Stimulating Skill Evolution in Market-Based Crowdsourcing

Benjamin Satzger, Harald Psaier, Daniel Schall, and Schahram Dustdar

Distributed Systems Group, Vienna University of Technology
Argentinierstrasse 8/184-1, A-1040 Vienna, Austria
{satzger,psaier,schall,dustdar}@infosys.tuwien.ac.at
http://www.infosys.tuwien.ac.at

Abstract. Crowdsourcing has emerged as an important paradigm in human problem-solving techniques on the Web. One application of crowdsourcing is to outsource certain tasks to the crowd that are difficult to implement in software. Another potential benefit of crowdsourcing is the on-demand allocation of a flexible workforce. Businesses may outsource tasks to the crowd based on temporary workload variations. A major challenge in crowdsourcing is to guarantee high-quality processing of tasks. We present a novel crowdsourcing marketplace that matches tasks to suitable workers based on auctions. The key to ensuring high quality lies in skilled members whose capabilities can be estimated correctly. We present a novel auction mechanism for skill evolution that helps to correctly estimate workers and to evolve skills that are needed. Evaluations show that this leads to improved crowdsourcing.

Keywords: Human-centric BPM, Crowdsourcing, Online communities, Task markets, Auctions, Skill evolution.

1 Introduction

Today, ever changing requirements force in-house business processes to rapidly adapt to changing situations in order to stay competitive. Changes involve not only the need for process adaptation, but also, require an additional inclusion of new capabilities and knowledge, previously unavailable to the company. Thus, outsourcing of parts of business processes became an attractive model. This work, in particular, focuses on a distinguished recent type of outsourcing called *crowdsourcing*. The term crowdsourcing describes a new web-based business model that harnesses the creative solutions of a distributed network of individuals [4], [18]. This network of humans is typically an open Internet-based platform that follows the *open world* assumption and tries to attract members with different knowledge and interests. Large IT companies such as Amazon, Google, or Yahoo! have recognized the opportunities behind such *mass collaboration systems* [8] for both improving their own services and as business case. In particular, Amazon focuses on a task-based marketplace that requires explicit collaboration. The most prominent platform they currently offer is *Amazon Mechanical Turk* (AMT) [3].

S. Rinderle-Ma, F. Toumani, and K. Wolf (Eds.): BPM 2011, LNCS 6896, pp. 66–82, 2011.

Requesters are invited to issue *human-intelligence tasks* (HITs) requiring a certain qualification to the AMT. The registered customers post mostly tasks with minor effort that, however, require human capabilities (e.g., transcription, classification, or categorization tasks [11]).

Reasons for companies to outsource tasks to a crowd platform are flexibility, high availability, and low costs. Workers from all over the world and different timezones are allowed to register with the platform. With 90% of tasks processed at a cost of $0.10 and less, the same task is usually also offered to multiple AMT workers. However, crowdsourcing platforms are loosely-coupled networks with members of different interests, working style, and cultural background. The flexible crowd structure allows workers to join and leave at any time. This hampers to predict or guarantee the quality of a task's result. There are crowd providers such as CrowdFlower [6] that broker crowd resources to customers to overcome quality and reliability issues.

However, managing and adapting the crowd's skills and resources in an automated manner remains challenging. Crowd customers prefer fully automated deployment of their tasks to a crowd, just as in common business process models. In this paper, we base our solution on the service-oriented architecture (SOA) paradigm. SOAs are an ideal grounding for distributed environments. With their notion of the participants as services and registries, resources can be easily and even automatically discovered for composing whole business processes. A plethora of standards supports seamless integration and registration of new services, and provides protocols for communication, interaction and control of the components. Altogether, we believe SOAs provide an intuitive and convenient technical grounding to automate large-scale crowdsourcing environments. Important to note, today SOA not only includes software-based services, but also *Human-Provided Services* [16] and *BPEL4People* [2] for human interactions in business processes and allow to host mass collaboration environments.

The main challenges addressed in this work relate to building and managing an automated crowd platform. It is not only of importance to find suitable workers for a task and to provide the customer with satisfying quality, but also, to maintain a motivated base of crowd members and provide stimulus for learning required skills. Only a recurring, satisfied crowd staff is able to ensure high quality and high output. As any crowd, fluctuations must be compensated and a *skill evolution* model must support new and existing crowd workers in developing their capabilities and knowledge. Finally, the standard processes on such a platform should be automated and free from intervention to handle the vast amount of tasks and to make it compatible with a SOA approach. Atop, the model should increase the benefit of all participants. In detail, we provide the following contributions in this paper:

- *Automated matching and auctions.* For providing a beneficial distribution of the tasks to the available resources we organize auctions according to novel mechanisms.

- *Stimulating skill evolution.* In order to bootstrap new skills and unexperienced workers we provide skill evolution by integrating assessment tasks into our auction model.
- *Evaluation.* Experiments quantify the advantages of a skill evolution based approach in comparison to traditional auctions.

The remainder of this paper is structured as follows: In Section 2 related work is discussed. Section 3 describes the design of our crowdsourcing system, including its actors and their interaction. Then, Section 4 details our adaptive auction mechanisms and Section 5 presents the conducted experiments and discusses their results. Section 6 concludes the paper and points to future work.

2 Related Work

In this work we position crowdsourcing in a service-oriented business setting by providing automation. In crowdsourcing environments, people offer their skills and capabilities in a service-oriented manner. Major industry players have been working towards standardized protocols and languages for interfacing with people in SOA. Specifications such as WS-HumanTask [10] and BPEL4People [2] have been defined to address the lack of human interactions in service-oriented businesses [14]. These standards, however, have been designed to model interactions in closed enterprise environments where people have predefined, mostly static, roles and responsibilities. Here we address the service-oriented integration of human capabilities situated in a much more dynamic environment where the availability of people is under constant flux and change [5]. The recent trend towards *collective intelligence* and crowdsourcing can be observed by looking at the success of various Web-based platforms that have attracted a huge number of users. Well known representatives of crowdsourcing platforms include Yahoo! Answers [20] (YA) and the aforementioned AMT [3]. The difference between these platforms lies in how the labor of the crowd is used.

YA, for instance, is mainly based on interactions between members. Questions are asked and answered by humans, thereby lacking the ability to automatically control the execution of tasks. The interesting aspect of YA relevant for our crowdsourcing approach is the role of *two-sided markets* [13]. In YA, users get 100 points by signing-up to the platform [19]. For each answer being provided, users get additional points (more points if the answer is selected as best answer). However, users get negative points if they ask questions, thereby encouraging members to provide answers. Based on the rewarding scheme in YA, users tend to have either role – being answerer or asker – instead of having both roles. In the context of YA and human-reviewed data, [17] provided an analysis of data quality, throughput and user behavior. In contrast, AMT offers access to the largest number of crowdsourcing workers. With their notion of HITs that can be created using a Web service-based interface they are closely related to our aim of mediating the capabilities of crowds to service-oriented business environments. According to one of the latest analysis of AMT [11], HIT topics include, first

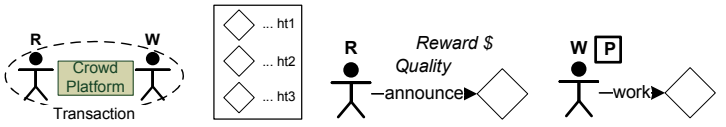
of all, transcription, classification, and categorizations tasks for documents and images. Furthermore, there is also tasks for collecting data, image tagging, and feedback or advice on different subjects. The major challenges are to find on request skilled workers that are able to provide high quality results for a particular topic (e.g., see [1]), to avoid spamming, and to recognize low-performers. To the best of our knowledge, these problems are still not faced by AMT. In this work we focus on those issues. The shortcoming of most existing real platforms is the lack of detailed *skill information*. Most platforms have a simple measure to prevent workers (in AMT, a threshold of task success rate can be defined) from claiming tasks. In [15], the automated calculation of expertise profiles and skills based on interactions in collaborative networks was discussed. In this work, we introduce a novel auction mechanism to promote skill evolution in crowdsourcing environments. Compared to open bidding systems used by popular online auction markets such as eBay [9], our auction mechanism is a closed bidding system similar to public procurement. In [7], a model of crowdsourcing is analyzed in which strategic players select among, and subsequently compete in, contests. In contrast, the presented approach in this paper supports the evolution of worker skills through auction mechanisms.

3 Design of Marketplaces in Crowdsourcing

The core activity in task-based crowd environments is members providing their labor by processing tasks. In this section, we explain our idea of task-based crowdsourcing on a market-oriented platform. The aim is to organize and manage the platform to the satisfaction and benefit of all participants; crowd members and platform provider. We will now introduce the basic design of the proposed crowdsourcing environment.

3.1 Skill-Based Task Markets

In task markets different stakeholders can be identified. Generally, there is the requesters and workers representing the registered members of a crowd marketplace. The task of the third stakeholder, the crowd operator in between, is to manage the crowd task auctions. To satisfy any of the stakeholders the operator must assure that the requesters obtain a result of high quality in a timely manner. On the other hand, the workers would like to have tasks available whenever they are motivated to work and are interested in a high reward for processing a task. The operator itself works towards a long-term profit. To bootstrap the skill-based system, each member interested in offering of processing tasks is required to create a profile containing information about her/his interests and skills. The basic interactions and an external view on the proposed crowdsourcing environment are depicted in Fig. 1. The crowdsourcing environment consists of members who can participate in transactions (see Fig. 1(a)). Within a particular transaction a member can either adopt the role of a requester **R**, who initiates a transaction by announcing tasks (see Fig. 1(b)), or the role of a worker



(a) Crowd (b) Tasks (c) Requester (d) Worker

Fig. 1. Crowd environment building blocks and interaction of stakeholders

W, who processes a task. We propose a crowdsourcing marketplace that handles transactions transparently for its members; requesters and workers do not communicate directly, but only with the crowdsourcing marketplace in between. We argue that this standardized style of interaction is less prone for misconceptions and more efficient because it allows members getting used to the system. Tasks (Fig. 1(b)) are created by requesters based on their current needs. Requesters initiate a transaction by submitting a task to the marketplace, with additional information about the amount of money he is willing to pay for the processing of the task and additional requirements (Fig. 1(c)). It is the responsibility of the marketplace operator to find a suitable worker, to submit the task to the worker, to collect the result, and to transmit it to the requester.

The interaction of a worker with the market platform is initiated by the latter by asking a member whether s/he is interested in processing a task (Fig. 1(d)). This interest can be expressed by bidding for the task. Workers have skill profiles denoted by the symbol **P**. These profiles are not statically defined, but are updated based on the level of delivered task quality. This procedure ensures an up-to-date view on workers' capabilities and skills. Based on the bids and background information about the bidders, the system selects one or multiple workers, who are then asked to process the task.

3.2 Towards Auction-Based Crowdsourcing

Auctions are a very old idea already used by the Babylonians but still an active area of research. The rise of e-commerce has drastically increased the number and diversity of goods traded via auctions. Many recently installed markets, such as energy or pollution permit markets, are based on auctions [12]. There are many different flavors of auctions differing in the number of items considered (single/multi item), the number of buyers and sellers (demand/supply/double auction), the bidding procedure (open/closed bids and ascending/descending), and how the price is determined (e.g., first/second price); however, four standard types are widely used [12]. They all assume a single seller and multiple buyers (demand auction) and, in their simplest forms, a single item to sell (single-item auction). The so-called English auction is an ascending open-bid auction, the Dutch auction is a descending open-bid auction. The other two standard auction types are closed-bid auctions, i.e., each bidder submits a single bid which is hidden for other buyers. We use an adapted version of a closed-bid auction; a single auction deals with the matching of one task to one or many crowd

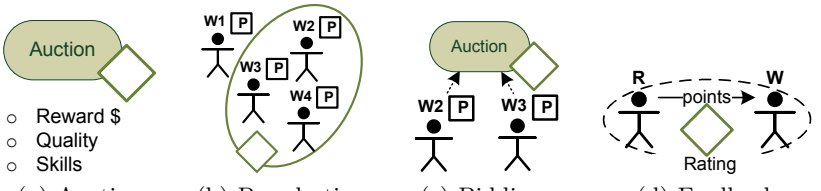
workers (single-item demand auction). We will introduce our crowdsourcing auction mechanisms in the following sections.

4 Auction-Based Task Assignment

In the following we discuss the steps involved in transaction processing and outline the novel idea of skill evolution.

4.1 Processing of Transactions

Figure 2 illustrates the steps involved in the internal processing of a transaction. In the qualification step the marketplace identifies all members capable of processing the task (Fig. 2(a)), based on the task description and the members' profiles. The preselection chooses a limited number of the most suitable workers (Fig. 2(b)) to have a reasonable amount of participants for the auction. The preselection step helps to avoid a flooding of auction announcements. In this way members are only exposed to tasks/auctions for which they actually have a realistic chance of being accepted as worker. Due to the voluntary, open nature of crowdsourcing environments, not all preselected workers may decide to follow the invitation to compete in an auction.



(a) Auction (b) Preselection (c) Biddings (d) Feedback

Fig. 2. Internal processing of a transaction

This fact is depicted by the transition phase between Fig. 2(b) and Fig. 2(c) where only a subset of preselected workers decides to participate. The auction phase (Fig. 2(c)) allows each participant to submit an offer and finally, a winner is determined who is supposed to process the task. In the case of a successful processing the marketplace returns the final result to the requester, handles the payment, and allows the requester to give feedback about the transaction in the form of a rating (Fig. 2(d)).

As mentioned before, tasks come with a description, a maximum amount of money the requester is willing to pay and further requirements, i.e., time requirements and quality requirements. The former is typically given in the form of a deadline, the latter could range from a simple categorization (e.g., low, high) to sophisticated quality requirement models. Each worker has a self-provided profile describing her/his skills and, additionally, the marketplace operator is keeping track of the actual performance. We propose to maintain a performance

value per user and skill, encoded as tuple consisting of the observed performance and the confidence in that value. The input used to generate these performance values comes from the ratings of the requesters and a check whether the deadline was met, which can be performed by the system without feedback from the requester. The qualification phase is based on a matching of the task description to the skills of the members considering their performance and confidence values. Higher requirements impose higher standards on the performance of the member. The result of this matching is a boolean value indicating whether a member is meeting the minimum requirements. In the next step, the preselection, the qualified members are ranked based on skill, performance, and the confidence in the performance; only the top-k members are chosen to participate in the auction. This helps to reduce the number of auction requests to members in order to avoid spamming members and to spare members the frustration caused by not winning the auction. The marketplace operator as the auctioneer hides parts of the task's data. Workers only see the task description and the time and quality requirements, but not the associated price determined by the requester. The auction is performed as a closed bid auction, whereas each participant is only allowed one bid. At the end of the auction a winner is determined based on the amounts of the bids and the performance-confidence combination of the bidders' skills. If all bids are higher than the amount the requester is willing to pay the auctioneer would typically reject all bids and inform the requester that the task cannot be assigned under the current conditions. In this case the requester could change the task by increasing the earnings, lowering the quality requirements or extending the deadline and resubmit the task. With a selection strategy outlined in more detail in the next section the marketplace assigns the task to the worker for processing. After the processing of the task by the worker and the receipt of rating information, the performance of the worker is adjusted and the confidence value is increased.

Technically, an aptitude function estimates how well workers are suited for handling a task. It is used as basis for qualification and preselection and can be formally defined as

$$aptitude : W \times T \rightarrow [0, 1], \tag{1}$$

where W is the set of workers and T represents tasks. $aptitude(w, t) = 1$ would mean that worker $w \in W$ is perfectly qualified for handling task $t \in T$. A mapping to zero would represent a total inaptness. Similarly, a ranking function is used to rank workers' bids:

$$rank : W \times T \times B \rightarrow [0, 1], \tag{2}$$

where B is the set of bids. In addition to the aptitude, the *rank* function also takes monetary aspects, contained in bid $b \in B$, into account. A property of a sound ranking function is that if two workers have the same aptitude for a task then the one with the lower bid will have a higher rank. The *aptitude* function is used for performing qualification and preselection. As auction admittance strategy you can either admit all workers with an aptitude higher than a certain threshold, the top-k workers according to aptitude, or a combination of the two

strategies. The ranking function is used to determine the winner of an auction: the highest ranked worker.

4.2 Skill Evolution

The concepts discussed so far provide the fundamentals for automated matching of tasks to workers. As outlined previously, a further major challenge hampering the establishment of a new service-oriented computing paradigm spanning enterprise and open crowdsourcing environments are quality issues. In our scenario this is strongly connected to correctly estimating the skills of workers. One approach for increasing the confidence in worker skills are qualification tasks, with the shortcoming that these tasks would need to be created (manually) by the requesters who have the necessary knowledge. This implies a huge overhead for the testing requester; s/he is also the only one who benefits from the gathered insights. Here, we take a different approach by integrating the capability of confidence management into the crowdsourcing platform itself. Instead of having point-to-point tests, we propose the automated assessment of workers to unburden requesters in inspecting workers' skills. We believe that this approach offers great potential for the (semi-)automatic inclusion of crowd capabilities in business environments. The first challenge one needs to address is to cope with the "hostile" environment in which computing is performed. Workers may cheat on results (e.g., copy and paste of existing results available in the platform), spam the platform with unusable task results, or even provide false information. A well-known principle in open, Web-based communities is the notion of *authoritative* sources that act as points of references. For example, this principle has been applied on the Web to propagate trust based on *good seeds*. Our idea of skill evolution is in a manner similar. We propose the *automatic assessment* of workers where confidence values are low. For example, newcomers who recently signed up to the platform may be high or low performers. To unveil the tendency of a worker, we create a hidden 'tandem' task assignment comprising a worker whose skills are known (high performer) with a high confidence and a worker where the crowdsourcing platform has limited knowledge about its skills (i.e., low confidence). The next step is that both workers process the *same* task in the context of a requester's (real) task. However, only the result of the high confidence worker is returned to the requester, whereas the result of the low confidence worker is compared against the delivered reference. This approach has advantages and drawbacks. First, skill evolution through tandem assignments provides an elegant solution to avoid training tasks (assessments are created automatically and managed by the platform) and also implicitly stimulates a learning effect. Of course, the crowdsourcing platform cannot charge the requester for tandem task assignments since it mainly helps the platform to better understand the true skill (confidence) of a worker. Thus, the platform must pay for worker assessments. As we shall show later in our evaluation, performing assessments provides the positive effect that the overall quality of provided results and thus requester satisfaction increases due to a better understanding of worker skills. We embed skill evolution in our crowdsourcing platform as follows.

After the winner of an auction has been determined it is evaluated whether an assessment task is issued to further workers. The function *assess* outputs 1 if an assessment task is to be assigned to a worker and 0 otherwise.

$$assess : W \times T \times B \times W \rightarrow \{0, 1\} \tag{3}$$

An input tuple (w, t, b, w_r) checks whether tasks $t \in T$ is to be assigned to $w \in W$ who offered bid $b \in B$. Worker $w_r \in W$ is the reference worker, in our case the worker who has won the corresponding auction and who will thus process the same task.

5 Implementation and Evaluation

In this section we discuss implementation aspects, introduce the detailed design of our experiments and present results.

5.1 Simulation Environment

We have implemented a Java-based simulation framework that supports all previously introduced concepts and interactions between requesters, the platform, and workers. All of the above introduced functions (1)-(3) have been implemented in our framework. Due to space limitations, we do not present a detailed discussion on implementation aspects. The interested reader can find details regarding the prototype as well as a Web-based demo online[1].

5.2 Experiment Design and Results

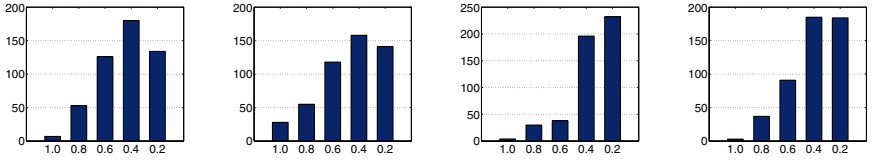
An evaluation scenario consists of a set of workers W and a set of requesters R. In every round of the simulation each requester usually announces a task. An auction is conducted for each announced task t, which consists of a description of the skills needed for its processing, an expected duration, a deadline, and the expected quality. High quality requirements indicate highly sophisticated and demanding tasks. For each worker w and skill s the platform maintains a performance value $pfmc(w, s)$ and a confidence in that value $cnfd(w, s)$. This observed performance value is derived from requester ratings; if it is based on many ratings the confidence is close to one, if there are only a few ratings available the confidence is close to zero. Based on task t's skill requirements and a worker w's performance/confidence values for these skills it is possible to calculate the expected performance $pfmc(w, t)$ and confidence $cnfd(w, t)$ for that task. For the evaluation we assume that each worker w has a certain performance $pfmc_{real}(w, s)$ for a skill s which is hidden but affects the quality of the results. Requesters rate the workers based on the results which in turn is the basis for the observed performance and confidence values. We assume that the processing of tasks demanding a certain skill causes a training effect of that skill,

[1] `http://www.infosys.tuwien.ac.at/prototyp/Crowds/Markets_index.html`

i.e., $pfmc_{real}(w, s)$ increases. For the sake of simplicity in the evaluation we assume that there is only one skill. This does not change fundamental system properties and one skill allows to extrapolate the behavior of multiple skills. Whether a single or multiple skills are considered indeed affects qualification, preselection, bidding, and rating but all the mechanisms are naturally extensible from one to multiple skills by performing the same computations for each skill and a final combination step. Hence, $pfmc : W \times S \to [0, 1]$ and $pfmc : W \times T \to [0, 1]$ are reduced to $pfmc : W \to [0, 1]$. The same holds for $cnfd$ and $pfmc_{real}$.

For the simulation we have created 500 workers with random values for $pfmc_{real}(w)$ and $cnfd(w)$ according to a normal distribution $\mathcal{N}(\mu, \sigma^2)$. The initial performance value $pfmc(w)$ is set according to the formula $pfmc_{real}(w) + \mathcal{N}(0, 1-cnfd(w))$ which ensures that for high confidence the expected deviation of $pfmc(w)$ from $pfmc_{real}(w)$ is small and for low confidence values it is high, respectively. All values are restricted to the range of $[0, 1]$. The following figures illustrate the simulation setup in detail.

Scenario 1 assumes that there are three requesters (i.e., typically three tasks are issued in every round). It is an environment in which skilled workers are rare and the confidence in the workers' performance is relatively low, i.e., there are many workers who have few ratings. The real performance $pfmc_{real}(w)$ for a worker w is drawn according to $\mathcal{N}(0.3, 0.25)$, the confidence value $cnfd(w)$ is randomly generated by $\mathcal{N}(0.2, 0.25)$. Given the two generated values $pfmc_{real}(w)$ and $cnfd(w)$ the observed performance is randomly drawn according to $\mathcal{N}(pfmc_{real}(w), 1-cnfd(w))$. Hence, a low confidence in the performance leads to highly distorted values for $pfmc(w)$, higher confidence values decrease the variance. Figure 3 gives a detailed view on the statistical distributions of the workers' performance and confidence in our experiments.

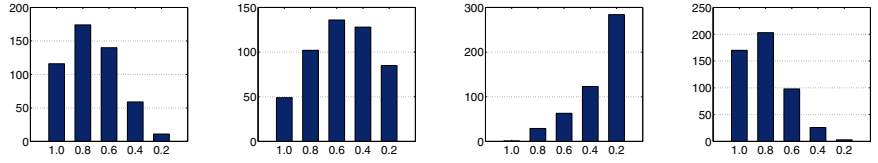


(a) real pfmc (b) obs pfmc (c) diff real/obs (d) cnfd

Fig. 3. Generated worker population according to Scenario 1

The histrograms count the number of workers in the buckets $[0, 0.2)$, $[0.2, 0.4)$, $[0.4, 0.6)$, $[0.6, 0.8)$, and $[0.8, 1]$ according to real performance $pfmc_{real}$ in Fig. 3(a) and according to the observed performance $pfmc$ in Fig. 3(b). Fig. 3(c) represents the difference of real to observed performance; the confidence $confd$ values is shown in Fig. 3(d).

Scenario 2 contains 20 requesters which results in a much more "loaded" system. The workers are relatively skilled; their real performance $pfmc_{real}(w)$ is generated according to $\mathcal{N}(0.7, 0.25)$, i.e., the mean performance value is 0.7 compared to 0.3 as in the previous scenario. We further assume that there is a



(a) real pfmc (b) obs pfmc (c) diff obs/real (d) cnfd

Fig. 4. Generated worker population according to Scenario 2

higher amount of ratings already available and $cnfd(w)$ is generated according to $\mathcal{N}(0.8, 0.25)$. Performance values are again generated as $\mathcal{N}(pfmc_{real}(w), 1 - cnfd(w))$. The figures of Fig. 4 illustrate the generated worker population for Scenario 2 in the same way as for the previous one. In both scenarios tasks are issued in the first 500 rounds and the simulation ends when all workers have finished all accepted tasks.

Tasks are generated randomly with the following methodology. The real hidden result of a task is set to a uniformly distributed random value $\mathcal{U}(0, 1)$ from the range $[0, 1]$. An expected duration is randomly drawn from the set $\{1, 2, \ldots, 24\}$. A randomly assigned quality requirement $\mathcal{U}(0.4, 1)$ states whether the task poses high requirements to its processing. This means that requesters never state that the quality of their tasks' results is allowed to be lower than 0.4. Last but not least a random deadline is generated for which is guaranteed that it is after the expected duration.

Requester Behavior. In every round of the simulation each requester is asked to submit a task. After processing requesters receive the result for the task. If they receive the result after the deadline they rate the transaction with a value of zero and suspend for 20 rounds, i.e., they would refuse to issue a new task due to the negative experience. If the result is transmitted on time requesters rate the quality of the received result. Computationally this is done by comparing the task's real result with the received result. It is assumed that task requesters are able to estimate whether the received result is close to what was expected. The best possible rating is one, zero is the worst result, all values in between are possible. Ratings for a worker w, be it negative or positive, increase the confidence $cnfd(w)$ and update the observed performance $pfmc(w)$. If the rating is below a threshold of 0.3 the worker suspends for ten rounds, similar to a deadline violation. Hence, requesters with negative experiences tend to make less usage of the crowdsourcing marketplace. In addition to the pure task description requesters announce the maximum price they are willing to pay for the processing of the task. Prices are also represented by random values within the range $[0, 1]$. Tasks with high quality and high expected duration are more likely to have costs close to the maximum value.

Worker Behavior. When asked for a bid during an auction a worker first checks whether it is realistic to finish the task before the deadline considering all tasks the worker is working on. Each task has an expected duration t and each

worker would only submit a bid if s/he has at least $1.5 \cdot t$ of time to work on the task before the end of the deadline considering already accepted tasks. The actual processing time is set to a random value within $[t, 2t]$. For workers with high real performance the processing time is more likely to be close to t. This value is set by the simulation environment and the workers do not know about the exact processing time in advance. For the evaluation we consider workers with two different bidding behaviors: conservative and aggressive. Conservative workers determine the price according to a linear combination of a tasks effort (i.e., a normalized value of the expected duration), the workers real performance, and her/his workload. The rationale is that workers want more money for work-intensive tasks; workers with a high real performance are aware of their capabilities which influences the price as well. Finally, the higher a worker's current workload the more payment is conceived.

$$bid(w,t)_{conservative} = 0.4 \cdot effort(t) + 0.4 \cdot pfmc_{real}(w) + 0.2 \cdot load(w)$$

Aggressive workers, in contrast to conservative workers, do not increase the bid's price based on the workload but are more strongly driven by their own real performance.

$$bid(w,t)_{aggressive} = 0.4 \cdot effort(t) + 0.6 \cdot pfmc_{real}(w)$$

Whether a worker is conservative or aggressive is chosen randomly with the same probability. The processing of a task has a positive influence on the real performance of the worker w, i.e, $pfmc(w)_{real,new} = pfmc_{real}(w) + 0.1 \cdot (1 - pfmc_{real}(w))$. This modeling of a training effect results in a high learning rate for workers with low real performance and a slowed down learning effect for workers who are already very good.

Auction Processing. Auctions are conducted for the purpose of matching a task to a worker. As described in Section 4 there is a qualification and preselection stage before the actual auction in order to avoid spamming a huge worker base with auction request for which many workers may not have the necessary skills. Since we only consider one skill and have a limited number of 500 workers it is reasonable to admit all of them to the auctions. To achieve that the aptitude function, see Eq. 1, is set as follows:

$$aptitude : w \mapsto 1$$

After receiving the workers' bids they are ranked by a ranking function as defined in Eq. 2. Since there is only one skill the function is slightly adjusted:

$$rank : (w,b) \mapsto 0.6 \cdot pfmc(w) + 0.3 \cdot cnfd(w) + 0.1 \cdot (1 - price(b)).$$

Workers may either return a bid or refuse to submit a bid. From the received bids all values are removed whose price is higher than the price the requester is willing to pay. The remaining valid bids are ranked such that a high observed performance, high confidence, and a low price of the bid positively influence the

rank. The emphasis at that stage clearly is on the performance and not on the price. It may happen that there is no valid bid; in that case the requester is informed that the task could not be processed.

Skill Evolution. In this work we want to investigate how crowdsourcing can benefit from skill evolution, which is achieved by assigning assessment tasks to workers. This is especially useful for workers with a low confidence value. For these workers only few or no ratings are available. An assessment task is a task that is assigned to a worker although another worker has won the auction and was assigned to the task as well. The workers are not aware of the fact that there are other workers processing the very same task; requesters are not either. The crowdsourcing provider is responsible for paying for the training tasks. As usual, the result of the highest ranked worker is returned to the requester but it is additionally used as a reference for the training task. This enables the marketplace to generate a rating for the assessed worker by comparing her/his result to the reference. A further positive effect is the training of the assessed worker. The assignment of training tasks is based on the received list of valid bids. For controlling the skill evolution Equation 3 needs to be set accordingly.

The following definition of the assessment function, which results in disabling skill evolution and leads to purely profit driven auction decisions, maps each combination of workers, bids, and reference workers to 0.

$$assess_{profit} : (w, b, w_r) \mapsto 0.$$

In the evaluation we have used the following setting for the skill evolution enabled auctions.

$$assess_{skill} : (w, b, w_r) \mapsto \begin{cases} 0, & \text{if working queue not empty} \\ & \text{or } pfmc(w_r) < 0.8 \\ & \text{or } cnfd(w_r) < 0.8 \\ select(w, b), & \text{otherwise} \end{cases}$$

The function $assess_{skill}$ guarantees that only workers with empty working queue are assessed and that reference workers have high performance and high confidence. This is crucial because the worker w is rated according to the result of the reference worker w_r. If workers with a performance lower than 0.8 or confidence lower than 0.8 win an auction a training task assignment is prohibited. If all prerequisites are met the *select* function determines the workers who are assigned a training task. It is possible that multiple training tasks are assigned.

$$select : (w, b) \mapsto \begin{cases} 1, & \text{with probability } (1 - cnfd(w)) \cdot urg \\ 0, & \text{otherwise} \end{cases}$$

The *select* function assigns a training task based on the confidence of the considered worker. A low confidence increases the likelihood for a training task. The constant urg can be used to finetune the training task assignment procedure.

Table 1. Tasks in Scenario 1 and Scenario 2

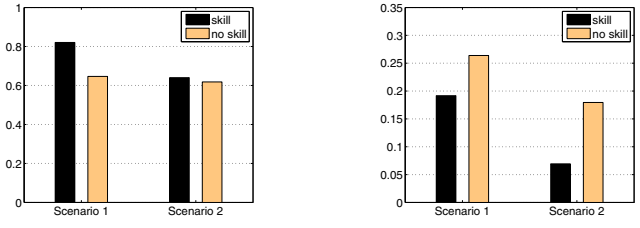
Tasks	Issued	No bid	Timeout	Training
Scenario 1	527/315	2/5	57/73	757/-
Scenario 2	1687/1641	19/26	556/579	1310/-

In our experiments it is set to a value of 0.01. A high value raises the probability of assessment tasks.

Discussions. Based on the introduced scenarios and simulation parameters, we test the benefit of skill evolution (*skill*) compared to regular auction processing (*no skill*). In the simulations, requesters issue a number of tasks to be processed by the crowd. However, requesters suspend their activity if the task quality is low (observed by low ratings) or task deadlines are violated. We hypothesize that a higher quality of task processing, and thus received ratings, also has positive effects on the profit of workers and the crowdsourcing platform. Table 1 gives an overview of the task statistics in each scenario. The number of *issued* tasks is influenced by the requesters' satisfaction. *No bid* means that a task could not be assigned to any matching worker.

The column *timeout* counts the number of tasks that were not delivered on time. Finally, the number of *training* tasks is depicted in the last column. All entries have the form *skill/no skill*.

In Fig. 5, we compare the results of Scenario 1 and 2 on the basis of total rating scores given by requesters and the average difference between the evolved real performance of the workers and the observed performance.

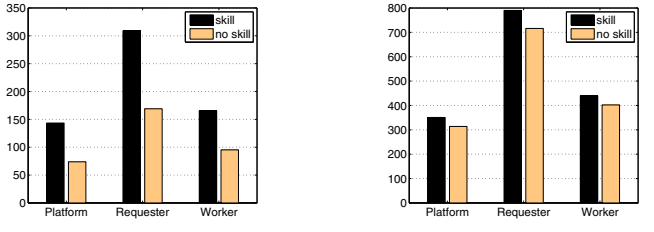


(a) Rating (b) Misjudgement

Fig. 5. Rating and skill misjudgement

For the *rating* (Fig. 5(a)) results in Scenario 1, the difference is more significant than in Scenario 2; 19% difference compared to 2%. In Scenario 1, whilst with no skill evolution support only an average rating score of slightly above 60% is given by the requesters, the advantage of skill evolution support is most apparent. In this scenario with low load, the system can optimally exploit the better accuracy of worker skill estimation, resulting in an average rating of 80%. Interestingly, the ratings are far lower in Scenario 2, although the average true performance of

the workers is higher. The reason is that the high number of requesters causes a heavily loaded system and increases the probability of deadline violations. Also, low performing workers are more likely to win an auction. The reaction of requesters to deadline violation and low quality is to give bad ratings, in this case an average rating of 60%. Due to the heavy load the benefit of skill evolution is small because for determining an auction's winning bid, performance and confidence become less decisive but free working capacity is more important. For the *misjudgement* of the workers in Fig. 5(b) (lower values are better), the results indicate clear benefits of skill evolution. Misjudgement is based on the average difference between the real and the observed performance. With more tasks, and in particular assessment-tasks being issued, worker capabilities can be estimated more correctly. For Scenario 1, the difference is below 20% with and below 30% without skill evolution support. For Scenario 2 with more load, the results considering misjudgement are evidently better. With more transactions, the average performance values' difference in the skill evolution support model is around 7% and around 19% otherwise. Thus, assessments provide remarkable good results for reducing misjudgements.



(a) Scenario 1 (b) Scenario 2

Fig. 6. Payments from requesters to workers and crowdsourcing platform

In our simulation requester try to minimize expenses; workers and crowdsourcing marketplace try to maximize earnings. Currently, we use a simple model in which the marketplace collects for each transaction the difference between the maximum expenses, as specified by the requester, and the minimum salary, as specified by the worker bid. In a real setting, the crowdsourcing platform could decide to charge less for its service. The quality of the results, and thus, their satisfaction directly influences the task offering tendency of the requesters. Again, with skill evolution applied, more tasks are processed since good ratings encourage requesters in offering tasks at a constant rate (see Fig. 6(a)). Altogether, the requesters spend almost twice as much money with skill evolution. This is only true for Scenario 1 and similar to the previous results, the difference is far smaller considering overload situations. With more than six times as many requesters, their expenses remain way below the sixfold amount as spent in Scenario 1 with skill evolution support. In total, the expenses in Scenario 2 are almost the same with and without skill evolution. The ratios are similar for the benefits of the

workers and the platform. As a summary, skill evolution generally performs better, however, is not antagonistic to overload scenarios. While with moderate task offering frequencies the model performs much better in all measurements, the differences become even when load increases and assessment-task further overload the platform. The results show that it is the responsibility of the platform to balance the task load and trade only with a fair amount of requesters.

6 Conclusions and Future Work

In this paper we present a novel crowdsourcing marketplace based on auctions. The design emphasizes automation and low overhead for users and members of the crowdsourcing system. We introduce two novel auctioning variants and show by experiments that it may be beneficial to employ assessment task in order to estimate members' capabilities and to train skills. Stimulating the demand for certain skills in such a way leads to skill evolution. As part of our ongoing research we plan to investigate how to allow for complex tasks and collaboration. Workers may, for instance, decompose tasks into subtasks, "crowdsource" the subtasks, and finally assemble the partial results into the final one. In such a setting the crowd would contribute the knowledge how to compose and assemble complex tasks. Furthermore, crowd members could provide higher level services such as quality control and insurance. Apart from auction-based mechanisms, specifications such as WS-HumanTask [10] and BPEL4People [2] for modeling human interactions in service-oriented business environments need to be extended to cope with the dynamics inherent to open crowdsourcing platforms. For example, providing skill and quality models based on prior negotiated service-level agreements (SLAs) that augment WS-HumanTask's people assignment model.

Acknowledgement. This work received funding from the EU FP7 program under the agreements 215483 (SCube) and 257483 (Indenica).

References

1. Agichtein, E., Castillo, C., Donato, D., Gionis, A., Mishne, G.: Finding high-quality content in social media. In: WSDM 2008, pp. 183–194. ACM, New York (2008)
2. Agrawal, A., et al.: WS-BPEL Extension for People (BPEL4People) (2007)
3. Amazon Mechnical Turk, `http://www.mturk.com` (last access March 2011)
4. Brabham, D.: Crowdsourcing as a model for problem solving: An introduction and cases. Convergence 14(1), 75 (2008)
5. Castellano, C., Fortunato, S., Loreto, V.: Statistical physics of social dynamics. Reviews of Modern Physics 81(2), 591–646 (2009)
6. CrowdFlower, `http://crowdflower.com/` (last access March 2011)
7. DiPalantino, D., Vojnovic, M.: Crowdsourcing and all-pay auctions. In: EC 2009, pp. 119–128. ACM, New York (2009)
8. Doan, A., Ramakrishnan, R., Halevy, A.Y.: Crowdsourcing systems on the world-wide web. Commun. ACM 54(4), 86–96 (2011)
9. eBay, `http://tinyurl.com/5w6zgfg` (last access March 2011)

10. Ford, M., et al.: Web Services Human Task (WS-HumanTask), Version 1.0 (2007)
11. Ipeirotis, P.G.: Analyzing the Amazon Mechanical Turk Marketplace. SSRN eLibrary 17(2), 16–21 (2010)
12. Klemperer, P.: Auctions: Theory and Practice. Princeton University Press, Princeton (2004)
13. Kumar, R., Lifshits, Y., Tomkins, A.: Evolution of two-sided markets. In: WSDM 2010, pp. 311–320. ACM, New York (2010)
14. Leymann, F.: Workflow-based coordination and cooperation in a service world. In: CoopIS 2006, pp. 2–16 (2006)
15. Schall, D., Dustdar, S.: Dynamic context-sensitive pagerank for expertise mining. In: Bolc, L., Makowski, M., Wierzbicki, A. (eds.) SocInfo 2010. LNCS, vol. 6430, pp. 160–175. Springer, Heidelberg (2010)
16. Schall, D., Truong, H.L., Dustdar, S.: Unifying human and software services in web-scale collaborations. IEEE Internet Computing 12(3), 62–68 (2008)
17. Su, Q., Pavlov, D., Chow, J.H., Baker, W.C.: Internet-scale collection of human-reviewed data. In: WWW 2007, pp. 231–240. ACM, New York (2007)
18. Vukovic, M.: Crowdsourcing for Enterprises. In: Proceedings of the 2009 Congress on Services, pp. 686–692. IEEE Computer Society, Los Alamitos (2009)
19. Yahoo: Answers scoring system, `http://answers.yahoo.com/info/scoring_system` (last access March 2011)
20. Yahoo! Answers, `http://answers.yahoo.com/` (last access March 2011)