Money, Glory and Cheap Talk: Analyzing Strategic Behavior of Contestants in Simultaneous Crowdsourcing Contests on TopCoder.com

Nikolay Archak
New York University, Leonard N. Stern School of Business
44 West 4th Street, Suite 8-185
New York, NY, 10012
narchak@stern.nyu.edu

ABSTRACT

Crowdsourcing is a new Web phenomenon, in which a firm takes a function once performed in-house and outsources it to a crowd, usually in the form of an open contest. Designing efficient crowdsourcing mechanisms is not possible without deep understanding of incentives and strategic choices of all participants. This paper presents an empirical analysis of determinants of individual performance in multiple simultaneous crowdsourcing contests using a unique dataset for the world's largest competitive software development portal: TopCoder.com. Special attention is given to studying the effects of the reputation system currently used by TopCoder.com on behavior of contestants. We find that individual specific traits together with the project payment and the number of project requirements are significant predictors of the final project quality. Furthermore, we find significant evidence of strategic behavior of contestants. High rated contestants face tougher competition from their opponents in the competition phase of the contest. In order to soften the competition, they move first in the registration phase of the contest, signing up early for particular projects. Although registration in TopCoder contests is non-binding, it deters entry of opponents in the same contest; our lower bound estimate shows that this strategy generates significant surplus gain to high rated contestants. We conjecture that the reputation + cheap talk mechanism employed by TopCoder has a positive effect on allocative efficiency of simultaneous all-pay contests and should be considered for adoption in other crowdsourcing platforms.

Categories and Subject Descriptors

H.4.m [**Information Systems**]: Miscellaneous; J.4 [**Social and Behavioral Sciences**]: Economics

General Terms

Measurement, Performance, Economics

Keywords

cheap talk, crowdsourcing, electronic markets, reputation, entry deterrence

1. INTRODUCTION

Web, in general, and electronic markets, in particular, are still evolving, periodically giving birth to new amazing market mechanisms like Amazon's product review system, Google's sponsored search engine [8], mashups [9] and cloud computing [4]. A recent, prominent and quite controversial example of such new mechanism is crowdsourcing. The term "crowdsourcing" was first used by Jeff Howe in a Wired magazine article [14] and the first definition appeared in his blog:

> *Simply defined, crowdsourcing represents the act of a company or institution taking a function once performed by employees and outsourcing it to an undefined (and generally large) network of people in the form of an open call. This can take the form of peer-production (when the job is performed collaboratively), but is also often undertaken by sole individuals. The crucial prerequisite is the use of the open call format and the large network of potential laborers.*

An important distinguishing feature of crowdsourcing, in addition to open call format and large network of contributors, is that it blurs boundaries between consumption and production creating a new *proactive* consumer type: the "working consumer" [16]. Whether these individuals are writing a blog or a product review, answering questions [21] or solving research problems [17], there is a significant economic value created by their actions; crowdsourcing platforms, that understand and successfully leverage this economic value, thrive, while others vanish. Yet little we know about incentives of "working consumers", and even less we know on how to shape them.

This paper presents an empirical analysis of determinants of individual performance in crowdsourcing contests using a unique dataset for the world's largest competitive software development portal: TopCoder.com. TopCoder is a crowdsourcing portal with strong market orientation [1] : users compete to design and develop software, which is later sold for profit by the sponsoring firm; monetary prizes are awarded to contestants with winning solutions. Not amazingly, we find that the prize amount is a strong determinant of the individual performance in the contest after controlling for the project complexity and the competition level. A very important distinguishing feature of our empirical study is that we analyze effects of the reputation system currently used by TopCoder.com on strategic behavior of contestants. In particular, we find that strong

[1]TopCoder is not simply a marketplace but also a strong community of interest. The social networking side of the platform is beyond the scope of this paper.

contestants (as measured by their reputations) face tougher competition in the contest phase, yet they strategically use cheap talk to deter competitor entry in the contest in the project choice phase.

Our contributions are as follows:

1. The major contribution of this paper is in demonstrating that online reputation of individuals in crowdsourcing contests have significant economic value and individuals may act strategically in order to extract profits from their online reputations.

2. The second contribution is in showing how individuals use cheap talk in multiple simultaneous contests to enhance their equilibrium payoffs and providing an estimate of the associated increase in user surplus.

3. The numerous minor contributions are in showing how individual performance in a contest is related to individual and project specific characteristics. For instance, we show that the length of the requirements document has no significant effect on the contest outcome, once we control for the number of distinct requirements.

2. PRIOR RESEARCH

Although crowdsourcing attracted significant attention in popular press, there are only few **empirical** studies of incentives and behavior of individuals in crowdsourcing environments. Yang et al. [21] examine behavior of users of the web-knowledge sharing market Taskn.com. They find significant variation in the expertise and productivity of the participating users: a very small core of successful users contributes nearly 20% of the winning solutions on the site. They also provide evidence of strategic behavior of the core participants, in particular, picking tasks with lesser expected level of competition. Huberman et al. [15] use data set from YouTube to conclude that the crowdsourcing productivity exhibits a strong positive dependence on attention as measured by the number of downloads. Brabham [3] performed online survey of the crowdsourcing community at iStockphoto. The survey results indicate that the desire to make money, develop individual skills, and to have fun were the strongest motivators for participation.

We emphasize that crowdsourcing is, by definition, related to a single firm extracting economic value from the enterprise. In that respect, crowdsourcing differs from open source (OSS) where the product produced is a public good. While some driving forces behind OSS phenomenon may have economic nature such as signaling of skills to prospective employees, it is currently acknowledged that intrinsic motivation is the crucial factor [2]. Another example is knowledge sharing in community forums such as “Yahoo! Answers": Wasko and Faraj [20] show that individuals perceiving that participation will enhance their reputation within the community, tend to contribute more. Although it is not clear whether value of such reputation is intrinsic or the individuals expect that the reputation can be “cashed” in the future, scale of such contributions [2] definitely favors the first explanation. As the example of “Yahoo! Answers" shows, making a clear distinction between crowdsourcing and other forms of peer production on the Web is often hard when the mechanism produces both a public good (publicly available knowledge forums) and private benefits (advertising profits).

A very important distinguishing feature of our empirical study is that we analyze effects of the reputation system currently used by TopCoder.com on strategic behavior of contestants. In particular, we find that strong contestants (as measured by their reputations) face tougher competition in the contest phase, yet they strategically use cheap talk to deter competitor entry in the contest in the project choice phase. These results are novel to the Information Systems literature due to the fact that the prior research on electronic markets predominantly concentrated on studying reputation effects for firms rather than individuals [5]. To the best of our knowledge, most empirical studies confirmed significant monetary value of online firm reputation. For instance, there is evidence that sellers with good reputation on eBay enjoy higher sales [18] and even price premiums [19]. In another example, sellers on Amazon.com with better record of online reputation can successfully charge higher prices than competing sellers of identical products [11]. In our study, the reputation premium is of different nature: contestants with higher reputations enjoy more freedom in the project choice phase and can successfully deter entry of their opponents in the same contest.

3. TOPCODER: THE WORLD’S LARGEST COMPETITIVE SOFTWARE DEVELOPMENT COMMUNITY

TopCoder.com is a website managed by the namesake company. The company hosts weekly online algorithm competitions as well as weekly competitions in software design and software development. The work in design and development produces useful software, which is licensed for profit by TopCoder. As of July 23, 2008 163,351 people have registered at the TopCoder website. 17.3% of those registered have participated in at least one Algorithm competition, 0.3% in Design, 0.7% in Development [3]. We are particularly interested in Design and Development competitions as they have tangible payments to competitors.

The business model underlying software Design and Development competitions is briefly summarized below. [4] TopCoder produces software applications for major clients. It interacts directly with the client company to establish application requirements, timelines, budget etc. Once the application requirements are defined, the application goes to the Architecture phase, where it is split into a set of components. Each component is supposed to have a relatively small scope and precise set of technical requirements defining the expected component behavior and interface for interacting with other components. For instance, an “Address Book” component can be required to implement certain address management functionality, moreover, it should be written in Java and provide a Web service interface. The set of requirements to each component is summarized in a single document (Requirements Specification) and posted on the website as a single Design competition. Any registered member of the website satisfying minimum legal requirements can submit a UML [5] design to any posted design competition. Winning design submission goes as input into the Development competition, which has similar structure, only the competitors are required to submit actual code implementing the provided UML design. Output from Development competitions is assembled together into a single application, which is later delivered to the customer.

Design and Development competitions are posted on TopCoder website on a weekly basis, Figure 1 shows a sample list of weekly Development competitions. Each competition belongs to a certain catalog. Four largest catalogs are Java (generic components

[2]For instance, as of December 2006, Yahoo!Answers had 60 million users and 65 million answers.

[3]`http://en.wikipedia.org/wiki/TopCoder`

[4]Due to space limitations, the model description is simplified and we omit some of the recent changes to the business process such as introduction of the Architecture and Specification contests.

[5]Unified Modeling Language

for Java platform), .Net (generic components for .Net platform), Custom Java (custom components for Java platform), Custom .Net (custom components for .Net platform), but there are other catalogs such as catalogs for major clients (AOL can be seen in Figure 1). Every competition has two associated deadlines: the registration deadline and the submission deadline. The registration deadline specifies the time by which all individuals willing to participate must register for the competition, it is usually two or three days after the competition posting date. The submission deadline specifies the time by which all solutions must be submitted, it is usually within five to seven day interval after the competition posting date. Every competition has associated payment that is given to the competition winner and 50% of this amount is given to the first runner-up. The registration information is public, so that competitors can see who else has registered for the component. This is achieved by clicking on items in the “Registrants Rated/Unrated” tab. The result may look like shown in Figure 2. Moreover, the information is updated instantly: as soon as one competitor has registered for the contest, others can see that.

Important component of the Design and Development process is its scoring and review system. Once the submission deadline has passed, all submissions enter the review phase. Each submission is graded by three reviewers according to a prespecified scorecard on dimensions varying from technical submission correctness and clarity of documentation to flexibility and extendability of the solution. After the review process is complete, submissions enter the appeals phase where the competitors get a chance to appeal the decisions made by the reviewers. Once all appeals have been resolved, the placement is determined by the average score across all three reviewers. A sample of results is shown in Figure 3.

TopCoder implements policy of maximum observability [6]. At first, competitors can always observe identities of their opponents, i.e., other members registered for the same contest (see Figure 2). Moreover, for every member TopCoder tracks all prior competition history and summarizes it in a single rating number [7]. The rating is provided for members who have submitted at least one solution in some contest. It is calculated via a relatively complex formula taking into account all prior submission history of the contestant and relative performance compared to other contestants [8]. Fortunately, the exact formula for calculating the rating value is not important, as, in fact, even more information is available for each rated competitor, including all prior competition history. This information can be revealed by clicking on the member’s handle; the sample is shown in Figure 4. Thus, we will simply think of ratings as proxies for the coder’s performance so far: the better the coder performed in the past, the higher the rating will be, with more recent performance having higher effect on the current rating.

Finally, there is also the reliability score, which is provided for members who have been registered for at least 15 contests and is equal to the fraction of the last 15 contests they registered for in which they delivered a solution satisfying the minimum quality requirement (received a score of at least 75 in the review phase). Members with the reliability rating equal to or exceeding certain threshold will receive a bonus for every prize they receive. Due to space limitations, we will not give any more attention to the reliability score, but only mention that we take it into account when calculating the expected coder’s payment.

Dataset

We obtained historical contest data from the TopCoder website. The dataset included 1,966 software Design contests and 1,722 software Development contests ran by TopCoder from 09/02/2003 to 08/23/2009. The total number of different TopCoder members, who participated in at least one of the contests in our dataset, was 1,660. Among these members, 301 individuals participated in Design competitions only, 1,106 individuals participated in Development competitions only, and 253 participated in at least one Design and at least one Development competition. Descriptive statistics of the data are given in Table 1.

The first two rows of this table summarize the design and development rating distributions extracted from the competition data; note that these distributions are different from the static snapshot of the rating distributions for all TopCoder members as they weigh active competitors more. The next two rows of Table 1 summarize the distribution of review scores extracted from the contest data. Due to the nature of the review scorecard, one can never get the score higher than 100.0. 75.0 is the official reservation score - the minimum score required for submission to be accepted and the prize (if any) to be paid to the competitor. Out of 5,113 design submissions and 7,602 development submissions in our dataset, 4,247 and 6,046 submissions respectively received at least the score of 75.0. The fifth and the sixth row of Table 1 summarize the distribution of the number of submissions *per contest* and the next two rows summarize the distribution of the number of contests *per active individual* [9] as of the last date in our dataset (08/23/2009).

Additionally, by crawling the website, we obtained project requirements data for 1,742 software Design projects from our sample. [10] For every Design contest, we obtained length in pages of the project’s requirement specification [11] as well as the number of distinct project requirements and the target compilation platform. The two most popular target languages were Java (994 projects) [12] and C# (605 projects), together they account for more than 90% of all projects. The rest of the projects (Ruby, PHP, other languages, and projects for which we failed to extract the target platform from the Requirements Specification) were classified as “other”.

Finally, we collected payment data for projects in our sample. Table 1 lists the first place prize amounts. Note that TopCoder also awards second place prizes equal to exactly half of the first place prize for the corresponding project.

4. EMPIRICAL ANALYSIS: COMPETITION PHASE

In this Section, we perform simple statistical analysis of the dataset to determine what factors influence individual performance in a software contest *once the registration phase of the contest is closed*. The usual suspects are individual specific characteristics and skills, project specific characteristics, experience with TopCoder competition platform, level of competition in a particular contest, current

[6]One important exception from this rule is that contestants cannot see scores given by the reviewers to their opponents until after the appeals phase is over.

[7]Ratings are different for every competition track. Thus, an individual participating in both Design and Development competitions will have two different ratings - one for Design and one for Development.

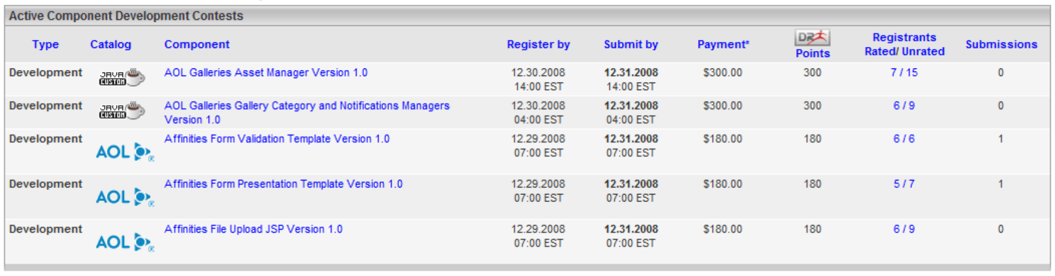
[8]Detailed description of the rating system can be found at `http://www.topcoder.com/wiki/display/tc/Component+Development+Ratings`

[9]Competed in at least one contest

[10]The rest of the projects had missing or broken Requirements Specification document.

[11]Note that the same Requirements Specification document, together with the design documentation produced during the design contest, goes as input to the corresponding Development contest.

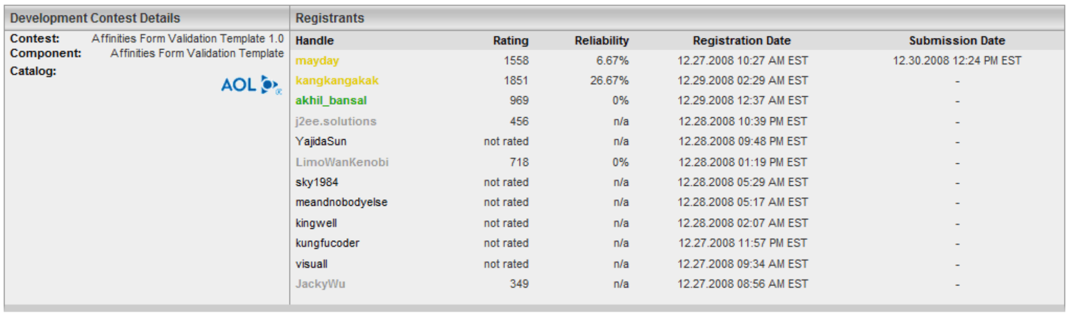
[12]including JavaScript



Active Component Development Contests

Type	Catalog	Component	Register by	Submit by	Payment*	Points	Registrants Rated/ Unrated	Submissions
Development		AOL Galleries Asset Manager Version 1.0	12.30.2008 14:00 EST	**12.31.2008** 14:00 EST	$300.00	300	7 / 15	0
Development		AOL Galleries Gallery Category and Notifications Managers Version 1.0	12.30.2008 04:00 EST	**12.31.2008** 04:00 EST	$300.00	300	6 / 9	0
Development		Affinities Form Validation Template Version 1.0	12.29.2008 07:00 EST	**12.31.2008** 07:00 EST	$180.00	180	6 / 6	1
Development		Affinities Form Presentation Template Version 1.0	12.29.2008 07:00 EST	**12.31.2008** 07:00 EST	$180.00	180	5 / 7	1
Development		Affinities File Upload JSP Version 1.0	12.29.2008 07:00 EST	**12.31.2008** 07:00 EST	$180.00	180	6 / 9	0

Figure 1: Sample list of Development competitions from TopCoder.



Development Contest Details

Contest: Affinities Form Validation Template 1.0
Component: Affinities Form Validation Template
Catalog:

Registrants

Handle	Rating	Reliability	Registration Date	Submission Date
mayday	1558	6.67%	12.27.2008 10:27 AM EST	12.30.2008 12:24 PM EST
kangkangakak	1851	26.67%	12.29.2008 02:29 AM EST	-
akhil_bansal	969	0%	12.29.2008 12:37 AM EST	-
j2ee.solutions	456	n/a	12.28.2008 10:39 PM EST	-
YajidaSun	not rated	n/a	12.28.2008 09:48 PM EST	-
LimoWanKenobi	718	0%	12.28.2008 01:19 PM EST	-
sky1984	not rated	n/a	12.28.2008 05:29 AM EST	-
meandnobodyelse	not rated	n/a	12.28.2008 05:17 AM EST	-
kingwell	not rated	n/a	12.28.2008 02:07 AM EST	-
kungfucoder	not rated	n/a	12.27.2008 11:57 PM EST	-
visuall	not rated	n/a	12.27.2008 09:34 AM EST	-
JackyWu	349	n/a	12.27.2008 08:56 AM EST	-

Figure 2: Sample list of registrants for a Development competition from TopCoder.

reputation of the individual. We start by providing evidence that individual's rating is a significant predictor of future performance.

Table 1 shows significant variation in the distribution of final project scores. For instance, for Design contests the median review score (88.6) is only slightly more than a standard deviation (10.1) away from the perfect review score (100), as well as from the minimum passing score (75.0). To further understand the factors behind the dispersion of scores, we first grouped all competitors in five different groups on the basis of their Design rating just before the contest [13]. The group boundaries were chosen in accordance with the color group assignments by TopCoder: "grey" (rating 0-899), "green" (rating 900-1199), "blue" (rating 1200-1499), "yellow" (rating 1500-2199), "red" (rating 2200+). Within each group, we estimated the distribution of the review scores. Density and cumulative density functions for each group are shown in Figure 5. The plot suggests the first order stochastic dominance ranking of score distributions for different groups. We formally tested this statement by Mann-Whitney-Wilcoxon stochastic dominance test on pairs of adjacent groups; in all 4 cases ("red" vs. "yellow", "yellow" vs. "blue" etc.), the test rejected the null (no stochastic dominance) hypothesis at 1% significance level. Similar results were obtained for the Development contests.

There might be several alternative explanations of why members with high rating today are expected to deliver higher scores in the future:

- *Hypothesis Ia:* Higher rated members are those with more *inherent* skills and abilities and therefore they deliver better solutions.
- *Hypothesis Ib:* Higher rated members are those who *inherently* care more about their rating and therefore consistently put more effort into the competition to keep the status high.

[13]Note that members ratings change over their lifetime, therefore the same person may be classified to two different categories at different moments of time.

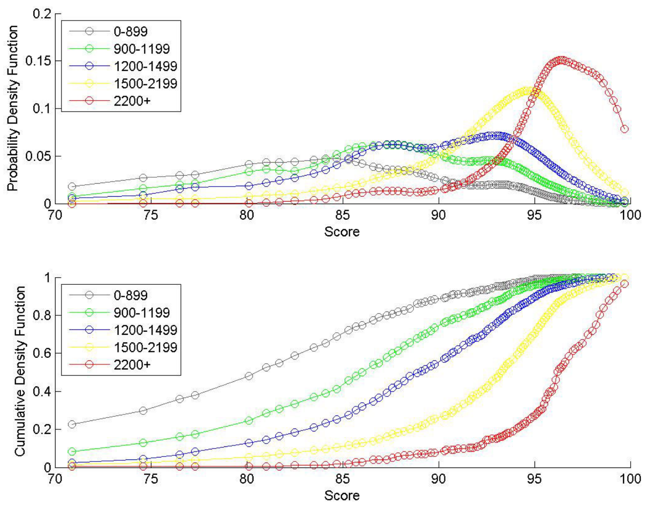
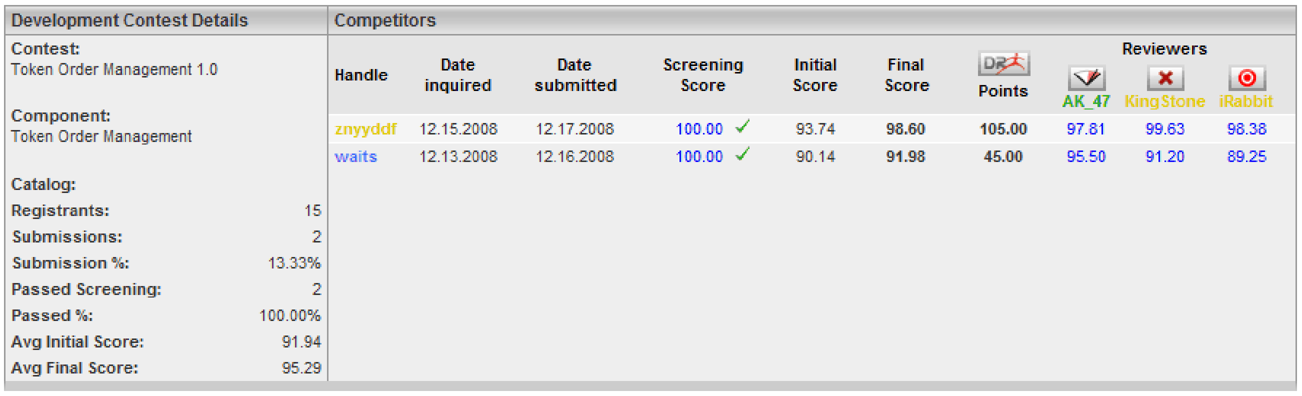


Figure 5: Distribution of Software Design Review Scores Clustered By Coder Rating

- *Hypothesis II:* Higher rated members are those with more *accrued* experience and therefore they deliver better solutions.
- *Hypothesis III:* The rating is "addictive", members that achieved high rating today tend to contribute more in the future to keep their status high (this is similar to Huberman et al. [15] statement that users that get more attention on YouTube tend to contribute more in the future).
- *Hypothesis IV:* Higher rated members experience less competition in the project choice phase, therefore they can afford to choose easier, better paying or less competitive projects and deliver higher scores.



Development Contest Details

Contest:
Token Order Management 1.0

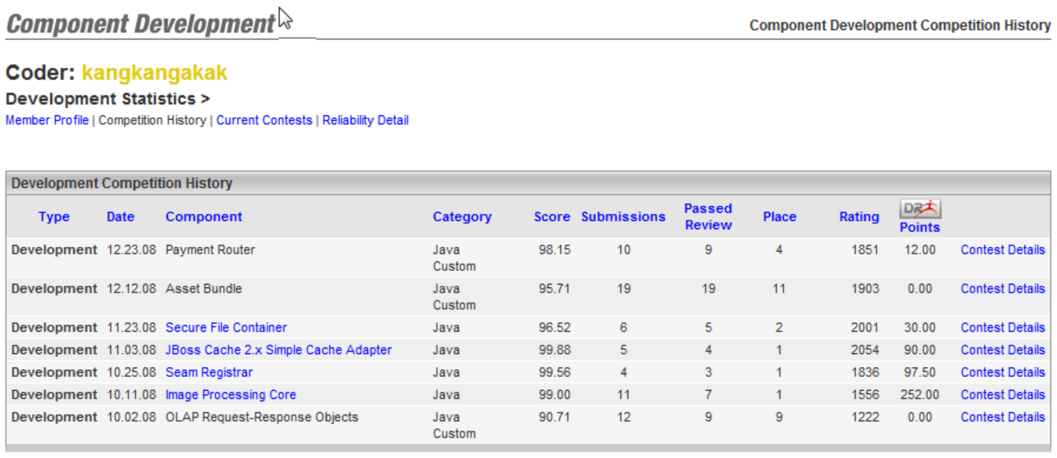
Component:
Token Order Management

Catalog:	
Registrants:	15
Submissions:	2
Submission %:	13.33%
Passed Screening:	2
Passed %:	100.00%
Avg Initial Score:	91.94
Avg Final Score:	95.29

Competitors

Handle	Date inquired	Date submitted	Screening Score	Initial Score	Final Score	Points	Reviewers AK_47	KingStone	iRabbit
znyyddf	12.15.2008	12.17.2008	100.00 ✓	93.74	98.60	105.00	97.81	99.63	98.38
waits	12.13.2008	12.16.2008	100.00 ✓	90.14	91.98	45.00	95.50	91.20	89.25

Figure 3: Sample contest results from TopCoder.



Component Development

Component Development Competition History

Coder: kangkangakak

Development Statistics >

Member Profile | Competition History | Current Contests | Reliability Detail

Development Competition History

Type	Date	Component	Category	Score	Submissions	Passed Review	Place	Rating	Points	
Development	12.23.08	Payment Router	Java Custom	98.15	10	9	4	1851	12.00	Contest Details
Development	12.12.08	Asset Bundle	Java Custom	95.71	19	19	11	1903	0.00	Contest Details
Development	11.23.08	Secure File Container	Java	96.52	6	5	2	2001	30.00	Contest Details
Development	11.03.08	JBoss Cache 2.x Simple Cache Adapter	Java	99.88	5	4	1	2054	90.00	Contest Details
Development	10.25.08	Seam Registrar	Java	99.56	4	3	1	1836	97.50	Contest Details
Development	10.11.08	Image Processing Core	Java	99.00	11	7	1	1556	252.00	Contest Details
Development	10.02.08	OLAP Request-Response Objects	Java Custom	90.71	12	9	9	1222	0.00	Contest Details

Figure 4: Sample Development competition history for a TopCoder member.

- *Hypothesis V:* Higher rated members expect fiercer competition from opponents and therefore have to deliver better solutions in order to win.

Note that these hypotheses are not mutually exclusive. In order to test the hypotheses, we estimate a set of econometric models. The first set of econometric models, analyzed in this paper, relates contestant's performance on a particular project to the set of observable contestant and project characteristics. Results of the first set of models for Design contests are given in Table 2 [14]. To ensure no "cold-start" effect for contestants who have not been rated yet or are not familiar enough with the TopCoder Software methodology, we dropped the first five contests for every contestant from our sample.

The first column of Table 2 presents a simple OLS model which specifies that

$$\begin{aligned}\text{score}_{ij} &= \text{constant} + \beta_1 \text{payment}_{ij} + \beta_2 \text{rating}_i \\ &+ \beta_4 \max \text{rating}\_{-i} + \beta_5 \text{experience}_i + \beta_6 \text{opponents}_j \\ &+ \beta_7 \max \text{spec. length}_j + \beta_8 \text{num. req.}_j + \beta_9 \text{is Java}_j \\ &+ \beta_{10} \text{is C\#}_j + \varepsilon_{ij},\end{aligned}$$

where

1. score_{ij} is the (final) submission score for the contestant i in the contest j on a scale of 0 to 100.
2. payment_{ij} is the first prize payment for the contest j measured in \$1,000 [15].
3. rating_i is the contestant i rating immediately before the contest j.
4. $\max \text{rating}\_{-i}$ is the maximum opponent rating for the contestant i in the contest j.
5. experience_i is the contestant i experience right before the contest measured in the number of previous contests for this contestant.
6. opponents_j is the number of opponents in the contest j.
7. spec. length_j is the length of the Requirements Specification document for the contest j measured in pages.
8. num. req._j is the number of distinct requirement items in the Requirements Specification document for the contest j.

[14]Results for Development contests are quantitatively similar. We omit them due to space limitations.

[15]Note the individual specific subscript i, indicating that we adjust the payment to take into account individual specific bonuses, such as the reliability bonus.

Variable	Min	Max	Mean	Median	S.Dev
design rating	1.0	2794.0	1406.6	1376.0	465.9
development rating	124.0	2397.0	1182.3	1166.0	365.5
design submission score	25.0	100.0	86.0	88.6	10.1
development submission score	21.8	100.0	86.6	88.9	10.2
number of submission per design contest	1.0	26.0	2.0	2.0	1.9
number of submission per development contest	1.0	61.0	4.0	3.0	4.4
number of design contests per member	1.0	408.0	9.0	2.0	27.1
number of development contests per member	1.0	76.0	5.0	2.0	8.6
requirements specification length	1.0	51.0	4.0	4.0	3.2
number of requirements per design contest	1.0	195.0	14.0	12.0	13.1
design contest 1st place payment	100.0	3000.0	705.5	750.0	319.8
development contest 1st place payment	50.0	3000.0	580.7	500.0	325.3

Table 1: Descriptive Statistics

9. is Java$_j$ is the boolean indicator if the contest j had Java as the target platform.

10. is C#$_j$ is the boolean indicator if the contest j had $C\#$ as the target platform.

The second column of Table 2 (GMM 1) presents the GMM (Generalized Method of Moments) [12] estimation results for Equation 1 controlling for potential endogeneity of the contest payment (payment$_j$), as well as the coder's rating (rating$_i$) and reliability (reliability$_i$). Endogeneity of the contest payment comes from the fact that there might be unobservable project characteristics that affect both the contest complexity as well the contest payment set by TopCoder.com. For instance, it is plausible to assume that the contest sponsor will set higher prizes for more complex contests, however such contests will also have lower average solution quality due to the complexity of the underlying project. OLS estimates will capture both the direct effect of the contest payment on the contestants' performance and the projection of the unobserved complexity variable, thus underestimating the causal effect of the contest prize. This is a classic endogeneity problem in economics. In order to account for endogeneity of the contest payment, we instrument this variable with the average payment in the contemporaneous contests [16]. This is essentially the Hausman et al. [13] approach of using prices from different markets as instruments for prices in the given market. Furthermore, contestant's rating might be correlated with unobservable individual traits which affect project choice and therefore, indirectly, contestant's performance. To account for potential endogeneity of the rating, we instrument it with 3 lags of differences. The assumption here is that, although the rating might be correlated with the individual specific characteristics, short term fluctuations of the rating are not; this is conceptually similar to the Anderson and Hsiao [1] estimator for the dynamic panel data.

The third column of Table 2 (GMM 2) presents the GMM estimation result for Equation 1 controlling additionally for potential endogeneity of the maximum opponent rating ($\max$ rating$_{-i}$) and the number of opponents in the contest (opponents$_j$): both can be correlated with unobservable project characteristics affecting the individual performance. Again, we use the Hausman et al. [13] approach and instrument the maximum opponent rating with the average of the maximum contestant rating in the contemporaneous contests and the number of contestants with the average of the number of contestants in the contemporaneous contests.

The fourth column of Table 2 extends the model by including the coder specific fixed effects in the previous model. Finally, for comparison purposes, the last column of Table 2 presents results of the fixed effects model without the rating variable.

[16]more precisely, average payment across the contests ran in the previous two weeks (note it excludes the instrumented contest)

4.1 Results

Payment is a strongly significant determinant of the individual contestant performance in all five models. From the first column of Table 2, we can see that the OLS significantly underestimates the effect of payment, as compared to the GMM 1 model, suggesting the project payment is in fact correlated with unobservable project characteristics, beyond the specification length, the number of requirements and the target platform. Durbin-Wu-Hausman endogeneity test using the average contemporaneous project payment as an instrument confirms that the OLS model is inconsistent ($p < 0.001$). Furthermore, endogeneity test for the maximum opponent rating also rejects the null hypothesis ($p < 0.001$), therefore in the rest of the Section we will concentrate on analyzing the results of the last three models (GMM 2, IV FE 1 and IV FE 2).

The last three columns of Table 2 suggest that the marginal effect of a $1,000 payment on the final quality of submission in a TopCoder Software Design contest lies somewhere between 6 to 10 project points. Specification length is not a significant determinant of the final project score, however the number of requirements is: there is approximately 1.2 point loss in quality for every 10 additional project requirements [17].

Java and C# contests seem to have lower average quality score than "other" contests, although this effect is barely statistically significant. We acknowledge that these two variables are also potentially endogenous as they depend on the project choice by the contestant. While we did not instrument for the target platform endogeneity, we performed a robustness check by dropping these regressors from the model and verifying that coefficient estimates for the rest of the variables are not significantly affected.

Experience of the contestant is not a significant predictor of the contestant's performance [18], therefore we suggest that the Hypothesis II does *not* hold in our sample.

While the number of opponents is a barely significant predictor of the contestant performance, the maximum rating of the opponent is strongly statistically and economically significant: the marginal effect of an extra 1,000 points of the opponent's rating is somewhere between 6 and 8 project points. Note that this is comparable with an effect of an extra $1,000 of the project payment. In par-

[17]While this value might seem to be very low for readers familiar with TopCoder Design contests, we should emphasize that we count every bullet point in the section 1.2 of the Requirements Specification document as a separate requirement.

[18]Note that we drop the first five observations for every contestant.

Table 2: Regressions of the final project score for Software Design contests

	(1)	(2)	(3)	(4)	(5)
	OLS	GMM 1	GMM 2	IV, FE	IV, FE 2
payment (in $1,000)	2.044***	3.351*	6.345***	6.425*	10.11**
	(3.81)	(2.08)	(3.52)	(2.35)	(2.82)
rating (in 1,000 pts)	9.306***	3.789*	3.408	3.942	
	(22.27)	(2.26)	(1.84)	(1.56)	
max opponent rating (in 1,000 pts)	-0.0435	-0.0241	7.485**	6.248*	8.834**
	(-0.10)	(-0.04)	(3.10)	(2.42)	(3.16)
experience (num contests)	0.0103**	0.00158	-0.000881	0.00729	-0.00134
	(3.01)	(0.42)	(-0.20)	(0.83)	(-0.13)
number of opponents	0.330**	0.417**	0.524	1.165*	1.199
	(2.92)	(3.22)	(0.81)	(2.05)	(1.84)
specification length	-0.0651	-0.135	-0.0944	-0.0722	-0.151
	(-0.96)	(-1.39)	(-0.65)	(-0.49)	(-0.89)
number of requirements	-0.0247	-0.0397*	-0.0813	-0.109**	-0.122*
	(-1.78)	(-2.24)	(-1.56)	(-2.66)	(-2.56)
Java	-1.895*	-0.285	-1.611	-2.612*	-2.778*
	(-2.07)	(-0.27)	(-1.39)	(-2.11)	(-1.99)
C#	-1.083	0.793	-1.568	-2.997*	-3.517*
	(-1.17)	(0.71)	(-1.30)	(-2.23)	(-2.31)
constant	73.95***	80.81***	69.20***	69.56***	69.59***
	(57.84)	(24.60)	(16.36)	(13.46)	(12.79)
N	1488	1174	1174	1174	1174

t statistics in parentheses
* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$

ticular, this result means that, *conditional on facing the same set of opponents*, a higher rated contestant will face fiercer competition from the opponents. We performed a robustness check of this result by considering two separate cases: the highest rated opponent is rated lower than the coder (i.e., the coder is the favorite) and the highest rated opponent is rated higher than the coder (i.e., the coder is the underdog). In both cases, there was a (similar in magnitude) positive effect of the opponent's rating on the performance of the coder. The news is not all bad for high rated coders: as the next Section shows, higher rated coders face less competition in the project choice phase.

We conclude this Section with discussion of the effect of rating on the contestant performance. As Figure 5 and the OLS results suggest, there is a lot of variation in performance between contestants of different ratings. The result persists even if we instrument all variables except for the rating properly, but it significantly decreases and becomes statistically insignificant when one puts contestant specific dummies in the model and instruments the contestant's rating by its short term fluctuations. This suggest that much if not all effect of ratings is due to inherent contestant specific traits (Hypothesis Ia and Ib) and we do *not* see empirical evidence for Hypothesis III that the rating is "addictive". We acknowledge that our current dataset does not allow us to test Hypothesis Ia and Ib separately, i.e, although we know that some individuals consistently perform better than others due to some individual characteristics, we cannot conclude whether they are simply more skilled or care more about their performance than other contestants.

5. EMPIRICAL ANALYSIS: REGISTRATION PHASE

Empirical results of the previous section show that higher rated contestants, ceteris paribus, face tougher competition from their opponents in the competition phase of the contest. We have hypothesized (Hypothesis IV) that this effect might be compensated by a competitive advantage in the project choice phase: TopCoder competition system allows contestants to register for projects in advance, thus letting strong contestants to signal their project choices in order to deter entry of their opponents. In this Section, we formally test this statement.

Our first test is based on a simple observation that, in order to deter entry of their opponents in the contest they like, higher rated contestants should register early in the contest, while lower rated contestants will prefer to wait until the higher rated opponents made their choices. If this is true, one should empirically observe a correlation between the contestant's rating and the probability of being the first registrant in a contest. As Figure 6 shows, the correlation is present in our dataset. The effect is also visible (although weaker) if one plots contestant's rating against the probability of NOT being the last registrant in the contest (due to space limitations, we omit this Figure).

Furthermore, we formally tested for presence of the "early registrant" effect by performing the Bernoulli (binomial probability) test of the first registrant being the highest rated coder in the contest conditional on the number of contestants. If the coder registration process is independent of the ratings, then the null hypothesis of

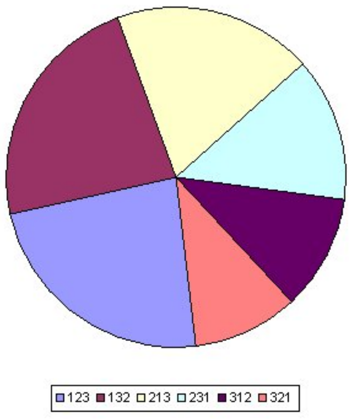


Figure 7: For three member contests, a fraction of all contests in which the registration order of contestants represents a certain permutation of the rating order. 1 represents the highest rated coder, 3 represents the least rated coder. Coders are order by the registration time.

the "fair" coin-flip model holds and, for contests with exactly N members, the fraction of contests, where the first registrant was the highest rated member, should be approximately $\frac{1}{N}$. In our sample, out of 153 two member contests, in 92 contests the first member was the highest rated; out of 130 three member contests, in 67 contests the first member was the highest rated; out of 79 four member contests, in 37 contests the first member was the highest rated. [19] In all three cases, the null hypothesis of the fair coin flip is rejected ($p < 0.001$).

In fact, we observe even stronger effect in our data. Again, consider all contests with exactly m participants. Encode every participant by a number from 1 to m in order of their ratings, with 1 being the highest rated coder and m being the lowest rated coder (assume no ties). Any registration sequence can be represented by a permutation of numbers from 1 to m. For instance, in a three-person contest, sequence 132 would represent that the highest rated coder was the first one to register and the lowest rated coder was the second one to register. For any particular registration sequence, one can count how many times did it occur in the data. Amazingly, not only the first registrant is most often the highest rated one but also the ordering of registration sequences by contest shares is similar to the lexicographic ordering. In case of $m = 3$ shown in Figure 7, it is in fact identical to the lexicographic ordering.

Next, we formally test whether an early entry of a high rated coder has a positive deterrence effect on entrance of other high rated coders in the contest. If the effect exists, then, controlling for the project complexity, there must be a negative correlation between the ratings of the coders who registered for the project so far and the rating of the coder who is going to register next (assuming that there is such). Table 4 presents results of a series of regressions, starting from a simple OLS model in the first column:

$$\begin{aligned}\text{next coder rating}_j &= \text{constant}\\ &+ \beta_1 \text{last coder rating}_j\\ &+ \beta_2 \text{ the number of coders}_j + \varepsilon_j,\end{aligned}$$

where the left hand side of the equation represents the rating of the coder who is going to register next and the right hand side represents the rating of the last registered coder (so far) and the total number of coders. The second column of Table 4 adds project specific dummies to the regression. The next column additionally includes the maximum rating of the coder who registered so far. Finally, the last column includes the coder specific dummies (for the last registered coder) in addition to the project dummies.

The simple OLS model (first column of Table 4) shows strong positive correlation between the rating of the last registered coder and the rating of the coder who will register next for the same project. We believe that this correlation picks up the fact that certain project types (like projects requiring specific set of skills) attract higher rated coders while other project types attract lower rated coders, therefore, as long as one does not control for the project specific effects, there is a positive correlation between ratings of the coders registered for the same project. Second column of Table 4 shows that, as soon as project specific dummies are included to the regression, the correlation turns negative. The next column explains that it is the maximum rating of the coder who registered so far, not the rating of the last registrant, that influences future registration process. Adding coder dummies in the last column does not make a significant difference.

We emphasize that registration for a contest in TopCoder is not legally binding as the registered contestants may choose not to participate; however, registering and not participating has a negative effect of reducing the contestant's reliability score, thus (potentially) decreasing the amount of future winnings. In our sample, only small fraction of contestants (about 20%) had reliability score high enough (at least 80%) to qualify for a project related bonus. We performed a robustness check by eliminating these contestants from the sample and performing estimation only for coders who had no significant cost of registering but not participating. Our results were quantitatively similar, thus indicating that the cost of signal (registration) had no significant effect on its value. We conjecture that the registration phase in TopCoder contests works largely as cheap talk [10] (communication between players which does not directly affect the payoffs of the game) rather than costly signaling.

We conclude this Section, by providing a back-of-the-envelope estimate of the effect of early registration of a high rated coder on the expected surplus in the contest. In order to do this, we need several other estimates:

1. We need an estimate of how early registration of a high rated coder affects the maximum opponent rating in the contest. We can infer this estimate from Table 4, which shows that each rating point of the highest rated registrant in the contest (so far) decreases the rating of *all* future registrants by approximately 0.5 rating point.

2. We need an estimate of how the level of effort in the competition phase decreases when the maximum opponent rating decreases. While Table 2 reports this value, we take a conservative estimate and assume that coders do not reduce the quality of their submission when the opponent ratings drop. This assumption ensures that our estimate will provide a robust lower bound on the coder surplus whether or not the coder behaves strategically in the competition phase.

3. We need an estimate of how the coder's probability of winning the first prize in a contest depends on the maximum rating of the opponent. Note that we assume that the highest rated coder in the contest is guaranteed to win at least the second place prize; this assumption is consistent with our dataset: we have only 9 observations where a member with rating more than $1,800$ did not win the first or the second

[19]In this sample, we have excluded unrated participants from consideration.

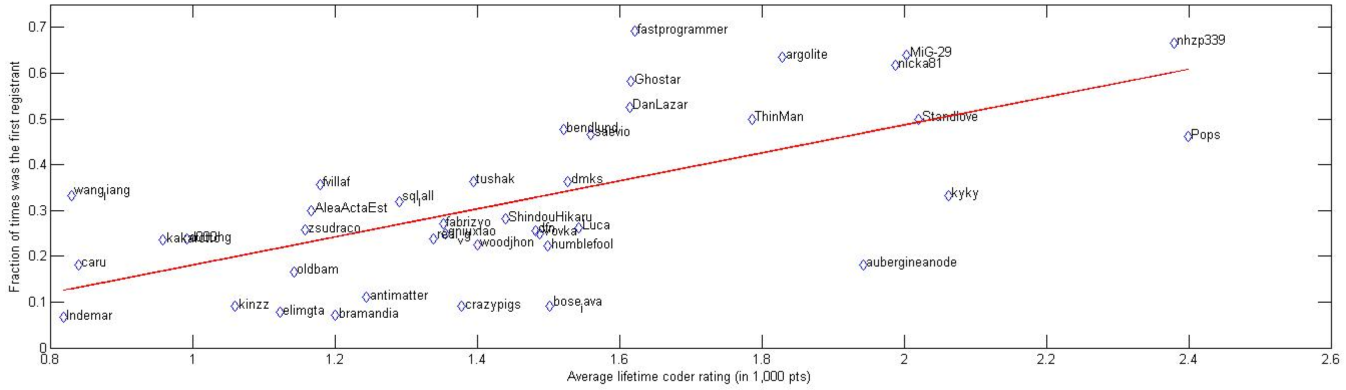


Figure 6: By member probability of being the first registrant in a Software Design competition. X axis represents the average lifetime member rating (excluding the first five contests). Y axis represents the sample probability estimate. The plot includes only members that participated in at least 15 Software Design contests.

Table 3: Regressions of the next registrant rating for Software Design contests

	(1)	(2)	(3)	(4)
	OLS	FE project	FE project	FE project coder
the last registratnt rating	0.0883**	-0.195***	-0.0759	0.145
	(2.73)	(-3.71)	(-1.04)	(1.04)
the number of registrants so far	-0.00284	-0.0101	0.0355	0.0590*
	(-0.30)	(-0.70)	(1.88)	(2.58)
the maximum registrant rating so far			-0.476**	-0.614**
			(-2.95)	(-3.11)
N	917	917	917	917

t statistics in parentheses * $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$

prize in a contest where he was the highest rated participant. In order to estimate the probability of winning the prize, we estimate a series of logistic regressions; results are shown in Table 5. In columns 2 to 4 of Table 5, we additionally control for unobservable project specific characteristics by including the project level random effects [20]. Random effects and additional regressors do not affect value of the coefficient on the maximum opponent rating variable significantly, therefore we use more conservative estimate from the simple logit model. The corresponding value of the marginal effect is -0.54.

Putting it all together, for a sufficiently high rated member, every additional rating point decreases the rating of the toughest opponent by half a point and increases the probability of winning a prize by $0.5 * 0.001 * 0.54 = 0.027\%$. For a $\$1,000$ contest, the difference between the first and the second prize is $\$500$ and therefore the change in the expected winnings per single rating point is 13.5 cents. For large rating values, this amount can be quite significant. For instance, the difference between the smallest rating in the “red” (the highest) rating category and the smallest rating in the “yellow” (the second highest) rating category is 700 points, what translates to the difference of $\$94.5$ per a $\$1,000$ contest. Overall, our empirical results support Hypothesis IV that higher rated members face less competition in the project choice phase and behave strategically to exploit this competitive advantage.

[20]We do not use fixed effects to avoid the incidental parameters problem.

6. SUMMARY

Crowdsourcing is a new Web phenomenon, in which a firm takes a function once performed in-house and outsources it to a crowd, usually in the form of an open contest. Designing efficient crowdsourcing mechanisms is not possible without deep understanding of incentives and strategic choices of all participants. This paper presents an empirical analysis of determinants of individual performance in multiple simultaneous crowdsourcing contests using a unique dataset for the world’s largest competitive software development portal (TopCoder.com). Special attention is given to studying the effects of the reputation system currently used by TopCoder.com on behavior of contestants. We find that individual specific traits together with the project payment and the number of project requirements are significant predictors of the final project quality. Furthermore, we find significant evidence of strategic behavior of contestants. High rated contestants face tougher competition from their opponents in the competition phase of the contest. In order to soften the competition, they move first in the registration phase of the contest by signing up early for particular projects. Although registration in TopCoder contests is non-binding, it deters entry of opponents in the same contest; our lower bound estimate shows that this strategy generates significant surplus gain to high rated contestants.

This study has a number of limitations. At first, while we find that better performance of higher rated coders should be attributed to some individual specific traits, we currently cannot say whether such differentiation is purely skill based or some coders care more about their reputation within community than others. Next, al-

Table 4: Logistic regressions for probability of winning the contest

	(1) logit	(2) logit + RE project	(3) logit + RE project	(4) logit + RE project
max opponent rating (in 1,000 pts)	-2.371***	-2.372***	-3.007***	-2.840***
	(-14.89)	(-14.89)	(-15.67)	(-14.50)
rating (in 1,000 pts)			2.871***	2.846***
			(15.33)	(15.15)
number of opponents				-0.138***
				(-3.59)
constant	3.247***	3.248***	-0.0344	0.120
	(12.91)	(12.92)	(-0.10)	(0.36)
N	1488	1488	1488	1488

t statistics in parentheses * $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$

though we find that strategic behavior of contestants in the registration phase increases the surplus of high rated coders, it is not clear what effect does it have on the sponsor's surplus and the overall social efficiency of the mechanism. We hypothesize that ability of skilled contestants to signal their intention to perform a particular project should result in more efficient allocation of the overall pool of contestants to particular contests, thus improving the overall social surplus. Indeed, imagine a situation in which the reputation system is not in place and contestants choose competitions in a setting of incomplete information. Theoretical analysis of such "choice game" for simultaneous crowdsourcing contests is provided in a recent paper of DiPalantino and Vojnovic [7] and also, in a setting of simultaneous online auctions by sellers of different reputation, by Dellarocas [6]. Both papers identify a unique symmetric equilibrium in which

1. Players (buyers) partition into several skill (value) zones, with the zone l covering a certain interval range $[v_{l+1}, v_l)$ of skills (values).

2. Players (buyers) in the zone k randomly choose between projects (auctions) with the top k highest rewards (seller reputation).

Such equilibrium is obviously inefficient due to the absence of coordination mechanism between selecting players; the inefficiency is particularly stark when the number of potential contestants (buyers) is relatively small compared to the number of offered contests (auctions) as it increases the probability of having a contest (auction) with zero level of participation. Our empirical investigation shows that a properly designed reputation system can, at least partially, mitigate this inefficiency by allowing coordination between individuals. Constructing a good structural model of individual behavior and using it to measure the social value of such signaling mechanism is a valuable direction for future research. We conclude the paper by conjecturing that the reputation + cheap talk mechanism employed by TopCoder has a positive effect on allocative efficiency of simultaneous all-pay contests and should be considered for adoption in other crowdsourcing platforms. [21]

7. REFERENCES

[1] T. W. Anderson and C. Hsiao. Formulation and estimation of dynamic models using panel data. *Journal of the American Statistical Association*, 76(375):598–606, Sep 1981.

[2] J. Bitzer, W. Schrett, and P. J. Schröder. Intrinsic motivation in open source software development. *Journal of Comparative Economics*, 35(1):160 – 169, 2007.

[3] D. Brabham. Moving the crowd at istockphoto: The composition of the crowd and motivations for participation in a crowdsourcing application. Working Paper, Available at SSRN: http://ssrn.com/abstract=1122462, 2008.

[4] R. Buyya, C. S. Yeo, and S. Venugopal. Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities. In *Proceedings of the 10th IEEE International Conference on High Performance Computing and Communications*, pages 5–13, Aug 2008.

[5] C. Dellarocas. The digitization of word-of-mouth: Promise and challenges of online reputation mechanisms. *Management Science*, 49(10):1407–1424, 2003.

[6] C. Dellarocas. Simultaneous auctions of imperfect substitute goods by sellers of different reputations. Working Paper, Available at SSRN: http://papers.ssrn.com/abstract=1115805, 2008.

[7] D. DiPalantino and M. Vojnovic. Crowdsourcing and all-pay auctions. In *EC '09: Proceedings of the tenth ACM conference on Electronic commerce*, pages 119–128, 2009.

[8] B. Edelman, M. Ostrovsky, and M. Schwarz. Internet advertising and the generalized second-price auction: Selling billions of dollars worth of keywords. *The American Economic Review*, 97(18):242–259, March 2007.

[9] R. J. Ennals and M. N. Garofalakis. Mashmaker: mashups for the masses. In *SIGMOD '07: Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 1116–1118, 2007.

[10] J. Farrell and M. Rabin. Cheap talk. *The Journal of Economic Perspectives*, 10(3):103–118, 1996.

[11] A. Ghose, P. Ipeirotis, and A. Sundararajan. The dimensions of reputation in electronic markets. Working Paper, New York University, 2006.

[12] L. P. Hansen. Large sample properties of generalized method of moments estimators. *Econometrica*, 50(4):1029–54, July 1982.

[13] J. A. Hausman, G. Leonard, and J. D. Zona. Competitive analysis with differentiated products. *Annales d'Economie et de Statistique*, (34):07, April 1994.

[14] J. Howe. The rise of crowdsourcing. Wired Magazine(`http://www.wired.com/wired/archive/14.06/crowds_pr.html`), 2006.

[15] B. Huberman, D. Romero, and F. Wu. Crowdsourcing, attention and productivity. Working Paper, Available at SSRN: http://ssrn.com/abstract=1266996, 2008.

[16] F. Kleemann, G. Voß, and K. Rieder. Un(der)paid innovators: The commercial utilization of consumer work through crowdsourcing. *Science, Technology & Innovation Studies*, 4(1), July 2008.

[17] K. R. Lakhani and J. A. Panetta. The principles of distributed innovation. *Innovations: Technology, Governance, Globalization*, 2(3):97–112, 2007.

[18] P. Resnick and R. Zeckhauser. Trust among strangers in internet transactions: Empirical analysis of eBay's reputation system. Working Paper, University of Maryland, 2001.

[19] P. Resnick, R. Zeckhauser, J. Swanson, and K. Lockwood. The value of reputation on eBay: A controlled experiment. *Experimental Economics*, 9(2):79–101, jun 2006.

[20] M. Wasko and S. Faraj. Why should i share? examining social capital and knowledge contribution in electronic networks of practice. *MIS Quarterly*, 29(1):35ÂŰ57, 2005.

[21] J. Yang, L. A. Adamic, and M. S. Ackerman. Crowdsourcing and knowledge sharing: strategic user behavior on taskcn. In *Proceedings of the 9th ACM conference on Electronic commerce*, pages 246–255, 2008.

[21]This advice is provided "as is" without warranty of any kind.