ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/300415424

Empirical Analysis on Parallel Tasks in Crowdsourcing Software Development

Conference Paper · November 2015

DOI: 10.1109/ASEW.2015.22

CITATIONS
6

READS
181

2 authors, including:

Razieh Saremi
Systems Engineering Research Center
19 PUBLICATIONS **87** CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Project Who Should Take This Task?: Dynamic Decision Support for Crowd Workers View project

Project Award vs. Worker Behaviors in Competitive Crowdsourcing Tasks View project

All content following this page was uploaded by Razieh Saremi on 23 March 2018.

The user has requested enhancement of the downloaded file.

Empirical Analysis on Parallel Tasks in Crowdsourcing Software Development

Razieh Lotfalian Saremi
School of Systems and Enterprises
Stevens Institute of Technology
Hoboken, NJ 07030, USA
rlotfali@stevens.edu

Ye Yang
School of Systems and Enterprises
Stevens Institute of Technology
Hoboken, NJ 07030, USA
Ye.yang@stevens.edu

Abstract— Crowdsourcing has become a popular option for rapid acquisition, with reported benefits such as shortened schedule due to mass parallel development, innovative solutions based on the "wisdom of crowds", and reduced cost due to the pre-pricing and bidding effects. However, most of existing studies on software crowdsourcing are focusing on individual task level, providing limited insights on the practice as well as outcomes at overall project level. To develop better understanding of crowdsourcing-based software projects, this paper reports an empirical study on analyzing four largest projects on Topcoder platform that intensively leverage crowdsourcing throughout the product implementation, testing, and assembly phases. The analysis results conclude that: (1) crowdsourcing task scheduling follows typical patterns including prototyping, component development, bug hunt, and assembly and coding (2) budget phase distribution patterns does not following traditional patterns, and uploading task rate is not representing same budget rate associated with them as about 75% of uploaded tasks would price under 67% of total project budget; (3) Higher degree of parallelism would lead to higher demand for competing on tasks and shorter planning schedule to complete the project consequently better resource allocation.

Index Terms—Parallelism, Software Worker, Tasks, Award, worker performance.

I. Introduction

The interest on crowdsourcing software development is rapidly increasing in both industry and academia. In crowdsourcing, jobs that were traditionally done in-house, would be distributed among a large distributed group of crowd workers [1]. Software projects have begun to accept crowdsourcing in several different phases of software design and production [1, 2].

Ideally, mass parallel production through Crowdsourcing could be an option to rapid acquisition in software engineering by leveraging on infinite worker resource on the internet. It is important to understand the patterns and strategies of decomposing and uploading parallel tasks in order to maintain stable worker supply as well as satisfactory task completion rate. However, much of existing studies are only focusing on individual task level, such as task pricing [6], and worker recommendation models [7, 9].

Crowd workers usually choose to register, work, and submit for certain tasks with satisfactory award and comfortable level of dedicated effort, as award typically represents degree of task complexity as well as required competition levels [4,5]. Although, sometimes Award is simply represents a specific required skill to perform the task, it is one of the main factors influence on crowd software workers in terms of number of registrants and consequently number of submissions [6]. Therefore, pricing tasks could be a huge challenge in decomposing the projects to mini-tasks and time of uploading them in the platform, yet by uploading more number of parallel tasks, crowd workers would have more choice of utilized tasks to register for and consequently the chance of receiving more number of completed tasks is higher. Since existing studies on general crowdsourcing reported limited or unpredictable results [4, 5, 7], it is a good opportunity to focus on parallelism on uploading same project tasks.

For software managers, utilizing external unknown, uncontrollable, crowd workers would put their projects under greater uncertainty and risk compared with in-house development [6, 7]. Understanding crowd worker's sensitivity to the project stability and failure rate, becomes extremely important for managers to make trade-offs among cost saving, degree of competition, and expected quality in the deliverables [6].

However, there are a lot of researches on crowdsourcing attributes and tasks uploading and trends, There is a lack of study on the impact of parallelism decomposing project in the crowdsourcing market and the time of uploading new tasks on the performance and project scheduling in the field of software crowdsourcing. We are trying to present a model of uploading decomposed tasks based on the average number of uploaded parallel tasks from analyzing competition history, and considering underlying factors including the challenge type and, and pre-arranged award of the competition.

In this paper we present an empirical study on analyzing the effects of number of uploaded parallel tasks on crowdsourcing performance, based on data gathered from Topcoder website, largest software development crowdsourcing platform with an online community of 750000 Crowd Software workers [8]. Topcoder started to explore crowdsourcing tasks in the form of competitions for software development, in which workers would independently create a solution and winner will be chosen [9].

The rest of the paper is organized as follows: Section II introduces the review of related literature; Section III Research design and Procedure; Section IV reports the empirical results to test the hypotheses; Section V discusses the results of experimental evaluation; finally, Section VI is the conclusion.

II. Research Design and Procedure

A. Metrics

In order to analyze parallel uploaded task in a project we categorized the available data to different project and defined following metrics which is summarized in table 1: 1) task attributes which describes the basic quantitative characteristics of a task including total associated award and total number of uploaded tasks in a limited period of time (Pi) ; 2) worker performance which measure the total number of workers registering for the task, i.e. number of registrants, and the total number of workers submitted work products for the task, i.e. number of Submissions, as well as Stability; and 3) task outcome which measures Task uploading failure rate. Different attributes used in this research are listed in table1.

B. Dataset

The used dataset contains 403 individual projects includes 4908 component development tasks from Jan 2014 to Feb2015, extracted from Topcoder website. We assume project with 100 uploaded tasks or more are big enough to perform parallel task competition, therefore, for the purpose of our study, we conduct comparison analysis on four biggest projects of this data set in different period of times in terms of uploading number of tasks.

Tasks are uploaded as competitions in the platform, where Crowd software workers would register for the challenges. On average most of the tasks have a life cycle of one and half months from first day of registration to the submissions deadline. When the workers submit the final files, it will be reviewed by experts to check the final results and grant the scores. The granted score of a submission is dependent on the task's level of complexity and the time it took to code a solution [9].We summarized four projects details including, challenge type, technology, and domain in table2 and also statistical variables regards to project attributes in table 3.

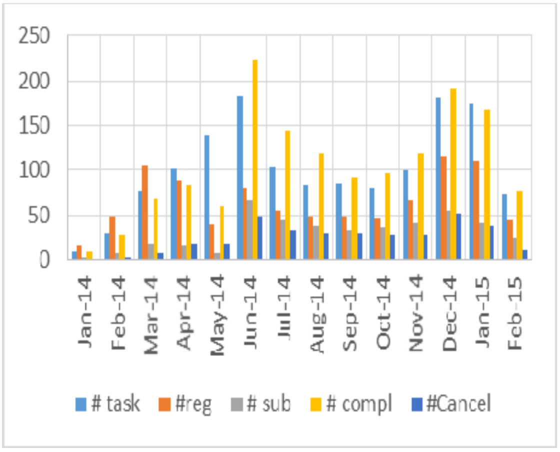


Fig. 2. Trend of uploading tasks and associated award

Table 1. Summary of metrics definitions

Category	**Metric**	**Measurement**
Task Attributes	Award	Dollars in task description
	# of parallel Task (n)	Total number of tasks uploaded in a limited period of time (month or week)
Worker Performance	# Registrants	Number of registrants that are willing to compete on total number of tasks in specific period of time
	# Submissions	Number of submissions that a task receives by its submission deadline in specific period of time
Task outcome	# Completed tasks	Number of Acceptable submissions that a task receives by its submission deadline in specific period of time
	# Cancelled tasks	Number of none-acceptable submissions that a task receives by its submission deadline in specific period of time
	Failure rate	# Cancel tasks/ n

C. Empirical Studies

To test the above hypothesis, we conduct the following empirical analysis:

- Analysis I: Parallelism and maximum task uploading trend;
- Analysis II: Distribution pattern of Parallel task uploading and task completion;

Analysis I and II are conducted on the four projects for comparison purpose. At the end, we apply an experimental evaluation on four sample projects according to the hypothesis to predict the number of submissions based on historical registrants, associated award with the tasks and also submissions timeline. The results are presented next.

III. Results

A. Parallelism and Maximum uploaded tasks (Analysis I)

As crowdsourcing is generally believed to be used for small pieces of work, it is interesting to observe that a significant number of companies are adopting crowdsourcing regularly and utilize the online workers to rapidly deliver product features end-to-end. These companies regularly decompose their tasks into hundreds of mini-tasks, follow certain task scheduling patterns to broadcast tasks online, and involve hundreds or thousands of internet workers to work on and complete the tasks within pre-specified, usually 2-4 weeks of task duration.

Figure 2 shows an overall profile of the distribution pattern of task uploaded and associated award rate per month for four sample projects. As it is shown by passing time the number of uploaded task raised to reach the maximum number, which included most number of small decomposed tasks per project, then it smoothly starts to decrease. And again, in September another increasing happens which lead us to the second peak in December, this can be due to new product launch, or a lot of debug tasks uploading in order to completing the projects. Obviously associated award would follow the same pattern.

As it is illustrated in Fig.3, total uploaded task rate and total associated award rate are not following the same trend, also the maximum uploaded task is not representing the maxim award associated with the projects, since tasks size and award have direct influence, number of decomposed tasks and total amount of associated award have negative influence [6, 7, 10], this fact makes the opportunity of better resource allocation since by decomposing the project to more number of tasks and following the parallel task uploading method not only the requestor would use minimum budget but also the pool of indefinite crowd workers is available to compete on the task.

Table 2: Sample Project details

	# tasks	Application	Top 3 Technology Involved	Top 3 Challenge type	Top Platform Type
Project I	156	Topcoder API	Java (21),Node.js (123) SQL(4), REST (4)	First2Finish (85) ,Assembly Competition (67), Code(2) , Bug Hunt (2)	HTML, DocuSign, Heroku, NodeJS (142), Other
Project II	306	Topcoder, Web Arena IDE	Node.js(45),Angular.js (53), CSS (53), JavaScript (53), Bootstrap (53), HTML5 (53), CSS (51)	First2Finish (114),Assembly Competition(36), Bug Hunt(120)	Facebook, HTML(158), Other, NodeJSEC2,Heroku,unknown
Project III	177	Topcoder Community	Angular.js (39), PHP (33, CSS (35)	Assembly Competition(51),First2Finish (93),Code (16)	NodeJS, Wordpress (151),Heroku,EC2,Beanstalk, AWS, HTML ,Unknown, Other
Project IV	277	Hercules Android APP	HTML5 (16), iOS (77), Android (177)	First2Finish(241),Bug Hunt (23), Assembly Competition (7)	iOS, Android(161), Mobile,HTML,EC2,Other,Unknown

One possible reason can be that, based on project scheduling methods, at first most of uploaded tasks are in the design category which are relatively big size, then it turn to implementation and development which makes the opportunity of decomposing the project to more number of mini tasks, next step would be test and deployment that develop bigger task size to work on, consequently cause reduction in total number of uploaded tasks and total award associated with them will increase due complexity of tasks or required skills to perform it [6, 12]. We focus on four biggest projects in the platform to analyze the result, Table 3 summarized the statistical analytics of these projects.

The uploading task rate in each of the four project are listed in Table 5. It shows that about 75% of all tasks are priced under the 67% of the total award, and maximum percentage of uploading task are not representing the maximum sum of award rate associated of the project, which can be due to task complexity or required skill to compete on the task. However increasing number of uploaded tasks at the same time may represent higher number of registrants and chance of acceptable submissions [7, 11]. While total number of uploaded tasks and associated award are not following almost the same pattern, it still supports individual task size and specific associated award follow negative influence [4, 6, 7].

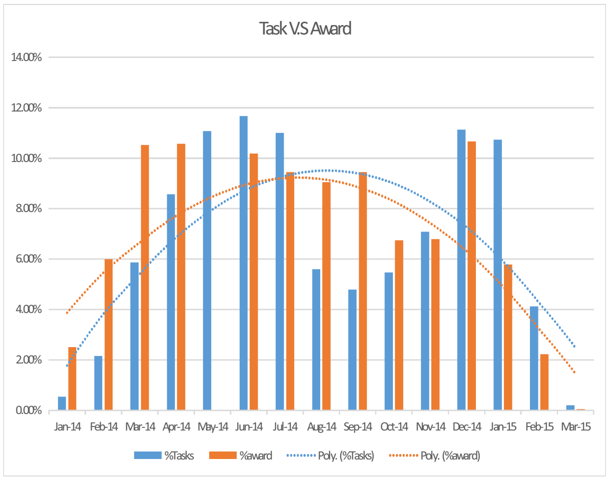


Fig. 3: Relationship between Award and Tasks uploading trend

Table 3: Statistical analytics of sample projects per month

		# Task	Award	# Reg	# Sub	# comp task	#Can tasks
Sample Project 1	Min	1	30	6	0	0	0
	Median	5.5	3850	69	12	5.5	0
	Max	57	43225	763	117	51	6
	Average	16.35	8233.21	155.28	30.57	15.21	1.14
Sample Project 2	Min	3	280	18	4	3	0
	Median	26	20910	274	37	23	3
	Max	95	34671	752	159	89	10
	Average	33.92	20853.84	318.15	62.07	30.84	3.076
Sample Project 3	Min	2	2300	30	5	2	0
	Median	25	10632	235	49	16	8
	Max	47	22426	410	97	38	10
	Average	23.81	12842.09	219.72	49.18	17.45	6.36
Sample Project 4	Min	2	355	43	5	2	0
	Median	26.5	8667.5	147	43.5	22	3
	Max	93	21045	464	173	82	25
	Average	35.64	10180.92	172.28	61.92	29.21	6.57

We further focus on the four biggest projects individually based on the total award rate and the uploading task rate to analyze deeper on task completion.

B. Distribution pattern of Parallel task uploading,the relationship between number of maximum task uploading in different period of time (Analysis II)

As it is illustrated in Fig 4. Higher number of uploaded tasks per period do not guarantee higher total amount of associated award in four different projects, however according to available empirical data there are the average positive correlation of 0.44 between total number of uploaded tasks and total prize associated with them, yet task type and project domain would influence this fact.

When analyzing individual projects, we saw that depends on project type and domain of the project, number of uploaded tasks are increasing to get the maximum number of tasks per project and then decreased based on the different phase and planning. Among all projects, project II took longer period of time to decompose the project to more number of uploaded tasks, which based on table 2, as maximum tasks type is bug hunt, the time making sense.

However unexpectedly, in project IV we are facing two peaks in number of uploaded tasks, which may happen due to different version of the software production and also the specific production phase that project started to be crowdsourced. In rest of sample project, task uploading almost following the total platform trend, meaning, in the beginning there are smaller number of tasks for compete on and by improving the project to further phases, project can be decomposed to more smaller task size, and more number of uploaded tasks available to be chosen from. In late production phase, again number of decomposed tasks would drop, due to the nature of deployment projects.

Fig.4 clearly shows that higher number of uploaded tasks in specific period of time do not significantly mean higher associated award rate and more budget to spend. For example in project III, higher uploading task rate happened in July which is not represents higher associated award rate.

Our analysis result shows that on the available project, there is a strong positive correlation of 0.99 between number of uploaded tasks and number of completed tasks in each period of time and also a positive correlation 0.94 between number of task uploaded and number of registrations. This argue insures that by increasing number of uploaded tasks in the platform, there would be higher demand to compete on them, and chance of getting more number of completed tasks would rise [4,5,6,7].

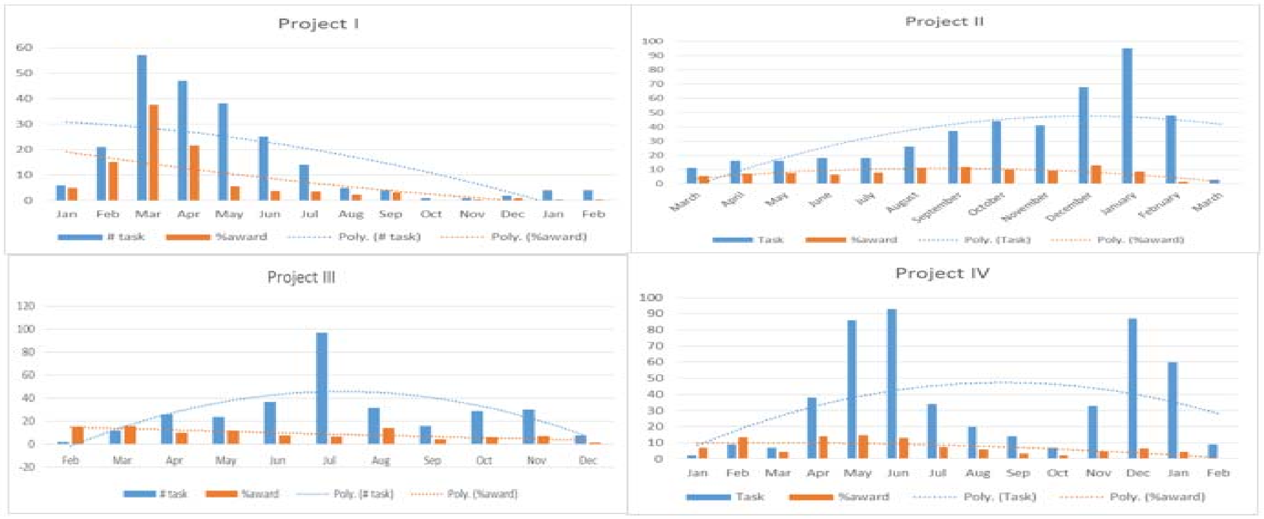


Fig. 4: Relationship between Award and Tasks and Registrants

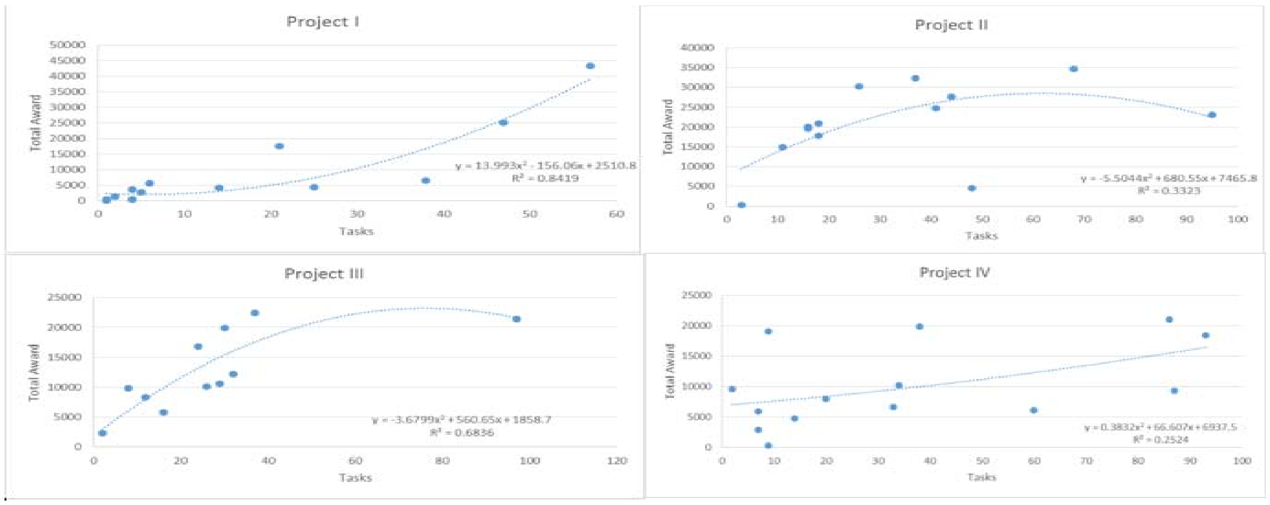


Fig. 5: Relationship between total number of uploading tasks and total associated award rate

Decomposing project to more number of tasks and parallel uploading them cause more choice for crowd workers to utilized tasks and compete on, and consequently better rate of resource allocation for project managers in terms of budget and project scheduling.

When further analysis is conducted on the four projects as listed in Table 3, we also observed that the relationship between number of uploaded tasks and associated award rate generally follows exponential distribution, Fig.5, still, it does not guarantee that by increasing number of uploading task, total associated award would increase, as it depends on task size and complexity [7].

Expectedly, it is possible that as number of uploaded tasks increases, some previously registered workers might perceive distracting factors such as competition pressure, insufficient time to complete the task or misunderstanding the task requirement while registering [4,7]. Hence by raising the number of uploaded tasks in parallel flow at the same time, number of registrants and consequently submissions would increase [6].

IV. Discussion and evaluation

A. Impact of Parallelism on task completion

According to the empirical analysis and assumption in pervious section, it is clear that although less than 25% of the total registrants will submit the final files, yet almost 87% of the uploading tasks will be completed. This result does not support Topcoder assumption of 90%-96% success rate with having more than 1 registrants [13]. Moreover the analysis suggest that uploading more number of parallel tasks makes the opportunity of different tasks being choose by more number of crowd workers at the same time hence shorter project time and higher chance of succeed.

Also based on the available empirical data, for large projects (decomposed to minimum 150 tasks) uploading on average 90 parallel tasks at the same period of time, may lead us to on average 86% success. Since in comparison with the in-house production there would be more number of crowd software workers available with wild variety of skills, there is no need to wait to finish one task and start the next on. Therefore in shorter time line higher rate of the tasks would be done parallel and consequently higher rate of resource allocation would happen.

Since decomposing project to more number of mini tasks means less complexity, easier task to work on and consequently less associated award, there would be higher demand to compete on the tasks [4]. These results would guide us to the fact that schedule the uploading tasks based on parallelism and off course decomposed project to more number of smaller tasks, cause the higher chance of project success in shorter period of time by achieving higher number of submissions and completion rate. Moreover, uploading parallel tasks makes the chance of receiving less rate of dropped tasks since the crowd worker is facing multiple available tasks with different range of size and complexity .

B. Presiction model of number of submissions

Since the result of empirical analysis of parallelism in uploading tasks shows positive effect on the tasks success and workers performance, we tried to come up with an empirical model to predict the number of submissions per tasks as the main factor of performance based on main projects attributes.

In order to build a linear regression model we use four main drivers in uploading parallel tasks: number of registrants, associated award, lead time for submissions, and number of parallel tasks in the same period.

The result of the linear regression model is described in table 4 . Further, to evaluate the model we compare the results via six different Machine Learning methods which will be explained in the next part.

Table 4: Model parameters

Source	Value	Standard error	t	Pr > |t|	Lower bound (95%)	Upper bound (95%)
Intercept	2.768	0.494	5.601	< 0.0001	1.791	3.745
# Parallel Tasks	1.000	0.386	2.593	0.025	0.151	1.850
# Reg	-0.001	0.001	-1.435	0.179	-0.003	0.001
Award	0.151	0.084	1.805	0.099	-0.033	0.335
Duration	0.294	0.799	0.368	0.720	-1.464	2.052

C. Experimenta Evaluation of the Model

To compare the model performance, we applied a leave one out cross validation on the dataset to predict the number of submissions based on six different approaches to predict models. The estimated submissions are used to compute four popular performance measures that are widely used in current prediction system for software development as follow: 1- Mean Magnetite of Relative Error (MMRE), 2- Median Magnetite Relative of Error (MdMER), 3- Standard Deviation Magnetite Relative of Error (StdMRE), 4- Percentage of the estimates with Relative Error less than or equal to N% (Pred (N)).

We investigate two research questions in order to analyze this experiment: A) which predictive model gives the best overall predictive performances assessed by performed analysis in pervious section? B) What actionable insight can we suggest to requestors to crowdsourced their project according to these emerging result?

The primary result of this analysis is shown in Fig.6, we try to address the answer of research questions by referring to the results.

It is clear that KNN analysis has a better predictive performance according to Pred (30) and also it has almost the lowest error rate, while CCR recreation has the worst performance based on Pred (30).

To answer second question, results indicate that for the best performer, KNN, almost 70% of estimations have an error less than 30%, this confirms the idea that it can be valuable to different predicting models for crowdsourcing software development projects. However KNN gave the best prediction among these models, Logistics regression produce the more reasonable result in all the predictions but Pred(30).

D. Limitations

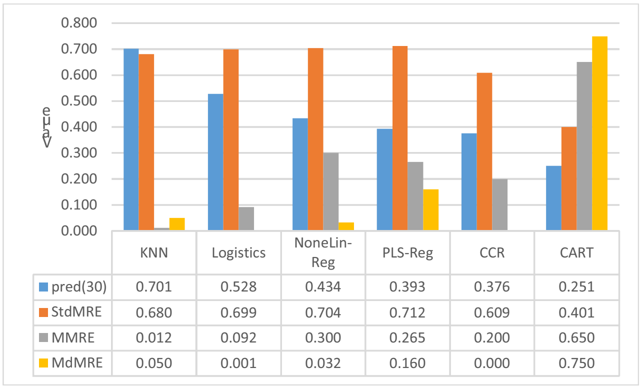
This approach is with some limitations, there is no grantee if same performance would happen in other platforms, also there are many different personal factors which will impact on competing on a task and have submissions, i.e domain familiarity, availability, requester company brand name, and new uploaded tasks in the platform.

In this research we only focused on uploading task time and percentage of parallel task uploading and associated award, and tried to analyze impact of parallelism on worker performance and also task stability, productivity and effectiveness in different period of time. The data we analyzed was limited to the four largest project uploaded in the platform from Jan 2014 to Feb 2015. Considering the characteristics of crowdsourcing software development, there might be a better task uploading representation for the same purpose.

V. Conclusions

Considering the highest rate for task completion and acceptable submissions, software mangers will be more concerned about risks for adopting crowdsourcing, and need to better decision support on analyzing and controlling the risk of insufficient competition and poor submissions. This paper reports an empirical study to address that end.

Based on the available empirical data and related researches, we came up with a set of hypotheses about impact of number of uploaded parallel tasks on tasks situation and



	KNN	Logistics	NoneLin-Reg	PLS-Reg	CCR	CART
🟦 pred(30)	0.701	0.528	0.434	0.393	0.376	0.251
🟧 StdMRE	0.680	0.699	0.704	0.712	0.609	0.401
⬜ MMRE	0.012	0.092	0.300	0.265	0.200	0.650
🟨 MdMRE	0.050	0.001	0.032	0.160	0.000	0.750

Fig. 6: Performance of number of submissions by each approach

workers' performance. These hypotheses are tested through empirical analysis on a set of 1482 competitive crowdsourcing tasks extracted from Topcoder platform. The analysis results conclude that: (1) crowdsourcing task scheduling follows typical patterns including prototyping, component development, bug hunt, and assembly and coding (2) budget phase distribution patterns does not following traditional patterns, and uploading task rate is not representing same budget rate associated with them as about 75% of uploaded tasks would price under 67% of total project budget; (3) Higher degree of parallelism would lead to higher demand for competing on tasks and shorter planning schedule to complete the project consequently better resource allocation.

In future we would like to focus on the crowd worker behavior in shorter period of time and try to analyze it based on task description and type to report more decision elements according to task size and date of uploading, achievement level, task utilization and performance.

VI. REFRENCES

[1] J. Howe, Crowdsourcing: Why the Power of the Crowd Is Driving the Future of Business. Crown Business, 2008.

[2] J. Surowiecki, The Wisdom of Crowds. Random House, Inc., 2005.

[3] He Zhang, Barbara Kitchenham, D. Ross Jeffery: Toward trustworthy software process models: an exploratory study on transformable process modeling. Journal of Software: Evolution and Process 24(7): 741-763 (2012)

[4] S. Faradani, B. Hartmann, and P.G. Ipeirotis, "What's the Right Price? Pricing Tasks for Finishing on Time", In Proc. Human Computation, 2011.

[5] N. Archak. Money, glory and cheap talk: analyzing strategic behavior of contestants in simultaneous crowdsourcing contests on topcoder.com. In Proceedings of the 19th international conference on World Wide Web, WWW '10, pages 21–30, New York, NY, USA, 2010. ACM.

[6] Y.Yang, R.L.Saremi, "Effects of Award on Worker Behaviors in Crowdsourcing ", ESEM 2015, Benjin,Chaina, In Process,Imperical software engineering and measures

[7] T.LAtoza,M.Chen, L.Jiang, M.Zhao,A.Van der Hoek, Y.Yang, "Borrowing from the Crowd: A Study of Recombination in Software Design Competitions", ICS-SE 2015, Florence, Italy, Crowdsourcing in Software Engineering

[8] Topcoder website: http://www.topcoder.com

[9] R.L.Saremi, Y.Yang, "Dynamic Simulation of Software Workers and Task Completion", ICS-SE 2015, Florence, Italy, Crowdsourcing in Software Engineering

[10] Kaufmann, N., Schulze, T. and Veit, D. More than fun and money. Worker Motivation in Crowdsourcing - A Study on Mechanical Turk. Proc. 17th AMCIS.(2011)

[11] Kulkarni, A., Can, M., and Hartman, B. Collaboratively Crowdsourcing Workflows with Turkomatic. In Proc. CSCW (2012).

[12] Dellarocas, C. Analyzing the economic efficiency of eBay-like online reputation reporting mechanisms. Proceedings of the 3rd ACM Conference on Electronic Commerce, (2001), 171–179.

[13] A. Begel, J. Bosch, M. A. Storey, "Social networking meets software development: perspectives from github, msdn, stack exchange, and topcoder." IEEE Software, vol. 30 (1), 2013, pp. 52-66.

View publication stats