Master Thesis

November 10, 2012

Crowd Manager

Experimental Analysis of an allocation and pricing mechanism on Amazon's Mechanical Turk

Bo Chen

of Changchun, China (07-726-581)

supervised by

Professor Abraham Bernstein

Patrick Minder

University of Zurich

Department of Informatics

Dynamic and Distributed Information Systems

Master Thesis

Crowd Manager

Experimental Analysis of an allocation and pricing mechanism on Amazon's Mechanical Turk

Bo Chen

University of Zurich
Department of Informatics

Dynamic and Distributed
Information Systems

Master Thesis

Author: Bo Chen, bo.chen@uzh.ch

Project period: May 10, 2012 - November 10, 2012

Department of Informatics, University of Zurich

Acknowledgements

I would like to thank my parents and Fei Liu for their constant support and believe in me. I would also like to thank Patrick Minder for his constant support, good guidance and crucial feedback. I appreciate the opportunity given to me by Professor Bernstein to write a master thesis about such an interesting topic. Finally, I would also like to express my thanks to Manuela Züger and David Caspar for their time to proof read my thesis.

Abstract

Before the invention of computers and calculation machines, any kind of computation was done by humans. The word *computer* was used to describe a person who would perform calculations as a profession. The rise of the internet and growing popularity of web 2.0 platforms like Wikipedia and Stackoverflow gave the word *human computation* a new dimension: It is not a couple of hundred people trying to solve simple mathematical calculations anymore, but millions of people trying to collaboratively solve complex problems, such as creating an all-encompassing encyclopedia or answering all kinds of questions on a specific topic in a timely manner.

New platforms like Amazon's Mechanical Turk (MTurk) have risen to support paid crowdsourcing work for micro-task markets and have quickly grown in size and popularity. With these kind of platforms, it has become possible to "program" the crowd and enable computer programs to perform complex tasks, such as intelligent text translation, intelligent text correction, intelligent image tagging, etc. However, the allocation of workers and the pricing mechanisms for such a big market are still very simple. If you have a set of tasks with certain time, quality and budget constraints, it is very hard for you to solve them, because currently you can only guess the "right" price for your tasks and hope for good solutions.

Minder et al. proposed an allocation and pricing mechanism that solves an Integer program incorporating the requestor's constraints to solve the allocation problem and a Vickrey-Clarke-Grooves payment mechanism to solve the pricing problem. In their initial simulation study they have shown that the CrowdManager mechanism leads to an overall better utility for the requestors in micro-task crowdsourcing markets compared to current fix price mechanisms. In order to test the results in the real world, we developed a prototype for the CrowdManager framework. We gathered various data through experiments on MTurk with the prototype and will answer the following research questions throughout this thesis:

(1) Do the assumptions in the CrowdManager model and the initial simulation hold in a real-world setting?

(2) How can we incorporate the observations of a real-world scenario in the CrowdManager's allocation and pricing model?

(3) How does the CrowdManager mechanism perform against the baseline mechanisms in a real world setting?

With our data analysis and hypothesis driven approach, we are able to conclude that the CrowdManager mechanism is a valid approach and worthwhile to be developed further. For this purpose, we have come up with several propositions for the enhancement of the CrowdManager's allocation and payment mechanisms.

Zusammenfassung

Vor der Erfindung von Computern und Rechenmaschinen wurde jede Art von Berechnung von Menschen ausgeführt. Das Wort *computer* wurde benutzt, um Personen zu beschreiben, die Rechnen als Profession ausübten. Mit dem Aufkommen des Internets und der steigenden Popularität von Web 2.0 Plattformen wie Wikipedia oder Stackoverflow gewinnt das Wort *human computation* eine neue Dimension: Es beschränkt sich nicht mehr auf hunderte Menschen, die versuchen simple mathematische Kalkulationen durchzuführen, sondern mehrere Millionen Menschen der ganzen Welt kommen zusammen, um komplexe Probleme anzugehen, wie dem Erstellen einer allumfassenden Enzyklopädie oder dem Beantworten von fachspezifischen Fragen innerhalb eines angemessenen Zeitraums.

Neue Plattformen wie Amazons Mechanical Turk (MTurk) sind entstanden, um bezahlte Micro-Task Crowsourcing Märkte zu bedienen. Mit dieser Art von Plattformen ist es möglich geworden die Crowd zu „programmieren", wodurch es Computerprogrammen erlaubt komplexe Aufgaben durchzuführen, wie z.B. intelligente Textübersetzungen, intelligente Textkorrektur oder intelligente Bildbeschreibung. Jedoch sind die Zuordnung der Arbeiter und die Preisfindungsmechanismen für einen solchen Markt noch sehr beschränkt. Wenn man ein Aufgabenset mit bestimmten Zeit, Qualität und Budgetbeschränkungen hat, dann kann man den Aufgaben momentan nur den „richtigen" Preis erraten und auf gute Ergebnisse hoffen.

Minder et al. haben CrowdManager Mechanismus vorgeschlagen, welches ein Integer Programm für das Zuordnungsproblem löst und ein Vickrey-Clarke-Grooves Zahlungsmechanismus für das Lösen des Preisfindungsproblems beinhaltet. In ihrer Initialen Simulationsstudie haben sie gezeigt, dass der CrowdManager Mechanismus zu einem höheren Nutzen für den Auftraggeber führt als herkömmliche Fixpreis Mechanismen. Um diese Resultate in der realen Welt zu testen, haben wir einen Prototypen für das CrowdManager Framework entwickelt. Wir haben diverse Daten durch Experimente auf Amazons Mechanical Turk Plattform gesammelt und werden in dieser Arbeit folgende Fragestellungen beantworten:

(1) Gelten die Annahmen im CrowdManager Modell sowie in der Simulationsstudie auch für die reale Welt?

(2) Wie können wir die Beobachtungen des realen Scenarios im Zuordnungs- und Preisfindungsmechanismus abbilden?

(3) Wie verhält sich die Performance des CrowdManager Mechanismus gegenüber der Baseline Mechanismen in einem realen Scenario?

Mittels unserer Datenanalyse und unserem hypothesengetriebenem Ansatz können wir schlussfolgern, dass der CrowdManager Mechanismus ein valider Ansatz ist und es sich lohnt, diesen weiterzuentwickeln. Dazu haben wir mehrere Verbesserungen für den Zuordnungs- und Preisfindungsmechanismus vorgeschlagen.

Contents

1 Introduction **1**
- 1.1 Motivation . . . 2
- 1.2 Research Questions and Description of Work . . . 2
- 1.3 Outcome/Contribution . . . 3
- 1.4 Outline . . . 3

2 Related Work **5**
- 2.1 Human Computation, Collective Intelligence, Social Computation and Crowdsourcing . . . 5
- 2.2 Managing Crowd Latency and the Relationship between Time, Price and Quality . . . 6
- 2.3 Pricing Mechanisms for Crowdsourcing Tasks . . . 7

3 The CrowdManager Framework **9**
- 3.1 Models and Assumptions . . . 9
  - 3.1.1 Requestor . . . 9
  - 3.1.2 Crowd Workers . . . 10
  - 3.1.3 The CrowdManager Mechanism . . . 10
- 3.2 Allocation Mechanism . . . 11
- 3.3 Pricing Mechanism . . . 11
- 3.4 Budget Constraints, Time Constraints and Infeasibility . . . 12
- 3.5 Limitations and Issues of the Current CrowdManager Framework . . . 12

4 Prototype **15**
- 4.1 Architectural and Technical Aspects . . . 15
  - 4.1.1 Software and Technologies Used for Development . . . 15
  - 4.1.2 Architecture . . . 16
- 4.2 Implementation of the CrowdManager Framework . . . 17

5 Evaluation **21**
- 5.1 Simulation . . . 21
- 5.2 Research Questions . . . 22
  - 5.2.1 The Recruiting Process . . . 22
  - 5.2.2 The Bidding Behavior . . . 23
  - 5.2.3 The Post-Allocation Process . . . 24
  - 5.2.4 Performance Comparisons . . . 24
- 5.3 Experimental Setup . . . 25
  - 5.3.1 Baseline Algorithms . . . 25
  - 5.3.2 Use cases . . . 25

5.3.3 Method 26
5.4 Results 26
5.4.1 The Recruiting Process 26
5.4.2 The Bidding Behavior 29
5.4.3 The Post-Allocation Process 31
5.4.4 Performance Comparisons 34

6 Discussion, Limitations and Future Work **39**
6.1 The Recruiting Process 39
6.2 The Bidding Behavior 40
6.3 The Post-Allocation Process 40
6.4 Performance Comparison 41

7 Summary and Conclusion **43**

List of Figures

4.1 A client-server architecture very common for web applications has been used for the prototype. . . . 16
4.2 Process model from the server's perspective . . . 17
4.3 Process model from a client's perspective . . . 18
4.4 The bidding interface. On the top the qualification test, on the bottom the bidding input. . . . 18
4.5 The retainer interface while the crowd worker is waiting. On the top a display of the current earnings and earnings in case of a leave. . . . 18
4.6 The job solving interface. On top a display of the current earnings, current status of jobs done and earnings in case of a leave. . . . 19

5.1 Histogram of the arrivals at the retainer in seconds. . . . 28
5.2 Distribution of bid price per job. Median: 0.05$, Average: 2.74$ . . . 30
5.3 Distribution of bid how many jobs the crowdworker wants to solve. Median: 10, Average: 1285457.45 . . . 30
5.4 Acceptance rate of the crowd workers for different price levels. . . . 31
5.5 Average (blue) and median (red) completion time of jobs for the x-th number of job done as well as a regression for an experience curve (green). . . . 32
5.6 Average quality delivered of tasks for the x-th number of task done. . . . 33
5.7 Average cost for successful allocations for each of the three methods for different budget levels . . . 35
5.8 Utility of different methods at different budget levels. Utility post-allocation = successful if successfully completed, Utility pre-allocation = successful if successful allocated. . . . 36

List of Tables

5.1 Experiments done for the various mechanisms and budget levels. CMM = CrowdManagerMechanism, T = Total, A = successful Allocation, C = successful Completion. 26
5.2 Bid prices and sizes for those that passed and those that failed the qualification test. 29
5.3 Number of successful allocations for each method and different budget levels. . . . 35

Chapter 1

Introduction

Before the invention of computers and calculation machines, any kind of computation was done by humans [1]. The word *computer* was used to describe a person who would perform calculations as a profession [1]. To process big calculation projects such as the Mathematical Table project in 1938 where human computers calculated mathematical key values, such as the value of pi or the values for sinus, people had to come up with error-free and efficient workflows [1]. Error detection mechanisms were applied by letting multiple people do the same calculations and other people compare the results [1]. In order to manage these kind of calculation processes, complex calculations and problems were broken down to simple calculations that could be done by human computers, similar to divide and conquer algorithms in modern computer science [1,2].

With the rise of computers in the last half of the 20th century, human computation was a long lost word. Modern computation machines became more and more powerful and were able to calculate pi with a 1'000 digit accuracy within a split-second. However, there are still many tasks (such as image tagging, text translation), that cannot be solved perfectly with machine computation, but are relatively easy to solve for humans. The rise of the internet and growing popularity of web 2.0 platforms like Wikipedia[1], YouTube[2], Flickr[3], Stackoverflow[4], etc., gave the word human computation a new dimension: It is not a couple of hundred people trying to solve simple mathematical calculations anymore, but millions of people trying to collaboratively solve complex problems, such as creating an all-encompassing encyclopedia or answering all kinds of questions on a specific topic in a timely manner.

A new word, called crowdsourcing, has been invented by Jeff Howe to describe some of these phenomenons [3]. He describes crowdsourcing as "the act of taking a job traditionally performed by a designated agent (usually an employee) and outsourcing it to an undefined, generally large group of people in the form of an open call" [4]. New platforms like Amazon's Mechanical Turk (MTurk)[5] have risen to support paid crowdsourcing work and have quickly grown in size and popularity. Currently, there are over 180'000 tasks (called HITs) available on the website. With these kind of platforms, it has become possible to "program" the crowd and enable computer programs to perform complex tasks, such as intelligent text translation [5], intelligent text correction [6], intelligent image tagging [7], etc.

[1] http://www.wikipedia.org
[2] http://www.youtube.com
[3] http://www.flickr.com
[4] http://www.stackoverflow.com
[5] http://www.mturk.com/

1.1 Motivation

However, the allocation of workers and the compensation approaches for such a big market are still very simple. If you have a set of tasks with certain time, quality and budget constraints, it is very hard for you to solve them, because the only constraint that you can manage is the budget constraint by assigning a fix payment per task. Imagine that you need a text translation of a certain quality level to be done within the next 10 minutes with a budget of only 10$. Would it be best to set a certain amount per word translated and let the crowd do it multiple times? Or would it be better to set the reward per sentence, and if so, how much? And how can you guarantee that you will have a feasible solution in 10 hours? Currently, you would be stuck to either go to a professional translation service, which would certainly cost more than 10$, or put your text on a popular crowdsourcing platform like MTurk for a fix price of 10$ and hope for good results within your time constraint. There are three main issues: crowd latency, lack of dynamic pricing mechanisms and lack of quality assurance mechanisms.

Crowd latency is one of the main reasons, why crowd sourcing has not found it's way into real time applications. For example the Word add-on Soylent [6] is such a real time application that suffers from crowd latency. If you want your text to be corrected an shortened, you mostly do not want to wait 20 or 30 minutes for the result, but you would expect the result to be there within seconds. This latency issue significantly limits the use of crowdsourcing markets for interactive real time applications.

The lack of dynamic pricing mechanisms causes the market to be very inefficient, as the requestor will always have to guess what a good price for his tasks might be. By setting the price too low, the tasks might never get finished on time. By setting the price too high, the requestor might waste a lot of money and in the end, just hiring a professional might have been cheaper and less stressful.

Quality assurance is an important topic and is in many cases a crucial factor. Imagine a text translation task for a law agency: one wrong translated sentence or word could result in a huge economic loss for the client. Currently, the possibilities to control the quality on crowdsourcing platforms like MTurk is very limited and many applications and frameworks like Soylent or CrowdLang develop their own mechanisms to control the quality [6, 8].

Minder et al. proposed an allocation and pricing mechanism that solves an Integer program incorporating the requestor's constraints to solve the allocation problem and a Vickrey-Clarke-Grooves payment mechanism to solve the pricing problem [9]. The crowd worker specific information would be collected via an auction interface. Auctions are very popular for solving allocation and pricing problems, for example allocation problems in the internet advertising market are solved using generalized second price auctions [10]. With a simulation study, Minder et al. have shown that their mechanism design can lead to overall more cost effective solutions and increases the requestors overall utility compared to fix price mechanisms [9].

1.2 Research Questions and Description of Work

Minder et al. have shown in their initial simulation study that the CrowdManager mechanism leads to an overall better utility for the requestors [9] in micro-task crowdsourcing markets compared to current fix price mechanisms. However, there are some real world problems, that the current state of the CrowdManager model does not consider. For example, that workers might drop out during their stay in the retainer or that they might not completely finish an assigned task package. To find out these and other real world problems and see whether the simulation results can be reproduced in the real world, a prototype for the current CrowdManager model has been developed. Amazon's Mechanical Turk platform has been chosen to perform the tests

on, as it offers a well established API to interact with, has the possibility to provide custom user interfaces and is a well established platform for payed crowdsourcing work. We have gathered various data through our experiments on MTurk with the prototype and will answer the following research questions throughout this thesis:

(1) Do the assumptions in the CrowdManager model and the initial simulation [9] hold in a real-world setting?

(2) How can we incorporate the observations of a real-world scenario in the CrowdManager's allocation and pricing model?

(3) How does the CrowdManager mechanism perform against the baseline mechanisms in a real world setting?

1.3 Outcome/Contribution

For this thesis, we collected data in various field experiments with our prototype. Using this data we were able to

- gain a better understanding of the recruiting process and prove that dropouts do exist throughout the whole process,
- gain a deeper insight into the bidding behavior of crowd workers and find hints to non-truthful bidding behavior,
- gain a better understanding of the post-allocation process and prove that completion times as well as delivered qualities are not constant over time,
- and gain performance measurements on the CrowdManager mechanism under real world constraints and show that the CrowdManager mechanism might be a viable approach.

We could conclude with several propositions for the enhancement of the CrowdManager's allocation and payment mechanisms:

- Create a model to predict the amount of workers needed given the requestor's constraints
- Incorporate decreasing completion times of crowd workers with experience curves
- Use a pricing mechanism that works well under the circumstances of low competition and non-truthful bidding

1.4 Outline

In the next chapter we will introduce the field of human computation, collective intelligence, social computing and crowdsourcing and discuss related work in this field of research. In chapter 3 we will describe the current state of the CrowdManager framework and highlight the current issues and limitations of it. In chapter 4 we will describe the prototype of the CrowdManager framework that we implemented for our experiments. In chapter 5 we will refine our research questions and define various hypotheses around these research questions. We will then use the collected data to validate our hypotheses with the help of various statistical methods. In chapter 6 we will discuss the results of the evaluation and their limitations. Finally, in chapter 7 we will summarize our findings, make a conclusion and suggest possible future work.

Chapter 2

Related Work

In this chapter, we will review the current state-of-the-art research in the field of human computation. In the first section we will define the terminology used for this master thesis and, by doing this, introduce the key concepts of human computation. In the second section we discuss current research on crowd latency. In the last section we will review the current research on pricing mechanisms for crowdsourcing tasks.

2.1 Human Computation, Collective Intelligence, Social Computation and Crowdsourcing

The term "human computation" has a long history and the terms "crowdsourcing", "collective intelligence" and "social computing" are used more or less synonymously. In the current research there is an ongoing discussion on how to distinct between these terms and serveral attemps at defining these terms have been undertaken [11–13]. In this work, we will consider human computation as a "computation that is carried out by humans" [9,13] and human computation systems as "intelligent systems that organize humans to carry out the process of computation" [13][p. 4] or "paradigms for utilizing human processing power to solve problems that computers cannot yet solve" [14].

As proposed by Howe, we will define crowdsourcing as "the act of taking a job traditionally performed by a designated agent (usually an employee) and outsourcing it to an undefined, generally large group of people in the form of an open call" [4]. Many platforms like Mturk or CrowdFlower[1] were created in the past couple of years to support this process of outsourcing tasks to an unknown crowd. These crowdsourcing platforms can be considered as human computation systems, but a human computation system can work without crowdsourcing by "assigning tasks to a closed set of workers hired through the traditional recruitment process" [13]. A good example for this is the software testing platform uTest[2], where uTest maintains its own crowd of software testers and "lends" it to companies who want their applications to be tested.

The term "social computing" is a very broad concept and can be defined as "the interplay between persons' social behaviors and their interactions with computing technologies" [15] or social communication between humans "mediated by technology" [11]. Good examples for this are technologies like blogs, wikis or online communities [11]. By this definition, human computation and social computing certainly have large intersections. While discussions like whether social computation technologies (e.g. wikis) also belong to human computation are very intriguing [11],

[1] http://crowdflower.com/
[2] http://www.utest.com/

it's quite clear that human computation does not always require social interactions between the workers [13][p. 5].

Finally, the term "collective intelligence" describes "the emergent intelligent behavior of individuals, which includes non-humans and non-living things" [13][p. 5]. By this definition, collective intelligence is a superset of social computing, crowdsourcing and human computation. Collective intelligence can produce very intelligent and astonishing results and multiple studies, such as the study by Wolfers & Zitzewitz where they revealed that the aggregated predictions of non-professionals consistently outperform predictions of professionals [16], have shown this.

The biggest difference of human computation to the other three concepts is that human computation research focuses on algorithms and "explicit control" (i.e. "explicit decomposition or assignment of tasks, etc.), "instead of focusing on studying human behavior" [13][p. 5] and thus this master thesis is classified as a work in the field of human computation, more specific human computation systems.

2.2 Managing Crowd Latency and the Relationship between Time, Price and Quality

Interactive and time-constrained crowdsourcing applications like Soylent [6], a Microsoft Word plugin which provides proofreading and editing services from the crowd, are limited to crowd latency. Crowd latency mostly arises due to the long process of recruiting suitable workers for the current task. On average, 56 seconds are needed to produce a single, unverified answer to a question [6] and more complex workflows take 22 minutes or more to produce a result [17]. Another approach suggested by Mason & Suri [18] is to build up a panel of crowd workers beforehand and notify them about upcoming tasks. However, most relevant to the CrowdManager framework is the retainer model introduced by [19]. The retainer model pays crowd workers a small wage to wait and to be ready to solve incoming tasks at any time. Using the retainer model, it is possible to have "on-demand synchronous crowds" and "nearly zeroes out wait times" [19]. Another possible approach is used in VizWiz [17], where the response time is improved by constantly reposting old tasks. While these approaches all address the crowd latency problem, they do not consider the possibilities of market inefficiencies caused by the lack of dynamic pricing models and the lack of selecting workers by the quality they deliver. This results in a non-optimal allocation of workers to tasks.

Practitioners and researchers are both looking for new techniques to influence or predict the remaining completion time. Because of the high importance of completion time for many applications, several prediction models have recently been developed by researchers. Huang et al. [20] try to predict the completion time by assessing the number of assignments per task and the financial reward. Wang et al. [21] predict completion times using survival analysis models. They assess the influence of various task parameters like price, time posted on MTurk, number of HITs, etc. on the completion time and come up with a model to predict completion times given these parameters. In contrast to these statistical methods, the CrowdManager framework uses a mechanism design approach. It elicits the opportunity costs from the current set of crowd workers, measures the quality and completion time for each task and then tries to find a cost-minimizing allocation of task to workers that guarantees a completion of all tasks within the specified time, quality and budget constraints [9].

Defining and finding the "correct" price for every worker for each task is very difficult. This has been researched heavily and several market studies have investigated the relationships between price, completion time and quality. Fehr and Goette [22] showed that there is no statistically significant impact on the quality of work with increased payments. Mason and Watts [23] have

come to the same conclusion, but they have also come to the interesting conclusion that increased payments do increase the quantity of tasks, and thus accelerates the completion time.

2.3 Pricing Mechanisms for Crowdsourcing Tasks

In an recent experiment Singer and Mittal [24] have proposed a new pricing mechanism for "Online Labor Markets". Their proposed mechanism assigns a task to a worker, when his bid is below a threshold that is calculated based on previous bids. This mechanism is designed for a domain where workers arrive sequentially and the requestor tries to maximize the number of tasks completed given a budget constraint. The CrowdManager framework in contrast uses the retainer model and does not try to maximize the number of tasks completed given a budget constraint, but instead tries to minimize the cost of the requestor given a quality constraint, a budget constraint and a number of task constraint [9].

Chawla et al. [25] study the mechanism design for crowdsourcing contest. A crowdsourcing contest is, where for example a company instead of hiring a specialist team to solve the problem, gives the problem to the crowd and pays pays a reward to the one who hands in the best solution. This is not the problem domain the CrowdManager framework is designed for. The CrowdManager framework focuses on problem domains, where possibly thousands of small and relatively simple tasks have to be solved, instead of one big task that requires a whole specialist team to solve, i.e. their auction design is made only for one item, whereas the auction design of the CrowdManager mechanism is for multiple items, as well as tasks that have time, budget and quality constraints, which might not always be the case for crowdsourcing contest tasks.

Another interesting approach to finding the "correct" price is the bargaining approach by Horton and Zeckhauser [26]. They implemented a dynamic bargaining interface and would let crowd workers solve image labeling tasks. Crowd workers would have the opportunity to solve an image labeling task for a fixed price and would then be asked to solve another image labeling task where the price was negotiable. They would be presented either a one cent or a five cent starting price and could negotiate from there up to a certain threshold. They found out that crowd workers "were generally reluctant to make counter-offers or end negotiations. As a result, subjects who received one cent offer ended up with far lower average wages than subjects in the five cents group" [26]. This approach, however, does not maximize social welfare. Once crowd workers find out the threshold wage, they will always counter offer at the threshold wage. The CrowdManager framework tries to maximize social welfare and collects multiple bids from multiple crowd workers instead of only negotiating with one crowd worker.

Horton and Chilton [27] tried to find out the general reservation wages of crowd workers on MTurk using labor and supply models from economics. However, this approach only finds out a general reservation wage, but some workers have lower reservation wages and some have higher reservation wages. The optimal wage at an individual level is not the general reservation wage. The CrowdManager framework tries to find the optimal allocation for each crowd worker with optimal price determination.

Chapter 3

The CrowdManager Framework

In this chapter, we will present the idea of the CrowdManager framework, which has been first introduced by Minder et al. [9]. We will first explain the idea of the CrowdManager by an example of a problem that can occur in the real world which could be solved using crowdsourcing. We will then explain the models introduced by Minder et al. to describe this process and show where we made extensions in order for the models to work in a real world application.

3.1 Models and Assumptions

Imagine a scenario where you would need to get 100'000 pictures tagged in the next hour and only have 1000$ to do so. You, the requestor, would probably want to recruit crowd workers via a crowdsourcing platform. But how do you want to allocate the tasks and what would be the payment scheme? If we want to create a platform that will handle the recruitment of workers, assignment of the tasks and payment of the workers, we will need a way to model the requestor, the crowd workers and the allocation and pricing mechanisms. In this section we will introduce the theoretical models for the requestor, the crowd worker and the CrowdManager mechanism made by Minder et al. [9] and extend them where needed.

3.1.1 Requestor

The requestor is modeled the exact same way as it is by Minder et al. [9]: Requestors submit work packages W that contain sets of m tasks to the CrowdManager application, i.e. $W = (w_1, ..., w_m)$ [9]. A requestor has a completion time constraint $T > 0$ for his tasks, which is the maximum time he can wait until all m tasks from W are solved [9]. He also has a quality constraint $Q \in [0, 1]$, which specifies the minimum level of quality that is demanded for each of the m tasks [9]. If all of the m tasks are solved with the minimum level of quality Q and are solved within the demanded time constraint T, then the requestor has a positive value $B > 0$ for this work package and 0 otherwise, where B is the requestors budget [9]. If we denote the costs as C and assume that the requestor has a quasi-linear utility of $U = B - C$ for a positive outcome of the work package (all tasks are solved within the quality and time constraints) and $U = -C$ for a negative outcome, then the requestor is willing to spend a maximum of B for all tasks to be solved and wants to minimize the costs C [9]. If we take all these parts together, then the requestor is defined as $R = (W, B, T, Q)$ [9].

3.1.2 Crowd Workers

The model of the crowd workers is also modeled the same way as it is by Minder et al. [9]: There is a set of n crowd workers $I = (i_1, ..., i_n)$ in the retainer [9]. Minder et. al. have assumed, that the crowd worker will not leave the retainer "unless they fail the qualification test or they have completed all tasks that have been allocated to them" [9]. This is clearly a shortcoming of the crowd worker model as it does not hold true in the real world. Crowd workers can leave the retainer at any time and may not be able to finish all the tasks allocated to them. This leads to two dropout rates, one while the crowd workers wait in the retainer and one after the crowd workers have been allocated their tasks. These dropout rates differ in their nature and will be analyzed in detail in chapter 5.

Each crowd worker has his own private cost of $c_i > 0$ for solving any of the tasks $w \in W$ and does not want to solve more than $j_i \geq 1$ tasks [9]. The workers quality level $q \in [0, 1]$ and completion time $t_i > 0$ is estimated by a qualification test [9]. As Minder et al. mentioned, there is a moral hazard problem that the quality of workers may drop over time and that completion time may increase. There is also the possibility of decreasing completion time and increasing quality, as learning effects might come into play. These behavioral effects will be analyzed in detail in chapter 5.

Putting everything together, the crowd worker specific information are denoted as $\theta_i = (c_i, j_i, t_i, q_i)$ and called "type" [9]. The joint type of all workers will be denoted as θ [9]. These so called types will be submitted by the crowd worker via the qualification test and bidding interface [9]. These type reports can be non-truthful if there is an incentive for the crowd worker to be non-truthful [9]. Thus the report by worker i will be denoted as $\hat{\theta}_i = (\hat{c}_i, \hat{j}_i, \hat{t}_i, \hat{q}_i)$ and $\hat{\theta}$ will denote the joint of all type reports of all crowd workers [9].

3.1.3 The CrowdManager Mechanism

The CrowdManager uses a mechanism M to allocate each of the tasks in the work package W based on the request R of the requestor and the joint type report $\hat{\theta}$ of all crowd workers that are currently waiting for an allocation in the retainer [9]. The mechanism defines two rules: an allocation rule, which maps all type reports in $\hat{\theta}$ to an allocation x, and a payment rule, which maps the vectors $\hat{\theta}$ and x to a price vector p [9]. The allocation vector x is defined as $x = (x_1, ..., x_n)$ where x_i is the number of tasks allocated to crowd worker i and the price vector is defined as $p = (p_1, ..., p_n)$ where p_i is the price paid per task solved to crowd worker i [9].

The CrowdManager mechanism has a slight extension in contrast to the way it is modeled by Minder et al. In order to give an incentive to crowd workers to participate in the bidding process and wait in the retainer, similarly to Bernstein et al. [19], we define a payment per minute waiting and an initial payment for their bid as p_{minute} and $p_{initial}$. We define their total stay time in the retainer as $t_{retainer}$ and the cost of retainer as $c_{retainer} = t_{retainer} * p_{minute} + p_{initial}$. In order to avoid too much dropout during the waiting time in the retainer, a minimum waiting time $t_{minimum}$ has been added. If a crowd worker leaves the retainer before $t_{minimum}$ has elapsed, he will receive no payment. In order to give an incentive to the workers to not leave immediately after the tasks have been allocated, a punishment multiplier α will be applied to the total payment if the crowd worker decides to leave before he finished all tasks. This part has not been modeled by Minder et al. in the initial CrowdManager model.

3.2 Allocation Mechanism

Given $\hat{\theta}, m, T, Q$ (joint type reports, number of tasks, time constraint and quality constraint) , solving the following integer program returns an optimal allocation that minimizes the total costs [9]. This combinatorial optimization problem is denoted as $x(\hat{\theta}, m, T, Q)$, where $x = (x_1, ..., x_n)$ denotes the total allocation of tasks [9].

$$\min_{x_1,\ldots,x_n} \sum_{i=1}^{n} \hat{c_i} x_i \tag{3.1}$$

$$s.t. \sum_{i=1}^{n} x_i = m \tag{3.2}$$

$$x_i * \hat{t_i} \leq T, \forall i \in I \tag{3.3}$$

$$x_i * \hat{q_i} \geq x_i * Q, \forall i \in I \tag{3.4}$$

$$x_i \leq \hat{j_i}, \forall i \in I \tag{3.5}$$

$$x_i \geq 0, \textbf{integer}, \forall i \in I \tag{3.6}$$

The first equation (3.1) denotes the objective to minimize the total costs. The second to sixth equation are constraints: (3.2) makes sure that exactly m tasks are assigned to crowd workers; (3.3) makes sure that each agent will be able to finish the allocated tasks x_i in the given time constraint T; (3.4) makes sure that each crowd worker with tasks assigned to him is able to fulfill the given quality constraint Q; (3.5) makes sure that no crowd worker gets more tasks assigned than he is willing to solve; and finally (3.6) makes sure that no fractional number of tasks are assigned [9].

3.3 Pricing Mechanism

The payment per task p_i for each crowd worker i has to be higher than the crowd workers opportunity cost, i.e. $p_i \geq \hat{c_i}$, and is called individual rationality constraint [9]. We also want a truthful mechanism, which means that crowd workers are best off reporting their true opportunity costs [9]. This means that p_i is strictly higher than $\hat{c_i}$ and will be the maximum price that crowd worker i is able to obtain with any other type report $\hat{\theta}'_i$ that is able to obtain an allocation [9]. These properties are obtained by using the Vickery-Clarke-Groves (VCG) payment rule, "where each agent's payment is the negative externality it imposes on the other agents by its presence in the mechanism" [9].

$$p_i(\hat{\theta}) = \sum_{k \in I, k \neq i}^{n} c_k * x_k^{-i} - \sum_{k \in I, k \neq i}^{n} c_k * x_k^* \tag{3.7}$$

Equation 3.7 shows such a VCG payment rule for each crowd manager i given the joint type reports $\hat{\theta}$. x^* is the optimal allocation with all crowd workers and x^{-1} is the optimal allocation without crowd worker i. Thus, the payment for crowd worker i is the total costs that would incur without crowd worker i, subtracted the costs that incurs if we considered crowd worker i. By using the VCG payment rule, "the CrowdManager's allocation and pricing mechanism is (1) truthful, (2) efficient, and (3) ex-post individually rational" [9]. These are well-known theoretical properties of the VCG mechanism and for a formal prove of these properties see Nisan et al. [28].

3.4 Budget Constraints, Time Constraints and Infeasibility

Algorithm 1 shows how the CrowdManager algorithm works. It will first gather the joint type reports $\hat{\theta}$ and then try to create an allocation by using the allocation mechanism of section 3.2. If the allocation can be done, then the algorithm will calculate the prices according to section 3.3 and test whether this solution is budget feasible. Currently, the cost of retainer $c_{retainer}$ is not beeing considered in the budget feasibility test.

Algorithm 1 CrowdManager's Task-to-Worker Allocation and Pricing Procedure

Require: I, W, B, T, Q
  $\hat{\theta} = runProcurementAuction(I, QualificationTest, m)$
  $x = allocationMechanism(\hat{\theta}, W, T, Q)$
  if x is feasible **then**
    $p = paymentMechanism(\hat{\theta}, x)$
  else
    return no completion time feasible allocation found
  end if
  $cost = \sum_{i \in I} p_i * x_i$
  if cost $\leq$ B **then**
    return x, p
  else
    return no budget-feasible solution found
  end if

3.5 Limitations and Issues of the Current CrowdManager Framework

As mentioned by Minder et al. and also discussed throughout the previous sections, the CrowdManager framework still has some issues if it will be implemented in the real world.

First, the initial model of the crowd worker assumes that there are no dropouts other than failing the initial qualification test. This is obviously not true, as there are many other reasons why the crowd worker can dropout: impatience, other liabilities, etc. This is a big issue and will lead to many problems if not dealt with properly. There are two dropouts that differ in nature and in their impact on the CrowdManager mechanism: (1) before the allocation has been done and (2) after the allocation. (1) can negatively impact the allocation, as too many pre-allocation dropouts will lead to a problem of not having enough bids and thus not enough competition and this will, in turn, negatively impact the outcome of the CrowdManager mechanism in total. (2) (as well as (1) to a certain extend) will have a very high negative impact on the requestors utility U, because $U = -C$ if not all m tasks are solved as defined in section 3.1.1. In chapter 5, we will examine the nature of the dropouts and propose solutions on how to handle dropouts in the different parts of the whole CrowdManager process.

Secondly, the quality and completion time of the crowd workers will not stay constant over time in the real world. There are various sources that can influence the human performance: learning effects, moral hazard problems, weariness, etc. This can either lead to decreasing or increasing completion times and qualities. Decreasing qualities and increasing completion times

will cause negative impacts to the CrowdManager mechanism, because this means that the completion time constraint T and quality constraint Q might not be kept anymore. Decreasing completion times can cause crowd workers with lower opportunity costs to not being considered as they might minimally break the completion time constraint. This, in turn, would break property (2) making the CrowdManager mechanism not efficient anymore. In chapter 5, we will examine whether or not there are significant performance decreases or increases and how to handle with those performance changes.

Thirdly, Minder et al. did not model the retainer costs. We added a payment scheme to the retainer, similarly to Bernstein et al. [19], by adding a payment per minute p_{minute} and an initial payment to participate in the bidding process $p_{initial}$. In order to avoid abusive behavior, i.e. intentionally failing the qualification test and getting $p_{initial}$ for "free" or leaving immediately after the qualification test, we will only pay out $p_{initial}$, if the qualification test has been successful and added a minimum stay time in the retainer $t_{minimum}$. The cost of the retainer $c_{retainer}$ has not been added to the overall algorithm of section 3.4. Adding a cost aspect to the retainer means that part of the budget has to be spent on keeping crowd workers in the retainer and might result in negative utility if the allocation fails, which was never the case in the simulations done by Minder et al. [9].

Fourth, assumptions of uniformly distributed bids prices, bid sizes, completion times and delivered qualities for tasks might not be true for the real world. All these variables could follow other distributions and other distributions might seriously affect the outcomes of the allocation mechanisms.

Finally, a VCG payment mechanism is a good choice if the crowd workers know their own true costs and know that by submitting their true costs, they can achieve the best possible outcome. This might not be the case in a real world setting, as many crowd workers might not even know what their true costs are. They might also choose to not trust the CrowdManager mechanism and report false costs, either ones that are higher or ones that are lower than their true costs. If the majority will not bid truthfully or there are not enough variations in the bids, then the VCG payment scheme might not be able to find competitive prices.

Chapter 4

Prototype

An integral part of this Master Thesis was the development of a prototype for the CrowdManager framework. In the first part of this chapter we will describe the architectural and technical aspects of the prototype. Then, in the second part of this chapter, we will explain the process model and the user interfaces of the CrowdManager prototype.

4.1 Architectural and Technical Aspects

In this first part of this section we will explain the various software and technologies used to develop the prototype. We will then explain the architectural design as well as the data model design of the prototype.

4.1.1 Software and Technologies Used for Development

The prototype has been implemented using the Google Web Toolkit (GWT)[1] as it offers a stable an uncomplicated way to develop real-time web applications. GWT allows to develop a web application using Java only. It translates the client code into HTML, JavaScript and CSS and translates client-server interactions into asynchronous JavaScript and XML (AJAX) calls. The program can be easily deployed on a JBoss[2] or Tomcat[3] server.

In order to be able to deal with data storage in an easy and uncomplicated way, PostgreSQL[4] in combination with Hibernate[5] has been chosen. Hibernate has become the de facto standard for database interaction and object-relational mapping (ORM) within Java. Hibernate allows easy access to database relations by facilitating the mapping process of Java objects to relational database models and vice versa.

The open source software Tomcat has been chosen as web application server to deploy the GWT application on. Tomcat has a very wide user base and is a well established web application server software. According to a market study of the online magazine Heise, in 2008 approximately 75% of the german companies and public administration used Tomcat as one of their application servers [29].

[1] https://developers.google.com/web-toolkit/
[2] http://www.jboss.org/
[3] http://tomcat.apache.org/
[4] http://www.postgresql.org/
[5] http://www.hibernate.org/

4.1.2 Architecture

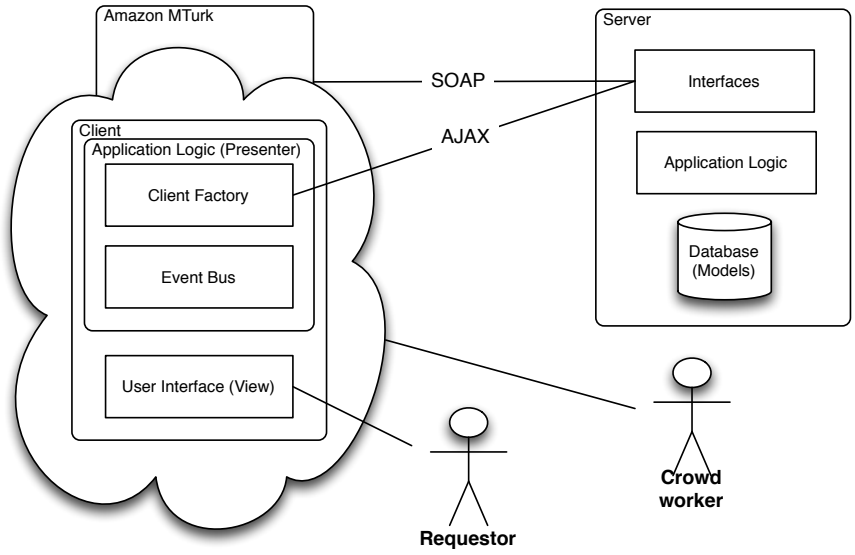


Figure 4.1: A client-server architecture very common for web applications has been used for the prototype.

The architecture of the CrowdManager prototype follows the simple client server model. Besides the client-server model, a GWT specific architectural pattern, called "Model-view-presenter" (MVP)[6] has also been applied to this prototype. It is very similar to the commonly known model-view-controller (MVC) pattern [30], but extends it with an event driven aspect by adding an event bus to it. Consider the following simple example using the MVC approach to model a cow: The model would describe that every cow has 2 eyes, one mouth, one nose, two ears, four legs, etc. The view would describe how the model looks from the outside, e.g. the four legs are attached to the body and the two eyes are attached to the head, etc. The controller would describe how our model functions, e.g. it uses her eyes to explore the area, uses her feet to walk, uses her mouth to eat, etc.

As you can see in figure 4.1, the server consists of three layers:

1. Interfaces: The interface layer is used to communicate with the clients and with MTurk. The communication is either via SOAP[7] or via AJAX.

2. Application Logic: The application logic, such as the allocation mechanisms, pricing mechanisms, quality grading, etc. is stored in this layer.

3. Database: The database layer stores the data and contains the model part of the MVP pattern.

The client, which is used by the crowd workers via MTurk and by the requestors directly, consists of two layers (see figure 4.1):

[6] https://developers.google.com/web-toolkit/articles/mvp-architecture
[7] http://www.w3.org/TR/soap/

1. User Interface (UI): The UI layer contains the user views and represents the view part of the MVP pattern. It is mainly HTML and some JavaScript (generated by GWT).

2. Application Logic: The application logic contains the presenters, which are used to manipulate the UI objects. It also contains the event bus as well as the client factory, which handles the communication with the server.

As MTurk provides the opportunity to create your own UI and embed it via an inline frame[8], the client code is also hosted on the server. The UI for the crowd workers is embedded into the MTurk UI and crowd workers use the MTurk platform in order to access the CrowdManager, whereas requestors will directly access the CrowdManager UI via the backend.

4.2 Implementation of the CrowdManager Framework

,

In this section we will describe the process model of our CrowdManager prototype. We will also show the important user interfaces for the various steps for our prototype.

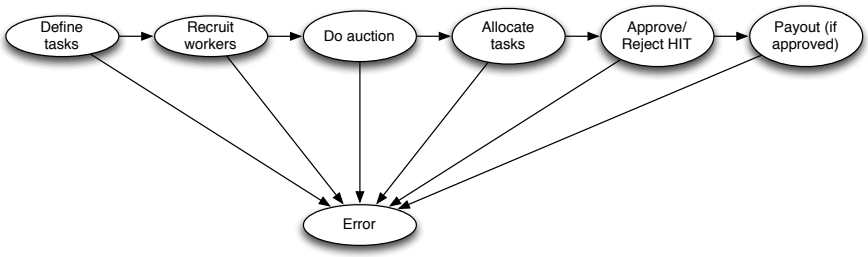


Figure 4.2: Process model from the server's perspective

The recruiting process starts by putting new HITs on MTurk via the administration interface (fig. 4.2). Parameters such as budget, completion time and quality constraints have to be defined beforehand. Once the HITs are online, crowd workers can preview the HIT by viewing an introduction page which explains the process and payment scheme of the CrowdManager (fig. 4.3). The crowd worker can then accept the HIT, solve the qualification test and submit his bid (fig. 4.3). Figure 4.4 shows how the bidding interface looks like: The qualification test is located at the top of the interface and the bidding interface is placed directly after the qualification test. The crowd worker has the opportunity to leave the bidding process without submitting the qualification test and bid.

If the crowd worker passed the qualification test, he will be put into the the retainer to wait for the allocation to take place. He will have a display of current earnings, earnings per minute and total time stayed in the retainer. If he did not pass the qualification test, the HIT will be closed and the crowd worker will earn nothing (fig. 4.3). This mechanism is to discourage crowd workers to intentionally fail the qualification test in order to earn "easy money". Figure 4.5 shows how the retainer interface is implemented in the prototype: The earnings display on the top, in the middle the total stay time and a warning to not leave or refresh the page and on the bottom a button to leave the retainer.

[8] see http://www.w3schools.com/tags/tag_iframe.asp for the definition of an inline frame

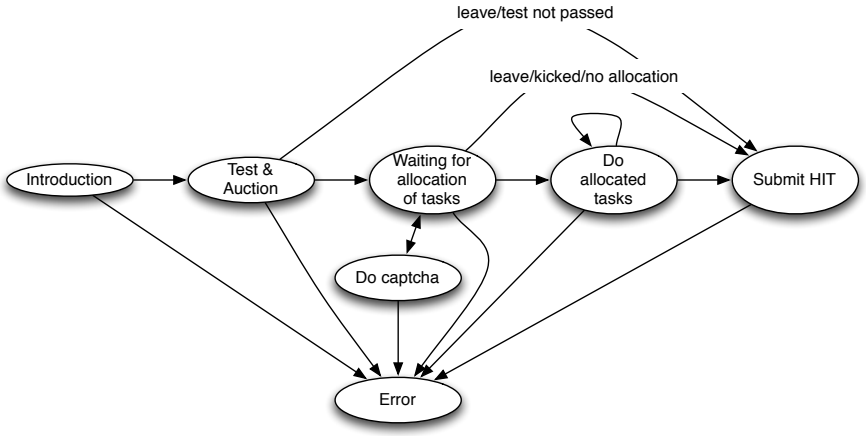


Figure 4.3: Process model from a client's perspective

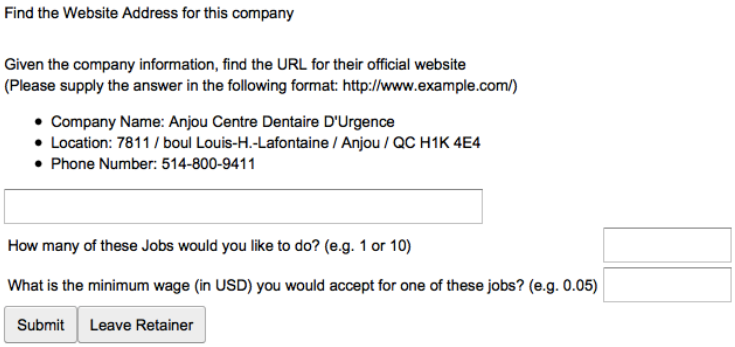


Figure 4.4: The bidding interface. On the top the qualification test, on the bottom the bidding input.

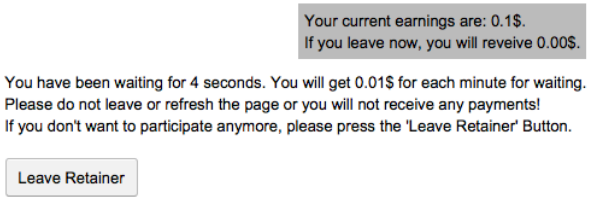


Figure 4.5: The retainer interface while the crowd worker is waiting. On the top a display of the current earnings and earnings in case of a leave.

During their stay in the retainer, the crowd workers have to solve captchas in a specified time period $t_{captcha}$ (fig. 4.3). Every time the crowd workers have to solve a captcha, a warning sound is played. There is a specific time frame $t_{solving_time}$ to solve the captcha and if they fail to solve the captcha within that specified time frame, they will be kicked out of the retainer and receive a penalty of $\alpha \in [0, 1]$ on their payment or no payment at all, depending on their total stay time. The same applies if they decide to leave themselves. This mechanism is supposed to prevent abusive

behavior like immediately leaving the retainer after submitting the qualification test and bid or leaving the browser window open and being allocated tasks without really being in front of the monitor or paying attention to it.

Once the specified time for the auction $t_{auction}$ is reached, the chosen auction mechanism will try to create an allocation with the joint type reports $\hat{\theta}$ of all crowd workers waiting in the retainer (fig. 4.2). The prototype has three different mechanisms implemented:

1. Fix price auction: A fix price p_{fix} is defined before the start of the recruiting process. Crowd workers can only submit how many tasks they want to solve, but not what the price is that they want per task. The final allocation is based on a "first come first serve" model.

2. First price sealed-bid auction with a specific price barrier: A price barrier $p_{barrier}$ is defined before the start of the recruiting process. Crowd workers who bid more than the barrier price will not be accepted to the retainer. The final allocation and pricing is based on a first price sealed-bid auction model.

3. CrowdManager mechanism: The allocation is done by implementing the integer program described in section 3.2 using the software Gurobi Optimizer[9]. The VCG payment scheme described in section 3.3 has also been implemented using the software Gurobi Optimizer.

Once the allocation is done, crowd workers who have been allocated tasks will be asked to solve the tasks (fig. 4.3). Crowd workers, who have not been allocated any tasks, will receive their full payment $c_{retainer}$. Figure 4.6 shows the job solving interface: The earnings and status display on the top and the task solving interface in the middle. The crowd worker can also leave here at any time, but will be punished with penalty multiplier α.

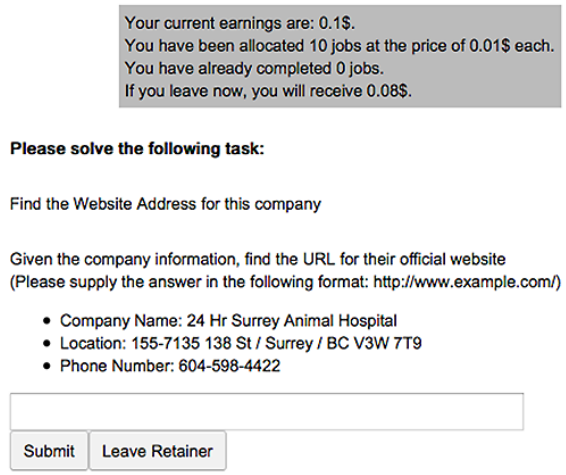


Figure 4.6: The job solving interface. On top a display of the current earnings, current status of jobs done and earnings in case of a leave.

[9]http://www.gurobi.com/products/gurobi-optimizer/gurobi-overview

Chapter 5

Evaluation

In this chapter, we will first look at the simulation results from Minder et al. [9] and define our research questions based on both the simulation results as well as the limitations and issues discussed in section 3.5. We will then formulate our research hypotheses and present our experimental setup to validate these hypotheses. Finally, we will evaluate the data gathered from our field experiments and validate our hypotheses based on the evaluations.

5.1 Simulation

Minder et al. showed in [9], by doing a simulation study, that the CrowdManager allocation and pricing mechanism outperforms a "first-completed first-served" allocation mechanism with a fix price p_{Task} per task (baseline 1) and a CrowdManager allocation mechanism with a fix price p_{Task} per task (baseline 2). This initial simulation showed, that the CrowdManager mechanism might be able to

(a) find allocations that are more cost effective than using a fixed-price mechanism,

(b) find the same or even more feasible allocations than the baseline mechanisms,

(c) and increase the requestor's utility, which depends on both the number of feasible allocations found and the total cost

In 10'000 distinct trials, they have benchmarked the CrowdManager mechanism against the two baseline mechanisms on various aspects across different budget levels: number of feasible solutions and average requestor cost. In a fix price setting, the mechanism would spend $p_{Task} = \beta * B/m$ per task, where β denotes the budget level. So for a 100% budget level, the fix price setting would spend all of the available budget evenly split on solving the m tasks. In their simulation, Minder et al. would also make some important assumptions:

(a) Work package: The number of tasks m was uniformly distributed between 50 and 500 tasks and an average completion time t' of 10 seconds per task was assumed [9].

(b) Requestor's constraints: The time constraint T and quality constraint Q were uniformly distributed between 60 and 1200 seconds and 0.65 and 0.85. The budget was set at $B = m * t' * p'$, where p' was uniformly distributed between 0.1 cents and 0.15 cents [9].

(c) Number of workers in the retainer: At least 10 workers would be put into the retainer for each run and depending on the number of tasks m, the expected completion time t', and the requestor's time constraint T, additional workers would be put into the retainer [9].

(d) Workers: The completion time of each worker t_i was uniformly distributed between 5 and 15 seconds and the opportunity cost was set at $c_i = t_i * r_i$, where $r_i$was the workers cost for working for one seconds and was uniformly distributed between 0.1 cents and 0.15 cents. The number of tasks j_i was uniformly distributed between 5 and $0.5 * m$ and the worker quality was uniformly distributed between 0.65 and 0.85 [9].

These assumptions assume that requestors know the average completion time and reservation price and would respect this in their budget calculations. This might be the case sometimes, but is certainly not the case every time in a real world setting. It would also assume, that the price per second p' set by requestors and cost per second r_i follow the same distribution and thus, are equal. This is certainly not the case in real world settings and certainly these two variables have different distributions. Other limitations are discussed in detail in section 3.5.

The success rate of allocations was approximately 55% overall, where the baseline mechanisms were able to find marginally more successful cases for higher budget levels, but the CrowdManager mechanism was clearly able to outperform the baseline mechanisms for the lower budget levels. The CrowdManager mechanism would spend significantly less than the baseline approaches for higher budget levels. However, these results are also attributable to the fact that crowd workers were bidding truthfully in the simulation, which might not be the case in the real world and has been discussed in section 3.5.

5.2 Research Questions

With the promising results of the simulation, the following questions remain unanswered:

(1) Do the assumptions in the CrowdManager model and the initial simulation [9] hold in a real-world setting?

(2) How can we incorporate the observations of a real-world scenario in the CrowdManager's allocation and pricing model?

(3) How does the CrowdManager mechanism perform against the baseline mechanisms in a real world setting?

In this thesis, we will focus on the following four areas in order to gain an answer to our three research questions: the recruiting process, the bidding behavior, the post allocation process, and the performance comparison between the three mechanisms.

5.2.1 The Recruiting Process

The recruiting process is an area that Minder et al. [9] did not model at all, they just assumed that the crowd workers were already in the retainer. Thus, our refined research question on the recruiting process is:

(2.a) How can we represent the recruiting process in the current CrowdManager model?

The recruiting process is an important process and knowing the arrival rate of crowd workers and their likelihood to drop out during this process is crucial for the whole CrowdManager mechanism, because initially we do not know how many workers we need to recruit to create a successful allocation and how long we have to wait to get a certain amount of workers into the retainer. The arrival rate and dropout rate are the two most important factors for the initial

recruitment of crowd workers and based on them (along with some other factors like quality distribution, bid distribution, etc.), the mechanism can predict how many workers to recruit in order to create a successful allocation within the given time constraint T. Other factors like the initial payment p_{intial} to take part in the auction or the impact of the payment per minute in the retainer p_{minute} have not been researched in this thesis. Examining the dropout rates and the arrival rate of crowd workers will help us to find an answer to our research question 2.a.

Dropouts in the Recruiting Process

Several other issues have already been identified by Minder et al. and have been covered within chapter 3, one of them being the dropout rate. Minder et al. have initially "assumed that crowd workers do not leave the retainer before all tasks are solved" [9]. This will clearly not be the case in a real world setting. As we have defined in section 3.5, we assume to observe two different dropout rates, one pre-allocation and one post-allocation.

Due to the nature of human becoming less patient over time, our initial hypothesis for the pre-allocation dropout rate is the following:

Hypothesis 1.1 *Pre-allocation dropouts do occur and they are proportional to the time stayed in the retainer.*

Arrival of Crowd Workers

We assume that the arrival rate $t_{arrival}$ of crowd workers at the retainer is a Poisson distributed variable, since the arrival rate of crowd workers given a fixed time interval x is independent of the arrival rate of the crowd workers since the last time interval $x - 1$. This will let us estimate arrivals for a given time period and leads us to a second hypothesis:

Hypothesis 1.2 *The arrival rate $t_{arrival}$ at the retainer is a Poisson distributed variable.*

5.2.2 The Bidding Behavior

The bidding behavior is a very important area, as the bidding behavior of the crowd workers will directly influence the success rate of the allocation done by the CrowdManager mechanism. Minder et al. [9] assumed in their simulation study that the bid prices and bid sizes are uniformly distributed variables. They also assumed, that crowd workers would bid truthfully. Thus, our research questions regarding the bidding behavior are:

(1.a) Are the assumptions of uniformly distributed bid prices and bid sizes made by Minder et al. true?

(1.b) Is the assumption of truthful bidding crowd workers made by Minder et al. true?

Unlike Minder et al., we assume that the bid price $\hat{c}$ as well as the bid for number of tasks $\hat{j}$ are log-normal distributed variables, as very high values have a very low likelihood of occurrence and the lower range values are are much more likely to occur. By gaining more insight into the bidding behavior, we will be able to do better predictions on how many bids are needed to fulfill certain budget restrictions and thus optimize the recruiting process. We can formulate the following two hypotheses:

Hypothesis 2.1 *The bid price $\hat{c}$ is a log-normal distributed variable.*

Hypothesis 2.2 *The bid amount $\hat{j}$ is a log-normal distributed variable.*

We also assume, that crowd workers are not bidding truthfully, but higher than their true private price c_i. This can be shown if crowd workers accept lower prices if you present them lower prices, but if they have the freedom of bidding, they will put in much higher prices.

Hypothesis 2.3 *The bid price $\hat{c}_i$ is not truthful.*

These hypotheses will help us to answer our research question 1.a and 1.b and gain a better understanding of the crowd workers' bidding behavior. Knowing the bid distributions will also help us to answer research question 2.a as the bid distributions can also be used as parameters in a prediction model for the recruiting process.

5.2.3 The Post-Allocation Process

Another issue mentioned by Minder et al. is that the completion time t_i and quality q_i of an individual crowd worker is "prone to various sources of variance" [9]. However, in their simulation study Minder et al. assumed that these two variables are constant over time. Another issue is the already mentioned dropout issue: crowd workers can leave the retainer even after they have been assigned tasks. Thus, our two research questions in the post-allocation process are are:

(1.c) Are the assumptions of constant completion time and qualities made by Minder et al. true?

(1.d) Is the assumption of no dropouts during the post-allocation process made by Minder et al. true?

Our hypothesis for the post-allocation dropouts is linked to the prices received per task, as the smaller the payment per task is, the less the crowd worker is willing to really solve the tasks.

Hypothesis 3.1 *post-allocation dropouts do occur and they are negatively proportional to the price crowd workers receive per task.*

We assume that an experience curve can be shown by analyzing delivered completion times of tasks over the course of completed tasks, as crowd workers will learn how to solve the allocated tasks faster over time. We assume that the experience curve will have a form of $C_n = C_1 * n^{-\lambda}$, where C_n is the cost of the n-th unit (measured as completion time in our case), C_1 is the cost of the first unit, n is the cumulative number of units and λ is the elasticity of unit costs with respect to cumulative volume [31]. We also assume, that a moral hazard problem and weariness effects can be measured by analyzing delivered qualities of tasks over the course of completed tasks, as decreasing the quality (i.e. finishing tasks "quick and dirty") is not being punished and will benefit the crowd workers. Once we know how these variables are distributed, we will be able to predict these variables. This leads us to the following two hypotheses:

Hypothesis 3.2 *Learning effects do occur and an experience curve can be derived from delivered completion times of tasks over the course of completed tasks.*

Hypothesis 3.3 *A moral hazard problem does occur and a decrease in delivered qualities of tasks over the course of completed tasks will be a strong indicator of it.*

5.2.4 Performance Comparisons

Given the results of the simulation study by Minder et al., we assume that these results also hold true in the real world, even with the extensions that have to be made to the models and the limitations of the model, especially regarding the dropout possibility and the assumption of the completion time t_i and the quality q_i to be constant. Thus, our three last hypotheses are:

Hypothesis 4.1 *The CrowdManager mechanism finds allocations that are more cost effective than the two baseline mechanisms.*

Hypothesis 4.2 *The CrowdManager mechanism is able to find at least the same amount of feasible solutions compared to the two baseline mechanisms.*

Hypothesis 4.3 *The CrowdManager's mechanism significantly outperforms the two baseline allocation and pricing mechanisms in terms of the overall requestor's utility.*

By comparing the three performance indicators cost effectiveness, allocation rate and requestor utility between the three mechanisms, we are able to answer our research question (2).

5.3 Experimental Setup

To test our hypothesis, we implemented two baseline mechanisms (as mentioned in section 4.2) and used address finding tasks as our sample tasks. The following section will explain the baseline algorithms, the address finding tasks and our methods.

5.3.1 Baseline Algorithms

As already mentioned in section 4.2, we have implemented two baseline algorithms in order to have a comparison:

1. Baseline 1: Analogously to Minder et al., we use a "first-completed first-served" [9] allocation. The price per task p_{Task} is fixed, and calculated as B/m. The crowd worker will have to solve the qualification test, will be shown the price per task and will be asked how many tasks he would like to solve.

2. Baseline 2: Similarly to Minder et al., we also first remove every worker from $\hat{\theta}$ where $\hat{c}_i > p_{Task}$ [9]. But after that, we will do a first price sealed-bid auction [32] instead of letting the CrowdManager algorithm define the optimal allocation.

5.3.2 Use cases

For the experiments, we used address finding tasks with data from the Canadian yellow pages. Crowd workers had to find the correct top-level url for a corporate website given the companies name, address, and phone number (a similar task-setting is also used in [33]) . These kind of tasks will make it relatively easy for us to calculate a quality from the solution, as there is only one right solution. We calculated the levenshtein distance [34] of the sample solution string and the answer string provided by the crowd worker and divided this by the maximum length of the two strings, in order to get a quality measurement between 0 and 1.

$q_i = 1 - levenshtein(solution, answer)/max(length(solution), length(answer))$

Consider "http://www.example.com" to be the correct solution for company abc. If the crowd worker submitted "http://www.example.com", the quality of the solution would be 1. If he submitted "http://www.example.com/", then the quality of the solution would be 0.96. If he submitted "http://www.example.com/index.html", then the quality would be 0.67.

5.3.3 Method

We conducted the experiments over the course of one week at different times throughout the day (GMT+1). We had a total of 87 runs, where 26 runs were done with the first baseline mechanism, 29 runs with the second baseline mechanism and 32 with the CrowdManager mechanism. Each run would put 15 HITs on MTurk as recruitment task and would have 30 address finding tasks to solve. Each HIT would give a initial payment $p_{initial}$ of 0.1$ for participating in the auction process and a payment per minutes p_{minute} of 0.01$ for staying in the retainer. The minimum stay time in the retainer $t_{minimum}$ has been set to 20 minutes and the punishment multiplier α has been set to 0.8. The auction would start 15 minutes after the HIT has been posted on MTurk. We assumed that solving these 30 tasks would give the requestor a utility of 3$. Analogous to Minder et al. [9], we ran the experiments for namely for the budget levels: 100% (3$), 75% (2.25$), 50% (1.5$), 25% (0.75$) and 10% (0.3$).

5.4 Results

Overall, 1130 crowd workers participated in a total of 87 experiment runs, 42 (48%) of them lead to a successful allocation and 19 (21.8%) of those 42 were successfully completed. Table 5.1 shows how many runs for which mechanism and which budget level was done and how many of them lead to a successful allocation and successful completion. In this section, we will go through the four areas recruiting process, bidding behavior, post-allocation process and performance measurement as previously defined in section 5.2. We will analyze each section and verify our hypotheses based on the data that we gathered through our experiments.

		Allocation Method								
		Baseline 1			Baseline 2			CMM		
Budget Level		T	A	C	T	A	C	T	A	C
	100% (3$)	5	4	3	6	4	1	7	2	1
	75% (2.25$)	5	2	1	6	6	2	7	6	1
	50% (1.5$)	4	3	1	5	5	2	6	2	1
	25% (0.75$)	6	4	2	7	1	1	7	0	0
	10% (0.3$)	6	3	3	5	0	0	5	0	0
	Total	26	16	10	29	16	6	32	10	3

Table 5.1: Experiments done for the various mechanisms and budget levels. CMM = CrowdManagerMechanism, T = Total, A = successful Allocation, C = successful Completion.

5.4.1 The Recruiting Process

In the recruiting process area, we have asked ourselves how to represent the recruiting process in the current CrowdManager model and have defined two hypotheses, one regarding the dropout rate and one regarding the arrival time of the crowd workers. In this section we will first examine our data gathered around the dropouts and verify hypothesis 1.1. We will then examine our data gathered for the arrival rate and verify our hypothesis 1.2. Finally, we will proceed to answer our research question 2.a on how to represent the recruiting process in the current CrowdManager model.

Hypothesis 1.1 *Dropouts pre-allocation do occur and they are proportional to the time stayed in the retainer.*

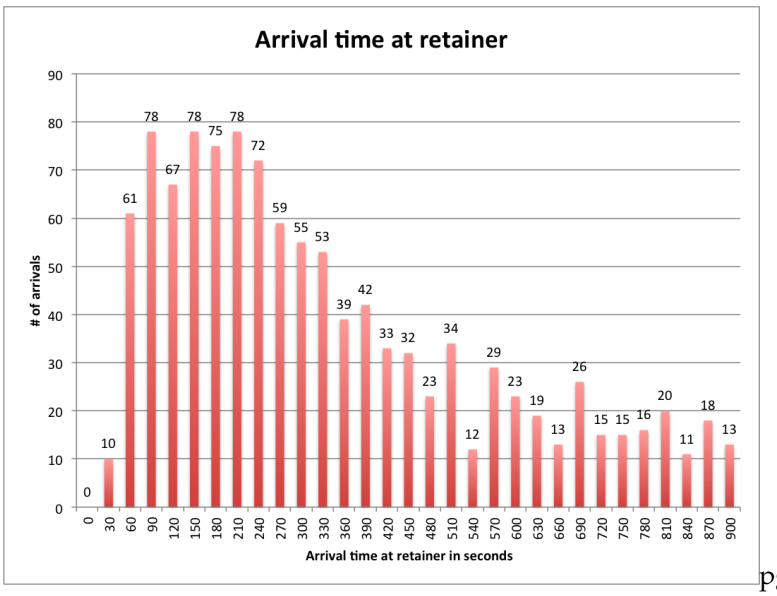
Out of the 1130 crowd workers, 304 (26.9%) did not agree to the terms and conditions of the CrowdManager prototype and dropped out during the introduction process (see fig. 4.3 for the process). 47 (5.7%) out of the remaining 826 dropped out during the auction process without solving the qualification test or submitting a bid. 160 (20.4%) out of the 783 remaining workers did not pass the initial qualification test and 83 (10.6%) were rejected for submitting a price that was higher than the price barrier p_{task} (this only occurred for the barrier method). Of the remaining 540 crowd workers, 84 (15.7%) dropped out during their waiting time in the retainer. Of these 84 crowd workers, 22 (26.2%) were kicked due to failing to submit one of the captchas in time. So overall, roughly every third crowd worker would drop out before submitting a bid or doing the qualification test. Of the remaining crowd workers, roughly every third will either not pass the qualification test or submit a price that was higher than the fix price p_{Task}. Roughly every seventh will drop out during their stay in the retainer. Comparing this dropout rate (15.7% at average 9.05 minutes stay time, median 9.3, σ 5.9 minutes, 3$^{\text{rd}}$ quartile 11.6 minutes) in the retainer to the dropout rate of Bernstein et al. (33.6% at 10 minutes [19] for baseline, 14.5% for reward system) we were able to achieve a significantly better dropout rate than their baseline mechanism and a similar dropout rate to their reward system. These dropouts did not cause any additional costs, as dropping out that early will not give the crowd worker any payment (see section 3.1.3). This first analysis validates the first part of our hypothesis 1.1 that drop outs do occur pre-allocation. However, two different drop outs can be observed: one pre-bidding, and one post-bidding, which is the drop out during the stay in the retainer.

Due to too many outliers in our data and one variable being a binary value, we had to recode our data, as Pearson's product-moment correlation coefficient [35][p. 33-46] should be used with data that has an underlying normal distribution [36]. We recoded the interval scale variable stay time in the retainer to a nominal scale variable of short and long stay times (see Stevens et al. [37] for a definition of interval scale and nominal scale). The phi coefficient ϕ_c [38][p. 282ff] can be used to measure the association between two binary variables, which is the case for us here with the recoded stay time variable and the drop out variable. A ϕ_c of 0.176 can be observed between the two variables and Pearson's chi-squared test returned a value of 16.181 and a p-value of 0.000, meaning that this is a significant observation. With this result, we can find strong support for our hypothesis 1.1: there is a significant weak positive linear correlation between the two variables stay time and dropout meaning that the longer the crowd workers stay in the retainer, the higher the chance is that he drops out.

Hypothesis 1.2 *The arrival rate $t_{arrival}$ at the retainer is a Poisson distributed variable.*

Figure 5.1 shows the frequency of the arrivals at the CrowdManager application. The Y axis shows the number of arrivals and the X axis shows the time elapsed since the HIT has been posted onto MTurk. The average arrival time of the crowd workers was 5.5 minutes (median 4.4 minutes, σ 3.9, 3$^{\text{rd}}$ quartile 7.9 minutes).

An initial analysis of our data on the arrival rate did not return a Poisson distribution and indicates that our hypothesis 1.2 does not hold true. A one-sample Kolmogorov-Smirnov test [39] successfully rejected the null hypothesis that the data followed the Poisson distribution. It returned a z-value of 5.854 and a p-value of 0.000. This means, that our data gathered from the experiments is not likely to be Poisson distributed as it significantly differs from a Poisson distribution. Looking at the histogram in figure 5.1, one can see that given the time intervals of 30 seconds, there is a very high difference in the frequency of the arrivals over time: the level of arrivals is high in the beginning and low in the end. This should not be the case for a Poisson



pg

Figure 5.1: Histogram of the arrivals at the retainer in seconds.

distribution, as the probability of a certain number of arrivals in each time frame should be the same, which should result in a similar number for each time frame.

The issue of the HIT attracting more crowd workers in the beginning may come from the fact that once a HIT is posted on MTurk, it will be shown on the first page if the crowd worker is sorting the HITs by creation date. Thus, in the beginning the HIT will attract more people and as time goes by, it will take more mouse clicks/page loads to find this specific HIT. Considering this fact, it means that our data is biased and we cannot say that the arrival rate is or is not Poisson distributed unless we can find a way to remove the bias.

By assuming that the bias is time dependent, as the more time elapses, the less visible the HIT will be on the front page, we can create a function to describe this bias. If we create 60 second time frames $X = (x_1, ..., x_n)$ for counting the frequency of arrivals (fig. 5.1 has 30 second time frames) and assume that the bias of the arrival rate can be removed by applying a multiplier of $1 - e^{-\delta x_n}$ to each of the frequencies ($e^{-\delta x_n}$ is a function used to calculate the half-life of radioactive chemicals and can be used here to calculate the "half-life" of a HIT) and do a one-sample Kolmorgorov-Smirnov test on this recoded data with various δ values, we still successfully reject the null hypothesis that the data followed the Poisson distribution.

Fadridani et al. [40] have shown that the arrival rate of the workers is a Non-Homogeneous Poisson Process, meaning that the "arrivals of two workers are not independent and they both depend on a latent variable, time t". This might be another factor that has to be considered and we are unable to do this analysis with our data, as we have only gathered data for arrivals during the day time and not for the night time.

(2.a) How can we represent the recruiting process in the current CrowdManager model?

We were able to prove that there is a significant, but weak linear correlation between the dropout rate and the stay time in the retainer. We were unable to prove that the arrival rate of the crowd workers followed a Poisson distribution, but Fadridani et al. [40] have already shown that the arrival rate of the workers at MTurk is a Non-Homogeneous Poisson Process. We propose to run new tests during day and night times in order to obtain a concrete function for the arrival rate.

This function can then be integrated into the CrowdManager model to predict the time needed in order to recruit n crowd workers. Another possible approach would be to record the pagination of the HIT on MTurk during the time of arrival and try to remove the bias with a function that contains the pagination as variable.

We also propose to gain more sample data on the relation between dropouts stay time in the retainer in order to be able to calculate a concrete function to describe the relationship between stay time and dropout rate in more detail. In addition, we propose to raise the competition level, i.e. raise the HIT to task ratio and put more HITs on MTurk due to the high dropout rate pre-bidding. For our experiments, we had a HIT to task ratio of 0.5, which might be good for bigger task numbers (and maybe even too high for very high task numbers), but are not optimal for lower number of tasks, as not having enough competition leads to having higher prices or no allocations being found at all.

5.4.2 The Bidding Behavior

In the bidding behavior area, we were curious whether the assumption of uniformly distributed bid prices and sizes made by Minder et al. [9] were true. We doubted the uniform distribution of these two variables and argued that they were rather log-normal distributed. In this section we will first analyze the two variables regarding their distribution and then proceed to answer our research question 1.a.

	Total		Passed Test		Failed Test	
	Price	Size	Price	Size	Price	Size
Average	2.74	1’285’457	0.073	6,325.17	13.06	6’250’010.70
σ (SD)	71.52	35’759’993.09	0.12	157’325.05	157.60	78’809’500.5
Median	0.05	10	0.05	10	0.075	10
1st quartile	0.03	5	0.03	6	0.04	5
3rd quartile	0.1	10	0.075	10	0.1	10

Table 5.2: Bid prices and sizes for those that passed and those that failed the qualification test.

A total of 783 bids were recorded. Table 5.2 shows some statistical key values for the the bid prices and sizes. Clearly, there were some non-serious bids, which have to be cleared out for the hypothesis testing, as the averages and standard deviations for total values are way off the chart. Interestingly, the average bid price for crowd workers who passed the qualification test was 0.073$ (median 0.05$) and average bid size was 6’325.17 tasks (median 10 tasks). The average bid price for workers who did not pass the test was 13.06$ (median 0.075$) and average bid size was 6’250’010.7 (median 10 tasks). This data tells us, that most of the non-serious crowd workers were mainly in the group of those that did not pass the test.

Hypothesis 2.1 *The bid price $\hat{c}$ is a log-normal distributed variable.*

A one-sample Kolmogorov-Smirnov test successfully rejected the null hypothesis that the logarithmized data followed the normal distribution and returned a z-value of 5,272 and a p-value of 0. This means, that the sample we gathered indicate that the variable bid price significantly differs from a log-normal distribution. It might be either due to the fact that the sample is biased or that the bid price simply does not follow a log-normal distribution. Looking at the histograms in figure 5.2, no apparent distribution is visible. However, there is a clear peak at 0.05$ and 0.1$.

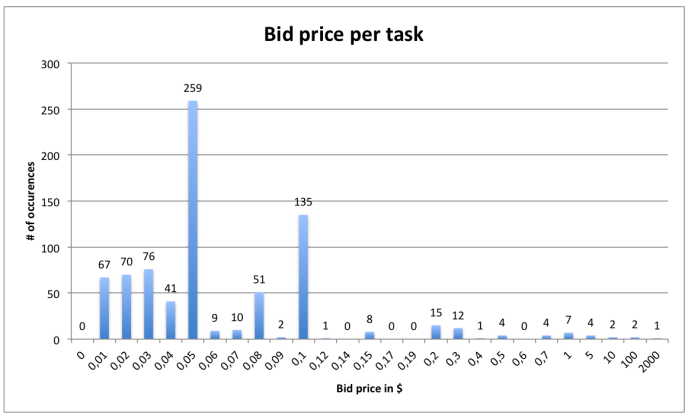


Figure 5.2: Distribution of bid price per job. Median: 0.05$, Average: 2.74$

Hypothesis 2.2 *The bid amount $\hat{j}$ is a log-normal distributed variable.*

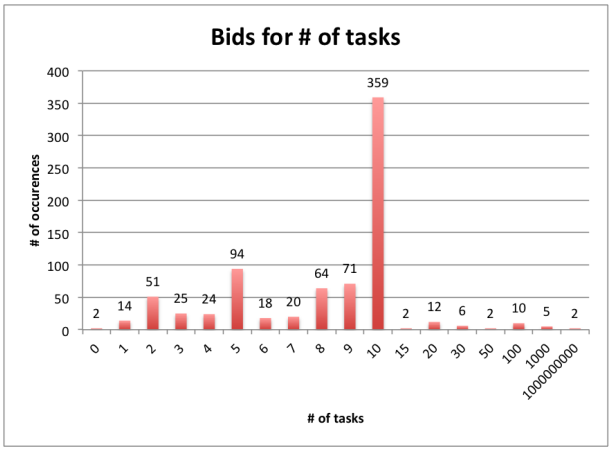


Figure 5.3: Distribution of bid how many jobs the crowdworker wants to solve. Median: 10, Average: 1285457.45

A one-sample Kolmogorov-Smirnov test successfully rejected the null hypothesis that the logarithmized data followed the normal distribution and returned a z-value of 10,328 and a p-value of 0.000. This means, that the sample we gathered indicate that the variable bid number significantly differs from a log-normal distribution. Looking at the histograms in figure 5.3, we can see a peak at 10 tasks. However, no distribution or other pattern is apparent.

We fail to validate our hypothesis 2.1 and 2.2 with the sample gathered through our experiments. However, the sample could be biased and possible biases sources could be the ethnicity, gender, age, etc. future experiments are to examine these various bias sources. Another possible bias source is the fact, that the bid interface suggested 0.05$ and 10 tasks as example (see fig. 4.4) and looking at the histogram, there is indeed a bias towards these two numbers.

Hypothesis 2.3 *The bid price $\hat{c}_i$ is not truthful.*

Figure 5.4 shows the acceptance rate of crowd workers for different price levels for our three

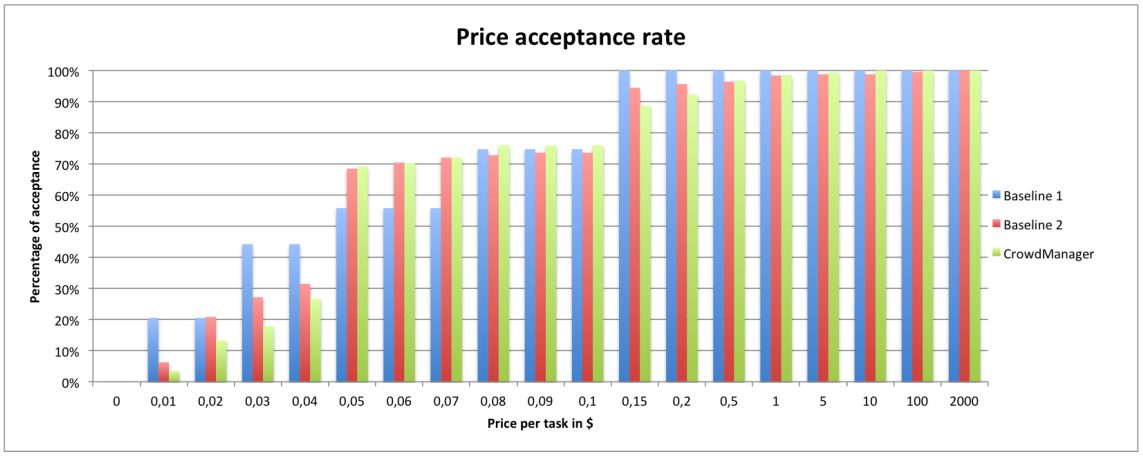


Figure 5.4: Acceptance rate of the crowd workers for different price levels.

mechanisms. We can see that more crowd workers were likely to accept prices lower than 0.05$ at the baseline 1 mechanism, where prices were fix and already predefined. 20% of the crowd workers participating in the experiments for baseline 1 mechanism were willing to accept a price of 0.01$ per task, whereas less than 10% were willing to accept the same price for baseline 2 and the CrowdManager mechanism, where they were free to report their reservation price. This is a strong indication that crowd workers were not bidding truthfully, as they tended to accept lower prices as they were bidding for, especially in the lower price areas.

(1.a) Are the assumptions of uniformly distributed bid prices and bid sizes made by Minder et al. true?

Due to the bias in our data we are unable to answer our research question 1.a. We propose to gather new samples with a "bias-free" interface design. Together with these samples, other possible bias sources such as ethnicity, gender and age should be recorded as well.

(1.b) Is the assumption of truthful bidding crowd workers made by Minder et al. true?

Due to a bias in our data, we cannot answer research question 1.b. The fact that the interface suggested 0.05$ as an example price may have led to the untruthful behavior of crowd workers, as more crowd workers were willing to accept a price of 0.05$ in the non-fix price mechanisms.

5.4.3 The Post-Allocation Process

For the post-allocation process we have formulated two research questions. First, we want to know whether there is a dropout rate during this process and second we want to know whether the qualities and completion times of crowd workers are constant. We assumed that the dropout in this process is negative proportional to the price per task. We also assumed that completion times and qualities of crowd workers would drop over time. In this section we will first analyze the dropouts and answer verify our hypothesis 3.1. We will then analyze the completion times and qualities and verify our hypotheses 3.2 and 3.3. Finally, with the knowledge gathered through our analysis, we will answer our research question 1.b and 1.c.

Hypothesis 3.1 *Dropouts post-allocation do occur and they are negative proportional to the price crowd workers receive per task.*

We have a very high dropout rate during the task solving process: 51 (31.1%) of the 164 crowd workers that had been allocated tasks, did not solve all of the tasks allocated to them. This indicates, that the punishment multiplier α might need some correction and an extra reward if all of the allocated tasks are finished, similarly to Bernstein et al. [19], could possibly improve the dropout rate post allocation.

Due to not having normal distributed data and the dropout variable being a binary value, we recode the variable allocated price per task from an interval scale to a nominal scale. After recoding, we obtain a phi coefficient ϕ_c of 0.098 and the chi-squared test returns a value of 1.59 and p-value of 0.452. This means, that there is no statistically significant correlation between the dropout and the price paid per task. This indicates that our hypothesis 3.1 is partly incorrect: the dropout is neither negative nor positive proportional to the price paid per task.

Hypothesis 3.2 *Learning effects do occur and and an experience curve can be derived from delivered completion times of tasks over the course of completed tasks.*

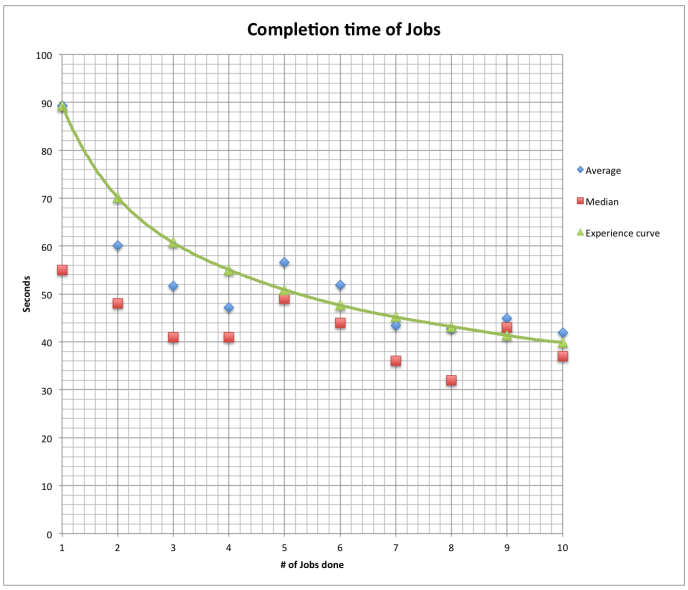


Figure 5.5: Average (blue) and median (red) completion time of jobs for the x-th number of job done as well as a regression for an experience curve (green).

As both our variables are not normal distributed, Spearman's rank-order correlation coefficient ρ is a good measurement for the correlation between the two variables [36]. A Spearman rank-order coefficient $\rho = -0.225$ can be observed from the two variables completion time and number of tasks completed. A 2-sided significance test returned a p-value of 0, which indicates that this is a statistically significant observation and the null hypothesis of no correlation between the two variables can be rejected. This means, that there is a significant weak negative linear correlation between the two variables completion time and the number of tasks completed. In figure 5.5 a regression for an experience curve has been done with an λ of 0.35 and a C_1 of 89.3, so the function for the experience curve would $C_n = 89.3 * n^{-0.35}$ (green curve). The average (blue) and median (red) completion times are also plotted alongside the experience curve in figure 5.5. A very strong explanation for this negative correlation of completion time and number of tasks

completed is the fact that crowd workers learn with each task how to solve the task faster. Thus, our experiments strongly support our hypothesis 3.2.

Hypothesis 3.3 *A moral hazard problem does occur a decrease in delivered qualities of tasks over the course of completed tasks will be a strong indicator of it.*

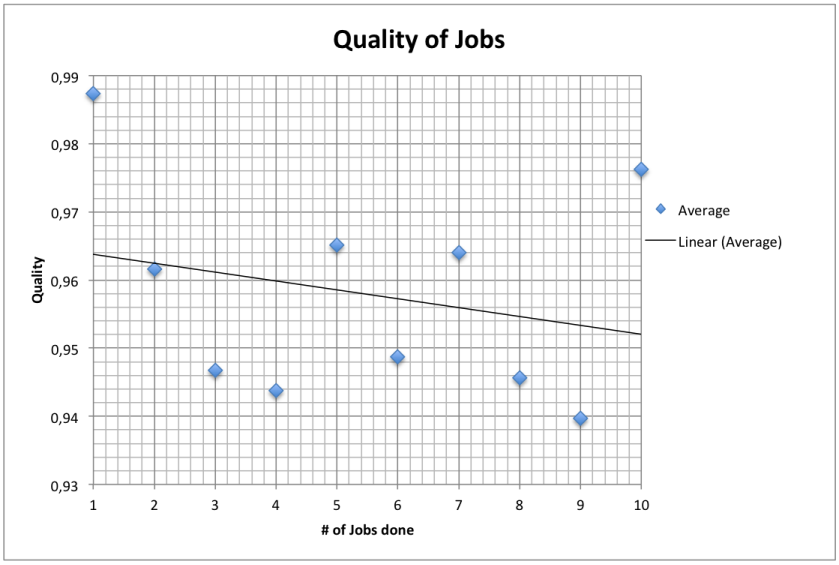


Figure 5.6: Average quality delivered of tasks for the x-th number of task done.

A Spearman rank-order coefficient $\rho = -0.105$ can be observer from the two variables quality and number of tasks completed. A 2-sided significance test returned a p-value of 0.001, which indicates that this is a statistically significant observation and the null hypothesis of no correlation between the two variables can be successfully rejected. This means, that a very weak, statistically significant, negative linear correlation can be observed between the two variables quality and number of tasks completed. If we plot the average quality for the number of tasks completed and do a linear regression (fig. 5.6), we can see that there is a very small decrease in quality over the course of tasks completed. This shows that there is a very small decline in quality over the course of tasks completed. A good explanation for this phenomenon is that there is a moral hazard issue: crowd workers do not get paid for better quality and solving the task in lower quality can lead to quicker completion times, which benefits the crowd worker. The results of our analysis supports our hypothesis 3.3.

(1.c) Are the assumptions of constant completion time and qualities made by Minder et al. true?

The results of our experiments clearly showed that the assumption for constant completion times and qualities as well as the assumption for no dropouts made by Minder et al. is not true. We propose to add the changing completion time and quality over time to the current allocation mechanism from section 3.2. This would mean that we refine equation 3.3 from the integer program for address solving tasks to:

$$\hat{t_i} * x_i^{-0.35} \leq T, \forall i \in I \tag{5.1}$$

For other kind of tasks we will need to find out the λ coefficient. Decreasing qualities are a problem as well and rather difficult to integrate into the integer program. We propose to discourage workers to submit sub-par work by randomly inspecting submitted tasks. A reputation system could also solve the problem of decreasing qualities. This needs further research and a definite solution for this problem cannot be drawn at this point as developing up with a solution for moral hazard problems lies beyond the scope of this thesis.

(1.d) Is the assumption of no dropouts during the task solving process made by Minder et al. true?

We have recorded various dropout rates during our experiments and the most problematic dropouts were the dropouts during the post-allocation process. We have come up with the following proposals on how solve research question 2:

- Create an incentive to fulfill all tasks by giving a bonus if all allocated tasks are completed.
- Raise the punishment multiplier α (currently 0.8).
- Over-allocate tasks according to new dropout rates with proposed changes.

Future experiments have to be designed to find out what α and what bonus payments create a healthy balance between increased success rates and increased costs. After that, the new dropout rates have to be recorded and tasks should be over allocated according to the new dropout rates for the post-allocation process.

5.4.4 Performance Comparisons

The performance of the CrowdManager mechanism is a very central research are. In particular, we want to know how the CrowdManager mechanism performs against the two baseline mechanisms. To analyze this, we defined three performances indicators: cost effectiveness, allocation rate and requestor utility. In this section, answer research question 2 by comparing the performance indicators for all three mechanisms.

Hypothesis 4.1 *The CrowdManager mechanism finds allocations that are more cost effective than the two baseline mechanisms.*

We do not have enough data for a statistical analysis of the budget used for each mechanism for the various budget levels. Figure 5.7 shows the average costs for each of the mechanisms and budget levels for successful allocations and looks very similar to the outcome of the simulations by Minder et al. [9], but since we do not have enough data, we cannot make any statistical statements about the cost effectiveness.

Hypothesis 4.2 *The CrowdManager mechanism is able to find at least the same amount of feasible solutions compared to the two baseline mechanisms.*

A one way analysis of variance (ANOVA) test [41][p. 457ff] test on the differences in successful allocations found, returns an F value of $F_{(2,84)} = 3.166, p = 0.047$, which means that there is a significant difference in the means of the number of successful assignments between the three mechanisms on a 5% significance level. Having a deeper look with Tukey's honestly significant difference (HSD) test [41][p. 471ff], we can observe no significant difference between either of the mechanisms and that the differences are mostly due to chance, i.e. not statistically insignificant. This may be due to the fact that the sample for each of the three mechanisms differ in size (26, 29, 32) and the Tukey test is more conservative and applies a correction on the data for this

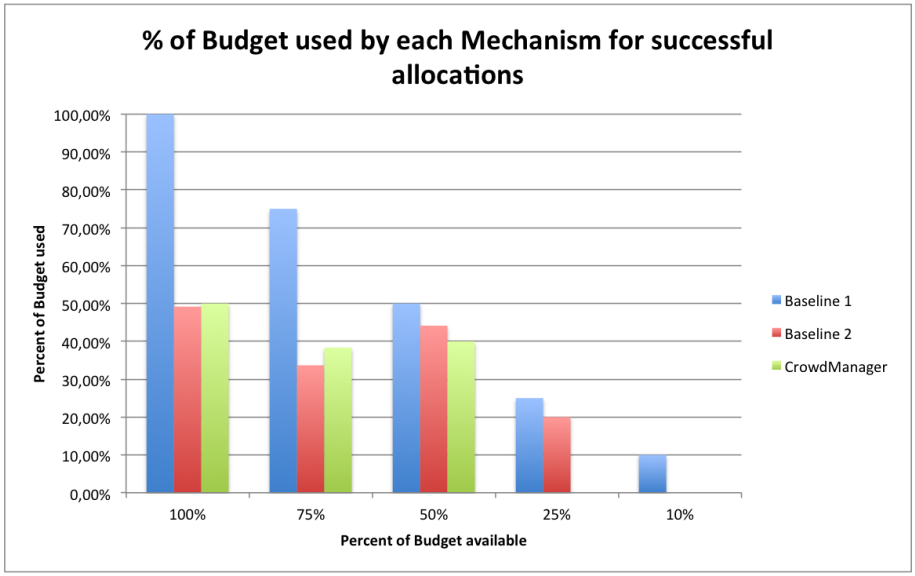


Figure 5.7: Average cost for successful allocations for each of the three methods for different budget levels

		Allocation Method		
		Baseline 1	Baseline 2	CrowdManager
Budget Level	100%	4 (80%)	4 (67%)	2 (29%)
	75%	2 (40%)	6 (100%)	6 (86%)
	50%	3 (75%)	5 (100%)	2 (33%)
	25%	4 (67%)	1 (14%)	0 (0%)
	10%	3 (50%)	0 (0%)	0 (0%)
	Total	16 (62%)	16 (55%)	10 (31%)

Table 5.3: Number of successful allocations for each method and different budget levels.

problem, whereas the ANOVA test does not. Considering the fact that the data sample is very small as well as differs in size, choosing the more conservative approach of Tukey's test, is the better solution. Thus, there are no statistical significant differences in the means of the number of feasible allocations found by each of the three mechanisms, meaning that our data shows support for hypothesis 4.1: The CrowdManager mechanism can find as much successful allocations as the other two baseline mechanisms.

Hypothesis 4.3 *The CrowdManager's mechanism significantly outperforms the two baseline allocation and pricing mechanisms in terms of the overall requestors utility.*

Figure 5.8 shows the various utilities $U = B - C$ for the requester for each method and the respective budget levels. For good comparability, we normalized the data by dividing the results by 3$ analogous to Minder et al. [9]. The two tables in the first row shows the utility with retainer costs in the calculations $C = c_{retainer} + c_{allocation}$. The utility post allocation considers an allocation successful and thus making $B = 3\$$ if all crowd workers have completed all allocated tasks to them. This makes utility post allocation consider the dropouts after the allocation, which was never the case in the simulations done by Minder et al. [9]. Utility pre allocation considers an allocation successful if the mechanism was able to find a successful allocation, thus making

$B = 3\$$ far more often, because this does not consider dropouts after the allocation. The numbers behind the utility is the number of observations for this case.

Utility with allocation costs post allocation

	Baseline 1		Baseline 2		CrowdManager	
100%	-49,08%	3	-39,61%	1	-61,40%	1
75%	-41,82%	1	-36,81%	2	-76,53%	1
50%	-37,46%	1	-46,20%	2	-30,73%	1
25%	-20,92%	2	-26,02%	1	-66,45%	0
10%	7,66%	3	-28,23%	0	-85,58%	0
Total	-26,30%	10	-34,93%	6	-63,84%	3

Utility with allocation costs pre allocation

	Baseline 1		Baseline 2		CrowdManager	
100%	-49,08%	4	-2,94%	4	-49,49%	2
75%	-22,82%	2	20,30%	6	-16,87%	6
50%	4,63%	3	3,13%	5	-17,56%	2
25%	8,11%	4	-26,02%	1	-66,45%	0
10%	7,66%	3	-28,23%	0	-85,58%	0
Total	-9,48%	16	-7,02%	16	-45,72%	10

Utility without allocation costs post allocation

	Baseline 1		Baseline 2		CrowdManager	
100%	0,00%	3	-2,78%	1	2,38%	1
75%	-9,00%	1	9,22%	2	-6,81%	1
50%	-4,58%	1	6,53%	2	6,83%	1
25%	20,97%	2	11,43%	1	0,00%	0
10%	45,00%	3	0,00%	0	0,00%	0
Total	12,79%	10	5,22%	6	0,31%	3

Utility without allocation costs pre allocation

Budget	Baseline 1		Baseline 2		CrowdManager	
100%	0,00%	4	33,89%	4	14,29%	2
75%	10,00%	2	66,33%	6	52,86%	6
50%	37,50%	3	55,87%	5	20,00%	2
25%	50,00%	4	11,43%	1	0,00%	0
10%	45,00%	3	0,00%	0	0,00%	0
Total	29,62%	16	33,13%	16	18,44%	10

Figure 5.8: Utility of different methods at different budget levels. Utility post-allocation = successful if successfully completed, Utility pre-allocation = successful if successful allocated.

We do not have enough data to make any statistical statements or conclusions regarding the utilities of the different mechanisms on a budget level. But we do have enough data to make statistical statements on a total level between the mechanisms. The ANOVA test returned the following f values for each of the utilities

- Utility with allocation costs post allocation: $F_{(2,84)} = 7.210, p = 0.001$
- Utility with allocation costs pre allocation: $F_{(2,84)} = 13.768, p = 0.000$
- Utility without allocation costs post allocation: $F_{(2,84)} = 1.310, p = 0.275$
- Utility without allocation costs pre allocation: $F_{(2,84)} = 1.841, p = 0.165$

These results tell us, that the mean values of the three mechanisms are significantly different for the utilities with allocation costs post and pre allocation, but not significantly different for utilities without allocation costs post and pre allocation. This means, that if we considered the allocation costs in the utility calculation, we can see a significant difference in the mean values of the three mechanisms. If we do a Tukey's HSD test to have more insight in the significant differences, we will get that the utilities with allocation costs post and pre allocation are only different for the CrowdManager mechanism in both cases. Looking at the tables in figure 5.8 we can see that the CrowdManager mechanism is significantly lower than the two other baseline mechanisms for the utilities with allocation cost post and pre allocation. This means that if we considered the allocation costs, the CrowdManager mechanism performs worse than the two baseline mechanisms. If we do not consider the allocation costs, there is no statistically significant difference in the means of the utilities between the three mechanisms. By comparing utility with allocation costs we could say that hypothesis 4.3 could be rejected. If we compare utility without allocation costs we could also reject hypothesis 4.3 as there is no significant difference between

the means of the three mechanisms overall. But comparing these two utilities across all budget levels would be wrong as the comparison of the utilities have to be done on a budget level. This is not possible as our sample size is too small and thus we can neither reject nor confirm hypothesis 4.3.

(3) How does the CrowdManager mechanism perform against the baseline mechanisms in a real world setting?

Due to the small sample size we cannot answer our research question 2 in a satisfactory manner. We can say that the CrowdManager mechanism was roughly able to find the same amount, but we cannot make any statistically relevant conclusions on the performance indicators cost effectiveness as well as requestor utility.

Chapter 6

Discussion, Limitations and Future Work

In this chapter, we are going to discuss the results of our evaluation and the limitations of it. We will discuss the results of our data analysis in section 5.4 as well as point out possible limitations and future work along our four core areas recruiting process, bidding behavior, post-allocation process and performance comparison.

6.1 The Recruiting Process

We were able to show with our data that dropouts do exist and they were spread throughout the whole process. Our work shows how important it is to know about the dropouts and to deal with them, as otherwise the overall requestor utility might considerably suffer from it, because without any dropouts we would have had more competition and more competition might have resulted into more successful allocations. We propose to include the pre-allocation dropouts in the current CrowdManager model by increasing the HIT to task ratio to compensate these dropout and competition issues.

Knowing the arrival rate is as important as knowing the dropout rates, because the optimal time to execute the allocation is certainly not exactly 15 minutes after the task has been posted on MTurk. By knowing the arrival rate, dropout rates and bid distribution, one can predict whether it is worth to wait more or whether the auction should be closed right at that moment. Due to not having enough data and having a bias in the data, we could not show that the arrival rate of the workers is Poisson distributed or a Non-Homogeneous Poisson Process [40]. However, given our initial recruiting price $p_{initial}$ of 0.1$, most of the crowd workers (75%) would arrive before 8 minutes. An important factor influencing this arrival rate is certainly $p_{initial}$. Future work has to determine other sources of influence and assess whether it is possible to create a model to predict the number of crowd workers needed, given the number of tasks m, the quality constraint Q, the time constraint T and the budget constraint B. New experiments could either try to show a Non-Homogeneous Poisson Process by recording the arrival rate evenly spread through day and night, or record bias variables such as pagination of the HIT and try to remove the bias out of the data.

Retainer costs have not been modeled by Minder et al. previously in their simulation study. As can be seen in figure 5.8, the utility values with retainer costs are not very overwhelming and can be below 0% if an allocation fails. Raising the competition level, as already proposed, can help to increase the success rate of the auctions and thus the utility, but will also negatively impact the utility as raising the competition level means to recruit more workers into the retainer. This is a

very subtle balance task and finding the right balance can be tough and shall be the task of future to find out.

Retainer costs need to be considered in the overall CrowdManager algorithm when checking the budget constraint. Our proposal for the retainer payment is, as explained in section 3.5, an initial payment $p_{initial}$ for participating in the bidding process and a payment per minute p_{minute} for each minute stayed in the retainer. A payment of 0.01$ per minute just for being ready and doing nothing could be too high, as many crowd workers participated just to drop out at the post-allocation process. A different possible model suggested by Bernstein et al. [19] would be a reward system, as financial bonuses can be powerful incentives [23]. Future work could consider decreasing the payment per minute p_{minute} and adding a reward at the end if crowd workers reacted quick enough to solve the first task.

6.2 The Bidding Behavior

Due to the low number of bids per auction (on average 5.25 crowd workers per auction that stayed until the allocation of the tasks), a median bid price of 0.05$ and a very high concentration of bid prices towards 0.05$, the CrowdManager mechanism was unable to find allocations for budget levels below 50% and the baseline 2 mechanism was unable to find allocations for budget levels below 25%. On the contrary, the baseline 1 mechanism, where the price per task was fix, was well able to find allocations in these lower budget levels. This means that either we were very unfortunate and chance was not in our favor or crowd workers were mostly not bidding truthfully. By increasing the competition level, this problem could potentially be solved, but this, in turn, would also increase the retainer costs.

If we look back at figure 5.4, we can see that for the baseline 1 approach, where prices were fix, crowd workers were much more likely to accept prices lower than 0.05$. This is a strong indication that crowd workers were not bidding truthfully and would also accept prices lower than their bid submitted for the baseline 2 and CrowdManager mechanisms. It also means that crowd workers generally tend to "accept their fate" when presented with a fix price. A similar conclusion was reached by Horton and Zeckhauser in [26], where crowd workers who got presented lower prices, would also end up with lower prices in the bargaining process. But, as already mentioned in section 5.4.2, we have to be careful in reaching premature conclusions, as the data might be biased. Thus, we propose to redesign the user interface to be bias-free and not suggest any prices or bid sizes to the crowd workers.

6.3 The Post-Allocation Process

Dropouts post-allocation were a huge problem and comparing the utilities post and pre-allocation we can see that they had a massive influence (see figure 5.8 in section 5.4.4). The utilities in many cases (19 of 42, roughly 45%) were positive in a pre-allocation assessment and turned negative in a post-allocation assessment. Increasing the punishment multiplier α or implementing another reward system can be possible solutions here.

We have found an experience curve for address finding jobs through analyzing our data. This means that over the course of completing address finding tasks, the crowd workers learn on how to complete these tasks quicker and we can model this as an experience curve. Our experience curve is valid for address finding tasks and future work is to find out similar experience curves for other tasks (e.g. translation, image tagging, etc.). We propose to incorporate this experience curve in the CrowdManager framework when trying to find allocations for address finding tasks and other experience curves when trying to find allocations for other tasks. A learning algorithm

can be implemented to automatically "learn" and readjust experience curves once more data has been gathered. Incorporating the decreasing completion times into the allocation mechanism can lead to the mechanism finding successful allocations that were previously not possible due to the completion time constraint. Certainly, these values should be calculated with great caution, as too optimistic parameters bear the risk of tasks not being completed on time.

Another topic is the issue with decreasing qualities over time. Solutions to this problem have to be found either in the current literature or developed and tested. Operations research or human resource might be good areas to start a literature review on moral hazard problems and how to handle this problem, as certainly other industries, especially manufacturing industries, face the same issues with decreasing qualities. Certainly, some of the approaches used in CrowdLang [8], or the "Find-Fix-Verify" pattern used for Soylent [6] can be reused here.

6.4 Performance Comparison

Due to the small sample size, we were unable to answer our second research question on how the CrowdManager mechanism performs against the baseline mechanisms. However, the utility with retainer cost shows that the CrowdManager mechanism performed significantly worse than the two baseline mechanisms. This is mainly attributable to the fact that in the baseline 1 approach, the crowd workers would already know what the price was beforehand and in the baseline 2 approach, all crowd workers with too high costs would be filtered out before they were accepted to the retainer and thus causing less retainer costs. The CrowdManager mechanism would accept all bids and would pay everybody who submitted a bid and waited in the retainer until the allocation was done. Some of the bids show that some crowd workers may have abused this fact and submitted unrealistic bids with prices of 0.2$ and more. In order to prevent this behavior, we propose the CrowdManager to implement a price barrier of $p_{barrier} = 2 * B/m$. This way, most of the non-serious bids can be filtered out and most of the relevant bids are still kept.

Another issue that sometimes occurred was that the CrowdManager mechanism was unable to compute the prices and thus failing to generate an allocation because it did not have enough bids. Consider the following simple example of $n = 2$, $m = 2$ and bid prices and sizes of $c_1 = 1$, $c_2 = 2$, $j_1 = 1$ and $j_2 = 1$. In this case crowd worker 1 and 2 would get allocated 1 task each. To calculate the price for crowd worker 1, we would need to calculate the total costs that would incur without crowd worker 1 subtracted the costs that incurs if we considered crowd worker 1 (see section 3.3). If we do not consider crowd worker 1 bid, then there would be no allocation, as there are not enough bids. This was sometimes a problem and caused the CrowdManager mechanism to fail an allocation although there was one possible.

Both the possibility of untruthful bidding prices of the crowd workers as well as the fact of low number of bids per auction make the currently implemented VCG payment rule suboptimal. In order to get better results, we would need to raise the competition level and gather more data through more experiments. Due to the budget and time restrictions of this thesis, we were unable to gather enough data and further experiments need to be conducted to gather enough data to make statistically relevant statements about the performance of the three mechanisms on a budget level. We propose for future work to also test out other pricing mechanisms that are less error prone under the circumstances of low competition and untruthful bidding.

Chapter 7

Summary and Conclusion

Minder et al. [9] introduced the CrowdManager framework to solve the current pricing and allocation problems in micro-task crowdsourcing markets. The CrowdManager framework relies on mechanism design and uses a reverse auction to determine prices and allocation of tasks. In their initial simulation study, a benchmark of the CrowdManager mechanism versus two baseline mechanisms, revealed promising results. In order to verify these results and the assumptions made for the model and simulation, we have programed a prototype for the CrowdManager framework. We chose MTurk as our crowdsourcing platform to to run our experiments on as it is a well established platform for paid crowdsourcing work. We defined the four research areas recruiting process, bidding behavior, post-allocation process and performance comparison as we felt that these areas were the most important areas to look at in order to answer our three research questions about the correctness of the assumptions and the repeatability of the results in the real world. For the research question regarding the assumptions, we defined four concrete research questions with each one of them focusing on specific assumptions that were made by Minder et al. [9] for the CrowdManager framework.

We have ran a total of 87 experiments over the course of one week during day times (GMT + 1) on MTurk. With the results of our experiments we were able to show that dropouts do exist and they can be categorized into a pre-bidding dropout, post-bidding dropout and post-allocation dropout. These dropouts, especially the post-allocation dropout, had a very big impact on the performance results. We proposed to take the dropouts into consideration for the CrowdManager mechanism. The pre-bidding dropout can be taken into account by increasing the HIT to task ratio. The post-bidding dropout can be taken into account by increasing the HIT to task ratio as well as implementing a reward system to the retainer payment scheme. The post-allocation dropout can be taken into account by increasing the punishment alpha, adding a reward system to the post-allocation process and over allocating tasks according to the dropout probability.

We were unable to show that the arrival rates at the CrowdManager prototype was a Poisson distributed variable or followed a Non-Homogeneous Poisson process. Our data was biased by the fact that when a HIT is posted on MTurk, it will appear on the front page, thus generating much more visits in the first few minutes. We were not able to remove this bias from our data sample and thus failed to prove that the arrival rate followed a Poisson distribution. We also failed to show that it followed a Non-Homogeneous Poisson process due to not having any data points for most of the night and morning times (GMT +1). We proposed to run new test distributed evenly on the 24 hours of a day in order to get a concrete distribution function for the underlying Non-Homogeneous Poisson process or to record the pagination of the HIT on MTurk in order to remove the bias in the data with the help of the pagination variable.

We were also unable to show that the bid prices and bid sizes had any kind of underlying distribution. We assume that the issue lies in biased data once again, as the user interfaces suggested

exactly the peak points of the bid prices and bid sizes (0.05$ and 10 taks). We proposed run the experiments again with "bias-free" interfaces and to also record other possible sources of bias like the gender, ethnicity or age of the crowd worker.

We were able to show that the completion times and delivered qualities for tasks were, contrary to the assumptions made by Minder et al., not constant over time. For the decreasing completion times for address finding tasks we were able to compute an approximate experience curve of the form $C_n = 89.3 * n^{-0.35}$. We proposed to take the decreasing completions times into consideration by changing equation 3.3 in the allocation mechanism to $\hat{t_i} * x_i^{-0.35} \leq T, \forall i \in I$. We proposed to further investigate the issue of decreasing qualities in order to find possible solutions to this problem.

We had strong signs in our data that crowd workers did not always bid truthfully. This coupled with the low amount of bids per auction (5.25 on average) created a non-competitive environment and made it very difficult for the allocation mechanisms to find successful allocations. Nevertheless, we were able to show that despite this fact the CrowdManager mechanism was able to find a similar amount of successful allocations compared to the two baseline mechanisms. Due to our small sample size we were not able to show any differences in cost effectiveness, although looking at the data indicates a favorable outcome for the CrowdManager mechanism. We were able to show that the utility with retainer costs was unfavorable for the CrowdManager mechanism. This is mainly attributable to the fact that the two baseline mechanisms had some sort of filter before the crowd worker would be accepted to the retainer, whereas the CrowdManager mechanism did not have any filters besides initial qualification test.

Overall, the experiments showed promising results and if the average competitiveness for each auction (i.e. more bids per auction) can be raised, the CrowdManager mechanism might outperform a fix price mechanism and proof to be a viable approach for real world crowdsourcing platforms. Due to the low competitiveness and the possible untruthfulness of crowd workers, a different pricing mechanism might be a better choice, but more data needs to be collected with "bias-free" user interfaces to validate this.

Bibliography

[1] Grier, D.: When computers were human. Volume 316. Princeton University Press (2005)

[2] Cormen, T., Leiserson, C., Rivest, R., Stein, C.: Introduction to algorithms. MIT press (2001)

[3] Howe, J.: The rise of crowdsourcing. Wired magazine **14**(6) (2006) 1–4

[4] Howe, J.: Crowdsourcing: A definition. http://crowdsourcing.typepad.com Online; accessed: 20/10/2012.

[5] Minder, P., Bernstein, A.: Crowdlang - first steps towards programmable human computers for general computation. In: 3rd Human Computation Workshop (HCOMP 2011), San Francisco, CA, USA, AAAI Publications (JAN 2011) 103–108

[6] Bernstein, M., Little, G., Miller, R., Hartmann, B., Ackerman, M., Karger, D., Crowell, D., Panovich, K.: Soylent: a word processor with a crowd inside. In: Proceedings of the 23nd annual ACM symposium on User interface software and technology, ACM (2010) 313–322

[7] Von Ahn, L., Dabbish, L.: Labeling images with a computer game. In: Proceedings of the SIGCHI conference on Human factors in computing systems, ACM (2004) 319–326

[8] Minder, P., Bernstein, A.: How to translate a book within an hour - towards general purpose programmable human computers with crowdlang. In: Web Science 2012, New Yortk, NY, USA (JUN 2012)

[9] Minder, P., Seuken, S., Bernstein, A., Zollinger, M.: Crowdmanager - combinatorial allocation and pricing of crowdsourcing tasks with time constraints. In: Workshop on Social Computing and User Generated Content in conjunction with ACM Conference on Electronic Commerce (ACM-EC 2012), Valencia, Spain (JUN 2012) 1–18

[10] Edelman, B., Ostrovsky, M., Schwarz, M.: Internet advertising and the generalized second price auction: Selling billions of dollars worth of keywords. Technical report, National Bureau of Economic Research (2005)

[11] Quinn, A., Bederson, B.: Human computation: a survey and taxonomy of a growing field. In: Proceedings of the 2011 annual conference on Human factors in computing systems, ACM (2011) 1403–1412

[12] Malone, T., Laubacher, R., Dellarocas, C.: The collective intelligence genome. IEEE Engineering Management Review **38**(3) (2010) 38

[13] Law, E., Ahn, L.: Human computation. Synthesis Lectures on Artificial Intelligence and Machine Learning **5**(3) (2011) 1–121

[14] Von Ahn, L.: Human computation. In: Design Automation Conference, 2009. DAC'09. 46th ACM/IEEE, IEEE (2009) 418–419

[15] Dryer, D., Eisbach, C., Ark, W.: At what cost pervasive? a social computing view of mobile computing systems. IBM Systems Journal **38**(4) (1999) 652–676

[16] Wolfers, J., Zitzewitz, E.: Prediction markets. Technical report, National Bureau of Economic Research (2004)

[17] Bigham, J., Jayant, C., Ji, H., Little, G., Miller, A., Miller, R., Miller, R., Tatarowicz, A., White, B., White, S., et al.: Vizwiz: nearly real-time answers to visual questions. In: Proceedings of the 23nd annual ACM symposium on User interface software and technology, ACM (2010) 333–342

[18] Mason, W., Suri, S.: Conducting behavioral research on amazon's mechanical turk. Behavior Research Methods **44**(1) (2012) 1–23

[19] Bernstein, M., Brandt, J., Miller, R., Karger, D.: Crowds in two seconds: Enabling realtime crowd-powered interfaces. In: Proceedings of the 24th annual ACM symposium on User interface software and technology, ACM (2011) 33–42

[20] Huang, E., Zhang, H., Parkes, D., Gajos, K., Chen, Y.: Toward automatic task design: A progress report. In: Proceedings of the ACM SIGKDD workshop on human computation, ACM (2010) 77–85

[21] Wang, J., Faridani, S., Ipeirotis, P.: Estimating the completion time of crowdsourced tasks using survival analysis models. Crowdsourcing for search and data mining (CSDM 2011) **31** (2011)

[22] Fehr, E., Goette, L.: Do workers work more if wages are high? evidence from a randomized field experiment. The American Economic Review (2007) 298–317

[23] Mason, W., Watts, D.: Financial incentives and the performance of crowds. In: Proceedings of the ACM SIGKDD workshop on human computation, ACM (2009) 77–85

[24] Singer, Y., Mittal, M.: Pricing mechanisms for online labor markets. In: Proc. AAAI Human Computation Workshop (HCOMP). (2011)

[25] Chawla, S., Hartline, J., Sivan, B.: Optimal crowdsourcing contests. In: Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SIAM (2012) 856–868

[26] Horton, J., Zeckhauser, R.: Algorithmic wage negotiations: Applications to paid crowdsourcing. In: Crowd Conf. (2010)

[27] Horton, J., Chilton, L.: The labor economics of paid crowdsourcing. In: Proceedings of the 11th ACM conference on Electronic commerce, ACM (2010) 209–218

[28] Nisan, N.: Introduction to mechanism design (for computer scientists). In Nisan, N., Roughgarden, T., Tardos, E., Vazirani, V., eds.: Algorithmic game theory. Cambridge University Press (2007) 209–241

[29] Diedrich, O.: Trendstudie open source: Wie open-source-software in deutschland eingesetzt wird. http://www.heise.de/open/artikel/Trendstudie-Open-Source-221696.html Online; accessed: 20/10/2012.

[30] Krasner, G., Pope, S., et al.: A description of the model-view-controller user interface paradigm in the smalltalk-80 system. Journal of object oriented programming **1**(3) (1988) 26–49

[31] Day, G., Montgomery, D.: Diagnosing the experience curve. The Journal of Marketing (1983) 44–58

[32] Milgrom, P., Weber, R.: A theory of auctions and competitive bidding. Econometrica: Journal of the Econometric Society (1982) 1089–1122

[33] Oleson, D., Sorokin, A., Laughlin, G., Hester, V., Le, J., Biewald, L.: Programmatic gold: Targeted and scalable quality assurance in crowdsourcing. Proc. HComp (2011)

[34] levenshtein, V.I.: Binary codes capable of correcting deletions, insertions, and reversals. In: Soviet Physics-Doklady. Volume 10. (1966)

[35] Edwards, A.: An introduction to linear regression and correlation. WH Freeman (1976)

[36] Kowalski, C.: On the effects of non-normality on the distribution of the sample product-moment correlation coefficient. Applied Statistics (1972) 1–12

[37] Stevens, S., et al.: On the theory of scales of measurement (1946)

[38] Cramér, H.: Mathematical methods of statistics. Princeton University Press, Princeton (1946)

[39] Massey Jr, F.: The kolmogorov-smirnov test for goodness of fit. Journal of the American statistical Association **46**(253) (1951) 68–78

[40] Faridani, S., Hartmann, B., Ipeirotis, P.: What's the right price? pricing tasks for finishing on time. In: Proc. of AAAI Workshop on Human Computation. (2011)

[41] Field, A.: Discovering statistics using SPSS. Sage Publications Limited (2009)