it is not possible to observe whether a task in unstructured or not, then it is optimal to set $W_1 = \{1M\}$ and $W_2 = \{\frac{1}{3}L, \frac{1}{3}H\}$. The reason is that the productivity of L workers on unstructured tasks is affected by them having to deal with the M tasks in the task pool, as some of the L workers will be lost to some M tasks that they cannot finish, and hence their output on U tasks decreases. Putting the M workers upfront ensures that all M tasks are removed from the task pool, and therefore these tasks would not constitute an undue burden on either L or H workers.

Of course, with the allowance of a larger number of groups and a larger supply of L workers, the optimal assignment in the example above may resort back to being a hierarchy. Thus, whether a solution is a hierarchy or not in the unobservable case is a function of several factors: the number of groups, the supply and type of workers, and the distribution of tasks and task types in the task pool. This is in contrast to any of the previous variations, where it was always optimal to have a hierarchical assignment regardless of these factors.

5.3 Task Reassignment

As mentioned in the discussion preceding Definition 1, a key assumption of our analysis has been that a worker can pick up and work on only one task, regardless of whether he or she was able to finish that task or not. When it comes to innovation tasks, it is not difficult to imagine settings where the time until a worker finishes a task is fairly long, making it reasonable that this worker can work on at most a single task during the process. However, workers who were unable to complete their tasks might be given another task to work on, a possibility we have not considered in our analysis, but one

that does not alter the hierarchical nature of the solution. This is a simple consequence of the fact that the new task that will be assigned to that worker will not be a task that was not finished by someone whose skill is higher than the worker in question, but rather a task that was unsuccessfully attempted by someone with lesser skill, and hence the bottom-up approach to task assignment is preserved.

5.4 Discounting

Discounting, where tasks that are finished later in the process count for less than those finished earlier, has no effect on any of our results. Depending on the discount factor, either a hierarchy is still an optimal solution, or the solution is simply a single group that comprises the most skilled workers. To see this, consider the case $k = 2$ and assume that under certain discounting the optimal assignment has two groups of workers w' and w'', both in W_1 and a group $w \in W_2$ such that $w' \succ w$ and $w \succ w''$, i.e. the assignment is not a hierarchy. Recall that the reason a hierarchy emerges as an optimal assignment is that later groups, instead of working on easy tasks, are more focused on working on tasks that are commensurate with their skills. But if the earlier group in an optimal solution contains some workers who are more skilled than those in the second group, then this indicates that discounting is high enough that it is not worth the improvement we get from placing those high-skilled workers in the second group, and in that case workers who are even less skilled are better utilized by including them in the first group as well (potentially displacing some even less-skilled workers in that group), especially since some of the tasks that they are meant to finish will be finished by the more skilled group in W_1 anyway.

5.5 Probabilistic Completion of Tasks

The model we presented assumes that a worker finishes a task if their skill is at least equal to the difficulty of the task. A more reasonable assumption might be that workers do not always finish tasks that are below their skills with certainty, but rather, that more skilled workers have a higher chance of finishing a task compared to their less-skilled peers. This means that for workers i and j with $s_i > s_j$ and a task t, we have $Pr(i \text{ finishes } t) > Pr(j \text{ finishes } t)$ for all t with $d_t \leq s_j < s_i$. This change does not affect Lemmas 1 and 2, and our results carry through in that modified setting as well.

6 Conclusion

Crowd innovation involves the sourcing of innovation to a broad set of workers of unknown skills using a virtual marketplace. In these large and anonymous virtual markets, employers cannot assign workers specific tasks and do not know the workers' specific skills. In addition, the nature of the tasks to be performed often preclude specific information about how difficult the given task is and what type of workers will be able to complete them.

Our main conceptual contribution in this paper is that, these informational and coordination difficulties not withstanding, crowd innovation can be organized as if it was taking place within an

organization, but with some constraints resulting from the market allocation. In particular, the optimal assignment involves a sequential matching in which workers are sorted into different groups of increasing skill level, and are successively assigned to tasks with endogenously increasing difficulties (endogenously because the remaining tasks assigned to later groups are those that earlier groups were unable to complete). This application can be supported by a simple pricing mechanism in which prices — rewards —for tasks increase the longer these tasks remain uncompleted.

This optimal assignment is notable because of its ability to economize on the scarce factor in this marketplace — the time of skilled workers. The effort to economize on these skills is reflected in a range of interesting, and perhaps surprising, results of our framework. First, though the optimal matching is hierarchical, this hierarchy may not look like a pyramid — which contrasts with existing results in organizational economics and on related assignment problems. This happens because sometimes pyramids waste the skills of middle-skill workers as they expose them excessively to difficult tasks which they fail to perform. Second, for similar reasons, increasing the number of layers of the hierarchy may not be beneficial — it may even lead to lower output. This is because more layers will increase the number of workers that are exposed to the most difficult tasks which will waste the skills of many types of workers. However, too few levels of hierarchy is also not optimal, because these will waste the skills of the most skilled workers on easy tasks.

There are several other tools that can be used in practice for ensuring an efficient allocation of scarce resources. The first is an explicit tournament, where workers are rewarded for completing tasks before others. This has not been a problem in our formulation because there is no moral hazard or effort decision — in our framework, a worker can either complete or not complete a given task regardless of effort or other choices she makes, and thus it is always better to sequentially assign workers to tasks and not waste the time of a worker on a task that will already be successfully completed by another worker. Enriching our framework with an additional effort decision is an interesting area for theoretical research.

Second, in practice, the asymmetry of information between employers and potential workers can be ameliorated if workers are able to build a reputation. Such reputation building may even be possible in online markets as shown by the creative recent work of Pallais (2013). Nevertheless, at this point crowd innovation platforms lack both a decent reputation tracking system and any meaningful way to enforce liabilities. Moreover, recent experimental work by Dulleck et al. (2011) suggests that workers have no incentive to build reputations as long as they can avoid liabilities, which further complicates the efforts to build platforms that leverage reputation to improve the assignment of heterogeneous workers to tasks. A dynamic analysis of these issues and work on the design of better platforms are other important areas for research.

Finally, it is important to note that this area would benefit from more detailed empirical evidence on how workers make their decisions of what types of tasks to work on and how firms design incentives for crowdsourcing and innovation. Using experimental methods and different treatments for workers on various online platforms is a possible way of tackling these questions and constitutes yet another area for interesting future research.

Appendix

A Proofs

This section starts by presenting a few straightforward lemmas that are used as building blocks for the main results. Throughout, we will refer in different ways to the output of a sequential match $\theta(\mu)$ depending on context. Sometimes using the shorthand $\theta(W_1, W_2, ...)$ to indicate a sequential matching that first assigns tasks uniformly at random to group W_1 then to group W_2, etc. Also, recall that even though a matching m randomly assigns tasks from the pool of available tasks to workers, we will assign different subscripts when we refer to matchings in order to distinguish between them based on the worker and tasks groups they comprise. This makes it easier to refer to matching m_i and m_j instead of $m(W_i, T)$, $m(W_j, T)$, $m(W, T_k)$, and so on.

Lemma A.1. Let W be a group of workers, and consider matchings $m_1(W, T_1)$ and $m_2(W, T_2)$, where T_1 and T_2 are two task pools with $\lambda(T_1) = \lambda(T_2)$ and $d_{T_1} \succ d_{T_2}$, then $\theta(m_1) \leq \theta(m_2)$.

Proof: Let $f_W(s)$ be the density function for skills in W, with support over $[s_1, s_2]$, and write the outputs of m_1 and m_2 as

$$\theta(m_1) = \pi(m_1)\lambda(W) = \int_{s_1}^{s_2} F_{d_{T_1}}(s) f_W(s) ds \tag{2}$$

$$\theta(m_2) = \pi(m_2)\lambda(W) = \int_{s_1}^{s_2} F_{d_{T_2}}(s) f_W(s) ds \tag{3}$$

Since $d_{T_1} \succ d_{T_2}$ implies $F_{d_{T_1}}(s) \leq F_{d_{T_2}}(s)$ for all s and $F_{d_{T_1}}(s) < F_{d_{T_2}}(s)$ for some s, we have $(1) \leq (2)$ and hence $\theta(m_1) \leq \theta(m_2)$. ■

Lemma A.1 states that the output of the same group of workers cannot improve when tasks become more difficult. The next lemma complements this observation by showing that a group's output on the same tasks can only increase if some workers in the group are replaced by an equal number of more skilled workers.

Lemma A.2. Consider matchings $m_1(W, T)$ and $m_2(W', T)$, where $W' = (W \setminus w) \cup w'$, $w \subset W$ and $w' \cap W = \emptyset$, $w' \succ w$, and $\lambda(w) = \lambda(w')$, then $\theta(m) \leq \theta(m')$.

Proof: Denote by S the set of skills in $W \cup w'$. Because $w' \succ w$, there is a set D such that $F_{s_W}(d_T) = F_{s_{W'}}(d_T)$ for $d_T \in D$ and $F_{s_W}(d_T) > F_{s_{W'}}(d_T)$ for $d_T \in \bar{D}$, where $D \cap \bar{D} = \emptyset$ and $D \cup \bar{D} = S$. By writing down the expressions for $\pi(m_1)$ and $\pi(m_2)$, we get $\pi(m_1) = \int_{d_1}^{d_2}(1 - F_{s_W}(\delta)) f_{d_T}(\delta) d\delta$ and $\pi(m_2) = \int_{d_1}^{d_2}(1 - F_{s_{W'}}(\delta)) f_{d_T}(\delta) d\delta$, where (d_1, d_2) is the domain of difficulties in T. It is straightforward to see that, for any $\delta \in (d_1, d_2)$

$$\begin{aligned} F_{s_W}(\delta) &\geq F_{s_{W'}}(\delta) \\ \int_{d_1}^{d_2}(1 - F_{s_W}(\delta)) f_{d_T}(\delta) d\delta &\leq \int_{d_1}^{d_2}(1 - F_{s'_W}(\delta)) f_{d_T}(\delta) d\delta \\ \pi(m_1) &\leq \pi(m_2) \end{aligned}$$

Since $\theta(m_1) = \pi(m_1)\lambda(W)$ and $\theta(m_2) = \pi(m_2)\lambda(W')$ and $\lambda(W) = \lambda(W')$, the result follows. ■

Lemma A.3. Consider matchings $m(W,T)$, $m'(W',T)$, and $m''(W,T')$. Let w and t be a group of workers and a group of tasks such that $W \cap w = \emptyset$, $W' = W \cup w$, $t \subset T$, and $T' = T \setminus t$, then

a. If $\lambda(T) = \lambda(W)$ and w is such that $\max_{i \in w} s_i < \min_{j \in W} s_j$, then $\theta(m) \geq \theta(m')$.

b. If $\lambda(T) > \lambda(W)$ and w is such that $\lambda(w) = \lambda(T) - \lambda(W)$, then $\theta(m) \leq \theta(m')$.

c. If $\lambda(T) \leq \lambda(W)$, then $\theta(m'') = \theta(m) - \theta(m(W,t))$.

Proof: For part a, note that $W \succ w$ implies $s_W \succeq s_{W'}$, i.e. $F_{s_W} \leq F_{s_{W'}}$, and hence by the same arguments of Lemma A.2 we have $\pi(m) \geq \pi(m') \rightarrow \theta(m) \geq \theta(m')$. For part b, let $\bar{T} \subset T$ be the set of tasks assigned to W in m, then by the assumption that matchings are uniformly random, the exact same set of tasks will be assigned to $W \subset W'$ in m', making $\theta(m') \geq \theta(m)$. The same argument applies to part c with the bigger set being W instead of T. ■

Lemma A.3a states that if the number of workers in a group is equal to the number of tasks given to that group, then there is no reason to add any more workers whose skills are less than the skills of everyone else in the group. The reason is that this will cause a redistribution of tasks in a way that results in some tasks going to less-skilled workers instead of the workers they were originally assigned to, while the remaining tasks have their original assignment, and hence the output can only go down. A simple corollary of this claim is that no optimal assignment will assign a group of tasks to a group of workers such that the number of workers is more than the number of tasks.

Corollary A.1. In an optimal solution, no tasks T are assigned to a group W such that $\lambda(W) > \lambda(T)$.

In contrast to Lemma A.3a, Lemma A.3b says that extending the size of a group of workers to make it equal to the number of assigned tasks in a matching cannot reduce output. This is because adding more workers in this scenario does not affect the distribution of tasks in the original smaller-sized group. Instead, the effect is that tasks that were not originally assigned due to a mismatch in group size will now have extra workers to attempt them.

Finally, Lemma A.3c states that when the number of workers is at least equal to the number of tasks, removing some tasks from a matching does not affect the output of the group of workers on the remaining tasks.

Proof of Lemma 1: Assume for simplicity that $\lambda(W) = \lambda(T)$. We can divide T into three regions as in Fact 2. Let γ_i be the probability that a worker in W finishes a task in region i, so that $\gamma_1 = 1$ and $\gamma_2 = 0$. From Fact 1, γ_3 is given by

$$\gamma_3 = \int_{s_l}^{s_h} (1 - F_{s_W}(\delta)) f_{d_T}(\delta) d\delta$$

The set $T' = T \setminus T^c$ has two regions:

- Region A: This is the same as R2 in Definition 2, with measure $\lambda(T)(1 - F_{d_T}(s_h))$.

- Region B: This contains the uncompleted tasks from R3, and has measure $(1-\gamma_3)\lambda(T)(F_{d_T}(s_h)-F_{d_T}(s_l))$.

This implies that the relative measure of tasks in R2 has increased. Tasks with difficulties over s_h had a proportion of $1 - F_{d_T}(s_h)$ in T, but in T', the same tasks now have a proportion

$$\begin{aligned}\frac{\lambda(T)(1 - F_{d_T}(s_h))}{\lambda(T)(1 - F_{d_T}(s_h)) + (1 - \gamma_3)\lambda(T)(F_{d_T}(s_h) - F_{d_T}(s_l))} &= \frac{1 - F_{d_T}(s_h)}{1 - F_{d_T}(s_h) + (1 - \gamma_3)(F_{d_T}(s_h) - F_{d_T}(s_l))} \\ &= \frac{1 - F_{d_T}(s_h)}{1 - F_{d_T}(s_l) - \gamma_3(F_{d_T}(s_h) - F_{d_T}(s_l))} \\ &> 1 - F_{d_T}(s_h)\end{aligned}$$

To understand the distribution of the tasks in Region B, note that the probability $1-F_{s_W}(\delta)$ of finishing a task in R3 is decreasing in δ. Consider intervals $d_1 = (\delta_1, \delta_2)$, and $d_2 = (\delta_3, \delta_4)$, where d_1 and d_2 are both in T, with $\delta_3 > \delta_2$ and $\lambda(d_1) = \lambda(d_2)$. The probability of finishing a task in (δ_1, δ_2) is given by

$$\begin{aligned}\gamma &= \int_{\delta_1}^{\delta_2} (1 - F_{s_W}(\delta)) f_{d_T}(\delta) d\delta \\ &\geq \int_{\delta_3}^{\delta_4} (1 - F_{s_W}(\delta)) f_{d_T}(\delta) d\delta \\ &= \gamma'\end{aligned}$$

where γ' is the probability of finishing a task in (δ_3, δ_4). Thus the measure of tasks in d_1 decreases by $\gamma\lambda(d_1)$ while the measure of tasks in d_2 decreases by $\gamma'\lambda(d_2)$. For $i = 1, 2$, let $d_i' \subset T'$ be the tasks in d_i that were not finished in the matching. We have $\frac{\lambda(d_1')}{\lambda(d_2')} = \frac{\lambda(d_1)(1-\gamma)}{\lambda(d_2)(1-\gamma')} = \frac{(1-\gamma)}{(1-\gamma')} \leq 1$ and therefore the relative measure of more difficult tasks in R3 is nondecreasing in T' compared to T. Let j and j' be two randomly chosen tasks from T and T', respectively, then for any value of difficulty d, $Pr(d_j \leq d) \geq Pr(d_{j'} \leq d))$ and $Pr(d_j \leq d) > Pr(d_{j'} \leq d))$ for $d > s_h$, implying that $d_{T'} \succ d_T$. ■

The following lemma derives a relationship between group dominance, group ordering, and output.

Lemma A.4. Let T be a group of tasks and consider two groups of workers W_1 and W_2 with $\lambda(W_1) = \lambda(W_2) \leq \frac{\lambda(T)}{2}$ and such that $W_2 \succ W_1$. Let $\mu^A = (W_1, W_2)$ and $\mu^B = (W_2, W_1)$, then $\theta(\mu^B) \leq \theta(\mu^A)$.

Proof: Let $\lambda(T)$ be normalized to 1 and let $\lambda(W_1) = \lambda(W_2) = l \leq \frac{1}{2}$. Assume wlog that the least difficult task in T and the lowest skill worker in W_1 have value zero. Let $d_1 = \max\{s_i : i \in W_1\}$ and $d_2 = \max\{s_j : j \in W_2\}$, and divide the difficulties in T into intervals $(0, d_1), (d_1, d_2)$, and (d_2, d_{max}),

where d_{max} is the maximum difficulty in T. Let

$$F_{d_T}(d) = \begin{cases} x_1 & d = d_1 \\ x_2 & d = d_2 \\ 1 & d = d_{max} \end{cases}$$

Denote by $T_{(\underline{d},\bar{d})}$ the set of tasks in T whose difficulties lie in $(\underline{d},\bar{d})$ and define matchings $m(W_1, T_{(0,d_1)})$ and $m'(W_2, T_{(d_1,d_2)})$. Recall that the probabilities of finishing a task in m and m' are given by $\pi(m)$ and $\pi(m')$, respectively. Because $W_2 \succ W_1$, $Pr(s_j \geq d) = 1$ for all $j \in W_2$ and $d \in (0, d_1)$. Now define $\mu^A = \left(m_1^A(W_1, T), m_2^A(W_2, T'_{m_1^A})\right)$ and $\mu^B = \left(m_3^B(W_2, T), m^4(W_1, T'_{m_3^B})\right)$, where $T'_{m_1^A}$ and $T'_{m_1^B}$ are the tasks remaining after the respective matchings. We have $\theta(m_1^A) = \theta(m) = lx_1\pi(m)$ and $\lambda(T'_{m^1}) = 1 - lx_1\pi(m)$, with $d_{T'_{m_1^A}}$ distributed as

$$F_{d_{T'_{m_1^A}}}(d) = \begin{cases} \frac{x_1(1-l\pi(m))}{1-lx_1\pi(m)} & d = d_1 \\ \frac{x_2-lx_1\pi(m)}{1-lx_1\pi(m)} & d = d_2 \\ 1 & d = d_{max} \end{cases}$$

and hence $\theta(m_2^A) = l\left(\frac{x_1(1-l\pi(m))+\pi(m')(x_2-x_1)}{1-lx_1\pi(m)}\right)$. This means that

$$\theta(\mu^A) = \theta(m_1^A) + \theta(m_2^A) = lx_1\pi(m) + l\left(\frac{x_1(1-l\pi(m)) + \pi(m')(x_2-x_1)}{1-lx_1\pi(m)}\right) \tag{4}$$

Proceeding in the same fashion, we compute $\theta(m_1^B) = l(x_1 + (x_2 - x_1)\pi(m'))$. The probability that a task has difficulty in $(0, d_1)$ in $T'_{m_1^B}$ is given by

$$F_{d_{m_1^B}}(d_1) = \frac{x_1(1-l)}{1-l(x+\pi(m')(x_2-x_1))}$$

Consequently, $\theta(m_2^B) = l\pi(m)\left(\frac{x_1(1-l)}{1-l(x+\pi(m')(x_2-x_1))}\right)$, and

$$\theta(\mu^B) = \theta(m_1^B) + \theta(m_2^B) = l(x_1 + (x_2-x_1)\pi(m')) + l\pi(m)\left(\frac{x_1(1-l)}{1-l(x+\pi(m')(x_2-x_1))}\right) \tag{5}$$

Since $d_2 > d_1$ implies $x_2 > x_1$, it can be verified that (4) is indeed greater than or equal to (5) for any values of l, $\pi(m)$, and $\pi(m')$, and hence $\theta(\mu^A) \geq \theta(\mu^B)$.

■

Lemma A.4 indicates that it is optimal to arrange groups with dominance relationships in ascending order, with dominant groups placed later in the process. Lemma A.5 formalizes the idea that an increase in the skill pool of the first group can only lead to the workers in the second group being worse off in terms of output.

Lemma A.5. Let T be a group of tasks and consider groups $W_3 \succ W_1 \succ W_2$ with $\lambda(W_1) = \lambda(W_2)$ and matchings $\mu^A = \left(m_1^A(W_1, T), m_2^A(W_3, T'_{m_1^A})\right)$ and $\mu^B = \left(m_1^B(W_2, T), m_2^B(W_3, T'_{m_1^B})\right)$, then $\theta(m_2^B) \geq \theta(m_2^A)$.

Proof: Let $s_l = \min_s s \in W_3$ and define $T^f_{m_1^i} = \{t \in T'_{m_1^i} | d_t \leq s_l\}$. It is enough to note that $\lambda(T'_{m_1^A}) \leq \lambda(T'_{m_1^B})$ and $T^c_{m_1^A} \cap T^c_{m_1^B} = T^c_{m_1^B}$ and hence $\lambda(T^c_{m_1^B}) = \lambda(T^c_{m_1^A}) - \tau$ and $\lambda(T^f_{m_1^A}) = \lambda(T^f_{m_1^B}) - \tau$ for some $\tau \geq 0$. This implies that

$$\begin{aligned} \frac{\lambda(T^f_{m_1^A})}{\lambda(T) - \lambda(T^c_{m_1^A})} &= \frac{\lambda(T^f_{m_1^B}) - \tau}{\lambda(T) - \lambda(T^c_{m_1^B}) - \tau} \\ &\leq \frac{\lambda(T^f_{m_1^B})}{\lambda(T) - \lambda(T^c_{m_1^B})}, \end{aligned}$$

and therefore the proportion of the tasks that W_3 can finish has shrunk. Since $\lambda(T^f_{m_1^A}) \leq \lambda(T^f_{m_1^B})$, W_3 now finishes a smaller proportion from a smaller set of tasks in m_2^A compared to m_2^B, and therefore $\theta(m_2^B) \geq \theta(m_2^A)$. ■

We use Lemmas A.4 and A.5 to prove the following lemma in preparation for proving Lemma 2. Roughly, Lemma A.6 states that any improvement on a two-group hierarchical arrangement can only be hierarchical, by either moving some of the most skilled workers from the first group to the second, or moving some of the least skilled workers from the second group to the first.

Lemma A.6. Let W_1 and W_2 be two worker groups such that $W_2 \succ W_1$ and denote by μ the matching $(m(W_1), m(W_2))$.

a. Consider $w \subset W_1$ and $w' \subset W_1$ such that $w \succ w'$ and $\lambda(w) = \lambda(w')$ and let $\mu' = \left(m_1(W_1 \setminus w, T), m(W_2 \cup w, T'_{m_1})\right)$ and $\mu'' = \left(m_2(W_1 \setminus w', T), m(W_2 \cup w', T'_{m_2}\right)$. If $\theta(\mu'') \geq \theta(\mu)$, then $\theta(\mu') \geq \theta(\mu'')$.

b. Consider $w \subset W_2$ and $w' \subset W_2$ such that $w \succ w'$ and $\lambda(w) = \lambda(w')$ and let $\mu' = \left(m_1(W_1 \cup w, T), m(W_2 \setminus w, T'_{m_1})\right)$ and $\mu'' = (m_2(W_1 \cup w', T), m(W_2 \setminus w', T'_{m_2}))$. If $\theta(\mu') \geq \theta(\mu)$, then $\theta(\mu'') \geq \theta(\mu')$.

Proof: To prove Part a, assume as in the statement of the lemma that $\theta(\mu'') \geq \theta(\mu)$. We can switch w and w' in μ' to obtain μ''. By Lemma A.4, w' followed by w has higher output than w followed by w', and because $w \succ w'$, by Lemma A.5, $\theta(W_2, T'_{m_1}) \geq \theta(W_2, T'_{m_2})$. Combining these two observations we get that $\theta(\mu') \geq \theta(\mu'')$. The argument for Part b is similar: if we swap w' for w in μ' we get μ''. Again by Lemmas A.4 and A.5 having w' precede w and $W_2 \setminus w$ can only increase output compared to w preceding w', and therefore $\theta(\mu') \leq \theta(\mu'')$. ■

It is important to note that is that Lemma A.6 applies when removing several, possibly disjoint intervals of workers simultaneously from one group to another. For example, consider two intervals

w_1 and w_2 of workers in W_1 and assume that there is an improvement in output when these intervals are moved to group W_2, then it is always at least as good to instead remove a measure w of workers from W_1 to W_2 with w being the most skilled workers in W_1 whose measure is equal to $\lambda(w) = \lambda(w_1 + w_2)$. The same idea holds when removing intervals from W_2 to W_1. This observation along with Lemma A.6 are used in the following proof.

Proof of Lemma 2: As in the statement of the lemma, assume that W_1 and W_2 are not hierarchical. Construct W_1' and W_2' such that $i \in (W_1 \cup W_2) \rightarrow i \in (W_1' \cup W_2')$, $\lambda(W_1) = \lambda(W_1')$, $\lambda(W_2) = \lambda(W_2')$, and $W_2' \succ W_1'$. There exists $w_1' \subset W_1'$ and $w_2' \subset W_2'$ –not necessarily continuous intervals– such that $W_1 = (W_1' \setminus w_1') \cup w_2'$ and $W_2 = (W_2' \setminus w_2') \cup w_1'$. Thus, by moving w_2' to W_1' and moving w_1' to W_2, we can reconstruct W_1 and W_2. However, by Lemma A.6, unless either: a) $w_1' = \emptyset$ and w_2' is such that $(W_2' \setminus w_2') \succ w_2'$ or b) $w_2' = \emptyset$ and w_1' is such that $w_1 \succ (W_1' \setminus w_1')$ – either of which imply that W_1 and W_2 are hierarchical and hence contradict our assumption– there exists $\bar{W}_1$ and $\bar{W}_2$ such that $\bar{W}_1$ and $\bar{W}_2$ are hierarchical and $\theta(\bar{W}_1, \bar{W}_2) = \theta(\bar{\mu}) \geq \theta(W_1, W_2) = \theta(\mu)$. ■

The optimal groups are contiguous:

Lemma A.7. Consider the optimal grouping $W^* = (W_1^*, W_2^*)$ for $k = 2$, and a worker i with skill s_i. If $i \in W^*$, then $\{j | s_j > s_i\} \in W^*$.

Proof: Assume that there is $w_1 \succ w_2 \succ w_3$ such that $w_1 \in W^*$ and $w_3 \in W^*$, and $w_2 \notin W^*$ and $\lambda(w_1) = \lambda(w_2)$[16]. The result follows from Lemma A.2 since the output of W^* improves by swapping w_1 and w_2, and hence W^* cannot be optimal. ■

Proof of Theorem 1: We prove the theorem by induction on the number of groups. Proposition 1 shows that the base case for $k = 2$ holds. Assume that the result holds for k and consider $k + 1$ groups. By the induction hypothesis, the optimal solution for the assignment of the remaining tasks from the matching $m_1(W_1, T)$ is a hierarchy. Furthermore, Lemma A.7 applies to k groups since the output of a group of workers can only increase by substituting a dominated subgroup within a group by a group of equal size that dominates it but is not in the current selection of workers. This implies that the optimal solution for k groups starts with a worker with skill $\bar{s}$ and includes all workers up to the maximum skill in $\mathcal{W}$. Therefore, the optimal solution for $k + 1$ groups uses at most one more measure of workers than the solution for k groups. This measure consists of workers whose skill lie in $(G^{-1}(G(\bar{s})-1), \bar{s})$. Assume that the optimal solution uses workers whose skills is less than $G^{-1}(G(\bar{s})-1)$, then by Lemma A.2 we can replace these workers by others whose skills are in $(G^{-1}(G(s)-1), \bar{s})$ to improve output. Let W_1 be the first group in the optimal $k+1$ groups solution. Groups W_2 through W_{k+1} form a hierarchy by the induction hypothesis. Consider moving $w \in W_i, i > 2$ to group W_1. By Lemmas A.4 and A.5, this will decrease output. Conversely, by Lemmas 1 and A.1, removing any workers from W_1 to later groups would decrease their output and exchanging them with any of

[16]We can always find groups that fulfill this measure requirement under the assumption $w_1 \in W^*$ and $w_2 \notin W^*$.

the more skilled workers in one of those groups is again output decreasing. By similar arguments to Lemma A.6, the only move with a potential for output maximization is one that removes a group w' from W_2 such that $(W_2 \setminus w') \succ w'$, and adds it to W_1, with possible adjustments to later groups (again, removing workers from the peripheries only), and hence the optimal $k+1$ groups solution is also hierarchical. ■

Proof of Proposition 2: The proof proceeds by induction. For $k = 1$, the optimal group W^* is the top measure of workers. Consider splitting this group into two at a worker whose skill is s, so that W_1 has workers whose skills are in (s_{min}, s) and W_2 has workers whose skills are in (s, s_{max}). Let $\mu = \big((m_1(W_1, T), m_2(W_2, T'_{m_1})\big)$. Notice that in this arrangement, workers in W_2 get tasks in T'_{m_1} whereas before they had tasks from T, and note that both $\lambda(T)$ and $\lambda(T'_{m_1})$ are larger than $\lambda(W_2)$. By Lemma 1, $T'_{m_1} \succ T$ and hence by Lemma A.1, $\theta(m_2) < \theta(W_2, T)$. The output of W_1 is the same under both scenarios, therefore, $\theta(W^*) > \theta(\mu)$. Now assume that the result holds for k groups. This means that in order to have $k+1$ groups we need more workers. Assume an incremental amount δ of workers are added and that the skills of these workers are such that they are less than everyone already used (but they can still perform some tasks). Let W_1 contain these δ workers and let the tasks finished in $m(W_1, T)$ be of size δ', then $\lambda(T'_{m_1}) = 1 - \delta'$. Set aside the least skilled δ workers from the original supply of workers and consider the assignment of the remaining $1 - \delta'$ tasks to the remaining workers. By the induction hypothesis, this assignment cannot simultaneously have more than k groups and be better than the original assignment of those workers on those tasks, since if that is the case then we can combine the discarded δ workers into the first group or into their own group and have a better assignment with more than k groups, violating the induction hypothesis. Thus in addition to the discarded δ workers we have group W_1, and at most k other groups. Breaking the (tiny) W_1 group into more than one group will only decrease output as we showed for the base case, and adding the discarded workers into that first group can only improve output since they face an easier difficulty distribution. Therefore the optimal arrangement cannot have more than $k+1$ groups. ■

Lemma A.8. (Monotonicity of success probabilities) For all groups i and all skills s, $\psi(s, W_i)$ is monotonically decreasing in i and monotonically increasing in s.

Proof: Let the distribution of difficulties in group i be given by $F_{d_{T_i}}$ and consider group $j > i$ with difficulty distribution $F_{d_{T_j}}$. A worker with skill s has $\psi(s, W_i) = F_{d_{T_i}}(s)$ in group i and $\psi(s, W_j) = F_{d_{T_j}}(s)$ in group j. By Lemma 1, $d_{T_j} \succeq d_{T_i}$ and therefore $F_{d_{T_i}}(s) \geq F_{d_{T_j}}(s)$, and $\psi(s, W_i)$ is monotonically decreasing in i, and since F is a distribution function, it is monotonically increasing in its argument s.

■

Corollary A.2. (Monotonicity of output) Denote by $\theta(s, i)$ the output of a worker with skill s in group

W_i. In the optimal hierarchical solution $\theta(s,i) \geq \theta(s,j)$ for $j > i$.

Proof of Lemma 3 Consider the optimal grouping and examine the i^{th} matching $m(W_i) = T_i$. Let $\lambda(W_i)$ and $\lambda(T_i)$ be normalized to 1 and assume wlog that the least difficult task in T and the lowest skill in W_i are normalized to zero. Let s_1 and s_2 be the skills of any two workers in W, with $s_2 > s_1$. Divide the difficulties in T into intervals $(0, s_1), (s_1, s_2)$, and (s_2, s_{max}), where s_{max} is the maximum difficulty in T. Let $F_{d_T}(s_1) = x_1$, $F_{d_T}(s_2) - F_{d_T}(s_1) = x_2$, $F_{s_{W_i}}(s_1) = l_1$, and $F_{s_{W_i}}(s_2) - F_{s_{W_i}}(s_1) = l_2$. Denote by π_1 the probability of finishing a task that is randomly assigned from $(0, s_1)$ to a worker whose skill is in $(0, s_1)$. Similarly, denote by π_2 the corresponding probability when a task with difficulty in (s_1, s_2) is assigned to a worker whose skill is in (s_1, s_2). Let T_{i+1} be the set of tasks remaining after $m(W_i)$ and let the measure of tasks in (s_2, s_{max}) in T' be equal to y. We have $\lambda(T_{i+1}) = y + x_1(l_1(1-\pi_1)) + x_2(l_1 + l_2(1-\pi_2))$. The distribution of T_{i+1} is therefore given by

$$F_{d_{T_{i+1}}}(d) = \begin{cases} \frac{x_1 l_1(1-\pi_1)}{y+x_1(l_1(1-\pi_1))+x_2(l_1+l_2(1-\pi_2))} & d = s_1 \\ \frac{x_1(l_1(1-\pi_1))+x_2(l_1+l_2(1-\pi_2))}{y+x_1(l_1(1-\pi_1))+x_2(l_1+l_2(1-\pi_2))} & d = s_2 \\ 1 & d = s_{max} \end{cases}$$

Using $F_{d_{T_{i+1}}}$ we can write

$$\phi(s_1) = \frac{\psi(s_1, W_i)}{\psi(s_1, W_{i+1})} = x_1 \frac{y + x_1(l_1(1-\pi_1)) + x_2(l_1 + l_2(1-\pi_2))}{x_1 l_1(1-\pi_1)}$$

and

$$\phi(s_2) = \frac{\psi(s_2, W_i)}{\psi(s_2, W_{i+1})} = (x_1 + x_2) \frac{y + x_1(l_1(1-\pi_1)) + x_2(l_1 + l_2(1-\pi_2))}{x_1(l_1(1-\pi_1) + l_2)) + x_2(l_1 + l_2(1-\pi_2))}$$

Finally, we have

$$\frac{\phi(s_2)}{\phi s_1} = \frac{(x_1 + x_2) l_1(1-\pi_1)}{x_1(l_1(1-\pi_1)) + x_2(l_1 + l_2(1-\pi_2))} < 1,$$

i.e. $\phi(s_2) < \phi(s_1)$, as required, and the result obviously holds if $i+1$ is replaced by any $j > i+1$. ■

Proof of Theorem 2 Denote by W_0 the group that includes workers whose skills are below the minimum skill for participation $\bar{s}$. Let $s_i = \{\min s | s \in W_i^*\}$, so that $s_1 = \bar{s}$. Consider an entry fee q and payments $p_1, ...p_k$ that satisfy the following system of equations

$$\begin{aligned} p_1 \psi(s_1, W_1^*) &= q \\ p_i \psi(s_{i+1}, W_i^*) &= p_{i+1} \psi(s_{i+1}, W_{i+1}^*) \quad i = 1, 2., , , k-1 \end{aligned}$$

The minimum payoff that can be obtained in W_1^* is equal to $p_1 \psi(s_1, W_1^*) - q = 0$. By Lemma A.8, $\psi(s', W_1^*) < \psi(s_1, W_1^*)$ for all $s' < s_1 = \bar{s}$, and therefore $p_1 \psi(s', W_1^*) - q < 0$. Similarly, for $s > \bar{s}$, $\psi(s, W_1^*) > \psi(\bar{s}, W_1^*)$ and $p_1 \psi(s, W_1^*) - q > p_1 \psi(s_1, W_1^*) - q \geq 0$. Consider a worker with skill $s < s_{i+1}$

in W_i^*. This worker makes $p_i\psi(s, W_i^*)$ and upon moving to W_{i+1}^* makes

$$\begin{aligned} p_{i+1}\psi(s, W_{i+1}^*) &= p_i \frac{\psi(s_{i+1}, W_i^*)}{\psi(s_{i+1}, W_{i+1}^*)} \psi(s, W_{i+1}^*) \\ &= p_i \psi(s, W_i^*) \frac{\psi(s_{i+1}, W_i^*)}{\psi(s_{i+1}, W_{i+1}^*)} \frac{\psi(s, W_{i+1}^*)}{\psi(s, W_i^*)} \\ &= p_i \psi(s, W_i^*) \frac{\phi(s_{i+1})}{\phi(s)} \\ &\leq p_i \psi(s, W_i^*) \end{aligned}$$

where the last inequality is obtained from Lemma 3 and the fact that $s < s_{i+1}$. The inequality still holds if we replace $i + 1$ with any $j > i$. A symmetrical argument shows that $p_{i+1}\psi(s, W_{i+1}^*) \geq p_i\psi(s, W_i^*)$ for any $s > s_{i+1}$. ■

References

Antràs, Pol, Luis Garicano, and Esteban Rossi-Hansberg (2006), "Offshoring in a knowledge economy." *The Quarterly Journal of Economics*, 121, 31–77.

Bansal, N., A. Gupta, J. Li, J. Mestre, V. Nagarajan, and A. Rudra (2010), "When LP is the cure for your matching woes: improved bounds for stochastic matchings." *Algorithms–ESA 2010*, 218–229.

Beggs, Alan W (2001), "Queues and hierarchies." *The Review of Economic Studies*, 68, 297–322.

Boudreau, K.J., N. Lacetera, and K.R. Lakhani (2011), "Incentives and problem uncertainty in innovation contests: An empirical analysis." *Management Science*, 57, 843–863.

Chandler, A.D. (1977), *The visible hand: The managerial revolution in American business*. Belknap Press.

Chawla, S., J.D. Hartline, and B. Sivan (2012), "Optimal crowdsourcing contests." In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms*, 856–868, SIAM.

Chen, N., N. Immorlica, A. Karlin, M. Mahdian, and A. Rudra (2009), "Approximating matches made in heaven." *Automata, Languages and Programming*, 266–278.

Chesbrough, H., W. Vanhaverbeke, and J. West (2008), *Open Innovation: Researching a New Paradigm: Researching a New Paradigm*. OUP Oxford.

Dulleck, U., R. Kerschbamer, and M. Sutter (2011), "The economics of credence goods: An experiment on the role of liability, verifiability, reputation, and competition." *American Economic Review*, 101, 526–555.

Faridani, S., B. Hartmann, and P.G. Ipeirotis (2011), "Whats the right price? pricing tasks for finishing on time." In *Proc. of AAAI Workshop on Human Computation*.

Feldman, J., A. Mehta, V. Mirrokni, and S. Muthukrishnan (2009), "Online stochastic matching: Beating 1-1/e." In *Foundations of Computer Science, 2009. FOCS'09. 50th Annual IEEE Symposium on*, 117–126, IEEE.

Fuchs, W., L. Garicano, and L. Rayo (2013), "Optimal contracting and the organization of knowledge." Working Paper.

Garicano, L. (2000), "Hierarchies and the organization of knowledge in production." *Journal of Political Economy*, 108, 874–904.

Garicano, Luis and Andrea Prat (2013), "Organizational economics with cognitive costs." In *Advances in Economics and Econometrics: Tenth World Congress*, volume 1, 342, Cambridge University Press.

Garicano, Luis and Esteban Rossi-Hansberg (2004), "Inequality and the organization of knowledge." *American Economic Review*, 94, 197–202.

Garicano, Luis and Esteban Rossi-Hansberg (2006), "The knowledge economy at the turn of the twentieth century: the emergence of hierarchies." *Journal of the European Economic Association*, 4, 396–403.

Horton, J.J. and L.B. Chilton (2010), "The labor economics of paid crowdsourcing." In *Proceedings of the 11th ACM conference on Electronic commerce*, 209–218, ACM.

Kamenica, E. and M. Gentzkow (2011), "Bayesian persuasion." *American Economic Review*, 101.

Moldovanu, B. and A. Sela (2006), "Contest architecture." *Journal of Economic Theory*, 126, 70–96.

Pallais, Amanda (2013), "Inefficient hiring in entry-level labor markets." Technical report, National Bureau of Economic Research.

Roy, Andrew Donald (1951), "Some thoughts on the distribution of earnings." *Oxford economic papers*, 3, 135–146.

Sattinger, Michael (1975), "Comparative advantage and the distributions of earnings and abilities." *Econometrica: Journal of the Econometric Society*, 455–468.

Teulings, Coen N (1995), "The wage distribution in a model of the assignment of skills to jobs." *Journal of Political Economy*, 280–315.

Tinbergen, Jan (1974), "Substitution of graduate by other labor." *Kyklos*, 27, 217–226.

Von Hippel, E. (1976), "The dominant role of users in the scientific instrument innovation process." *Research policy*, 5, 212–239.

Von Hippel, E. (2009), "Democratizing innovation: the evolving phenomenon of user innovation." *International Journal of Innovation Science*, 1, 29–40.

Wolinsky, A. (1993), "Competition in a market for informed experts' services." *The RAND Journal of Economics*, 380–398.