CS908 Assignment 4 - Dissertation specification

Thomas Archbold
1602581
Department of Computer Science
University of Warwick

February 20, 2020

1 Introduction

Prediction markets are exchange-traded markets[1] that trade on the outcomes of events rather than traditional financial instruments. Since actors in the market participate by putting up their own money in the form of betting on an unknown future outcome, the market prices can indicate the beliefs held by the market. What can make these markets more interesting is the ability to combine bets in complex ways, giving users the freedom to make predictions over a range of different unknown outcomes and have them affect the odds in interrelated ways. This is the approach taken by the combinatorial prediction market *Predictalot* [8], which provides a platform on which to bet on over 9.2 quintillion outcomes in the NCAA Men's College Basketball playoffs. This specification outlines the plans to implement a similar combinatorial prediction market for betting on events in the 2020 United States presidential election. The focus is on implementing efficient ways for agents to interact with the market as well as methods to combine collections of bets.

The rest of the specification is structured as follows. In Section 2 we outline in greater detail the motivation for the project and present our goals for the platform. In Section 3 we discuss the underlying theory of the model, including relevant recent results in mechanism design for combinatorial auctions and prediction markets. We also discuss current software that implements other prediction markets and how this project will fit in with the current body of work. In Section 4 we outline the approaches we plan to take in order to complete this project, including a brief discussion of the tools and data that we expect to use. Project management is discussed in Section 5. We conclude with a brief discussion of progress made thus far in Section 6, followed by closing thoughts and plans for the next stages in Section 7.

2 Project overview

2.1 Problem Statement

The goal of this project is to produce a combinatorial prediction market for the upcoming 2020 US presidential election and its related events, such as the primary elections and caucuses. Participants within this market will be able to place and trade bets based on what they think is likely to happen, and these bets may be traded among users up until the outcome of the event is realised. Throughout this time, the platform will update the odds accordingly, taking into account the bets that have been made on related events. Winnings will then be paid out to those who "own" a bet.

The project will be implemented as a web application written in Lisp, and in particular the Common Lisp dialect. The platform will allow for different auction mechanisms from the literature to be implemented and explored, with the secondary aim of measuring the impact of the underlying model on the market's functionality and performance.

2.2 Motivation

Prediction markets provide ways in which to bet on the outcomes of events in the future, and are often used to bet in a variety of situations. Since there is an incentive to do well, as players stake their own money, people are inclined to bet how they truly feel and hence learning public sentiment on these events can effectively be crowdsourced. Already this provides an exciting avenue to learn of public knowledge and belief on certain events. Combinatorial prediction markets allow for bets to be bought, sold, and combined in various ways, and we may learn more accurately how events are related. With this project, we seek to implement a combinatorial prediction market for the run up to the 2020 US presidential election, as there there is global importance to the outcomes of these events.

We want to implement a way to allocate collections of bets to competing bidders in a market. From the mechanism design literature it is well-known that allocating a set of goods to buyers that maximises, for example, social welfare or revenue, requires solving an NP-hard optimisation problem [26]. Furthermore, for a market offering bets on m separate events, there are 2^m possible ways of combining such bets – how can we expect users to enter an exponential number of bids before they even get any items? These two problems are the focus of much of the literature within algorithmic mechanism design [25], a field within algorithmic game theory that is concerned with designing the ways in which self-interested agents act within a strategic environment. These mechanisms aim to achieve certain economic properties – truthfulness, budget balance, and individual rationality, for example – while

[1]A market in which all transactions are routed through a central source.

ensuring that they are practical to implement. It hence makes use of techniques found in theoretical computer science, most notably asymptotic analysis, randomisation, and approximation. The goal is therefore to compute a solution that is approximately-optimal, or "good enough", rather than an optimal one that is computationally infeasible. With this in mind we wish to explore the challenges associated with implementing such a complex market.

Even restricting ourselves to approximately-optimal solutions, however, there is the problem of defining "good enough": what makes, say, a 4-approximation mechanism better than a 7-approximation that also achieves group-strategy proofness?[2] The answer is that it depends and that trade offs must be made on a variety of assumptions on the market. These parameters may be tweaked depending on the setting, and each of these changes will lead to mechanisms with different performance. Given that there is no single metric by which we can judge a mechanism's performance, this project also aims to allow for various different mechanisms to be used, to measure any impact they have on the system as a whole. Thus the project can serve as a platform to implement and analyse recent mechanisms in the literature in a practical environment.

We will give a brief justification for the choice of using the Lisp programming language, and in particular the Common Lisp dialect, to implement the project; the reasons are three-fold. Firstly, since the project involves creating a web application we will need to use a language for which there are sufficient libraries to do so: Quicklisp [10] is a library manager for Common Lisp that provides an extensive selection of libraries and is compatible with many Common Lisp implementations. In particular, there are many options available to us for writing a dynamic and responsive web application. The specific libraries we expect to use are detailed in Section 4. Secondly, Lisp is an expressive high-level programming language. Paul Graham argues that the style of "bottom-up" programming that Lisp encourages helps to make code easier to understand and more extensible [21], an important feature of our project. Unique to Lisp is the Read-Evaluate-Print-Loop (REPL), and effective use of this allows for quick and dynamic prototyping and testing. Changes may even be made while a Lisp program is executing, which can streamline the development process. Finally, Lisp nowadays seems a somewhat unconventional choice of language and it would be intriguing to use in a practical setting, even if just to build on my limited exposure to it so far.

2.3 Aims

As stated, the main aim of this project is to create a combinatorial prediction market for betting on outcomes in the forthcoming US presidential election. The underlying exchange will be modelled as a two-sided combinatorial auction (see Section 3 for a more in-depth discussion of this). We detail the features of the project below under the categories of **core**, **additional**, and **stretch** features. The core features will implement all of the functionality to qualify the software as a prediction market, and will cover making and buying basic bets, combining them in simple ways, and calculating somewhat accurate odds based on the trades. It will likely operate on dummy data for testing. Additional features will extend this functionality to explore the effects using different mechanisms from the literature, and we will begin to integrate real-world data; we will also begin to combine bets in more complex ways. Stretch features will focus on extending the types of possible bids, applying the market to other domains, and improving usability.

2.3.1 Core features

The project will have:

- A web application written in Common Lisp on which to host a combinatorial prediction market for the run up to the US presidential election, modelled as a two-sided combinatorial auction
- The ability for multiple users to interact within this market, and specifically:
  - Propose bets on a number of events
  - Buy bundles of bets from other users, transferring the right to the winnings should the outcome be realised
  - Compute approximately-accurate odds based on the trades which take place, and distribute winnings accordingly
- The ability for users to make bets on the outcomes of the following events:

[2]These are just examples to illustrate the point – these mechanisms may or may not exist.

  - the 2020 presidential winner
  - the party that wins the presidency
  - each state's Democratic primary elections and caucuses
  - each state's Republican primary elections and caucuses
  - the Democratic nominee
  - the Republican nominee
- A database for persistent storage of user information
- The web application run on dummy data, so that the market mechanisms may be tested for correctness
- The ability to react asynchronously to user input

2.3.2 Additional features

The project should have:

- Functionality to collect real-world data, moving the application away from simulation and into reality
- Various ways in which users may interact in trades, including the traditional method of selecting a single item, as well as assuming single- and k-minded bidders
- Implementations of various two-sided market mechanisms from the literature, including deferred acceptance auctions
- The provision for more complex types of bets

2.3.3 Stretch features

The project may:

- Be extended to other domains, such as sports tournaments or other political campaigns
- Incorporate additional types of (more complex) bids
- Take additional aesthetic and usability considerations

3 Background

3.1 Combinatorial auctions

In a single-item auction, we have a collection of n agents who each possess a valuation for acquiring a single item on offer. A *mechanism* is any method for allocating this item to the bidders: typically, this involves specifying an *allocation rule* and a *payment rule.* The former may be responsible for collecting information from the participants – often a bid – to compute the allocation, while the latter is used to ensure that agents act truthfully.

The real number v_i denotes agent i's valuation for acquiring the item. This information is private to each agent (referred to as "bidders" or "buyers"), meaning neither the mechanism nor the other agents know this value. Naturally, we wish to allocate this item to the bidder who truly values it the most – that is, allocate it to agent $i^* = \arg\max_{i \in [n]} v_i$.[3] It is a typical for an auction to elicit a bid b_i from each agent so as to acquire their valuation v_i, but since the agents are strategic they may lie (i.e., submit bid $b_i \neq v_i$) if they believe it is in their best interests. The Vickrey mechanism [29] achieves such an optimal allocation by "believing" the bidders, giving the item to the agent with the highest bid, and charging him the next highest bid.

In combinatorial auctions there are m items to allocate. Buyers now have a collection of valuation function, $v_i(S)$ that associate to each subset of items $S \subseteq [m]$ a value for acquiring this subset. However, since the number of subsets of this set is 2^m, we cannot hope to either collect all these valuations from bidders nor use them in sub-exponential time, unless we impose some restrictions. In the literature, one common way to deal with this obstacle is by restricting the bidders to be k-minded – each bidder submits a maximum of k bids for k different subsets, and effectively a bid of zero for every other set. There are

[3] The notation $[k]$ is used to represent the set $\{1, \ldots, k\}$

also subtleties involved if we assume items to be homogeneous, in which case the auction contains m copies of the same item, or heterogeneous, where each item is unique. The idea of this project is to model bets on future outcomes as items, and since we will be combining different bets to learn about these outcomes, we assume the items in the auction to be heterogeneous. Furthermore, since we are keen to explore the practical performance of some of the mechanisms in the literature, we plan to use k-minded bidders in the underlying auction mechanism.

Instead of simply maximising over n numbers as in the single-item setting, i.e. selecting the single bidder with the highest bid, we must maximise over all possible allocations. In this setting, we focus on maximising social welfare, the sum of the agents' valuations. If a mechanism computes allocation $A = (A_1, \ldots, A_n)$, where agent i is allocated subset A_i, then the value we are maximising as a function of this allocation is $W(A) = \sum_{i \in [n]} v_i(A_i)$. The analogue to the Vickrey auction in the multi-item setting, the Vickrey-Clarke-Groves mechanism [29, 13, 22], again believes the bidders, awarding items to agents who value them most and charging each bidder their *externality*, or the extra cost in social welfare incurred by that agent participating in the auction.

Until now we have assumed that the auction itself holds the items, but this will not be the case in our prediction market since the agents will propose and trade the bets. We will therefore use a *two-sided* market. This comprises a set of two distinct types of agents: sellers, who initially hold the items for sale, and buyers, who are interested in buying the items from the sellers. Formally a two-sided market is a tuple (n, m, k, I, G, F) [14], where $[n]$ is the set of buyers, $[m]$ is the set of sellers, $[k]$ is the set of items, and $I = (I_1, \ldots, I_m)$ is the *initial endowment*, such that I_j is the set of items initially held by seller j. Vectors $G = (G_1, \ldots, G_n)$ and $F = (F_1, \ldots, F_m)$ are the distributions from which the buyers' are sellers' valuation functions are assumed to be drawn, respectively. Given a two-sided market, our aim is to redistribute the items among the agents so as to maximise the social welfare. An allocation is a pair of vectors $(X, Y) = ((X_1, \ldots, X_n), (Y_1, \ldots, Y_m))$ such that the union of X and Y is the set of items $[k]$, and $X_1, \ldots, X_n, Y_1, \ldots, Y_m$ are mutually disjoint, meaning no two agents are allocated the same item. As before, buyers have a valuation $v_i(S)$ for each subset of items S; we have a similar concept for the sellers, denoted $w_j(S)$. The goal is still to maximise the social welfare, which is:

$$W(X, Y) = \sum_{i \in [n]} v_i(X_i) + \sum_{j \in [m]} w_j(Y_j)$$

Some of the sought-after economic properties in mechanism design are as follows:

- **Incentive Compatibility (IC):** agents are incentivised to bid truthfully (i.e. submitting $b_i = v_i$) as they can do no better by lying – truth-telling is a *dominant strategy.*
- **Individual Rationality (IR):** it is not harmful for any agent to participate in the market, meaning in any trade there is a strategy that yields a utility that is no less than their initial utility. Note that this says nothing about the outcome of the event – agents may still experience a net loss for having purchased a bet which turned out to be false.
- **Budget Balance (BB):** the sum of all payments is at least zero, meaning no extra funds have to be supplied to subsidise the market.

3.2 Related work - combinatorial auctions

Algorithmic mechanism design enjoys much attention in the setting of combinatorial auctions, where there is ample opportunity to make use of techniques from theoretical computer science. Much of this is inspired by a question posed shortly after the field's inception by Nisan and Ronen [25], which asks whether the requirement for dominant strategy incentive compatibility inherently degrades a mechanism's approximation ratio – put another way, it asks if mechanism design is inherently harder than algorithm design. This question is answered in part by Daniely, Schapira, and Shahaf [15], who present various inapproximability results for deterministic, truthful mechanisms.

To sidestep the obstacles faced by deterministic mechanisms, Colini-Baldeschi et al. [14] make use of randomisation to present three constant-factor approximation mechanisms for two-sided combinatorial markets, under various assumptions about agent valuations. In particular, they provide three 6-approximation mechanisms which achieve individual rationality, incentive compatibility, and a stronger notion of budget balance, termed Direct Trade Strong Budget Balance (DSBB). Two-sided combinatorial auctions are also considered for single-minded bidders by Dütting, Gkatzelis, and Roughgarden [19], in which they study Deferred Acceptance (DA) auctions. These auctions compute an allocation by repeatedly rejecting the least promising candidate in a greedy manner, in this case the lowest bid. DA auctions have preferable incentive guarantees over simple VCG or greedy mechanisms in that they are (weakly) group strategy proof, meaning there is no way for a coalition to act such that each member of the coalition is better off. They introduce a polynomial-time mechanism that achieves the incentive-compatibility of a DA auction and a $O(\log m)$-approximation. In a similar vein to Daniely et al. [15], they prove inapproximability results for DA auctions in the setting of single-minded bidders; specifically, that they cannot achieve a better approximation ratio than $o(m)$ if the way in which the mechanism ranks bids is of a particular form.

Dughmi, Roughgarden, and Yan [18] use randomisation to provide a mechanism that is truthful in expectation and achieves a constant factor approximation of $O(1 - 1/e)$ for welfare maximisation with heterogeneous items. They show that this is a polynomial-time mechanism in expectation, further proving that the approximation ratio is the best possible assuming P $\neq$ NP. Dobzinski and Nisan [17] make use of Maximal-In-Range and Maximal-In-Distributional-Range [16] mechanisms to introduce several DSIC approximation mechanisms for computing multi-unit auctions in polynomial time. These are mechanisms which search over a pre-specified range for a maximal solution. They present a 2-approximation for general valuation functions, and a Polynomial Time Approximation Scheme (PTAS) for bidders with k-minded valuations. Although this assumes there are multiple copies of items, and hence likely unsuitable to our setting, it would be interesting to explore the use of such a framework.

3.3 Related work - prediction markets

In the literature, prediction markets have been used in a variety of ways to predict the outcomes of a range of events, such as in business and politics [28, 12]. Their power is shown by Nay, Van der Linden, and Gilligan [24] who analyse prediction markets empirically to study the how the market parameters may affect the agents' beliefs on the real-world outcome. They make the argument for a prediction market for betting on the cause of global warming, with two events: either climate change is caused through CO_2 emissions, or as a result of variations in solar intensity. They show that increased agent participation causes public feeling to shift towards the "true" climate change model, they suggesting that a prediction market in this domain would be useful for getting people to agree on its (correct) cause.

In order for these prediction markets to function, they require a central trusted entity to verify the true outcome of the event. Freeman, Lahaie, and Pennock [20] identify this as a bottleneck of what can be predicted, motivating their study of decentralised prediction markets. Outcomes are determined by a group of arbiters who may themselves participate in the market. Given the greater incentive these agents may have to sabotage the mechanism, they derive mechanisms under which they are incentivised to act truthfully. Peterson et al. [27] also study this setting in their paper and use it to implement the decentralised oracle at the heart of their prediction market platform *Augur* [1].

In the centralised setting, Kroer, Dudík, Lahaie, and Balakrishnan [23] present a combinatorial prediction market that makes use of a cost-based market maker, in which all bets are bought from and sold to the market maker, rather than the traders, which they argue can be desirable in combinatorial auctions due to the possibility of low liquidity. This is the case in which there are no sellers interested in trading with a buyer; in contrast, a market maker is always available to trade. Our platform will consider a smaller space of events compared to the NCAA Men's College Basketball playoffs however, hence our decision to use a more natural two-sided market to model the interactions between agents.

3.4 Existing systems

There are several examples of recent implementations of prediction markets in various settings. This project's conception was initially inspired by *Predictalot* [8], a combinatorial prediction market for betting on the outcomes of a wide range of events in the NCAA Men's College basketball playoffs. It launched as a Yahoo! app in 2010, though it appears to have since been abandoned. Nonetheless, it boasts the ability to bet on "almost anything you can think of, like *Duke will advance further than UNC*". Specifically, the tournament is structured such that the top 64 college basketball teams play 63 games in a single elimination tournament, while *Predictalot* keeps track of the odds, computing them by scanning through all the predictions placed so far. They note that an exact computation of the odds is #P-hard – as hard as counting the number of variable assignments that satisfy a given CNF formula, or how many subsets of a list of integers sum to zero – hence justifying their decision to approximate the odds by random sampling of the outcome space. They also emphasise that they keep track of all different possible outcomes, resulting in odds that are interconnected in natural ways – betting on a team to make the final will have an effect on the odds of each stage, since the team must win at each of these stages to reach the final in the first place. Given the apparent power behind this idea, it would be interesting to apply it to a setting with perhaps more significance, such as the upcoming presidential elections.

As discussed, *Augur* is more recent example, launching in 2018 on the Ethereum blockchain. Similarly to how this project will be structured, *Augur* hosts the prediction market in two phases: the market stage, where users are able to trade shares between each other, and the arbitration stage, where the market's outcome is determined. However, *Augur* makes use of fees to subsidise the arbitration process, breaking the budget balance property discussed earlier. This makes it less appealing, as money is required to be fed into the system in order for it to function.

Both *PredictIt* [9] and the Iowa Electronic Markets (IEM) [5] provide the ability to make bets on various political events, including on outcomes within the 2020 US presidential election. However, like *Augur*, they do not provide the ability to combine these bets in complex ways and hence this is a gap that this project aims to fill.

This project will differ from these mainly in the fact that we will implement a combinatorial prediction market, with a focus on making bets on a combination of outcomes. The only similar existing software implementing this is *Predictalot*, which is no longer accessible. Betting on collections of events better incorporates conditional information into the market price, giving a more realistic view of how events are related. The project will also differ in the means by which users will interact with the markets. Specifically, on each of the existing systems discussed, the user trades bets on an individual basis. This project will model these interactions more akin to sealed-bid auctions, with the intention that bidders will place and buy bets that align more closely with their true beliefs and valuations. Placing such additional restrictions these interactions with the market allows for stronger performance guarantees, and results from the current literature can be implemented more easily. Of course, it will be an important design decision to trade off between these guarantees while ensuring the platform feels natural to use.

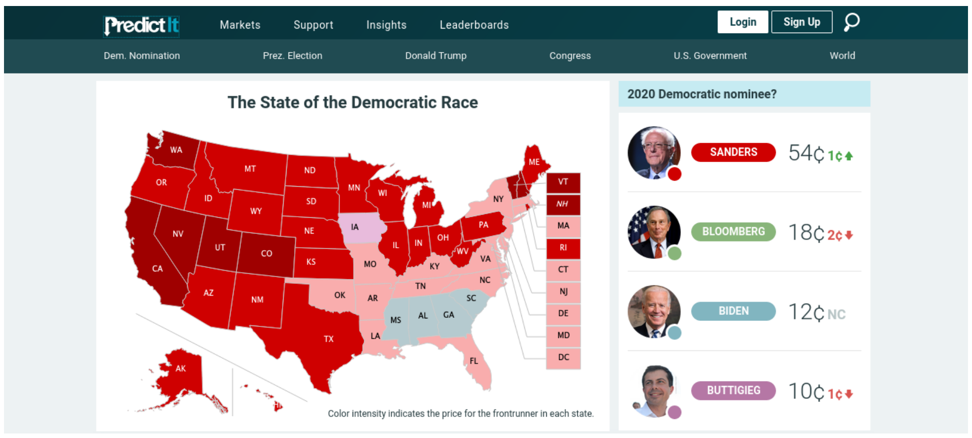


Figure 1: The *PredictIt* prediction market for the 2020 Democratic nominee

4 Research Processes

4.1 Approach

The approach to the project will be through software development, as we are looking to implement mechanisms from the literature in a different way than has been done previously. However, the project requires being familiar with recent results in two main areas, combinatorial auctions and prediction markets, and so wider reading will make up a significant portion of the work. Although much reading has been done in algorithmic mechanism design for combinatorial auctions, more will be done over the course of the project's completion to stay abreast of recent results and additional mechanisms to implement. The body of research on two-sided markets is relatively extensive, and hence there should be no shortage of material to cover. More research needs to be done, however, on prediction markets in general in order to better understand how they function. Given that *Augur* is open-source it is likely to prove a key source of information about implementing such markets – more qualitative observations on how prediction markets work will also be gathered by getting familiar with the various existing systems out there whose code is not open-source. The e-Print archive *arXiv.org* will be a key tool for further exploring recent developments in prediction markets.

4.2 Tools

As mentioned we will be implementing the project in Common Lisp. We plan to use the following libraries: CL-WHO [2], for converting Lisp expressions into HTML; Hunchentoot [4], a web server written in Common Lisp and a toolkit for hosting dynamic websites; Parenscript [7], which will allow us to compile Lisp expressions into JavaScript, for client-side validation; and SmackJack [11], a small AJAX framework for Common Lisp which allows us to create asynchronous functionality in our web application. The REPL is a key tool available to Lisp, and we plan to make full use of it to quickly and dynamically implement changes. LaTeXwill be used for all written documentation, and most code is likely to be written in Vim.

We plan to make extensive use of Git [3] by way of version control software – previous experience dictates this as an essential aspect to our development pipeline. The project will be hosted on Github, meaning development can take place essentially from anywhere (and at least on computers with a Common Lisp implementation), and all the necessary files are stored where they are guaranteed not to be lost. Github also allows for the progress to be more easily tracked (an example is given in Fig 2). Progress can more easily be checked against the project timetable in this way, not only by viewing how many commits are being made, but also which features are being added each time. This will simplify the evaluation process for the interim report and final write-up, as one can see exactly what features are added and when.

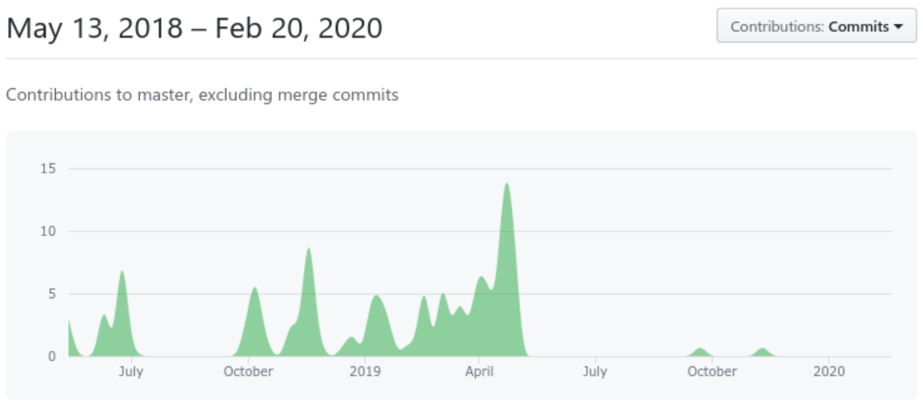


Figure 2: Visualising commits in Github

To allow for multiple users to interact within the market the web server will eventually need to be hosted somewhere, as opposed to simply being run locally. A possible (free) tool for hosting the server is therefore Ngrok [6], which provides the ability expose a local web server for public use. This will suffice for the small scale scope of this project, and can easily be hosted elsewhere should the need arise.

5 Project Management

5.1 Overview

Regarding the development of the software, we plan to mix between a plan-driven and agile approach. Initially there is a set order in which features are to be added, such as setting up the development environment, getting the web server up and running, and designing the systems that will store user information and allow for interactions between agents. Once the basics of the prediction market are implemented, the areas on which to spend development may open up, and hence focus may shift to an agile approach. Specifically, decisions will need to be made at the time on which features are most in need of attention. An initial outline of the timeline for the project is provided in Appendix A.

On the administrative side, weekly meetings with project supervisor Dr Matthias Englert will be held in order to track and discuss progress regularly. Moreover, they provide the opportunity to bring up issues as they occur so that solutions may be discussed and attempted quickly. Meetings will take place on Tuesdays between 5pm and 6pm.

5.2 Constraints

Since the project will mostly be developed and tested locally, there are few external constraints that could hinder progress. Once the project begins to take fuller shape, it will be required to be tested using multiple (non-simulated) users, and there may be additional unforeseen difficulties in providing for this. Testing in this phase is likely to be done with informal quality assurance testing with colleagues. A risk associated with this is that they may be busy (perhaps testing their own dissertations), but this is unavoidable.

There is little progress likely to be made during the exam period and the weeks leading into it as most attention will be towards revising. This is an acceptable constraint given the vacuum of activity over the summer that follows, which provides ample time to compensate for this deficit.

One potential constraint on the project's viability is the time period over which we plan to implement it. Since the US election is being held on 3rd November, the events in the run up to it will happen much earlier than this, and it could be that we miss the window to have certain bets and systems implemented before they happen "for real". This will need to be kept in mind, as although nothing too significant needs to be made specific to the election, aside from the bids themselves, it does provide an interesting area to apply a prediction market.

5.3 Ethical concerns

All development and initial testing will be done independently, hence there is no reliance on using external data from the outset. Once we integrate real-world data into the platform, all data we gather regarding the US election will all be public knowledge, meaning there is no ethical concern here. Since testing will be performed informally with the help of colleagues, there are no legal or professional issues to worry about in this regard, either.

5.4 Project Timeline

The rest of the term will be spent getting up to speed with the current literature surrounding prediction markets, in order to see how they have been implemented by others and what is considered state-of-the-art. Given the course load towards the end of this term from other modules, relatively less focus will be given towards this project. However, regular work will be done in setting up the necessary tools and environments so that the foundations for the prediction market itself to be implemented will be in place by the end of the term. This involves becoming better acquainted with Common Lisp; setting up the web server using Hunchentoot, and designing the database to store all the relevant data. Furthermore, it will involve implementing a basic two-sided combinatorial auction mechanism that acts on dummy data, again to build foundations for the work ahead.

During the second half of the Easter break, most time will be dedicated to revising for exams, and this will last until the end of exams around mid-June. Work will still be done during this period, and should be simple to implement given that all necessary foundations have been put in place by this time. In particular, during this time we will be implementing the necessary computations for displaying accurate odds for bets, based on the trades that have taken place so far. Initially these odds will be calculated in a simple manner, that is, not in a way analogous to *Predictalot*'s method. We will then write the system to pay winnings out to the appropriate agents.

After the exam period we will work on refining the computation of the odds, so that a more accurate idea of public sentiment may be achieved by combining the bets made in complex ways. During this point we will also begin to experiment with implementing more advanced mechanisms from the literature, including a deferred acceptance auction. Since the project presentation will be delivered around this period, work will begin roughly a month prior to provide ample time to prepare.

Throughout the summer we will begin to gather real-world data for the market and begin testing the application. Much of this time will be spent writing up the final report; we aim to finish a draft of this well in advance of the 10th September deadline, to give time to gather and implement feedback. The main bulk of the software will need to be done by the beginning of August, and the first draft on the report done by mid-August.

6 Progress

6.1 Current progress

Some progress has been made thus far with regards to implementing the project. The development environment is currently being set up, which includes downloading the aforementioned packages from using Quicklisp and ensuring they run correctly. In particular, it is now possible to run an instance of the server via Hunchentoot with several basic interactions with the page using Parenscript. Moreover, we have begun to define macros that implement several of the basic pages required using CL-WHO.

6.2 Next steps

Next steps involve fleshing out the skeleton code written so far and becoming familiar with the Common Lisp environment. Furthermore, the SmackJack library will require familiarisation in order to achieve asynchronous communication with the server. A database will also need to be implemented in order to store user information. After the completion of these tasks, we will begin to implement a basic two-sided combinatorial auction in order to achieve the core functionality of the market. Initially, this will only operate on dummy data so that we may perform targeted testing throughout the early stages of its development – integration with real-world data will come much later.

On the theory side of the project, more research needs to be undertaken on the state of the literature surrounding prediction markets. This will be the main focus over the coming weeks, to ensure that there is a clear understanding of how they operate. Attempting to write one before such an understanding may lead to delays in the project, and is hence the task with the most immediate importance.

7 Conclusion

In this specification we have detailed our plans to create a combinatorial prediction market for the 2020 US presidential election and its associated events, including primaries and caucuses, election of the nominees, and the presidential election itself. Although numerous prediction markets exist, this project aims to implement a *combinatorial* prediction market. This involves letting users bet on combinations of events and combining these bets in complex ways to compute accurate odds. This will allow us to see how various events are interconnected and learn about the public's belief on these outcomes. To the best of our knowledge, the only software that implements a similar idea is the Yahoo! app *Predictalot*, which is now defunct. Whereas this platform allowed for betting on college basketball, this project will allow for analysis of public sentiment of the US election.

References

[1] *Augur: the world's most accessible, no-limit betting platform.* `https://www.augur.net`. Accessed: 2020-02-16.

[2] *CL-WHO - Yet another Lisp markup language.* `https://edicl.github.io/cl-who`. Accessed: 2020-02-12.

[3] *git –everything-is-local.* `https://git-scm.com`. Accessed: 2020-01-09.

[4] *Hunchentoot - The Common Lisp web server formerly known as TBNL.* `https://edicl.github.io/hunchentoot`. Accessed: 2020-02-12.

[5] *IEM: Iowa Electronic Markets.* `https://iemweb.biz.uiowa.edu`. Accessed: 2020-02-16.

[6] *Ngrok.* `https://ngrok.com`. Accessed: 2020-02-09.

[7] *Parenscript.* `https://common-lisp.net/project/parenscript`. Accessed: 2020-02-12.

[8] *Predictalot! (and we mean a lot).* `http://blog.oddhead.com/2010/03/05/predictalot/`. Accessed: 2020-02-16.

[9] *Predictit.* `https://www.predictit.org`. Accessed: 2020-02-16.

[10] *Quicklisp beta.* `https://www.quicklisp.org/beta`. Accessed: 2020-02-12.

[11] *The smackjack Reference Manual.* `https://quickref.common-lisp.net/smackjack.html`. Accessed: 2020-02-12.

[12] J. Berg and T. Rietz, *The iowa electronic markets: Stylized facts and open issues*, AEI- Brookings Joint Center for Regulatory Studies, (2006), pp. 142–169.

[13] E. H. Clarke, *Multipart pricing of public goods*, Public Choice, 11 (1971), pp. 17–33.

[14] R. Colini-Baldeschi, P. W. Goldberg, B. de Keijzer, S. Leonardi, T. Roughgarden, and S. Turchetta, *Approximately Efficient Two-Sided Combinatorial Auctions*, in Proceedings of the 2017 ACM Conference on Economics and Computation - EC '17, Cambridge, Massachusetts, USA, 2017, ACM Press, pp. 591–608.

[15] A. Daniely, M. Schapira, and G. Shahaf, *Inapproximability of Truthful Mechanisms via Generalizations of the Vapnik–Chervonenkis Dimension*, SIAM Journal on Computing, 47 (2018), pp. 96–120.

[16] S. Dobzinski and N. Nisan, *Limitations of VCG-Based Mechanisms*, Proceedings of the thirty-ninth annual ACM symposium on Theory of computing (STOC '07), (2007), pp. 338–344.

[17] ——, *Mechanisms for Multi-Unit Auctions*, Journal of Artificial Intelligence Research, 37 (2010), p. 14.

[18] S. Dughmi, T. Roughgarden, and Q. Yan, *Optimal Mechanisms for Combinatorial Auctions and Combinatorial Public Projects via Convex Rounding*, Journal of the ACM, 63 (2016), pp. 1–33.

[19] P. Dütting, V. Gkatzelis, and T. Roughgarden, *The Performance of Deferred-Acceptance Auctions*, Mathematics of Operations Research, (2016), p. 24.

[20] R. Freeman, S. Lahaie, and D. M. Pennock, *Crowdsourced outcome determination in prediction markets*, in Thirty-First AAAI Conference on Artificial Intelligence, 2017.

[21] P. Graham, *On LISP: Advanced Techniques for Common LISP*, Prentice-Hall, Inc., USA, 1993.

[22] T. Groves, *Incentives in Teams*, Econometrica, 41 (1973), p. 617.

[23] C. Kroer, M. Dudík, S. Lahaie, and S. Balakrishnan, *Arbitrage-free combinatorial market making via integer programming*, in Proceedings of the 2016 ACM Conference on Economics and Computation, 2016, pp. 161–178.

[24] J. J. Nay, M. Van der Linden, and J. M. Gilligan, *Betting and belief: prediction markets and attribution of climate change*, in 2016 Winter Simulation Conference (WSC), IEEE, 2016, pp. 1666–1677.

[25] N. Nisan and A. Ronen, *Algorithmic Mechanism Design (Extended Abstract)*, in Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing, STOC '99, New York, NY, USA, 1999, Association for Computing Machinery, p. 129–140.

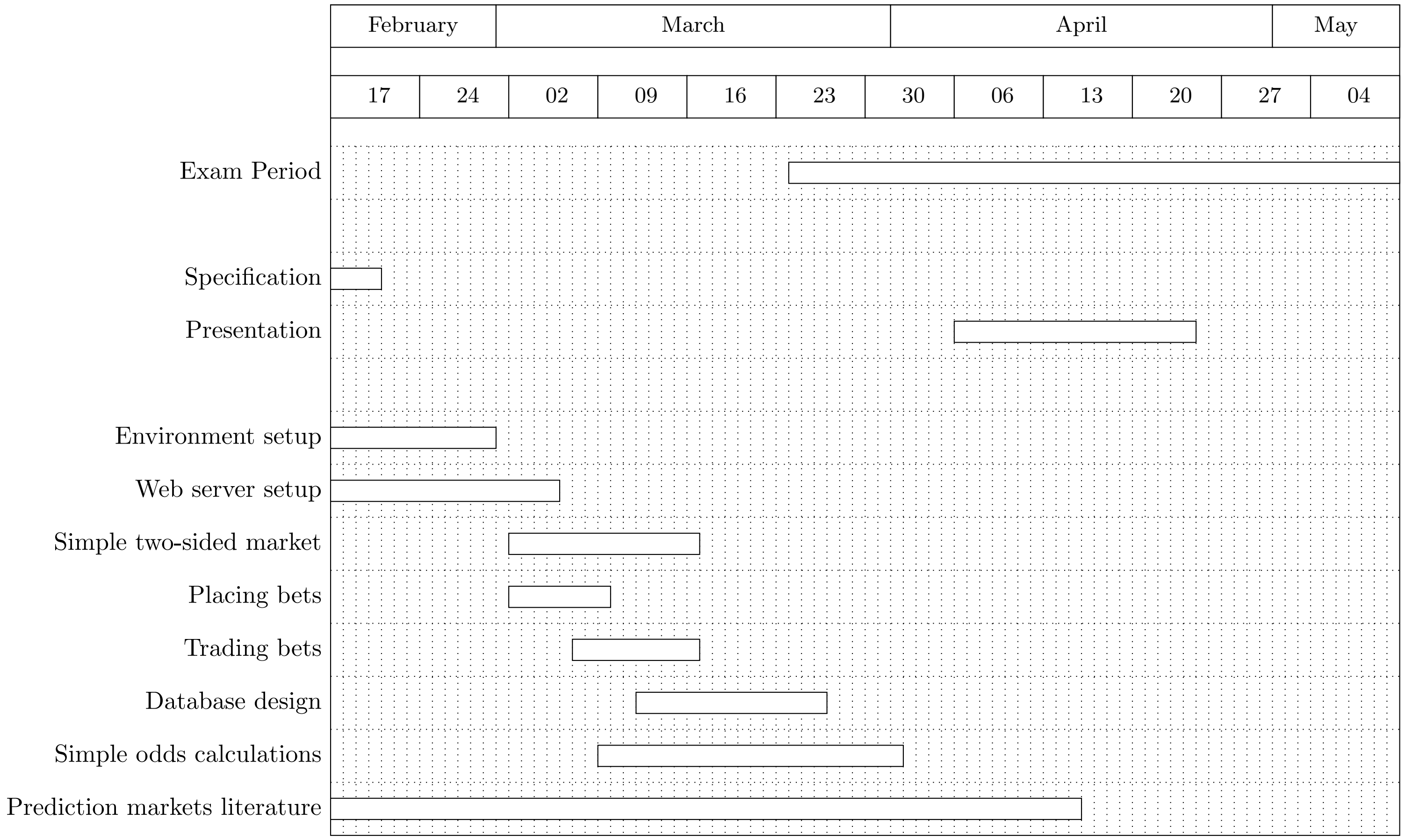
[26] ——, *Computationally Feasible VCG Mechanisms*, J. Artif. Int. Res., 29 (2007), p. 19–47.

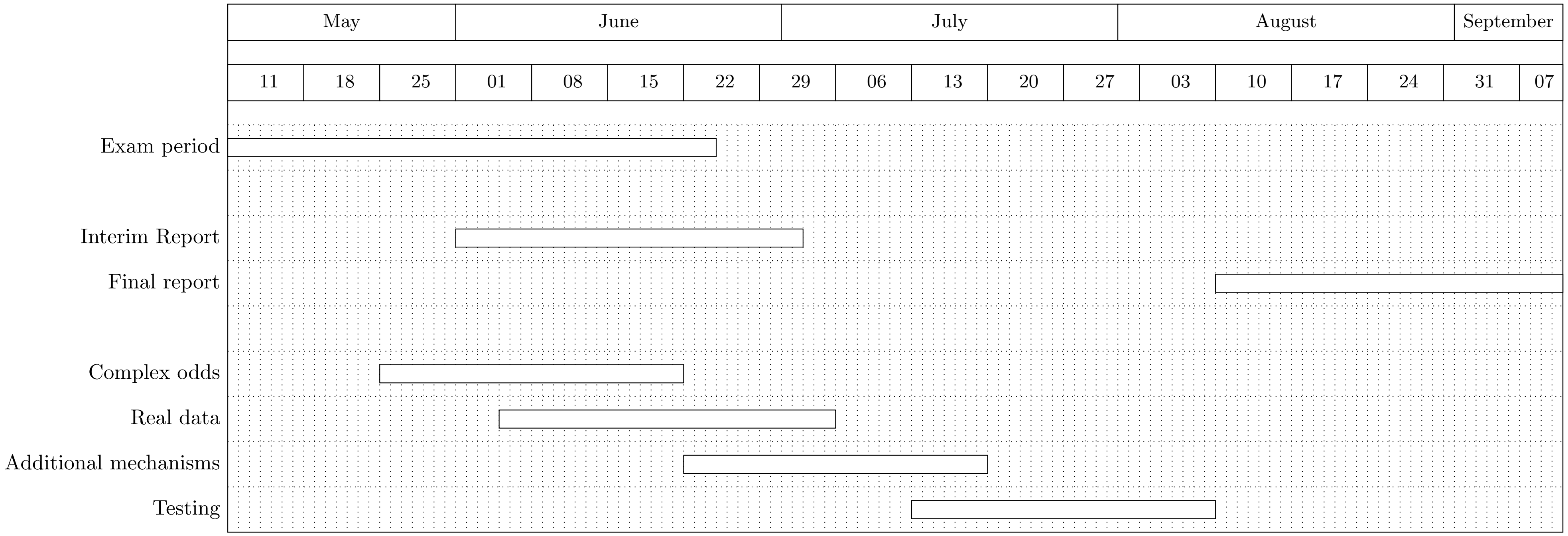
[27] J. Peterson, J. Krug, M. Zoltu, A. K. Williams, and S. Alexander, *Augur: a decentralized oracle and prediction market platform*, arXiv preprint arXiv:1501.01042, (2015).

[28] B. Skiera and M. Spann, *Internet-based virtual stock markets for business forecasting*, Management Science, 49 (2003), pp. 1310–1326.

[29] W. Vickrey, *Counterspeculation, Auctions, and Competitive Sealed Tenders*, The Journal of Finance, 16 (1961), pp. 8–37.

Appendix A Timetable




May
June
July
August
September
11
18
25
01
08
15
22
29
06
13
20
27
03
10
17
24
31
07
Exam period
Interim Report
Final report
Complex odds
Real data
Additional mechanisms
Testing