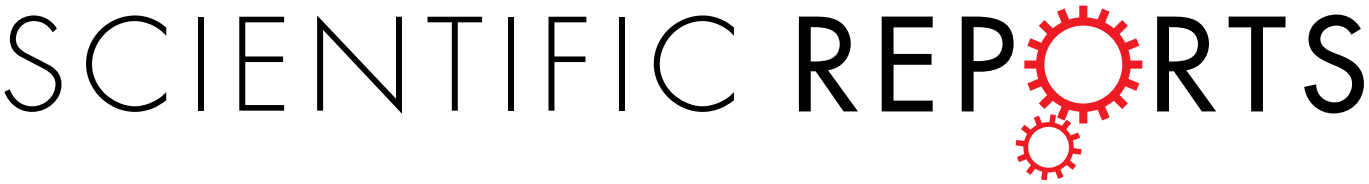

OPEN

Mitigating Herding in Hierarchical Crowdsourcing Networks

Han Yu[1], Chunyan Miao[1,2], Cyril Leung[1,3], Yiqiang Chen[1,4], Simon Fauvel[1], Victor R. Lesser[5] & Qiang Yang[6]

Received: 19 April 2016
Accepted: 24 August 2016
Published online: 05 December 2016

Hierarchical crowdsourcing networks (HCNs) provide a useful mechanism for social mobilization. However, spontaneous evolution of the complex resource allocation dynamics can lead to undesirable herding behaviours in which a small group of reputable workers are overloaded while leaving other workers idle. Existing herding control mechanisms designed for typical crowdsourcing systems are not effective in HCNs. In order to bridge this gap, we investigate the herding dynamics in HCNs and propose a Lyapunov optimization based decision support approach - the Reputation-aware Task Sub-delegation approach with dynamic worker effort Pricing (RTS-P) - with objective functions aiming to achieve superlinear time-averaged collective productivity in an HCN. By considering the workers' current reputation, workload, eagerness to work, and trust relationships, RTS-P provides a systematic approach to mitigate herding by helping workers make joint decisions on task sub-delegation, task acceptance, and effort pricing in a distributed manner. It is an individual-level decision support approach which results in the emergence of productive and robust collective patterns in HCNs. High resolution simulations demonstrate that RTS-P mitigates herding more effectively than state-of-the-art approaches.

The organization of social and economic activities to efficiently coordinate participants' effort is an important topic of economic theory. Thanks to the Internet, social media and online social networks, social mobilization through crowdsourcing has achieved unprecedented success. Crowdsourcing refers to the process whereby clients (a.k.a. *crowdsourcers*) obtain needed services by soliciting contributions from a large group of people (a.k.a. *workers*)[1]. Crowdsourcing communities based around social networks tend to have hierarchical structures[2,3]. These hierarchical crowdsourcing networks (HCNs) have been used to mobilize the masses in many significant real-world applications including political rallies[4], scientific research[5], mapping out natural environment features[6,7], and large-scale search-and-rescue missions[8].

In essence, crowdsourcing systems can be treated as resource allocation ecosystems containing a large number of interacting workers (i.e., resources) and crowdsourcers. Crowdsourcers are typically self-interested; their primary intention is to maximize their own utilities. This will usually lead them to only select workers with high perceived reputation, leading to the emergence of *herding*[9]. Herding refers to the situation in which a large number of task requests concentrate on a small group of reputable workers, causing them to be overloaded while leaving other workers idle. It can lead to cascading failures and eventually result in catastrophic system breakdown[10]. The risk of herding is especially pronounced in HCNs in which crowdsourcers lack global knowledge and workers have limited resources to be tapped into[11].

Mitigating herding in HCNs is important to ensure sustainable operation of these problem solving ecosystems[12]. In general, workers in an HCN make three important decisions in a distributed manner: 1) *how much new workload to accept*, 2) *how much existing workload to sub-delegate to others in the HCN (and to whom)*, and 3) *how to price their services*. The collective effect of these joint decisions made by all HCN participants determines

[1]Joint NTU-UBC Research Centre of Excellence in Active Living for the Elderly (LILY), Nanyang Technological University, Singapore, Singapore. [2]School of Computer Science and Engineering, Nanyang Technological University, Singapore, Singapore. [3]Department of Electrical and Computer Engineering, The University of British Columbia, Vancouver, BC, Canada. [4]Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China. [5]School of Computer Science, University of Massachusetts Amherst, Amherst, MA, USA. [6]Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong, China. Correspondence and requests for materials should be addressed to H.Y. (email: han.yu@ntu.edu.sg) or C.M. (email: ascymiao@ntu.edu.sg) or Y.C. (email: yqchen@ict.ac.cn)

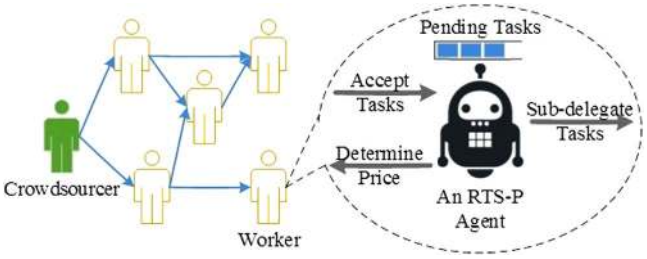

Figure 1. An RTS-P agent in an HCN.

whether herding will emerge. Therefore, herding mitigation mechanisms need to influence these three decisions by each worker in order to improve the overall efficiency of an HCN. The human nature of the HCN participants imposes additional complexities on this already challenging problem:

1. **Worker heterogeneity**: Workers have different skill levels and productivity. They may produce results of different quality when assigned the same task, and may not be able to maintain the same level of productivity everyday.
2. **Timing and targets for sub-delegation**: It is difficult for a worker to quantify when sub-delegation is needed and who the suitable candidates for sub-delegation are. This is further complicated by the fact that different workers may incur different costs to complete the same task. Sub-delegation to a worker resulting in a loss for the sub-delegator is not a rational choice.
3. **Workers' commitment**: Workers may not be fully committed to an HCN. Their eagerness to work (which may change over time) will affect their availability.

Recently, computational approaches for mitigating herding in crowdsourcing systems have emerged. In the Pinning control method[10], the pinning method is used to control the collective dynamics in complex networks. The study focuses on situations where multiple agents try to decide individually which one of two available resources to use. Thus, this method cannot be directly applied to crowdsourcing systems in which many crowdsourcers need to engage a large number of workers to accomplish their objectives. The Global Considerations (GC) approach uses a worker's current pending workload as a guide to adjust his reputation[13]. GC adjusts the probability for a task to be assigned to a worker based on the worker's reputation standing among all other workers using the *softmax* approach. In Yu *et al.*[14], a centralized task allocation approach was proposed to make dynamic trade-offs between the need for engaging trustworthy workers and obtaining task results on time. A fully distributed variant of this method that helps workers determine which incoming tasks to accept was studied in Yu *et al.*[15]. All of these approaches allow workers to be automatically assigned to tasks, saving them time spent on exploring open task calls and improving their collective productivity. Nevertheless, these existing approaches are not designed for HCNs. They do not support task sub-delegation, an essential mechanism to avoid herding in HCNs. The aforementioned complexities due to human nature have also not been accounted for by existing approaches.

This paper investigates the herding dynamics in HCNs and proposes the Reputation-aware Task Sub-delegation approach with dynamic worker effort Pricing (RTS-P) to mitigate herding through enhancing the efficiency of manpower utilization in HCNs. It is an individual-level decision-making approach based on Lyapunov optimization[16] with objective functions aiming to achieve superlinear time-averaged collective productivity in an HCN[17]. By considering a worker's current reputation, workload, willingness to work, and his trust relationships with others, RTS-P provides a systematic approach for a worker to make joint decisions on task acceptance, sub-delegation, and effort pricing, so as to maximize his income while avoiding significant fluctuations in workload. The approach is distributed and can be implemented as a personal decision support agent for a worker in an HCN (Fig. 1). RTS-P is an extension of our previous model - RTS[18]. The addition of the dynamic worker effort pricing function allows operation in systems which permit workers to set the price of their service. In doing so, substantial modifications to the original system model[18] and the joint task acceptance and sub-delegation decisions are required.

RTS-P is compared with 4 existing methods through extensive experiments based on a large-scale real-world dataset - the *Epinions* trust network dataset. The results show that RTS-P effectively mitigates herding through efficiently harnessing the available human resources. We also show that RTS-P workers achieve significantly higher total income compared with other state-of-the-art approaches, especially under high workload conditions. RTS-P not only automates key decisions in the situation-task-others triad[19] surrounding a worker, but also sheds light on the long-standing quest for an individual-level decision support approach which results in productive and robust collective patterns in human crowds[20]. Our work provides a general framework to optimally harness the collective productivity of a complex network of human resources in order to mitigate herding, with potential applications in many social and economic systems.

Methods

Our key results include (1) a formulation of the problem of mitigating herding through efficiently harnessing the productivity of workers in an HCN as a constrained optimization problem which minimizes drastic fluctuations in workers' workloads while maximizing their expected earnings; (2) a distributed algorithm which solves the

problem by jointly controlling the task acceptance, task sub-delegation, and effort pricing decisions for each worker; and (3) experimental evaluations of the performance of the proposed algorithm against state-of-the-art approaches in a large-scale HCN.

Proposed Framework. Our focus in this paper is to address the problem of delegating/sub-delegating a task, τ_j, proposed by a crowdsourcer j, to workers in an HCN. In general, the effort required to complete a task (i.e. the workload of the task) can be expressed in *effort units* which can be defined by crowdsourcing system operators. For example, the effort required to complete a software programming task can be measured by the expected number of lines of code. A task must be completed before its stipulated deadline and with quality acceptable to the crowdsourcer. A worker i has a limited effort output rate which can be up to $\mu_i^{\max}$ effort units per time slot. Tasks waiting to be completed by i are stored in his pending tasks queue. Let $q_i(t)$ be worker i's pending workload at the beginning of time slot t; the queuing dynamics of $q_i(t)$ can be formulated as:

$$q_i(t+1) = \max\,[q_i(t) + \lambda_i(t) - \mu_i(t) - s_i(t),\ 0] \tag{1}$$

where $\lambda_i(t)$ is the new workload accepted into $q_i(t)$ during time slot t, $\mu_i(t)$ represents the actual workload completed by i during time slot t, and $s_i(t)$ is the sub-delegated workload by worker i during time slot t.

With crowdsourcing analytics tools such as Turkalytics[21], workers' performance can be tracked in detail. A worker i's past performance as measured by the quality and timeliness of his productive output can be used to estimate i's reputation, $r_i(t) \in (0, 1)$, using a reputation evaluation model[22]. Reputation acts as a sanctioning mechanism affecting future demand for a worker's services. Using this information, a worker can establish trust relationships with a set of other known workers, $\mathbf{n}_i^{sub}$. $\mathbf{n}_i^{sub}$ is the set of trusted workers to whom worker i's tasks can potentially be delegated or sub-delegated. As a task can be iteratively sub-delegated through a *delegation chain*, it is reasonble for all workers in the delegation chain to be accountable (to various degrees) for the outcome of the task. A possible model for sharing responsibility in a delegation chain is the *Decreasing Weighting* (DW) reputation update mechanism[23] which assigns the last worker in the delegation chain (i.e. the one who actually completed the task) the highest share of responsibility and decreasing weight values to other workers higher up along the delegation chain. The future expected demand for a worker i's service, $\mathbb{E}\{\lambda_i(t)\}$, is affected by his $r_i(t)$ value and the current price he charges for his service, $p_i(t)$ (e.g., measured in dollars per effort unit), i.e.,

$$\mathbb{E}\{\lambda_i(t)\} = f(p_i(t),\ r_i(t)). \tag{2}$$

Automating Task Sub-delegation Decisions. An RTS-P agent takes only local knowledge as input, and automatically offers recommendations to a worker i concerning three key decisions in an HCN at any given point in time: 1) the timing, amount of workload, and the target workers for sub-delegation, 2) how much new workload shall be accepted by i, and 3) how to price his services. If an RTS-P agent determines that its owner i's risk of not completing all pending tasks before the respective deadlines is high, it will attempt to sub-delegate some of the pending tasks to other workers. The selection of candidate workers for sub-delegation takes into account how trusted the workers are and how much they charge for their services (i.e., so that the act of sub-delegating does not incur financial loss for its owner). These heuristics can be converted into a computational task sub-delegation mechanism as follows.

A conceptual queue, $Q_i(t)$, is used to quantify the urgency for a worker i to sub-delegate pending tasks. $Q_i(t)$ is updated by an RTS-P agent in conjunction with $q_i(t)$ as follows:

$$Q_i(t+1) = \max[Q_i(t) - \mu_i(t) - s_i(t) + \bar{\lambda}_i 1_{[q_i(t)>0]},\ 0]. \tag{3}$$

In this formulation, the symbol $\bar{\lambda}_i$ represents the average amount of new workload accepted by worker i per time slot. $1_{[\text{condition}]}$ is an indicator function. Its value is 1 if and only if [condition] is satisfied; otherwise, it evaluates to 0. The dynamics of $Q_i(t)$ are as follows:

- The workload in $Q_i(t)$ is increased in such way that if $q_i(t)$ is non-empty at the time when the value of $Q_i(t)$ is updated, then $Q_i(t)$ grows by $\bar{\lambda}_i$. This ensures that $Q_i(t)$ keeps increasing if there are tasks in $q_i(t)$ which have not been completed for some time.
- The value of $Q_i(t)$ is reduced by the task servicing process $[-\mu_i(t) - s_i(t)]$.

In order to efficiently utilize the productivity of a crowdsourcing network, RTS-P must ensure that the upper bounds of both $q_i(t)$ and $Q_i(t)$ are finite for all workers involved.

Let $X_i(t) = (q_i(t), Q_i(t))$ be a concatenated vector of worker i's physical and conceptual pending tasks queues. We adopt the *Lyapunov* function[16] to measure the level of congestion in both $q_i(t)$ and $Q_i(t)$ for all workers in a given HCN. It can be expressed as $L(X_i(t)) = \frac{1}{2}[q_i^2(t) + Q_i^2(t)]$. Then, the amount of change in worker i's pending workload can be measured using the conditional *Lyapunov* drift as:

$$\begin{aligned}\Delta(X_i(t)) &= \mathbb{E}\{L(X_i(t+1)) - L(X_i(t))|X_i(t)\}\\ &= \mathbb{E}\left\{\left[\frac{1}{2}q_i^2(t+1) + \frac{1}{2}Q_i^2(t+1)\right] - \left[\frac{1}{2}q_i^2(t) + \frac{1}{2}Q_i^2(t)\right]|X_i(t)\right\}\\ &= \mathbb{E}\left\{\frac{1}{2}q_i^2(t+1) - \frac{1}{2}q_i^2(t) + \frac{1}{2}Q_i^2(t+1) - \frac{1}{2}Q_i^2(t)|X_i(t)\right\}.\end{aligned} \tag{4}$$

Based on equation (3), we have:

$$\begin{aligned}\frac{1}{2}Q_i^2(t+1) - \frac{1}{2}Q_i^2(t) &= \frac{1}{2}[\bar{\lambda}_i - (\mu_i(t) + s_i(t))]^2 + Q_i(t)[\bar{\lambda}_i - \mu_i(t) - s_i(t)] \\ &= \frac{1}{2}\{(\bar{\lambda}_i)^2 - 2\bar{\lambda}_i[\mu_i(t) + s_i(t)] + [\mu_i(t) + s_i(t)]^2\} \\ &\quad + Q_i(t)[\bar{\lambda}_i - \mu_i(t) - s_i(t)] \\ &\leqslant \frac{1}{2}[(\lambda_i^{\max})^2 + (\mu_i^{\max} + s_i^{\max})^2] + Q_i(t)[\bar{\lambda}_i - \mu_i(t) - s_i(t)] \end{aligned} \tag{5}$$

where $\lambda_i^{\max}$ and $s_i^{\max}$ are the respective upper bounds of $\lambda_i(t)$ and $s_i(t)$ for a given worker i.

Based on the same approach, the conditional Lyapunov drift for the physical queue can be expressed as:

$$\frac{1}{2}q_i^2(t+1) - \frac{1}{2}q_i^2(t) \leqslant \frac{1}{2}[(\lambda_i^{\max})^2 + (\mu_i^{\max} + s_i^{\max})^2] + q_i(t)[\lambda_i(t) - \mu_i(t) - s_i(t)]. \tag{6}$$

From equations (5) and (6), equation (4) can be expressed as:

$$\Delta(X_i(t)) \leqslant \mathbb{E}\{(\lambda_i^{\max})^2 + (\mu_i^{\max} + s_i^{\max})^2 + Q_i(t)[\bar{\lambda}_i - \mu_i(t) - s_i(t)] + q_i(t)[\lambda_i(t) - \mu_i(t) - s_i(t)]|X_i(t)\}. \tag{7}$$

For simplicity of notation, let $C_i = (\lambda_i^{\max})^2 + (\mu_i^{\max} + s_i^{\max})^2$. From a worker i's view point, he would wish to minimize both the cost incurred by task sub-delegation as well as drastic changes in his pending workload. Thus, we formulate a $\{drift + cost\}$ expression to capture this dual goal as follows:

$$\begin{aligned}&\Delta(X_i(t)) + \rho_i(t)\mathbb{E}\{\varphi_i(t)s_i(t)|X_i(t)\} \\ &\quad \leqslant C_i + Q_i(t)\mathbb{E}\{\bar{\lambda}_i - \mu_i(t) - s_i(t)|X_i(t)\} + q_i(t)\mathbb{E}\{\lambda_i(t) - \mu_i(t) - s_i(t)|X_i(t)\} + \rho_i(t)\mathbb{E}\{\varphi_i(t)s_i(t)|X_i(t)\}. \end{aligned} \tag{8}$$

where $\varphi_i(t) = \frac{1}{|\mathbf{n}_i^{sub}|}\sum_{k\in\mathbf{n}_i^{sub}} p_k(t)$ represents the average price of service charged by worker i's known trusted workers at time slot t, and $\rho_i(t) > 0$ represents i's general eagerness to work. A large value of $\rho_i(t)$ indicates that a worker is highly motivated to work. It adjusts the relative importance given to the two components in the $\{drift + cost\}$ expression. It can be inferred by keeping track of the worker's productivity over a period of time, or be explicitly declared by the worker to control how the RTS-P agent behaves.

At the beginning of each time slot, the RTS-P agent observes $q_i(t)$ and $Q_i(t)$, as well as its owner i's current context tuple $\langle \mu_i(t), \lambda_i(t), \varphi_i(t)\rangle$, to determine the value of $s_i(t)$ which minimizes the $\{drift + cost\}$ expression. This form of combined value maximization and surprise minimization complies with the latest findings in human choice behaviours[24]. By only considering the terms containing the decision variable $s_i(t)$ which can be controlled by the RTS-P agent in equation (8), the $\{drift + cost\}$ objective function can be re-expressed as:

Minimize:

$$\frac{1}{T}\sum_{t=0}^{T-1}\sum_{i=1}^{N} s_i(t)[\rho_i(t)\varphi_i(t) - q_i(t) - Q_i(t)] \tag{9}$$

Subject to:

$$0 \leqslant s_i(t) \leqslant q_i(t) \tag{10}$$

$$\exists\, k \in \mathbf{n}_i^{sub}, \exists\, \tau_j \in q_i(t),\ r_k(t) \geqslant r_{\min}(t) \wedge p_k(t) \leqslant p_{\tau_j}. \tag{11}$$

p_{τ_j} is the price the worker i charges for task τ_j. $r_{\min}(t) \in [0, 1]$ is a pre-determined reputation threshold value. In order to minimize equation (9), $\hat{s}_i(t)$, the target value of $s_i(t)$, is:

$$\hat{s}_i(t) = \begin{cases} 0, & \text{if } \rho_i(t)\varphi_i(t) - q_i(t) - Q_i(t) \geqslant 0 \\ q_i(t) - \mu_i(t), & \text{otherwise.} \end{cases} \tag{12}$$

Intuitively, equation (12) means that when worker i is highly willing to work, the cost of sub-delegating is high, the current workload is low, and tasks in the pending tasks queue have not been pending for too long, worker i should not sub-delegate any tasks. Otherwise, worker i should try to sub-delegate as many tasks as possible. Nevertheless, the actual $s_i(t)$ value also depends on the satisfaction of Constraint (11), which requires at least one worker k in $\mathbf{n}_i^{sub}$ whose reputation is higher than the threshold, and who charges a price no higher than what worker i charges for the task (i.e., worker i does not incur any loss by sub-delegating the task to worker k).

Automating Task Acceptance and Effort Pricing Decisions. Taking the cost of task sub-delegation into account, the expected income for a worker i at time slot t becomes $\mathbb{E}\{\xi_i(t)|p_i(t), r_i(t)\} = a_i(t)p_i(t)r_i(t)f(p_i(t), r_i(t)) - \varphi_i(t)s_i(t)$ where $a_i(t)$ is a binary decision variable which controls if worker i accepts new tasks at the beginning of time slot t. In this case, a worker i would wish to maximize his income while minimizing drastic fluctuations in his pending workload. Similar to equation (8), this objective function can be formulated as an $\{income - drift\}$ function:

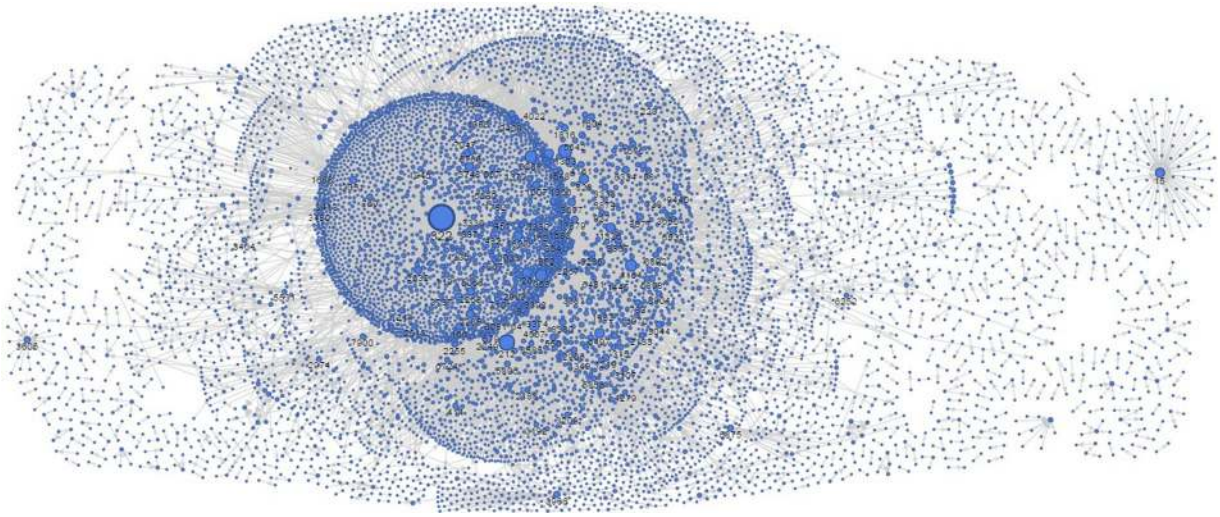

Figure 2. The HCN based on the *Epinions* trust network structure. Nodes represent worker agents and arrows represent trust relationships. The larger the size of a node, the more trusted a worker agent is.

$$\begin{aligned}\rho_i(t)\mathbb{E}\{\xi_i(t)|X_i(t)\} - \Delta(X_i(t)) \leqslant\ & \rho_i(t)\mathbb{E}\{a_i(t)p_i(t)r_i(t)f(p_i(t), r_i(t)) \\ & - \varphi_i(t)s_i(t)|X_i(t)\} - C_i - q_i(t)\mathbb{E}\{a_i(t) \\ & \times f(p_i(t), r_i(t)) - \mu_i(t) - s_i(t)|X_i(t)\} \\ & - Q_i(t)\mathbb{E}\{\bar{\lambda}_i - \mu_i(t) - s_i(t)|X_i(t)\}\end{aligned} \tag{13}$$

which is to be maximized.

A recent large scale empirical study in e-commerce, involving sellers from both eBay and Taobao[25], suggests the following expression relating new demand (i.e., workload) for a worker to his price and reputation:

$$\ln f(p_i(t), r_i(t)) = c_0 - c_1 \ln r_i(t) + c_2 \ln N_i^p(t) + c_3 d_i + \ln p_i(t) \tag{14}$$

where c_0 to c_3 are positive constants, $N_i^p(t)$ is the number of positive ratings received by i over a given period of time, and d_i represents how similar the quality of service provided by i is to what he promises. In this paper, we adopt equation (14) for modeling the dynamics of the demand for a worker's service to derive the joint task acceptance and effort pricing strategy. Nevertheless, equation (14) can be replaced by other functions suitable for different systems without affecting the principle on which RTS-P operates.

By taking exponents on both sides of equation (14), we have:

$$f(p_i(t), r_i(t)) = \vartheta_i \frac{p_i(t)}{[r_i(t)]^{c_1}} \tag{15}$$

where $\vartheta_i = e^{(c_0 + c_3 d_i)}[N_i^p(t)]^{c_2}$. As RTS-P only controls the decision variables $p_i(t)$ and $a_i(t)$ for effort pricing and task acceptance, we only consider the terms containing these decision variables on the right hand side of equation (13) and substitute $f(p_i(t), r_i(t))$ with equation (15). Thus, we have:

Maximize:

$$\frac{1}{T}\sum_{t=0}^{T-1}\sum_{i=1}^{N}\frac{a_i(t)\vartheta_i p_i(t)}{[r_i(t)]^{c_1}}[p_i(t)r_i(t)\rho_i(t) - q_i(t)] \tag{16}$$

Subject to:

$$p_i(t) \geqslant p_i^{\min}, \forall t, \forall i \tag{17}$$

where $p_i^{\min}$ is the minimum price to cover i's cost of service. We assume the value of $p_i^{\min}$ does not change frequently and can be treated as a constant with respect to i. The solution for maximizing this objective function can be obtained by finding the first order derivative of equation (16) and equating it to 0:

$$\frac{d}{dp_i(t)}\left\{\frac{a_i(t)\vartheta_i}{[r_i(t)]^{c_1}}[r_i(t)\rho_i(t)p_i^2(t) - q_i(t)p_i(t)]\right\} = 0. \tag{18}$$

Solving equation (18) yields:

$$p_i(t) = \max\left[p_i^{\min}, \frac{q_i(t)}{2\rho_i(t)r_i(t)}\right]. \tag{19}$$

The result means that i should increase the price he charges for new tasks if his current workload is high, his current reputation is low, or his eagerness to work is low (and vice versa), while ensuring that his price is always no less than $p_i^{\min}$. If i's reputation is low, he is less likely to receive a large number of task requests. Thus, whenever others are willing to solicit i's service, from i's perspective, he should charge a higher price in order to capitalize on these opportunities.

To maximize equation (16):

$$a_i(t) = \begin{cases} 1, & \text{if } p_i(t)r_i(t)\rho_i(t) - q_i(t) > 0 \\ 0, & \text{otherwise.} \end{cases} \tag{20}$$

In this paper, we ensure that a worker is never assigned more workload than the maximum workload he can handle within one time slot. Thus, when $a_i(t) = 1$, RTS-P accepts up to $\mu_i^{\max}$ effort units worth of new workload into i's pending tasks queue.

The core RTS-P algorithm is presented in Algorithm 1. It can be implemented as a personal decision support agent for each worker in an HCN. Multiple RTS-P agents can then communicate on their respective owners' behalf to automate the task acceptance, sub-delegation, and pricing decisions to maximize the overall productivity of the given crowdsourcing network.

Algorithm 1 RTS-P

Require: $q_i(t)$, $Q_i(t)$, $p_i^{\min}$, $r_i(t)$, $r_{\min}(t)$, $\rho_i(t)$, $\varphi_i(t)$, and the amount of new tasks demanding worker i's service at time slot t, $q_i^{new}(t)$.

$p_i(t) \leftarrow \max\left[p_i^{\min}, \frac{q_i(t)}{2\rho_i(t)r_i(t)}\right]$;
if $a_i(t) = 1$ **then**
  if $q_i^{new}(t) > \mu_i^{\max}$ **then**
    $\lambda_i(t) \leftarrow \mu_i^{\max}$
  else
    $\lambda_i(t) \leftarrow q_i^{new}(t)$ (t);
  end if
else
  $\lambda_i(t) \leftarrow 0$;
end if
Return the $\left[q_i^{new}(t) - \lambda_i(t)\right]$ unaccepted tasks to the original sub-delegator(s);
if $\rho_i(t)\varphi_i(t) - q_i(t) - Q_i(t) \geqslant 0$ **then**
  $s_i(t) \leftarrow 0$;
else
  $s_i(t) \leftarrow q_i(t) - \mu_i(t)$;
  for each worker k in $\mathbf{n}_i^{sub}$ with $r_k(t) \geqslant r_{\min}(t)$ **do**
    Set $d_k(t)$ to denote the amount of workload from sub-delegating tasks in $s_i(t)$ which satisfies $p_k(t) \leqslant p_{\tau_j}$;
    if $d_k(t) > 0$ **then**
      $u_k(t) \leftarrow$ Invoking worker k's RTS-P agent with $q_k^{new}(t) += d_k(t)$;
    end if
    $s_i(t) -= [d_k(t) - u_k(t)]$;
    if $s_i(t) = 0$ **then**
      Exit the **for** loop;
    end if
  end for
end if
Update $q_i(t+1)$ following equation (1)
Update $Q_i(t+1)$ following equation (3)

In our previous work[18], we have proved that the joint task acceptance and sub-delegation decisions made under prices of service fixed by the crowdsourcers are asymptotically optimal compared to an oracle that knows the exact situation of each worker at all times. Although the addition of dynamic worker effort pricing in RTS-P allows workers to adjust their prices according to their changing situations, the joint task acceptance and

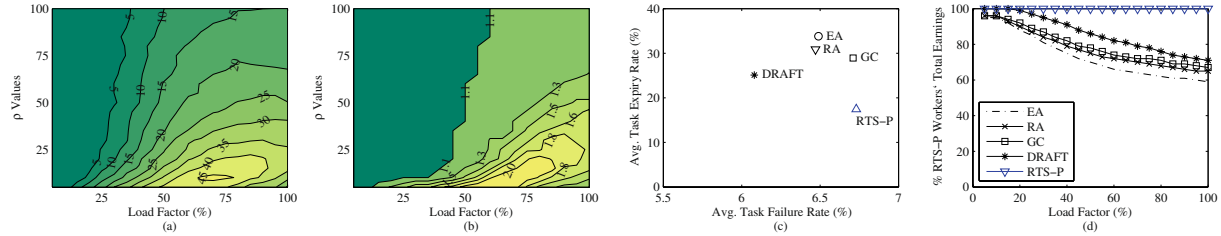

Figure 3. Experimental results under the worker behaviour characteristic setting $R_{\mu}^{h} = +$: (**a**) The percentage of all tasks successfully sub-delegated by RTS-P agents; (**b**) The average sub-delegation chain length; (**c**) The average task expiry rates vs. the average task failure rates; (**d**) The total income as a percentage of the total income of RTS-P agents.

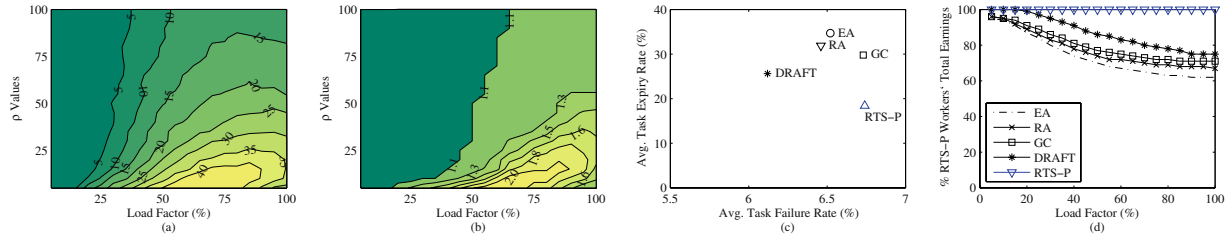

Figure 4. Experimental results under the worker behaviour characteristic setting $R_{\mu}^{h} = 0$: (**a**) The percentage of all tasks successfully sub-delegated by RTS-P agents; (**b**) The average sub-delegation chain length; (**c**) The average task expiry rates vs. the average task failure rates; (**d**) The total income as a percentage of the total income of RTS-P agents.

sub-delegation decisions are made only after the prices have been set. Thus, the original theoretical analysis is still valid for RTS-P. Interested readers may refer to the Analysis section in Yu *et al.*[18].

Results

To evaluate the performance of RTS-P under realistic settings, it is compared with four state-of-the-art approaches through extensive numerical experiments in an HCN based on the *Epinions* trust network dataset[26]. This real-world dataset allows us to construct realistic scenarios for performance comparison. The simulations facilitate understanding of the behavior of RTS-P under different situations.

Model Implementation on a Real Network. The *Epinions* trust network dataset used in the experiments contains $N = 10{,}476$ workers, each represented by a node in the network structure. These nodes are connected by weighted and directed edges. A weight of "+1" represents a trust relationship, while a weight of "−1" represents a distrust relationship. The dataset contains 15,742 trust relationships and 2,170 distrust relationships. Based on this dataset, we construct an HCN populated by worker agents with different characteristics. For a worker agent i in the experiment, $\mathbf{n}_i^{sub}$ consists of other worker agents connected with i through a directed "+1" edge originating from i. We assume that agents do not have global awareness. Thus, a worker agent i may only delegate or sub-delegate tasks to other worker agents in $\mathbf{n}_i^{sub}$. Each worker agent i has an innate trustworthiness $h_i \in [0, 1]$ which dictates its probability of producing satisfactory results for tasks delegated to it in simulations. This value is computed using the number of other agents trusting and distrusting agent i in the dataset following the Beta Reputation Model[27].

Figure 2 illustrates the crowdsourcing network derived from the dataset. The size of a node in the figure reflects the worker agent's h_i value. The larger the size of a node, the more trusted the worker agent is. Let ρ be the workers' average eagerness to work in a given crowdsourcing network:

$$\rho = \frac{1}{N}\sum_{i=1}^{N}\rho_i(t). \tag{21}$$

The value of ρ is varied from 1 to 100 to simulate different levels of workers' general eagerness to work.

The relationship between worker agents' ${\mu_i}^{max}$ values and h_i values is varied in three different ways as denoted by a setting variable $R_{\mu}^{h} \in \{-,\ 0,\ +\}$. Under $R_{\mu}^{h} = -$, ${\mu_i}^{\max}$ values are inversely proportional to h_i values (i.e., workers who produce high quality results have small task processing capacities). Under $R_{\mu}^{h} = 0$, ${\mu_i}^{\max}$ values are independent from h_i values (i.e., workers' task processing capacities are not related to the quality of their work). Under $R_{\mu}^{h} = +$, ${\mu_i}^{\max}$ values are directly proportional to h_i values (i.e., workers who produce high quality results have large task processing capacities). These settings are created to simulate different worker behaviour

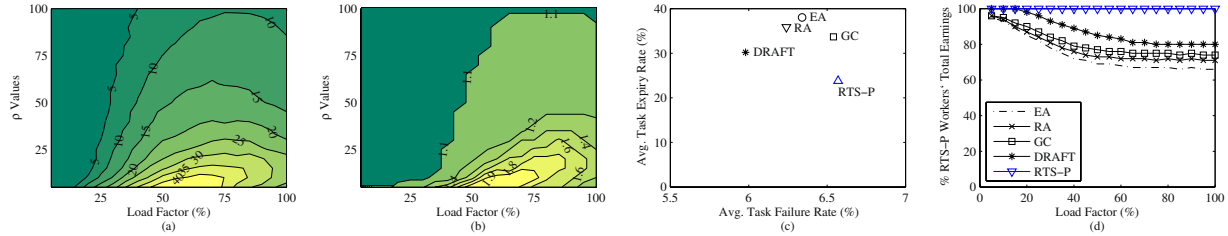

Figure 5. Experimental results under the worker behaviour characteristic setting $R_{\mu}^{h} = -$: (**a**) The percentage of all tasks successfully sub-delegated by RTS-P agents; (**b**) The average sub-delegation chain length; (**c**) The average task expiry rates vs. the average task failure rates; (**d**) The total income as a percentage of the total income of RTS-P agents.

characteristics to study how effectively RTS-P copes with these situations. Files containing the HCNs used in the experiments can be downloaded from http://goo.gl/QyRjTs.

In the experiments, we assume that the outcome for each task is binary (i.e., a task is regarded to be *successful* if the worker agent produces the correct result before the stipulated deadline; otherwise, it is considered *unsuccessful*). A worker agent is only paid if it completes a task successfully. Five duplicate crowdsourcing networks are created in the experiments to study the relative performance of the 5 approaches. They are:

1. The Equality-based Approach (EA): tasks are uniformly distributed among worker agents in a crowdsourcer agent j's $\mathbf{n}_j^{sub}$ without regard to their reputations.
2. The Reputation-based Approach (RA): the probability for a worker agent i to be selected by a crowdsourcer agent j is determined by its reputation standing among all worker agents in $\mathbf{n}_j^{sub}$ following the softmax choice rule[28].
3. The Global Considerations (GC) Approach: a crowdsourcer agent j adjusts the probability for tasks to be delegated to each worker agent in $\mathbf{n}_j^{sub}$ following the approach in Grubshtein *et al.*[13].
4. The DRAFT Approach: worker agents make task request acceptance decisions following the approach in Yu *et al.*[15].
5. The RTS-P Approach: worker agents follow the approach proposed in this paper.

Approaches 1 to 4 do not support task sub-delegation.

The overall workload level in the experimental HCN is adjusted to simulate different operational conditions. As the workload is measured in relative terms to the collective task processing capacity of the worker agents, we compute the maximum throughput θ of a given crowdsourcing network as $\theta = \sum_{i=1}^{N} h_i \mu_i^{\max}$. At each time step, a proportion of the agents, p_a, from the network are selected at random to act as crowdsourcers from which tasks originate. Based on empirical studies of the mTurk crowdsourcing system[29,30], the ratio between crowdsourcers and workers is close to 1:20. Thus, we set $p_a = 5\%$. The workload for a given crowdsourcing network is measured by the *Load Factor* (LF). It is calculated as $LF = \frac{w_{req}}{\theta}$, where w_{req} is the amount of new workload generated by crowdsourcer agents at each time slot. In the experiments, the LF value ranges from 5% to 100% in 5% increments. Under each LF setting, the simulation is run for $T = 10{,}000$ time slots. Task deadlines are randomized. On average, a task must be completed within 5 time slots after it is first assigned to a worker agent.

Simulation Results. As shown in Figs 3(a), 4(a) and 5(a), as LF increases, RTS-P agents sub-delegate an increasing percentage of their workload to other trusted worker agents in the network to mitigate delays for all worker behaviour characteristic settings. If the worker agents are more eager to work as indicated by larger ρ values (i.e., worker agents prefer working on their tasks instead of sub-delegating them), fewer tasks are sub-delegated. The highest percentage of tasks is sub-delegated under low general eagerness to work and high workload conditions. As ρ increases, this peak sub-delegation percentage shifts towards higher workload conditions. Under $R_{\mu}^{h} = +$ (Fig. 3(a)), as LF approaches 100%, an increasing percentage of tasks are sub-delegated. However, when ρ values are small and LF values are large (i.e. the general eagerness to work is low while the overall workload is high), the trend reverses. Under $R_{\mu}^{h} = 0$ (Fig. 4(a)), fewer trustworthy RTS-P agents are able to accommodate new tasks. Thus, there appears to be a systemic "downward shift" of the contour lines in the figure, indicating fewer tasks are being successfully sub-delegated even when agents show high willingness to work (i.e., larger ρ values). Under $R_{\mu}^{h} = -$ (Fig. 5(a)), workers who produce good results have lower productivity. The systemic downward shift of the contour lines is more pronounced compared to Fig. 4(a). The same trends can be observed for the average sub-delegation chain lengths in Figs 3(b), 4(b) and 5(b). This indicates that as the R_{μ}^{h} value decreases (i.e., the dichotomy between workers' productivity and quality of work increases), RTS-P adapts its strategy by reducing task sub-delegation throughout the HCN, especially in cases in which workers are highly eager to work and overall workload is high.

The trade-off between the average task failure rates and the average task expiry rates achieved by all five approaches under different R_{μ}^{h} settings is shown in Figs 3(c), 4(c) and 5(c). The overall effect of the RTS-P strategy is to significantly reduce the average task expiry rate (i.e., improving the timeliness of obtaining task results). This comes at the expense of a slightly lower average task result quality. The average task failure rates of RTS-P under

all R_μ^h settings are comparable to those of GC and consistently stay below 7%. The average task expiry rates of RTS-P under all R_μ^h settings are significantly lower than all other approaches (more than 20% lower than the best performing approach - DRAFT - under the most challenging situation of $R_\mu^h = -$). By making workers sacrifice task quality to a small extent, RTS-P significantly increases the total number of tasks completed by an HCN, putting the *Parrondo's Paradox*[31] to work on a large scale.

Figures 3(d), 4(d) and 5(d) illustrate the total earnings derived by worker agents following the five different approaches under different R_μ^h settings. The results are averaged over all ρ value settings in the experiments. As RTS-P worker agents consistently achieve the highest total earnings, RTS-P is used as the benchmark for all other approaches. It can be observed that the total earnings achieved by EA, RA and GC worker agents as a percentage of those achieved by RTS-P worker agents start to drop under low LF conditions. As LF approaches 100%, EA, RA and GC worker agents achieve around 70% of RTS-P worker agents' total earnings. DRAFT worker agents earned the same amounts as RTS-P under low LF conditions. The performance of DRAFT worker agents starts to deteriorate under medium LF conditions. As LF approaches 100%, DRAFT worker agents achieve around 75–80% of RTS-P worker agents' total earnings. Under the challenging situation of $R_\mu^h = -$ (Fig. 5(d)), the advantage of RTS-P over other approaches stalls under high LF conditions. However, when averaged over all LF conditions, RTS-P consistently maintains at least a 14.9% advantage on average over the best performing approach - DRAFT - under all R_μ^h settings. Overall, RTS-P significantly outperforms existing approaches and its performance is robust in the face of different worker behaviour characteristics.

Discussion

To summarize, the proposed RTS-P approach leverages Lyapunov stochastic network queueing theory to make joint decisions on task acceptance, sub-delegation, and effort pricing. To our knowledge, RTS-P is the first principled computational approach to assist hierarchical crowdsourcing workers to dynamically sub-delegate tasks and adjust the price of their services based on changing situational factors while ensuring efficient utilization of their collective productivity. High resolution numerical experiments show that RTS-P is robust under various worker behaviour characteristics and significantly outperforms state-of-the-art approaches, especially under conditions of high workload. As recent empirical results show that such conditions are common among crowdsourcing projects[32], RTS-P can be a useful tool to help HCNs mitigate the adverse effects of herding through efficiently harnessing the available human resources.

Furthermore, a worker can adjust the $\rho_i(t)$ variable value of his RTS-P agent to take on different roles in a crowdsourcing network. Since each worker can establish his trust relationships with a set of known workers, a worker can focus on tracking workers' historical performance and building up his list of trusted workers. With such a list, he could reduce the $\rho_i(t)$ value of his RTS-P agent so as to sub-delegate most of the accepted tasks to other trusted workers, thereby deriving most of his earnings from sub-delegation. By doing so, these workers can serve as task brokerage agents and provide a useful service to the crowdsourcing network. Other workers who are able to spend more time and effort completing tasks can increase the $\rho_i(t)$ values of their RTS-P agents so as to accept more tasks and sub-delegate only when absolutely necessary, thereby deriving most of their earnings through completing tasks.

RTS-P helps each worker compute a suitable effort price under different situations so that their collective benefits can be maximized. As a task propagates through a sub-delegation chain, subsequent price proposals are subject to Constraint (11) which dictates that workers with prices exceeding the current price for the task being considered for sub-delegation should not be selected (as this will cause the sub-delegator to incur a loss). Thus, there will never be a situation in which a crowdsourcer is forced to accept prices higher than what he can afford. Rather, prices reflect the current demand placed on the workers, and crowdsourcers can decide to either wait or increase their budgets. Such a signal helps coordinate the crowdsourcers' actions to reduce herding in the crowdsourcing network.

Following this work, we foresee a series of interesting research directions. RTS-P works well for workers who have accumulated some historical performance data in the system. For workers new to a system, there is a large body of literature on reputation bootstrapping[33–35]. Methods from these works can be put in front of RTS-P as a module to build up a system workflow to help new workers build up their track records. The most important direction of this field lies in understanding the dynamics of how the volume of task requests for a worker varies with his reputation and effort pricing. Large-scale user studies in crowdsourcing networks will be needed to investigate this topic. Furthermore, this field will also benefit from more detailed empirical evidence on how workers decide on what types of tasks to accept and what incentive mechanisms are effective.

In conclusion, the proposed approach and results provide a stepping stone towards more efficient management of large-scale hierarchical crowdsourcing based on evidence about workers' behaviours, and ultimately help improve the collective productivity of our connected world.

References

1. Silberzahn, R. & Uhlmann, E. L. Crowdsourced research: Many hands make tight work. *Nature* **526**, 189–191 (2015).
2. Derex, M., Beugin, M.-P., Godelle, B. & Raymond, M. Experimental evidence for the influence of group size on cultural complexity. *Nature* **503**, 389–391 (2013).
3. Dankulov, M. M., Melnik, R. & Tadić, B. The dynamics of meaningful social interactions and the emergence of collective knowledge. *Sci. Rep.* **5**, doi:10.1038/srep12197 (2015).
4. González-Bailón, S., Borge-Holthoefer, J., Rivero, A. & Moreno, Y. The dynamics of protest recruitment through an online network. *Sci. Rep.* **1,** doi:10.1038/srep00197 (2011).
5. Khatib, F. *et al.* Crystal structure of a monomeric retroviral protease solved by protein folding game players. *Nat. Struct. Mol. Biol.* **18**, 1175–1177 (2011).

6. Kyba, C. C. M. *et al.* Citizen science provides valuable data for monitoring global night sky luminance. *Sci. Rep.* **3,** doi:10.1038/srep01835 (2013).
7. Breuer, L. *et al.* Hydrocrowd: A citizen science snapshot to assess the spatial control of nitrogen solutes in surface waters. *Sci. Rep.* **5,** doi:10.1038/srep16503 (2015).
8. Hellerstein, J. M. & Tennenhouse, D. L. Searching for jim gray: A technical overview. *Commun. ACM* **54**, 77–87 (2011).
9. Chen, J.-J., Tan, L. & Zheng, B. Agent-based model with multi-level herding for complex financial systems. *Sci. Rep.* **5,** doi:10.1038/srep08399 (2015).
10. Zhang, J.-Q., Huang, Z.-G., Wu, Z.-X., Su, R. & Lai, Y.-C. Controlling herding in minority game systems. *Sci. Rep.* **6,** doi:10.1038/srep20925 (2016).
11. Rutherford, A. *et al.* Limits of social mobilization. *Proc. Nat. Acad. Sci. USA* **110**, 6281–6286 (2013).
12. Michelucci, P. & Dickinson, J. L. The power of crowds. *Science* **351**, 32–33 (2016).
13. Grubshtein, A., Gal-Oz, N., Grinshpoun, T., Meisels, A. & Zivan, R. Manipulating recommendation lists by global considerations. In *ICAART*, 135–142 (2010).
14. Yu, H., Shen, Z., Miao, C. & An, B. A reputation-aware decision-making approach for improving the efficiency of crowdsourcing systems. In *AAMAS*, 1315–1316 (2013).
15. Yu, H., Miao, C., An, B., Leung, C. & Lesser, V. R. A reputation management model for resource constrained trustee agents. In *IJCAI*, 418–424 (2013).
16. Neely, M. J. Stochastic Network Optimization with Application to Communication and Queueing Systems (Morgan and Claypool Publishers, 2010).
17. Sornette, D., Maillart, T. & Ghezzi, G. How much is the whole really more than the sum of its parts? $1 \boxplus 1 = 2.5$: Superlinear productivity in collective group actions. *PLoS ONE* **9,** doi:10.1371/journal.pone.0103023 (2014).
18. Yu, H., Miao, C., Shen, Z., Leung, C., Chen, Y. & Yang, Q. Efficient task sub-delegation for crowdsourcing. In *AAAI*, 1305–1311 (2015).
19. Guazzini, A., Vilone, D., Donati, C., Nardi, A. & Levnajić, Z. Modeling crowdsourcing as collective problem solving. *Sci. Rep.* **5,** doi:10.1038/srep16557 (2015).
20. Moussaïd, M. *et al.* Experimental study of the behavioural mechanisms underlying self-organization in human crowds. *Proc. R. Soc. B* **276**, 2755–2762 (2009).
21. Heymann, P. & Garcia-Molina, H. Turkalytics: Analytics for human computation. In *WWW*, 477–486 (2011).
22. Yu, H., Shen, Z., Leung, C., Miao, C. & Lesser, V. R. A survey of multi-agent trust management systems. *IEEE Access* **1**, 35–50 (2013).
23. Burnett, C. & Oren, N. Sub-delegation and trust. In *AAMAS*, 1359–1360 (2012).
24. Schwartenbeck, P. *et al.* Evidence for surprise minimization over value maximization in choice behavior. *Sci. Rep.* **5,** doi:10.1038/srep16575 (2015).
25. Ye, Q., Xu, M., Kiang, M., Wu, W. & Sun, F. In-depth analysis of the seller reputation and price premium relationship: A comparison between eBay US and Taobao China. *J. Electron. Commer. Res.* **14**, 1–10 (2013).
26. Leskovec, J., Huttenlocher, D. & Kleinberg, J. Signed networks in social media. In *CHI*, 1361–1370 (2010).
27. Wang, Y. & Singh, M. P. Formal trust model for multiagent systems. In *IJCAI*, 1551–1556 (2007).
28. Vogiatzis, G., MacGillivray, I. & Chli, M. A probabilistic model for trust and reputation. In *AAMAS*, 225–232 (2010).
29. Ipeirotis, P. G. Analyzing the Amazon Mechanical Turk marketplace. *XRDS* **17**, 16–21 (2010).
30. Ross, J., Irani, L., Silberman, M. S., Zaldivar, A. & Tomlinson, B. Who are the crowdworkers?: Shifting demographics in Mechanical Turk. In *CHI EA*' **10,** 2863–2872 (2010).
31. Shu, J.-J. & Wang, Q.-W. Beyond Parrondo's Paradox. *Sci. Rep.* **4,** doi:10.1038/srep04244 (2014).
32. Sauermann, H. & Franzoni, C. Crowd science user contribution patterns and their implications. *Proc. Nat. Acad. Sci. USA* **112**, 679–684 (2014).
33. Burnett, C., Norman, T. J. & Sycara, K. Bootstrapping trust evaluations through stereotypes. In *AAMAS*, 241–248 (2010).
34. Nguyen, H. T., Yang, J. & Zhao, W. Bootstrapping trust and reputation for web services. In *CEC*, 41–48 (2012).
35. Tibermacine, O., Tibermacine, C. & Cherif, F. Regression-based bootstrapping of web service reputation measurement. In *ICWS*, 377–384 (2015).

Acknowledgements

This research is supported by the National Research Foundation, Prime Minister's Office, Singapore under its IDM Futures Funding Initiative; the Lee Kuan Yew Post-Doctoral Fellowship Grant; and the National Natural Science Foundation of China (NSFC) under Grant No. 61572471, No. 61502456, and No. 61572004.

Author Contributions

H.Y. conceived the framework. C.M., C.L., S.F., Y.C., Q.Y. and V.R.L. verified the framework. H.Y., C.M. and C.L. conceived and conducted the experiments. H.Y., Y.C., S.F. and V.R.L. drafted and revised the manuscript. All authors have reviewed the manuscript.

Additional Information

Competing financial interests: The authors declare no competing financial interests.

Publisher's note: Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.