Depth-Workload Tradeoffs for Workforce Organization

Hoda Heidari and **Michael Kearns**
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, Pennsylvania 19104
{hoda,mkearns}@cis.upenn.edu

Abstract

We introduce and consider the problem of effectively organizing a population of workers of varying abilities. We assume that arriving tasks for the workforce are homogeneous, and that each is characterized by an unknown and one-dimensional *difficulty* value $x \in [0, 1]$. Each worker i is characterized by their *ability* $w_i \in [0, 1]$, and can solve the task if and only if $x \leq w_i$. If a worker is unable to solve a given task it must be forwarded to a worker of greater ability. For a given set of worker abilities W and a distribution P over task difficulty, we are interested in the problem of designing efficient forwarding structures for W and P. We give efficient algorithms and structures that simultaneously (approximately) minimize both the maximum workload of any worker, and the number of workers that need to attempt a task. We identify broad conditions under which workloads diminish rapidly with the workforce size, yet only a constant number of workers attempt each task.

1 Introduction

In crowdsourcing applications and many other settings, a large workforce is available for a series of relatively homogeneous tasks — for instance, labeling images for the presence or absence of a particular object such as cars. Most existing crowdsourcing systems treat the tasks as if they were of uniform difficulty, treat the workers as if they were of uniform ability, and simply assign the next task to the next available worker [1]. In reality, it is quite natural to assume there may be significant variation in both task difficulty and worker ability, and to incorporate this variation into the design of the system.

We thus consider the problem of effectively organizing a population of workers of varying abilities. We consider a simple model in which we assume that arriving tasks for the workforce are homogeneous, and that each is characterized by an unknown and one-dimensional [2] *difficulty* value $x \in [0, 1]$. Each worker i is characterized by their *ability* $w_i \in [0, 1]$, and can solve the task if and only if $x \leq w_i$ If a worker is unable to solve a given task then it must be passed or *forwarded* to a worker of greater ability. We assume that the worker abilities w_i are known (perhaps via prior observation or experience)[3].

Our primary interest is in the design of efficient forwarding structures for a given workforce. While the problem of efficiently *incentivizing* a workforce (Horton and Chilton 2010; Ho et al. 2012; Ghosh and McAfee 2012) is also interesting and important, we separate it from our concerns here. It is perhaps fair to think of our results as applicable to settings where workers are either salaried employees willing to attempt the tasks given to them, or are paid a fixed amount per task, as is common on Amazon's Mechanical Turk. In the latter case we can view a worker's ability value as a conflation of their true underlying ability, and their willingness to perform quality work at the offered rate.

For our notion of efficient workforce organization, we consider a bicriteria approach. More specifically, we seek to find forwarding structures that simultaneously keep both *maximum workload* and *depth* small. Maximum workload measures the largest workload of any worker in the workforce under a given forwarding structure, while depth measures the number of forwarding steps required before a task is solved. Small maximum workload ensures that no single worker is (unnecessarily) overburdened, and can be viewed as a sign of efficient use of a large workforce — we might hope (and indeed shall show) that under natural assumptions, individual workloads can diminish rapidly with the workforce population size n, meaning that the system is scaling well. Small depth is motivated by the desire to solve individual tasks as rapidly as possible by keeping forwarding chains short (which also minimizes expenditures if workers are paid per task). We note that these two criteria may be in tension with each other, leading to tradeoffs. For example, if we optimize only for depth we would choose to give all tasks to our best worker, who would then have the highest possible workload. It is also easy to create examples in which optimizing the workload results in large depth. We are interested in the problem of designing forwarding structures

[1]See (Salek, Bachrach, and Key 2013) for an exception that is directly related to the models we examine here.

[2]The important generalization to the multi-dimensional case carries a number of definitional and technical challenges, and is left to future work.

[3]For example, by standard arguments if we draw $O(\log(n)/\epsilon^2)$ tasks at random and give them to all n workers, we can estimate all the w_i to within an additive ϵ, which suffices for our results.

that approximately minimize both workload and depth, and the types of structures required to do so.

We consider two variants of the framework above that differ in precisely how they measure workloads. In the *Pay-to-Forward (P2F)* model, the only way for workers to discover whether they are able to solve a given task is to attempt it, which then counts against their workload. In other words, in the P2F model, attempting a task and failing is just as onerous as solving it. In the *Free-to-Forward (F2F)* model, we assume that workers are able to quickly assess whether they are able to solve a given task *without* actually attempting it, and thus tasks they must forward do not count against their workload. Mathematically, in the P2F model a given task is charged against the workload of *every* worker along the forwarding chain that ends in a worker able to solve it, whereas in the F2F model only this final successful worker's load is charged.[4]. We note that a number of our results are the same or similar in both models, but there are sufficient differences that it is worth identifying them separately.

As more concrete motivations for these two variants, first consider a crowdsourcing system in which workers are asked to locate specific objects in images (Salek, Bachrach, and Key 2013). While task difficulty and worker ability may vary considerably, no worker can simply glance at an image and immediately determine whether they will succeed or fail — a certain amount of time must be spent studying and scanning the image either way. In this example the P2F model is appropriate. In contrast, consider a system in which workers are asked to judge the accuracy of technical articles. A worker whose specialty is computer science might be able to immediately see that they have been assigned an article on organic chemistry, and know they cannot solve the task before investing any effort. In this case the F2F model seems more fitting.

Results: Omniscient Workload. We begin our results by considering the optimization of the maximum workload, regardless of depth.[5] We first consider the optimal *omniscient* algorithm for task assignment that is allowed to observe the actual difficulty x of each arriving task, and assign it to any worker in the worker population W capable of solving it; since workloads in the weaker P2F and F2F models can clearly only be higher, the optimal omniscient algorithm provides an important baseline or lower bound. For any W and task difficulty distribution P, we give a precise characterization of the optimal omniscient maximum workload $M_{W,P}$, and give an efficient algorithm for computing its forwarding policy.

Results: P2F and F2F Workloads. We then consider how closely $M_{W,P}$ can be approximated in the P2F and F2F models, and by what types of forwarding structures. Following a preliminary result showing that it always suffices to consider probabilistic DAGs as the forwarding structure in both models, we then show that in the F2F model, $M_{W,P}$ can always be achieved, and give an efficient algorithm for computing the witnessing DAG. We also show that the weaker forwarding structure of trees cannot always achieve $M_{W,P}$ in F2F in general, and that $M_{W,P}$ cannot always be achieved in the P2F model regardless of the structure.

Results: Near-Optimal Depth-Workload Tradeoffs. We conclude by giving our main positive approximation results for both models in the bicriteria setting. More specifically, we show that by hierarchically arranging the workers in balanced b-ary trees, we can simultaneously come within a multiplicative factor of b^2 of $M_{W,P}$ in terms of maximum workload, with a resulting depth of $\frac{\log(n)}{\log(b)}$ in both the P2F and F2F models. Thus, if $b = 2$ we obtain a 4-approximation to the optimal omniscient workload with only $\log(n)$ depth. More generally, we can tune b to decrease the depth at the expense of workload. We also provide a lower bound for this class of structures, showing that maximum workloads must be at least $\sqrt{b}M_{W,P}$ in the worst case. We apply the upper bound to the natural model where worker abilities are distributed as a power law, and show that the maximum workload diminishes with the population size, even for constant depth.

We freely acknowledge that the models studied here make a variety of strong and simplifying assumptions that prevent their immediate applicability to real-world crowdsourcing systems and other labor organization problems. These include:

- One-dimensional task difficulty; surely in reality most settings demand a multi-dimensional notion.
- Lack of detailed treatment of incentive issues.
- No modeling of variable costs for forwarding tasks.
- No modeling of weaker workers being able to solve any task, but with greater effort.
- No modeling of imperfect task completion.

Despite these simplifications, we shall see that there is already a fair amount to say in our streamlined model; we leave the consideration of the important extensions above (some of which we suspect are both conceptually and technically nontrivial) to future work.

1.1 Related Work

To our knowledge, this work is the first to study a model of a workforce with varying ability, and to examine workload-depth tradeoffs and associated algorithmic and representational issues. There are, however, several tangentially related strands of existing research that study labor organization design and various other tradeoffs.

Organizational Design and Hierarchical Structures: There is a large body of work on the design and architecture of organizations; see (Hax and Majluf 1981) for a survey. In most of these works, hierarchical organizational structures are considered and their properties are studied. In (Garicano 2000) the focus is on the tradeoff between communication and knowledge acquisition costs. In (Cremer, Garicano, and Prat 2007), a theory of optimal organizational languages is developed, and a tradeoff between facilitating internal communication and encouraging communication with other organizations is identified. The authors study the optimal communication structure and language, and the organizational

[4]Note that the acronyms could equally well stand for "Pay to Fail" and "Free to Fail".

[5]As noted above, optimizing depth alone is trivial, since we can simply assign every task to the best worker.

structure that supports them. (Ferreira and Sah 2012) studies organizations with individuals whose expertise differ in content and breadth, and derives implications for the trade-off between specialization of knowledge and communication costs, the role of top and middle managers, and the optimal design of hierarchies. (Prat 1997) studies the properties of a hierarchy of processors where the organization must pay each employed processor a wage which is an increasing function of the processor's capacity.

Crowdsourcing: While our models are not specifically limited to crowdsourcing settings, they are related to the growing literature on this subject, as we too study efficient ways of organizing and utilizing a large pool of workers. We discuss some of the works more relevant to our interests. In (Salek, Bachrach, and Key 2013), the authors employ a model that shares our emphasis on the variability of task difficulty and worker ability. They propose a probabilistic graphical model to localize objects in images based on responses from the workforce, and improve upon natural aggregation methods by simultaneously estimating the difficulty and skill levels. In (Ho and Vaughan 2012), the goal is to assign tasks to a set of workers with unknown skill sets in a way that maximizes the benefit to the crowdsourcing system. The authors show that the algorithm they propose to this end is competitive with respect to the optimal offline algorithm which knows the skill levels when the number of workers is large. This model differs from ours in that its focus is on workers with unknown skill levels who arrive in an online fashion, and does not consider depth-workload tradeoffs. In (Horton and Chilton 2010) the authors study the reservation wage one needs to offer to a workforce to incentivize them to perform the task. In (Ho et al. 2012), incentive mechanisms are devised to discourage workers from putting in as little effort as possible, and crowdsourcers from denying payments. In (DiPalantino and Vojnovic 2009) and (Ghosh and McAfee 2012), a game-theoretic model of crowdsourcing is proposed, in which workers act strategically and seek to maximize their total benefit. (Karger, Oh, and Shah 2011) studies achieving a desired level of reliability cheaply, using the fact that crowdsourcers often increase their confidence in the result of crowdsourcing by assigning each task multiple times and combining the results. (Zhang et al. 2012) studies mechanisms for task routing that aim to harness people's abilities to both contribute to a solution and to route tasks to others, who they know can also solve and route. See also (Law and Ahn 2011) for a thorough discussion of different task routing methods.

Load Balancing. Somewhat indirectly related to our work is the literature on load balancing, where the goal is to distribute jobs among a set of machines to optimize the overall performance of the system; for instance, see (Azar et al. 1999; Adler et al. 1995; Ghosh et al. 1999). Our model differs from these works in that task assignments are driven by worker abilities, while in the load balancing scenario, the decision to assign a task to a machine is usually based on its current workload only. Also, in our model there are no arrival and departure times defined for the tasks, rather they stay in the system until they are solved.

2 Workforces with Varying Abilities: Models

In our model, an organization will consist of a set of n workers $W = \{1, 2, \dots n\}$, where the *ability* of worker i is denoted by $w_i \in [0, 1]$; we use i and w_i interchangeably to denote workers. The ability of a worker determines the *difficulty* of the tasks she is capable of solving: a worker with ability w can solve all the tasks with difficulty less than or equal to w [6]. We assume without loss of generality that $w_1 \le w_2 \le \dots \le w_n = 1$, and the difficulty of every task lies in $[0, 1]$; the best worker w_n can thus solve any task.

Any task given to the organization has an *unknown* difficulty denoted by $x \in [0, 1]$ and sampled from a known distribution P. Without loss of generality, we may represent the task distribution P by $\langle A_1, A_2, \dots, A_n \rangle$ where $A_i = Pr_{x \sim P}[w_{i-1} < x \le w_i]$ is the mass of tasks solvable by w_i but not by w_{i-1} (where we define $w_0 = 0$).

In order to solve a given task, an algorithm assigns it to one of the workers, who attempts to solve it. If the worker is unsuccessful, the algorithm forwards it to another worker of greater ability. This chain of forwarding continues until the task is solved by the first worker of sufficient ability. Note that the most general notion of forwarding would allow an arbitrary, centralized algorithm that (probabilistically) decides which worker to forward the task to, given the sequence of failures so far. However, as we shall show, it turns out that such an algorithm can always be represented by a decentralized probabilistic DAG over the workers themselves, and that in many cases near-optimal performance is possible with even simpler decentralized forwarding schemes, such as trees over the workers.

In this paper, we are interested in algorithms and representations for task forwarding, and their performance according to two criteria: workload and depth. We discuss and define workload first.

In the *Pay-to-Forward (P2F)* model, we assume that any worker who receives a task along a forwarding chain has their workload charged for attempting that task. Thus, for any algorithm A that determines how to forward tasks for W and P, we define the workload ℓ_i of worker i to be the probability that i receives a task drawn from P and forwarded according to A. In the *Free-to-Forward (F2F)* model, we assume that only the worker who actually *solves* the task is charged, and thus define the workload ℓ_i of worker i to be the probability that i is the *last* worker to receive a task drawn from P and forwarded according to A. Clearly for any A all workloads ℓ_i in the P2F model are greater than or equal to those in the F2F model.

As noted in the Introduction, the P2F model is appropriate for settings in which the only way workers can determine whether they are able to solve a task is to attempt it, which consumes the same effort whether they succeed or fail; and the F2F model is appropriate for settings in which, despite the actual task difficulty value x being unobservable, work-

[6]Our notion of ability can also incorporate some unobservable mixture of a worker's actual ability, and the quality of work they are willing to do at a given rate — for instance, a high-ability worker may choose to be lazy at the given rate and do no or low-quality observed work.

ers can immediately assess whether they have the requisite skills to solve a problem, and forward it if not.

In both the P2F and F2F models, we define the *depth* of a forwarding algorithm A as the maximum number of workers in any forwarding chain of A under W and P. Our interests here are in forwarding algorithms and schemes that simultaneously achieve small maximum workload and depth. As noted in the Introduction, these two criteria may often be in conflict with each other, necessitating the tradeoffs that we study here. In general, we are most interested in cases in which workloads diminish rapidly with n, with depth growing only very slowly with n (logarithmic or even constant); we shall eventually see this is possible under fairly broad conditions.

3 The Omniscient Model

Regardless of W and P, it is clear how to minimize depth alone in task forwarding: simply assign every task to $w_n = 1$, who can solve all tasks (at the expense of the worst possible maximum workload of 1). In contrast, if we ask what the smallest possible maximum workload is, the answer depends strongly on W and P, and is far from obvious. Since we need to compare the workload performance of algorithms in the P2F and F2F models to some baseline, we consider an idealized *omniscient model*, in which a forwarding algorithm is actually allowed to observe the true task difficulty $x \sim P$, and immediately assign it to any worker capable of solving the task. Obviously workloads in the P2F and F2F models, where x is not observed, can only be worse than the workload-minimizing algorithm in the omniscient model. We now give a precise characterization of the optimal maximum workload in the omniscient model, and show that it can be computed efficiently. Perhaps surprisingly, later we shall see that this ideal can actually be achieved or well-approximated in the P2F and F2F models under fairly general circumstances.

Theorem 1 *In the omniscient model (and therefore in both the P2F and F2F models), any algorithm for task assignment has maximum workload greater than or equal to* $\max_i \frac{\sum_{j=i}^{n} A_j}{n-i+1}$.

Proof Every task $x > w_{i-1}$, must eventually be solved by one of the workers $i, i+1, ..., n$. So at least one of these $(n-i+1)$ workers, say w, must have workload greater than or equal to $\frac{Pr[x>w_{i-1}]}{n-i+1} = \frac{(A_i+...+A_n)}{n-i+1}$. The maximum workload among all workers cannot be less than w's workload, therefore we have $\max_j \ell_j \geq \frac{(A_i+...+A_n)}{n-i+1}$. This holds for any i, therefore we can conclude $\max_j \ell_j \geq \max_i \frac{(A_i+...+A_n)}{n-i+1}$. ∎

We next show that in the omniscient model, there is an efficient algorithm for assigning the tasks that achieves this lower bound.

Theorem 2 *In the omniscient model, there is a task assignment algorithm whose maximum workload is equal to* $\max_i \frac{\sum_{j=i}^{n} A_j}{n-i+1}$*, and the assignment policy used by this algorithm can be computed in time* $O(n^2)$.

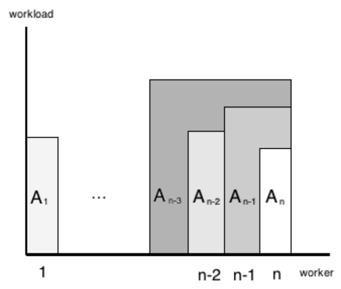

Figure 1: Workload-optimizing assignment in the omniscient model.

We omit the full proof due to space considerations but we sketch the algorithm here. The algorithm uses a policy that determines how to distribute tasks among workers, that is, it computes the probability with which a task $w_{i-1} < x \leq w_i$ is given to worker j ($j \geq i \geq 1$). It does this inductively from the hardest to easiest tasks, always maintaining the invariant that the workloads of workers above the current difficulty level are equal. Note that the (pre-)computation of this policy occurs only once, and then is used to assign arriving tasks.

The policy computation first sets all the workloads to 0. If a task is in $(w_{n-1}, w_n]$, there is no choice other than giving it to worker n. So the current workload of worker n is increased to A_n. Now if a task is in $(w_{n-2}, w_{n-1}]$, either worker n or worker $n-1$ must eventually solve it. If $A_{n-1} \leq A_n$, the task is given to worker $n-1$ with probability 1, making its current workload A_{n-1}. If not, in order to minimize the maximum current workload, we split A_{n-1} between these two workers such that their current workload becomes equal. Similarly at each step when we are deciding how to distribute a task in $(w_{i-1}, w_i]$, we do it in a way that the maximum current workload is minimized. This continues until all the tasks assignment probabilities are computed. In Figure 1, a visualization of this procedure is given.

For the remainder of the paper, we denote the maximum workload of the omniscient algorithm on a distribution P, by $M_{W,P}$, which we have established is equal to $\max_i \frac{(A_i+...+A_n)}{n-i+1}$.

4 F2F and P2F: DAGs Suffice

In this section, we present a useful result that we shall employ in the subsequent sections. We show that in both the F2F and P2F models, the class of arbitrary randomized forwarding algorithms is no more powerful than the class of probabilistic DAGs, with respect to both workload and depth.

We require the following notation. In a DAG (and therefore a tree) structure, we denote the probability that a task is initially assigned to worker i by I_i, and thus $\sum_{i=1}^{n} I_i = 1$. In addition, the weight on edge uv, denoted by $p_{u,v}$, specifies the probability that worker u forwards a given task to v, in the case she fails to solve it herself. Thus $\sum_{v=u+1}^{n} p_{u,v} = 1$. Note that throughout, when talking about a "structure", whether DAG or tree, we not only refer to the specific network structure, we also consider the edge weights (if applicable) and initial probabilities on the vertices to be part of the structure.

Theorem 3 *In both the F2F and P2F models, for any randomized task forwarding algorithm A, there is a probabilistic DAG G such that the workload of every worker, and the depth, are the same in A and G.*

Proof Observe that upon initialization, A has no information about the value of the given task x, and therefore it must have a fixed distribution P_0 over the first worker the task is given to. Now consider the point where A has forwarded the task to $k \geq 1$ workers $w_{i_1} < w_{i_2} < ... < w_{i_k}$, they have all failed, and A must decide who to forward the task to in the next step.

Note that the only information A has about x so far is that $x \in (w_{i_k}, w_n]$ and is distributed with respect to P in that range, since the information previous workers provided about x by failing is subsumed by the fact that w_{i_k} failed to solve it, that is:

$$Pr_{x \sim P}[x \mid (x > w_{i_1}) \wedge (x > w_{i_2}) \wedge ... \wedge (x > w_{i_k})] = Pr_{x \sim P}[x \mid (x > w_{i_k})]$$

So A has to forward the task based on a fixed distribution, call it P_{i_k}, over $w_{i_k+1}, w_{i_k+2}, ..., w_n$. It is now easy to see that A can be represented as a DAG: Let G be a complete DAG[7] in which for every worker u and every worker v with higher ability than u, the weight on edge uv is the probability that A forwards a task to v right after u's failure. This probability can be easily obtained from the fixed distribution P_u. Also the probability that a node gets selected in the initial step can be obtained from P_0. It is clear that on any given task, G and A have exactly the same forwarding behavior, and therefore the workload of every worker and the depth are the same in both of them. ▮

5 Workload-Minimizing Structures

We now consider the optimization of workload alone in the P2F and F2F models. We compare the workloads of optimal DAGs and trees with the omniscient algorithm, and show that in the F2F model there always exists an efficiently computed DAG with maximum workload equal to $M_{W,P}$. We show that that the same is not possible for trees. For the P2F model, we show that even the class of DAGs cannot always achieve maximum workload equal to $M_{W,P}$. We then proceed to consider workload-depth tradeoffs in Section 6.

5.1 The F2F Model

We first show that in the F2F model, there is a DAG that achieves maximum workload equal to the omniscient optimal $M_{W,P}$.

Theorem 4 *In the F2F model, there always exists a DAG G whose maximum workload is equal to $M_{W,P}$. In addition, there is an efficient algorithm for constructing G.*

[7]A DAG is complete if in its topological sort, each node points to all the nodes with higher indices.

Proof Consider the smallest index, say j, for which we have $\frac{(A_j+...+A_n)}{n-j+1} = \max_i \frac{(A_i+...+A_n)}{n-i+1}$. If there exists a DAG with maximum workload equal to $M_{W,P}$, then in that DAG, $w_j, ..., w_n$ should never get any task with difficulty $x \leq w_{j-1}$, otherwise at least one of them will have workload larger than $\frac{(A_j+...+A_n)}{n-j+1} = \max_i \frac{(A_i+...+A_n)}{n-i+1}$, resulting in max workload larger than $M_{W,P}$. An immediate consequence is that the initial probability of workers $j, ..., n$, i.e. $I_j, ..., I_n$ must be zero and the only node that can have edges with non-zero probability to them is worker $(j-1)$.

That being said, if $j > 1$, in order to build the optimal DAG, we do the following: first we build the optimal DAG G_1 for $w_1, ..., w_{j-1}$ and $\langle A_1, .., A_{j-1} \rangle$, then build the optimal DAG G_2 for $w_j, ..., w_n$ and $\langle A_j, ..., A_n \rangle$. To combine G_1 and G_2, we use the initial probability of vertex $i \geq j$ in G_2 as edge weight on the edge from worker $(j-1)$ to i, and we set I_i to 0. It is easy to see that combining the two DAGs this way results in the optimal DAG for $w_1, ..., w_n$ and P.

This suggests a recursive procedure for building the optimal DAG when $j > 1$. However, we still need to deal with the case where $j = 1$ and the workload must be divided equally among all workers.

For this remaining case $w_1, ..., w_n$ and $\langle A_1, ..., A_n \rangle$ are such that the omniscient algorithm gives every worker the same workload, equal to $\frac{(A_1+...+A_n)}{n}$. Algorithm 1 (which we call WED) presents an efficient procedure with running time $O(n^2)$ that generates a DAG with max workload $\frac{(A_1+...+A_n)}{n}$ for this case.

ALGORITHM 1: Algorithm WED for Computing the Workload Equalizing DAG in the F2F Model

```
Data: A task distribution $P = \langle A_1, A_2, ..., A_n \rangle$ with
      $M_{W,P} = \frac{(A_1+...+A_n)}{n}$
Result: A weighted DAG $G$
$I_1 \leftarrow \frac{(A_1+A_2+...+A_n)}{nA_1}$;
$V(G) \leftarrow \{w_1\}$;
$E(G) \leftarrow \emptyset$;
if $n=1$ then
    return;
else
    $P' \leftarrow \langle (1-I_1)A_1 + A_2, A_3, ..., A_n \rangle$;
    $H \leftarrow$ WED($P'$) i.e. the workload optimal DAG for $P'$;
    for $i \leftarrow 2$ to $n$ do
        $I'_i \leftarrow$ the initial probability of worker $i$ in $H$;
    end
    $V(G) \leftarrow V(G) \cup V(H)$;
    $E(G) \leftarrow E(H)$;
    Add edges from $w_1$ to every vertex in $V(H)$;
    for $i \leftarrow 2$ to $n$ do
        $p_{1,i} \leftarrow I'_i$;
        $I_i \leftarrow (1-I_1)I'_i$;
    end
end
```

This algorithm is the recursive form of the following inductive construction:

Induction statement: for $k \in \mathbb{N}$ workers, if P is such that $M_{W,P} = \frac{(A_1+...+A_k)}{k}$, then there is a DAG G for which the maximum workload in G is equal to $M_{W,P}$.

Induction basis: For $k = 1$ workers, the optimal DAG is trivial. It is a single node and we have to give all the workload A_1 to the only worker available i.e. $I_1 = 1$.

Induction step: Suppose for a set of $k < n$ workers, we know how to build the optimal DAG to have the workload of every worker equal to $\frac{(A_1+...+A_k)}{k}$. Now suppose we are given a set of n workers with ability $w_1, w_2, ..., w_n$, and the task distribution $\langle A_1, .., A_n\rangle$ such that $M_{W,P} = \frac{(A_1+...+A_n)}{n}$.

Note that in order to have a DAG in which everyone has workload equal to $\frac{(A_1+...+A_n)}{n}$, this must hold for w_1 as well. Since w_1's workload is equal to $I_1 \times A_1$, I_1 must be equal to $\frac{(A_1+...+A_n)}{nA_1}$ then.

To build the rest of the optimal DAG, observe that for workers $w_2, ..., w_n$, there is no difference between a task in $[0, w_1]$ and one in $(w_1, w_2]$ i.e. all of them can solve both. Therefore we can assume the remaining mass on $[0, w_1]$ is actually on $(w_1, w_2]$ and find the optimal DAG for the remaining workers and task distribution.

According to the induction hypothesis, for the task distribution $\langle A_1(1-I_1)+A_2, A_3, ..., A_n\rangle$, we can find the optimal DAG say H, that gives equal workloads to $w_2, ..., w_n$. Suppose the initial probability on node i in H is I'_i. Now we construct G, in the following way: Add a vertex for w_1 to H and set the initial probability of this vertex to $\frac{(A_1+...+A_n)}{nA_1}$. Then set the probability on the edge from w_1 to w_i, to I'_i. Also set the initial probability of vertex $i > 1$ in G to $I_i = I'_i(1-I_1)$.

We claim that in G, $w_1, ..., w_n$ all get equal workloads. For w_1 this obviously holds due to the way we chose I_1. For the nodes in H let's look at the task distribution they receive from outside of H, i.e. either initially or from w_1. With probability $(1 - I_1)$, a task is initially forwarded to a node in H; we know that such task, has distribution $\langle A_1, ..., A_n\rangle$. With probability I_1 the task is forwarded initially to w_1, and if w_1 forwards the task to some one in H, all we know about the task is that it is from the distribution $\langle 0, A_2, ..., A_n\rangle$. Combining the above distributions with respect to the probability that each happens, we get the distribution $I_1\langle 0, A_2, ..., A_n\rangle + (1 - I_1)\langle A_1, A_2, ..., A_n\rangle = \langle (1 - I_1)A_1, A_2, ..., A_n\rangle$.

Now according to the induction hypothesis, H is optimal for this task distribution, giving $w_2, ..., w_n$ equal workload. Therefore we can conclude G is giving equal workload to everyone. ∎

We next show that in the F2F model, the weaker class of trees does *not* suffice to achieve maximum workload $M_{W,P}$.

Theorem 5 *In the F2F model, for $n > 2$ workers, there are worker abilities W and a task distribution P for which the maximum workload in any tree is strictly larger than $\frac{6}{5}M_{W,P}$.*

Proof Let $P_\epsilon = \langle A_1, A_2, A_3\rangle$ where $A_1 = \frac{1}{3}$, and $A_2 = (\frac{1}{3} + \epsilon)$ and $A_3 = (\frac{1}{3} - \epsilon)$ with $0 < \epsilon \leq \frac{1}{3}$. We claim that for this task distribution the optimal DAG is unique, and because this unique DAG is not a tree, we conclude that there exists no tree with the same (optimal) max workload.

Obviously $M_{W,P} = \frac{1}{3}$ in this case. To have a DAG with the same max workload, we must initially give every task to worker 1. So the workload of worker 2 is $p_{1,2}(\frac{1}{3} + \epsilon)$ which must be equal to $\frac{1}{3}$, therefore $p_{1,2} = \frac{1/3}{1/3+\epsilon} < 1$, and as a result $p_{1,3} > 0$, meaning that the optimal DAG is not a tree. It is easy to see that for $P_{\frac{1}{3}}$, the maximum workload of the optimal tree is $\frac{2}{5}$, proving the claimed lower bound. ∎

The above theorem confirms that in the F2F model, the class of trees is not as powerful as the class of DAGs in terms of the maximum workload. One question we would like to address is how large this gap can be. Later in Section 6, we will see that for any task distribution P, we can find a tree T such that the maximum workload in T is less than or equal to $4M_{W,P}$, yielding a constant-factor approximation to the optimal DAG.

5.2 The P2F Model

We next see that for the P2F model, even the optimal DAG cannot always achieve the omniscient optimal maximum workload. This already holds for a very simple case of two workers, and the task distribution $P = \langle \frac{1}{2}, \frac{1}{2}\rangle$.

Theorem 6 *In the P2F model, there are worker values W and a distribution P for which the maximum workload of the optimal DAG is strictly larger than $\frac{4}{3}M_{W,P}$.*

In Section 6 we will see that in the P2F model, for any task distribution P, the maximum workload in the optimal tree (and as a result optimal DAG), is less than or equal to $4M_{W,P}$.

6 Near-Optimal Depth-Workload Tradeoffs

Armed with a firm understanding of workload-only optimization in the omniscient model, and for DAGs and trees in the P2F and F2F models, we now finally turn our attention to workload-depth tradeoffs. We shall need the following definition:

Definition A *well-balanced b-ary* tree T is a tree with branching factor b $(1 \leq b \leq n - 1)$, such that:

1. Worker n is located at the root (first layer); the next b best workers are direct children of worker n; the next b^2 best workers are the children of the nodes in second layer, and so on. The worst workers are the leaves in the tree.
2. If we sort the nodes in any layer, say d, in a decreasing order, and then sort the children of each of those nodes in a decreasing order as well, it must be case that nodes in layer $(d + 1)$ are properly sorted in a decreasing order.

Less formally, we simply take a balanced b-ary tree, and place the workers from best to worst in a top-to-bottom, left-to-right fashion. It is easy to see that the depth of the well-balanced b-ary tree is $\frac{\log n}{\log b}$. One example of a well-balanced b-ary tree is illustrated in Figure 2 for $b = 3$.

Unless otherwise specified, we assume that only leaves have non-zero probabilities of being initially assigned a task,

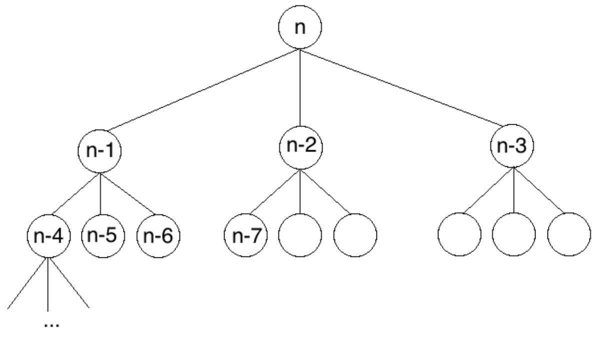

Figure 2: The b-ary tree structure with $b = 3$. Vertices are ordered by worker index, from the best (n, $w_n = 1$) down to the worst (1).

and that these probabilities are such that every node at the same depth has the same probability of having a task initially assigned to a worker in their subtree.

We show that in a well-balanced b-ary tree of size n, not only do we achieve a depth that increases very slowly with n, the maximum workload we get is a constant factor approximation of $M_{W,P}$. The intuition for why such trees provide a good trade-off is that for a given depth, they (approximately) minimize the difference in ability between the task sender and the immediate receiver. In this way we avoid the possibility of missing too many qualified workers in between, which in turn could result in large maximum workload.

Theorem 7 *In the F2F and P2F models, for any task distribution P, the maximum workload in a well-balanced b-ary ($b \geq 2$) tree of depth $\frac{\log n}{\log b}$ is less than or equal $b^2 M_{W,P}$.*

Proof We prove the theorem for the P2F model. Since workloads are only lower in the F2F model, the theorem holds there as well.

Consider the well-balanced b-ary tree T where the only nodes with non-zero initial probabilities are the leaves. Note that here we assume $n = \frac{b^{m+1}-1}{b-1}$ ($m \in \mathbb{N}$), i.e. the last layer of the well-balanced tree is complete. If it is not the case, we can add sufficient number of workers with ability zero to the set of workers to make n of the desired form. Note that by doing so, we may at most multiply the number of workers by a factor b, but as those additional workers cannot solve any task, $M_{W,P}$ remains the same. We also set the initial probability on every leaf to $\frac{1}{b^m}$.

We claim that for any P, in this tree the workload of each worker is less than or equal $b^2 M_{W,P}$. First note that our claim holds for the leaves in T. The workload of each leaf say l is equal to I_l. Since $I_l = \frac{1}{b^m} \leq \frac{b}{n} \leq b M_{W,P}$, we have $\ell_l \leq b^2 M_{W,P}$. Note that here we used the fact $M_{W,P} \geq \frac{1}{n}$ which holds by definition of $M_{W,P}$.

We shall make use of the following assertion.

Proposition 8 *In the P2F model, the workload of worker i in a tree T is equal to:*

$$\ell_i = I_i + \sum_{j \to i} H_j \times Pr_{x \sim P}[w_j < x]$$

where H_j is the probability that a task x is initially assigned to a worker in the subtree rooted at node j, and $j \to k$ means node j is a direct child of node k.

Observe that according to Proposition 8, for a non-leaf worker i, we can write the following:

$$\begin{aligned} \ell_i &= \sum_{j \to i} (H_j \times Pr_{x \sim P}[w_j < x]) \\ &\leq \max_{k \to i} Pr_{x \sim P}[w_k < x] \times \sum_{j \to i} H_j \\ &= Pr_{x \sim P}[w_{k^*} < x] \times \sum_{j \to i} H_j \\ &= Pr[w_{k^*} < x] \times H_i \end{aligned}$$

where $k^* = \arg\min_{k \to i} w_k$.

Using the above inequality, we next find an upper bound for the workload of a node at depth d in T. Note that at depth d, $H_i = \frac{1}{b^d}$. Also $k^* \geq i - b^{d+1}$. So:

$$\ell_i \leq H_i \times Pr[w_{k^*} < x] \leq \tfrac{1}{b^d} \times Pr[w_{i-b^{d+1}} < x].$$

Now note that since i has depth d, we have $i \geq n - (b^1 + b^2 + ... + b^d) = n - \frac{b^{d+1}-1}{b-1} + 1$. Therefore $i - b^{d+1} \geq n - \frac{b^{d+2}-b}{b-1}$, and we can write:

$$\begin{aligned} \ell_i &\leq \frac{1}{b^d} \times Pr_{x \sim P}[w_{i-b^{d+1}} < x] \\ &\leq \frac{1}{b^d} \times Pr_{x \sim P}[w_{n-b^{d+2}} < x] \\ &\leq b^2 \left(\frac{1}{b^{d+2}} \times Pr_{x \sim P}[w_{n-b^{d+2}} < x] \right) \\ &\leq b^2 M_{W,P} \end{aligned}$$

where the last inequality follows from Theorem 1 since we have:

$$\begin{aligned} M_{W,P} &= \max_r (A_r + ... + A_n)/(n - r + 1) \\ &= \max_r Pr_{x \sim P}[w_{r-1} < x]/(n - r + 1) \\ &\geq Pr_{x \sim P}[w_{n-b^{d+2}} < x]/(b^{d+2}) \end{aligned}$$

This completes the proof. ∎

In Figure 3 we illustrate the depth-workload tradeoff provided by Theorem 7. For a large value of n, each curve is parameterized by the branching factor b — i.e. each curve is traced out by varying b from n (smallest depth of 1) to 2 (largest depth of $\log(n)$. The x axis then measures the resulting depths, and the y axis the corresponding maximum workloads given by Theorem 7. The curves differ in the value of $M_{W,P}$ that is assumed, ranging from $1/n$ (rapidly diminishing omniscient optimal workload) to a constant independent of n. Clearly smaller workloads are desirable; each curve can thus be thought of as providing an upper bound on the Pareto optimal curve for the corresponding bicriteria problem. We see that for small $M_{W,P}$, there is essentially threshold behavior of our bound — unless we choose a

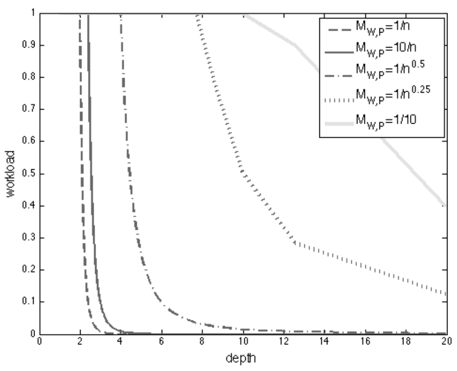

Figure 3: Illustration of Theorem 7.

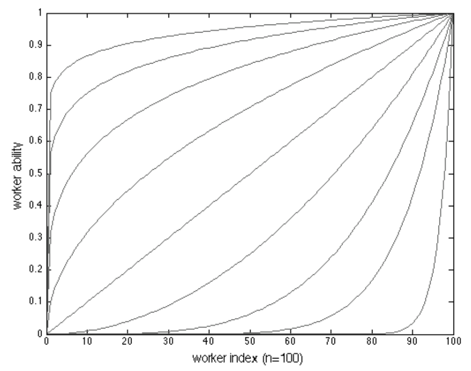

Figure 4: Worker ability as a function of index under the model $w_i = (i/n)^a$ for various a.

sufficiently large depth (small branching factor), our bound is vacuous, but then rapidly falls to 0. At larger $M_{W,P}$, the decay of workload with depth is more gradual.

The maximum workload upper bound given in Theorem 7 degrades with the branching factor b at a rate of b^2; it is not clear in general whether any such dependence is necessary. The following theorem, however, shows that at least $\sqrt{b}$ dependence is necessary within the class of balanced trees and the P2F model.

Theorem 9 *In the P2F model, for any* $n \in \mathbb{N}$ *and* $b = 2, ..., (n-1)$*, there is a task distribution* P *and a set of* n *workers* W *for which any well-balanced* b*-ary tree results in maximum workload greater than or equal* $\frac{\sqrt{b}}{3} \times M_{W,P}$.

Proof (Sketch) Given n and b, let P be the uniform distribution and let $w_i = 0$ for $i \leq (n-b-1)$ and $w_i = \frac{(i-n+b+1)}{(b+1)}$. It is easy to see that in this case $M_{W,P} = \frac{1}{(b+1)}$. By solving a suitable optimization problem, it can be shown that for P, W, a well-balanced b-ary tree results in maximum workload larger than $\frac{1}{3\sqrt{b}}$, even if we allow the initial distribution of tasks over the leaves to be arbitrary, and also allow an arbitrary probability that the root receives an assigned task directly.

6.1 Constant Depth and Diminishing Workloads

We conclude with an application of Theorem 7 to a natural parametric model of worker abilities that results in perhaps the best tradeoff we could hope for. Consider the case where $M_{W,P} = 1/n^\alpha$ for some constant $0 < \alpha \leq 1$ — that is, the maximum workload diminishes rapidly as a function of the workforce size n. (We shall shortly give natural assumptions on W for which this holds.) By choosing $b = n^\beta$, Theorem 7 yields maximum workload at most $b^2/n^\alpha = 1/n^{\alpha-2\beta}$. Thus as long as $2\beta < \alpha$, the well-balanced tree will also give an inverse polynomial workload decay with n, while the depth $\frac{\log(n)}{\log(b)} = 1/\beta$ will only be *constant*.

For example, consider the case where $w_i = (\frac{i}{n})^a$ for some $a \geq 1$, and P is the uniform distribution. In this parametric family for W, if $a < 1$ we get concave improvement of workers with i, so successive workers are improving rapidly; while if $a > 1$ we have convex improvement, so the best workers may be far better than the average. See Figure 4 for examples.

Note that when $a > 1$ since $\{A_i\}_{i=1}^n$ is an increasing sequence here, $M_{W,P}$ is equal to A_n. And $A_n = 1-(\frac{n-1}{n})^a \approx \frac{a}{n}$ using the Taylor expansion of the function $f(y) = y^a$. Thus we can immediately apply the observations above to obtain workloads bounded by $O(1/n^{1-2\beta})$ with only constant depth $1/\beta$.

Also for the case where $a < 1$ since $\{A_i\}_{i=1}^n$ is a decreasing sequence here, $M_{W,P}$ is equal to $\frac{1}{n}$. Applying Theorem 7, we obtain workloads bounded by $O(1/n^{1-2\beta})$ again with depth only $1/\beta$.

7 Future Directions

Here are some interesting generalizations and open questions raised by the results presented here.

- Perhaps the most important generalizations would address all the unrealistic assumptions of our model mentioned in the Introduction: multi-dimensional difficulty, incentive issues, variable task forwarding costs, imperfect task completion, and others.
- We showed that the optimal DAG in the P2F model is not necessarily as efficient as the omniscient algorithm in terms of maximum workload. One interesting problem is to determine how large the gap can be.
- The gap between the b^2 workload factor given in Theorem 7 and the corresponding lower bound of $\sqrt{b}$ given in Theorem 9 is large; can it be closed or improved?
- It would be interesting to conduct behavioral experiments designed to quantify the performance benefits of organizational schemes like those suggested here.

Acknowledgements

We thank Sanjeev Goyal for early discussions of the problems studied here, and to the anonymous reviewers for helpful suggestions on the exposition. Research was supported by ARO MURI no. W911NF-12-1-0509.

References

Adler, M.; Chakrabarti, S.; Mitzenmacher, M.; and Rasmussen, L. 1995. Parallel randomized load balancing. In *STOC*, 119–130.

Azar, Y.; Broder, A. Z.; Karlin, A. R.; and Upfal, E. 1999. Balanced allocations. *SIAM Journal on Computing* 29(1):180–200.

Cremer, J.; Garicano, L.; and Prat, A. 2007. Language and the theory of the firm. *Quarterly Journal of Economics* 122(1):373–407.

DiPalantino, D., and Vojnovic, M. 2009. Crowdsourcing and all-pay auctions. In *EC*, 119–128.

Ferreira, D., and Sah, R. K. 2012. Who gets to the top? generalists versus specialists in managerial organizations. *RAND Journal of Economics*.

Garicano, L. 2000. Hierarchies and the organization of knowledge in production. *Journal of Political Economy* 108(5).

Ghosh, A., and McAfee, P. 2012. Crowdsourcing with endogenous entry. In *WWW*, 999–1008.

Ghosh, B.; Leighton, F. T.; Maggs, B. M.; Muthukrishnan, S.; Plaxton, C. G.; Rajaraman, R.; Richa, A. W.; Tarjan, R. E.; and Zuckerman, D. 1999. Tight analyses of two local load balancing algorithms. *SIAM Journal on Computing* 29(1):29–64.

Hax, A. C., and Majluf, N. S. 1981. Organizational design: A survey and an approach. *Operations Research* 29(3):417–447.

Ho, C. J., and Vaughan, J. W. 2012. Online task assignment in crowdsourcing markets. In *AAAI*.

Ho, C. J.; Zhang, Y.; Vaughan, J. . W.; and van der Schaar, M. 2012. Towards social norm design for crowdsourcing markets. In *AAAI*.

Horton, J. J., and Chilton, L. . B. 2010. The labor economics of paid crowdsourcing. In *EC*, 209–218.

Karger, D. R.; Oh, S.; and Shah, D. 2011. Iterative learning for reliable crowdsourcing systems. In *NIPS*, 1953–1961.

Law, E., and Ahn, L. 2011. *Human Computation*. Morgan and Claypool Publishers.

Prat, A. 1997. Hierarchies of processors with endogenous capacity. *Journal of Economic Theory* 77(1):214–222.

Salek, M.; Bachrach, Y.; and Key, P. 2013. Hotspotting – A probabilistic graphical model for image object localization through crowdsourcing. In *AAAI*.

Zhang, H.; Horvitz, E.; Chen, Y.; and Parkes, D. 2012. Task routing for prediction tasks. In *AAMAS*, volume 2, 889–896.