Computation of Market Equilibria: Algorithm for Experiment-as-Market Mechanism [*]

Devansh Tandon [†]

Advised by: Yusuke Narita

Abstract

Randomized Controlled Trials (RCT) enroll hundreds of millions of subjects and randomize treatment with a large impact on human lives. *Experiment-as-Market (EXaM)* is an alternative design for RCT that respects subjects' welfare by incorporating information on predicted treatment effects and willingness to pay. This paper presents an algorithm developed to implement the EXaM mechanism by computing a market equilibrium, solving a problem similar to Fisher's general equilibrium model. The algorithm efficiently computes a market equilibrium using iterative random local search to find a market clearing price that maximizes subject utility. To evaluate the algorithm and EXaM, I use experimental data from a RCT to find an alternate treatment assignment matching (under EXaM), which leads to an increase in predicted subject welfare while replicating the treatment effects. The algorithm correctly and efficiently computes a market equilibrium, empirically verifies properties of EXaM, and provides an implementation viable for practical use.

[*]I am sincerely grateful to Professor Yusuke Narita for his thoughtful advice, encouragement and patience throughout this project. I also thank the other research assistants who worked on the larger EXaM project, especially Vincent Tanutama and Shreyas Ravishankar.

[†]B.S. Computer Science, Economics; Yale University, 2018. Email: `devansh.tandon@yale.edu`

Contents

1 Introduction **4**

2 Motivation **5**

2.1 Experiment-as-Market 5

2.2 Computation of Market Equilibrium 7

3 Experiment-as-Market (EXaM) Mechanism **8**

3.1 Setting 8

3.2 Experiment as a Market 9

3.3 Randomized Controlled Welfare Property 10

3.4 EXaM and General Equilibrium 11

4 Algorithm **13**

4.1 Setup and Useful Subroutines 14

4.2 Price Search 16

4.3 Alternative Approaches Considered 18

5 Results **21**

5.1 Example Problem 21

5.2 Simulated Problem 22

5.3 Empirical Application: EXaM vs. RCT 24

5.4 Welfare 27

5.5 Information 30

6 Literature Review **32**

7 Conclusion **35**

Bibliography **37**

A Python Implementation of Algorithm **39**

1 Introduction

The Randomized Control Trial (RCT) is an essential economic tool used to establish the causal effect of an intervention. RCTs are considered to be the gold standard of causal inference and evidence-backed policy making (Athey and Imbens (2017)). As RCTs continue to grow in importance, they are increasingly determining the fate of many people for high-stakes outcomes, such as treatment in critical medical trials for new cancer drugs, basic income and health insurance policies, cash transfer programs, and HIV testing. This has led to many calls for alternatives and improvements to the RCT mechanism to include some measure of subject well-being in the treatment assignment process.

In order to incorporate subject welfare into the RCT mechanism, Professor Yusuke Narita has proposed a new experimental design called Experiment-as-Market (EXaM) (Narita (2017)). EXaM would optimally assign each treatment to subjects with good predicted treatment effects or high willingness-to-pay for the treatment, thus respecting subject welfare. This Experiment-as-Market uses a perfectly competitive market behind the scenes to match treatments to subjects, which gives this mechanism superior welfare and incentive properties. In order to effectively match treatments to subjects using the EXaM mechanism, one must find the market equilibrium for the competitive market simulated behind the scenes by solving for a market clearing price that maximizes each subject's individual utility.

For the EXaM mechanism to be applied in practice, it is crucial to be able to compute the correct assignment of subjects to treatment arms. A critical component of computing this assignment is computing the market equilibrium where the market clearing price will give every subject the utility maximizing probability distribution over the treatment arms. In this paper, I will describe the algorithm I developed to compute a market equilibrium, which has been applied in simulations of the EXaM mechanism, and makes the theoretical mechanism viable for practical use by researchers. The algorithm finds a market clearing equilibrium by conducting an iterated price search using local random search.

The paper is structured as follows: Section 2 provides an overview of the motivation for the project, followed by section 3 describing the details of the EXaM mechanism and its properties. Section 4 describes the algorithm developed, provides a psuedocode implementation and discusses alternatives considered. Section 5 presents results: first, from simulated problems to get some insight into how the algorithm works, followed by an empirical application using data from a RCT by Kremer et al. (2011) where the algorithm is used to compare EXaM against RCT and verify properties of EXaM. Section 6 reviews relevant literature from economics and computer science, and section 7 concludes.

2 Motivation

2.1 Experiment-as-Market

The Experiment-as-Market mechanism design places an emphasis on subject welfare. To understand why a subject's well being is important, I will discuss some normative and some practical considerations.

As RCTs grow in size and importance, they are beginning to affect a large number of subjects. For example, let us consider clinical trials that randomize high-stakes treatments through RCTs. Based on WHO data, over 360 million patients were involved in clinical trials in the 2007 to 2017 [1]. There are numerous examples of RCTs with large treatment effects in situations with sensitive and potentially life-threatening outcomes. For example, in a clinical trial studying the prevention of HIV infection with antiretroviral therapy, Cohen et al. (2011) found a drastic relative reduction of 96% in the number of HIV transmissions in the treatment arm. In another example from development economics, Haushofer and Shapiro (2016) ran a RCT in Kenya that led to a 22% increase in household monthly consumption, 9 months after receiving an unconditional cash transfer as treatment. Given the tremendous impact such interventions have on the lives of participants, it is critical to find ways to incorporate subject welfare into the design of RCTs.

From a practical perspective, a successful RCT requires successful subject cooperation on a number of fronts: subjects must choose to participate, use the assigned treatment correctly, and stay in contact during the monitoring periods. A RCT will produce credible information only if subjects are active and willing participants in the process. Unfortunately, many RCTs are lacking on this front and suffer from numerous related problems: subject non-participation, non-compliance, and dropouts before, during and after experiments. A design that incorporates subject welfare can be expected to tackle some of these challenges. Chan and Hamilton (2006) analyzed dropout behaviour in an AIDS clinical trial and found that subjects experiencing better treatment effects are less likely to drop out. Thus, treatment assignments

[1] Calculated using WHO International Clinical Trials Registry Platform (ICTRP) data from `http://www.who.int/ictrp/en/`

that incorporate welfare can be expected to increase compliance.

2.2 Computation of Market Equilibrium

The market equilibrium problem is to compute a price vector and a feasible assignment of goods to buyers such that no buyer is induced to change his assignments with respect to the given set of prices, and the market is cleared i.e. there is no surplus or deficit of goods. This problem is of considerable interest in Economics and has a long history. It was first proposed by Irving Fisher in 1891 and Leon Walras independently proposed the general equilibrium model in 1894 (Walras (1954)). Herbert Scarf hailed the proof of the existence of a solution for the neoclassical model of economic equilibrium as the major triumph of mathematical economics in his time, and with it arose the natural question of establishing an efficient computation process to find the proposed equilibrium. Scarf described "a general method for the explicit numerical solution of the neoclassical general equilibrium model" in his 1973 monograph on 'The Computation of Economic Equilibria' (Scarf and Hansen (1973)).

The equilibrium for EXaM is closely related to Fisher's formulation, and the algorithm proposed here could possibly be extended and guide development of algorithms to solve other general equilibrium problems. The problem of computation of economic equilibrium has a long history, and has been studied in mathematical economics and recently in algorithms theory. In the literature review, I draw on both fields to understand this problem at intersection of computer science and economics. The final algorithm used to implement EXaM was chosen based on efficiency, interpretability, and practical implementation. The open source package and code should enable researchers to easily use EXaM for experiments, and develop new approaches to solving market equilibrium.

3 Experiment-as-Market (EXaM) Mechanism

The experimental design problem tackled by the Experiment-as-Market (EXaM) mechanism is described below (based on Narita (2017)):

3.1 Setting

- Experimental subjects: $i_1, i_2, ..., i_n$
- Experimental treatments: $t_0, t_1, t_2, ..., t_m$ where t_0 is a placebo or control
- Each treatment t has capacity or supply: $c_t \in N$ with $\sum_{t=t_0}^{t_m} c_t \geq n$
- Each subject i has willingness to pay for treatment t: $w_{it} \in R$ with $w_{it} \geq w_{it'} \Leftrightarrow i$ weakly prefers t over t'
- Each treatment t's predicted treatment effect for subject i: $e_{ti} \in R$ with $e_{ti} \geq e_{ti'} \Leftrightarrow t$ is predicted to have a weakly better effect for subject i over i'

The w_{it} and e_{ti} values are normalized by assuming that $e_{t_0 i} = w_{i_0 t} = 0$ for every subject i. Thus, the values of w_{it} and e_{ti} are the willingness to pay (WTP) and predicted treatment effects (PTE) for t relative to the control or placebo treatment t_0. The willingness to pay and predicted treatment effects can be freely correlated.

The willingness to pay and predicted treatment effect described above can be obtained from a few possible sources. Willingness to pay can be estimated using data on treatment choices made by subjects, or by asking subjects to self-report the WTP. Predicted treatment effect is best estimated from prior experimental or observational data. Sequential RCTs with the same treatment are common in medicine, thus a prior RCT of a similar treatment is the most reliable data source for PTE.

3.2 Experiment as a Market

As a benchmark, we define the Randomized Control Trial (RCT) mechanism as follows: RCT assigns each subject i to treatment t with an impersonal treatment assignment probability p_{it}^{RCT} that can be written as $p_{it}^{RCT} = c_t/n$ where $c_t \in N$. c_t is the capacity or supply of treatment, and $\sum_i p_{it} \leq c_t$ for all treatments.

The Experiment as a Market (EXaM) mechanism works as follows:

1. Using a computer, distribute a common artificial budget $b > 0$ to every subject.
2. Find a 'price-discriminted market equilibrium' – i.e. treatment assignment probabilities (p_{it}^{*}) – and their prices π_{te} with the following properties:
   - Effectiveness discriminated treatment pricing:
     There exists $\alpha < 0$ and $\beta_t \in R$ for each treatment t such that the price of a unit probability of assignment to t for subjects with $e_{ti} = e \in R$ is:

$$\pi_{te}^{*} = \alpha_t e + \beta t$$

- Subject utility maximization subject to the budget constraint: For all subjects i,

$$(p^*_{it})_t \in argmax_{\text{ feasible } (p_{it})_t} \sum_t p_{it} w_{it} \text{ s.t. } \sum_t p_{it} \pi^*_{te_{ti}} \leq b$$

  Here $\pi^*_{te_{ti}}$ is the price of a unit probability of assignment to treatment t for subject i.

- Meeting capacity constraints:
  $\sum_i p^*_{it} \leq c_t$ for all $t = t_1, ..., t_m$

3. Require each subject to get each treatment with a probability strictly between 0 and 1. To do this, we compute

$$p^*_{it}(\epsilon) \equiv (1-q)p^*_{it} + qp^{RCT}_{it}$$

   where $q \equiv \inf\{q' \in [0,1] | (1-q')p^*_{it} + q'p^{RCT}_{it} \in [\epsilon, 1-\epsilon]$ for all i and $t\}$. Here $\epsilon \in [0, \bar{\epsilon})$ is a parameter fixed by the experimenter where $\bar{\epsilon} \equiv \min\{\min_t p^{RCT}_{it}, 1 - \max_t p^{RCT}_{it}\}$ is the largest possible value of ϵ.

4. Draw final treatment assignments from $(p^*_{it})_{i,t}$

3.3 Randomized Controlled Welfare Property

Experiment-as-Market always exists and satisfies:

1. Randomly assigning treatments,

$$p^*_{it} = p^*_{i't} \ \forall\, t, i, i' \ , (w_{it}, e_{ti})_t = (w_{i't}, e_{ti'})_t$$

2. Generating as much causal information as the usual RCT, i.e. anything identified by a vanilla RCT can also be identified by this mechanism

3. As far as possible, assigning treatment to subjects with

   - better predicted treatment effects or
   - stronger preferences for the treatment

   i.e. there are no assignment probabilities (p_{it}) for i with

   - $\sum_t p_{it} e_{ti} \geq \sum_t p^*_{it} e_{ti}$ (expected predicted effect)
   - $\sum_t p_{it} w_{it} \geq \sum_t p^*_{it} w_{it}$ (expected willingness to pay)

EXaM is a generalization of a RCT. To see this, suppose that WTP and PTE are unkown or ignored, so that $w_{it} = w_{jt} > 0$ and $e_{ti} = e_{tj}$ for all subjects i and j and treatments t. In this case, EXaM reduces to RCT, i.e. for every $\epsilon \in [0, \bar{\epsilon})$, subject i and treatment t, we have $p^*_{it}(\epsilon) = p^{RCT}_{it}(\epsilon)$

3.4 EXaM and General Equilibrium

In the EXaM mechanism subjects are randomly assigned to treatment through an artificial, centralized market, building on the idea of competitive market equilibrium from equal incomes. To start, EXaM gives each subject an endowment of a common artificial budget. The mechanism allows each subject to use the budget to purchase the most preferred bundle of treatment assignment probabilities, given a set of prices. The prices faced by subjects are partially personalized so that a treatment is cheaper for those subjects with better predicted treatment effects from the treatment – the price faced by a subject for a treatment is a linear function of the predicted treatment effect. EXaM computes the treatment assignment probabilities as those that subjects purchase at the market clearing price, where the total demand for each treatment across all subjects is balanced with its given supply or capacity. Some constraints must also be met: the total demand for each treatment must be balanced with its supply (capacity constraint) and each subject can get each treatment with a probability strictly between 0 and 1 (feasibility constraint).

The EXaM mechanism is similar to the classical general equilibrium model, but has three crucial differences that deserve to be highlighted: the role of PTE, partial equilibrium, and absence of supply side modeling. EXaM can be seen as an extension of the classic general equilibrium, since removing the component of Predicted Treatment Effect (PTE) reduces EXaM to the traditional exchange economy problem. This marks a crucial difference in the EXaM mechanism that allows each subject to face a unique price for a unit probability instead of a market price common to all subjects in a traditional exchange economy. The EXaM mechanism only models the partial equilibrium setting, focusing on a single market instead of the general equilibrium – the process of optimizing which RCT to enroll in is not considered. The supply side behavior is also not modeled and the supply of each good (treatment) is considered to be fixed and exogenous. Overall, the EXaM mechanism extends the traditional general equilibrium exchange economy model by adding the Predicted Treatment Effect, but also has limitations of focusing on the partial equilibrium and taking the supply side to be fixed.

4 Algorithm

The central problem of EXaM is to find the market equilibrium where each subject's individual utility is maximized, and the constraints on capacity and probability are satisfied. In the algorithm, this is modeled as a price search problem, exploring a number of possible price matrices to find the market clearing price. Since price and demand are interconnected in computing the equilibrium, we cannot find a direct solution to the problem. Instead, we aim to compute an approximate equilibrium with iterative steps. The algorithm is able to find very good solutions with low clearing error – thus it is able to approximate the true global equilibrium for all situations.

If we were to imagine a discrete price space up to 100 (the artificial budget given to each subject) and limit to 100 treatments and subjects, then an exhaustive enumeration of the potential prices would have a cardinality on the order of 100^{100}. In contrast, when we run a similar sized problem, the number of linear programming problems computed is much smaller – on the order of 10^6. To give some perspective, 100^{100} is greater than the number of atoms in the universe, and 10^6 is smaller than the number of atoms in a grain of sand. This illustrates that the algorithm is a massive improvement over a brute force approach, and applies a reasonable heuristic to the search problem.

4.1 Setup and Useful Subroutines

The algorithm is detailed in pseudocode below, and a script with an implementation in python is attached in the appendix and available on GitHub. Before describing the main computation of the algorithm, I will highlight some features of EXaM and describe some useful subroutines for the computation.

The inputs to the algorithm are:

1. n the number of subjects, m the number of treatments
2. A list of capacities for each treatment of size m, ($(c_t) \in \mathbb{N}$ such that $\sum_t c_t = n$) and budget constraint b
3. A Willingness-To-Pay (WTP) matrix of size n x m, where (w_{it}) is subject i's WTP for treatment t
4. A Predicted-Trreatment-Effect (PTE) matrix of size n x m, where (e_{it}) is subject i's PTE for treatment t

The algorithm outputs:

1. A n x m matrix of equilibrium demand – the probability matrix. (p^*_{it}) is treatment t's assignment probability for subject i
2. A n x m matrix of the market clearing equilibrium price π^*_{te}, which is defined by 2 parameters as $\pi^*_{te} = \alpha^* e + \beta^*_t$. (π^*_{te}) is the treatment t's equilibrium price, defined for a subject with a given e_{it}.

The first feature to highlight is that since the EXaM mechanism is a generalization of the general equilibrium, each (subject, treatment) pair has a price. In the usual case, each treatment (analogous to a good) would have a price, but here the price is more complex. Thus, in the algorithm we use a price matrix, not a simple vector. The price is modeled as a linear function of the predicted treatment effect (PTE), by the equation $\pi^*_{te} = \alpha^* e + \beta^*_t$. An initial price matrix is computed by the `price` function which uses the `init_alpha` and `init_beta` functions to randomly initialize the parameters and returns a n x m price matrix.

Given a price, we can calculate the demand in the market by solving each subject's utility maximization problem. The utility maximization problem is modeled as a linear programming problem, with all the necessary constraints of the mechanism applied. In the implementation, this is solved using the `linprog` package, which relies on the simplex method. The simplex algorithm has a polynomial time complexity in the average case and is very efficient in practice, but has a theoretical worst-case exponential complexity. This computation happens in the `demand` function, which returns a n x m demand matrix.

Given the demand in the market, we can find the excess demand by aggregating the demand for a given treatment and subtracting the treatment capacity. This computation happens in the `excess_demand` function, which returns a m dimensional excess demand vector. I further simplify this excess demand into a singular measure of clearing error – the market clearing error relative to the total capacity. The clearing error is $\sqrt{\sum_t d_t^2} / \sum_t c_t$. This serves as the loss metric to guide the optimization core to computing the equilibrium.

4.2 Price Search

The key idea behind the algorithm is to model the market equilibrium as a search problem for the right prices, core to which is the optimization problem for the right prices that minimize the clearing error in the market. The price here is a matrix since each treatment has a different price for each subject in EXaM. In the optimization problem, we want to find the right set of prices (parameters to change) that minimizes the clearing error (loss function). For practical computational reasons, so that the computation is completed in a reasonable time, we want to find an approximate solution. In the algorithm we can simply set a hyper-parameter for the clearing error threshold which controls the accuracy of the approximation. The algorithm essentially needs to try a number of different price matrices until the clearing error drops low enough to satisfy our needs. We can visualize this problem geometrically by considering a geography of the price matrix and loss function – each point in the space would correspond to a given price matrix, and have a height given by the clearing error. Our goal is to find a point where the error drops below a threshold, where the height of the point would be lower than a cutoff.

There is an enormous amount of computational work required at each step of the algorithm, since computing the clearing error of the market is a computationally intensive process. Each element of the demand matrix requires solving a linear programming problem with many constraints, and this process must be repeated many times. To give a sense of the computational effort required, a problem of 10 treatments and 100 subjects in the EXaM would require computing about 1 billion linear programming problems. If it takes one millisecond to solve each linear programming problem – an optimistic assumption in practice – finding an approximate competitive equilibrium would still more than a week. A brute force search would increase the time by several orders of magnitude. It is clear that the search process needs a good heuristic based on the EXaM mechanism, and careful tuning of the hyper-parameters involved for performance that would enable practical use.

After experimenting with a number of approaches, we chose random local search as the best search strategy for the problem. The key idea here is to start with a random initialization of the price vector, generate some random perturbations to the price and if the loss at the perturbed price is lower, we can update to follow that path. Continuing the geometric analogy, we can think of this as starting at a random point in the space, extending our foot out in a random direction, but then only taking a step forward if it leads downhill (to a lower loss/clearing error). If our random local step leads to a higher clearing error, we simply try another random local step.

This random local search works well, but has the danger of getting caught in a local minima that never meets our approximation criteria. We might get stuck in a local minima where the clearing error increases in all directions, but is too high to be an approximate equilibrium. To make a robust algorithm, we need to be able to abandon a search that leads us down the wrong path. This leads us to a crucial hyper-parameter to the algorithm that controls the number of iterations we evaluate down a path before abandoning it for a new random start, defined as `Iteration_Threshold`.

4.3 Alternative Approaches Considered

I also considered a number of alternative approaches to the price search mechanism when developing the algorithm. In the end, practical computation considerations led to choosing random local search as the best option since it led to a fast solution for the equilibrium and was easy to understand, interpret and explain.

A promising alternative algorithm I developed relied on a tabu search instead of a random local search. Glover (1986) created tabu search as a meta-heuristic search method for optimization, in order to improve on local search methods. In tabu search, the basic rule of local optimization – every step must lead to a lower error – is relaxed. If no improving step is available, such as when the search is stuck in a local optimum, some worsening moves can be accepted. To prevent the search from entering an infinite loop, some prohibitions or 'tabu' states are remembered by the algorithm.

I developed a tabu search and a gradient descent algorithm to implement EXaM in addition to the random local search algorithm finally chosen. For the gradient descent approach, the computation of the gradient proved to be too costly since it required a multiplicative increase in LP problems to be solved at each step. Tabu search also required much more computation per step, since the best candidate price had to selected at each step after considering a number of candidates. The process of evaluating each candidate – to calculate the clearing error – is an expensive operation and thus tabu search also took more time to compute the equilibrium. While tabu search took fewer steps to reach the equilibrium price since each step was toward a considered direction (best of candidates) instead of random, the trade-off in increased computation per step was not favourable.

Algorithm 1 Experimental as Market (EXaM)

Require: n the number of subjects, m the number of treatments, $(c_t) \in \mathbb{N}$ treatment t's capacity with $\sum_t c_t = n$, (w_{it}) subject i's WTP for treatment t, (e_{ti}) treatment t's predicted treatment effect for subject i, b the budget constraint, $0 < \epsilon$ the bound on treatment probabilities.

Ensure: (p^*_{it}) treatment t's assignment probability for subject i, (α^*, β^*_t) parameters determining treatment t's equilibrium price of the form $\pi^*_{te} = \alpha^* e + \beta^*_t$.

```
1:  function INIT_ALPHA( )
2:      $\alpha \leftarrow$ generate $\sim$ Uniform$(-b, 0)$                 ▷ set the value of $\alpha$
3:      return $\alpha$
4:  function INIT_BETA( )
5:      $\beta_{t_0} \leftarrow 0$
6:      for each $t$ do
7:          $\beta_t \leftarrow$ generate $\sim$ Uniform$(-b, b)$         ▷ set the initial value of $\beta_t$
8:      return $(\beta_t)$                               ▷ return an $m$-dimensional vector

9:  function PRICE($\alpha$, $(\beta_t)$)                  ▷ get the price of treatment $t$
10:     for each $i, t$ do
11:         $\pi_{te_{ti}} = \alpha e_{ti} + \beta_t$
12:     return $(\pi_{te_{ti}})$                           ▷ return the $n \times m$ price matrix

13: function DEMAND($\epsilon, (\pi_{te_{ti}})$)       ▷ get the subject $i$'s demand for treatment $t$
14:     for each $i$ do              ▷ perform utility maximization for each subject $i$
15:         $(p_{it})_t \leftarrow \arg\max_{(p_{it})_t} \sum_t w_{it} p_{it}$
              s.t. $\sum_t \pi_{te_{ti}} p_{it} \le b, \ \sum_t p_{it} = 1, \ \epsilon \le p_{it} \le 1 - \epsilon$
16:     return $(p_{it})$                            ▷ return the $n \times m$ demand matrix

17: function EXCESS_DEMAND($(p_{it})$)           ▷ get the excess demand for treatment $t$
18:     for each $t$ do
19:         $d_t \leftarrow \sum_i p_{it} - c_t$
20:     return $(d_t)$                ▷ return the $m$-dimensional excess demand vector

21: function ClearingError($(d_t)$)     ▷ get the market clearing error relative the total
    capacity
22:     if $d_t < 0$ for all $t$ then
23:         return 0
24:     else
25:         error $\leftarrow \sqrt{\sum_t d_t^2} / \sum_t c_t$
26:         return error                          ▷ return the market clearing error
```

```
function BetaNew($\alpha$, $(d_t)$)                         ▷ recalibrate $\beta_t$'s to set new prices
    for each $t$ do
        $\beta_t^{new} \leftarrow \beta_t + d_t \delta_\beta$
    return $(\beta_t^{new})$

function ClearMarket( )                                      ▷ the main function
    $\alpha \leftarrow$ InitialAlpha( )
    $(\beta_t) \leftarrow$ InitBeta( )
    $(\pi_{te_{ti}}) \leftarrow$ Price$(\alpha, (\beta_t))$
    $(p_{it}) \leftarrow$ Demand$((\pi_{it}))$
    $(d_t) \leftarrow$ ExcessDemand$((p_{it}))$
    error $\leftarrow$ ClearingError$((d_t))$
    error$_{min} \leftarrow$ error                          ▷ initialize the min of clearing error
    $\delta_\beta \leftarrow b/50$                          ▷ scaling factor for $\beta_t$'s to set new prices
    ClearingThreshold $\leftarrow$ 0.01                     ▷ threshold for market clearing error
    IterationThreshold $\leftarrow$ 10                      ▷ threshold for iteration times
    iterations $\leftarrow$ 0                               ▷ initialize iteration time count
    while True do
        if iterations > IterationThreshold then
            $\alpha \leftarrow$ InitialAlpha( )             ▷ start new equilibrium research
            $(\beta_t) \leftarrow$ InitBeta( )
            iterations $\leftarrow$ 0
        else
            $(\beta_t) \leftarrow$ BetaNew$((\beta_t), (d_t))$
        $(\pi_{te_{ti}}) \leftarrow$ Price$(\alpha, (\beta_t))$
        $(p_{it}) \leftarrow$ Demand$(\epsilon, (\pi_{te_{ti}}))$
        $(d_t) \leftarrow$ ExcessDemand$((p_{it}))$
        error $\leftarrow$ ClearingError$((d_t))$
        if error < error$_{min}$ then
            error$_{min} \leftarrow$ error
            $\alpha^* \leftarrow \alpha$                    ▷ the new prices reduce the error
            $(\beta_t^*) \leftarrow (\beta_t)$
            $(p_{it}^*) \leftarrow (p_{it})$
        if error$_{min}$ < ClearingThreshold then
            break
        iterations += 1
        return $((p_{it}^*), \alpha^*, (\beta_t^*),$ error$_{min})$     ▷ return the outputs
```

5 Results

The algorithm performs well and is able to efficiently compute the market equilibrium and market clearing price. It provides a practical implementation of the EXaM mechanism that will enable researchers to use it to match subjects to treatments in an experiment. In this section, I will first describe an example problem to show that the algorithm solves for the market equilibrium correctly – finding a market equilibrium that satisfies all constraints. I then use a simulated problem based on artificially generated data to test the algorithm on a larger dataset and give some insight into how the algorithm works. The main results come from an empirical application of this algorithm using real experimental data from a RCT by Kremer et al. (2011). Here I use data from the RCT and a simulated counterfactual EXaM study to compare the two mechanism designs. I find that the algorithm outputs a treatment assignment that is different from that under the vanilla RCT, and which leads to an increase in subject welfare while replicating the treatment effects of the original paper by Kremer et al. (2011). These results provide a practical implementation of the theoretical EXaM mechanism, and show the improvement in subject welfare using real experimental data.

5.1 Example Problem

An example problem is presented here, just to demonstrate the algorithm's output for a simple situation. The algorithm takes 2 matrices as inputs – first with willingness to pay, and second with predicted treatment effects. Here, subject A prefers treatment 1 in both willingness to pay and the RCT organizers also prefer treatment 1 for subject A in terms of predicted treatment effect.

WTP table	Treatment 1	Treatment 2
Subject A	2	1
Subject B	1	2

PTE table	Treatment 1	Treatment 2
Subject A	2	1
Subject B	1	2

Final Demand	Treatment 1	Treatment 2
Subject A	1	0
Subject B	0	1

The algorithm is able to assign subjects to treatments correctly, with a clearing error of 0.0.

5.2 Simulated Problem

To see the algorithm in action, we created [20 x 20] matrices of willingness to pay and predicted treatment effect where the values were drawn randomly from $\{1, 2\}$. In this case, the algorithm solves about $40,000$ linear programming problems. This shows that the heuristic search works very well – computing the whole problem space would involve billions of problems and take months. We are able to find a very good solution by only computing a tiny fraction of problems and get to a result in ≤ 1 min.

Below I present some figures to show how the algorithm explores the search space. When plotting the clearing error at each candidate price, we see a noisy graph that shows the rugged search landscape and the regular peaks in clearing error when the search is re-started at a new random point. The second graph shows a consistently decreasing overall best clearing error, as the algorithm explores the search space while keeping track of the best solution so far. Figure 1 plots the clearing error for each candidate price, showing all 100 iterations. Figure 2 plots the best error, which is updated only when we find a lower clearing error than the best found so far. This updates 8 times, with the 8th update corresponding to the 32nd iteration plotted in Figure 1. Due to the random initialization of the search start, the correspondence between number of updates and number of candidate price matrices explored is dependent on the input data.

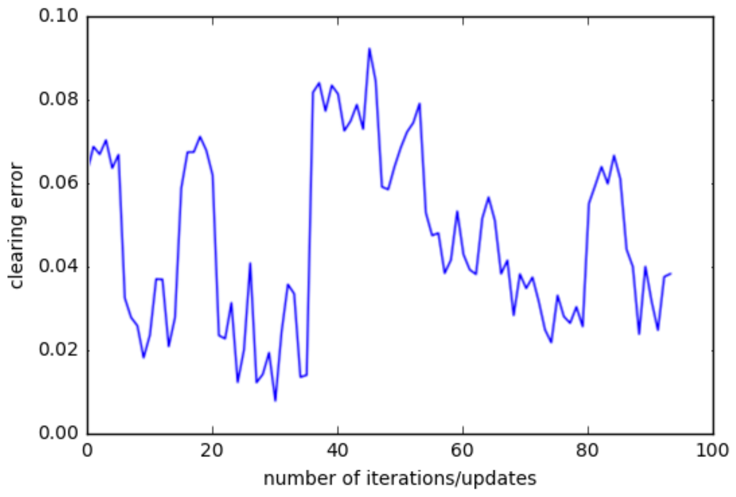


Figure 1: This graph plots the clearing error for each candidate price matrix explored. We can see that the plot is very noisy – demonstrating that the search landscape is rugged. We can also see how the search is re-starting at a new random initialization point where clearing error would be expected to be high by the regular peaks in the graph.

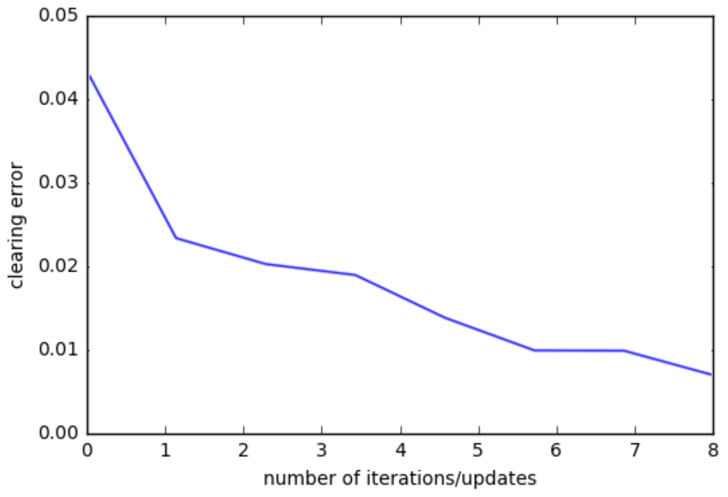


Figure 2: This graph plots the overall best clearing error, i.e. the lowest clearing error found so far which corresponds to the best price matrix. This is updated when we find a new best error, and the 8th update corresponds to the 32nd iteration of the candidate price plotted in figure 1. We can see a steady decrease till convergence.

5.3 Empirical Application: EXaM vs. RCT

I will now demonstrate the application of the algorithm to find market equilibrium in a real, empirical setting – the comparison of EXaM vs. a RCT. Imagine that a researcher is planning a new empirical study and wants to choose the better experiment design between EXaM and a vanilla RCT. A perfect comparison of the two designs would require some meta-experiment where different designs would be assigned to experimental studies. This is an unrealistic scenario, but we can propose a credible alternative – simulate a counterfactual EXaM study using data from a RCT. We can use experimental data from a previous RCT study, estimate the willingness to pay and predicted treatment effects, and then use some bootstrapped data to simulate the EXaM. We can then implement the EXaM mechanism, using the algorithm to find the optimal assignment of subjects to treatment, and compare the results of EXaM against the RCT in terms of information and welfare properties.

To empirically evaluate EXaM, we use experimental data from a RCT study by Kremer et al. (2011) which studied the health impact of spring protection in Kenya. This experiment conducted by International Children Support (a NGO in Kenya), randomly selected springs to receive protection from the universe of 200 local unprotected springs. The experimental subjects were a representative sample of 1500 households that regularly used some of the 200 springs. Kremer et al. (2011) found that spring protection substantially improves source water quality and is moderately effective at improving household water quality after recontamination. Diarrhea among children in treatment households fell by about a quarter of the baseline level.

To execute the EXaM mechanism, we first need to know the w_{it} and e_{ti} values required. These were estimated using experimental data from Kremer et al. (2011) by Professor Yusuke Narita and some research assistants as part of a larger project, and is described in detail in Narita (2017). They estimated the e_{ti} treatment effect in a similar way as Kremer et al. (2011), where households with better diarrhea prevention knowledge or mother's education tend to have better treatment effects. For the w_{it} estimate, a discrete choice model of households' water source choices was used, where households trade off water quality against other source features such as proximity to household. This model produces revealed preference estimates of household valuation of the spring protection treatment. Further details of this estimation procedure are described in the draft paper – Narita (2017).

Given the estimates of w_{it} and e_{ti} based on experimental data, we can simulate the EXaM mechanism using the algorithm, and compare results against the RCT in terms of information and welfare properties. Throughout, the set of subjects and treatments are fixed as in Kremer et al. (2011): there are 1540 households as subjects that can be assigned to either the single water source protection treatment t_1 or the control treatment t_0. The treatment capacity $c_{t_1} = 663$, the number of households that can be assigned to treatment t_1 is also the same as Kremer et al. (2011). The WTP and PTE values are drawn using a parametric bootstrap from the estimated distribution of $\hat{w_{it}}$ and $\hat{e_{ti}}$. After simulating (w_{it}, e_{ti}), we can run the algorithm described above to find the market equilibrium for the EXaM mechanism. The algorithm outputs us the treatment assignment probabilities $p_{it}^{EXaM} = p_{it}^{*}(\epsilon)$. These treatment assignment probabilities are often different from the constant probabilities in the vanilla RCT p_{it}^{RCT}. In Figure 4 we can clearly see that using the EXaM mechanism would give us a different assignment of subjects to treatment than the vanilla RCT.

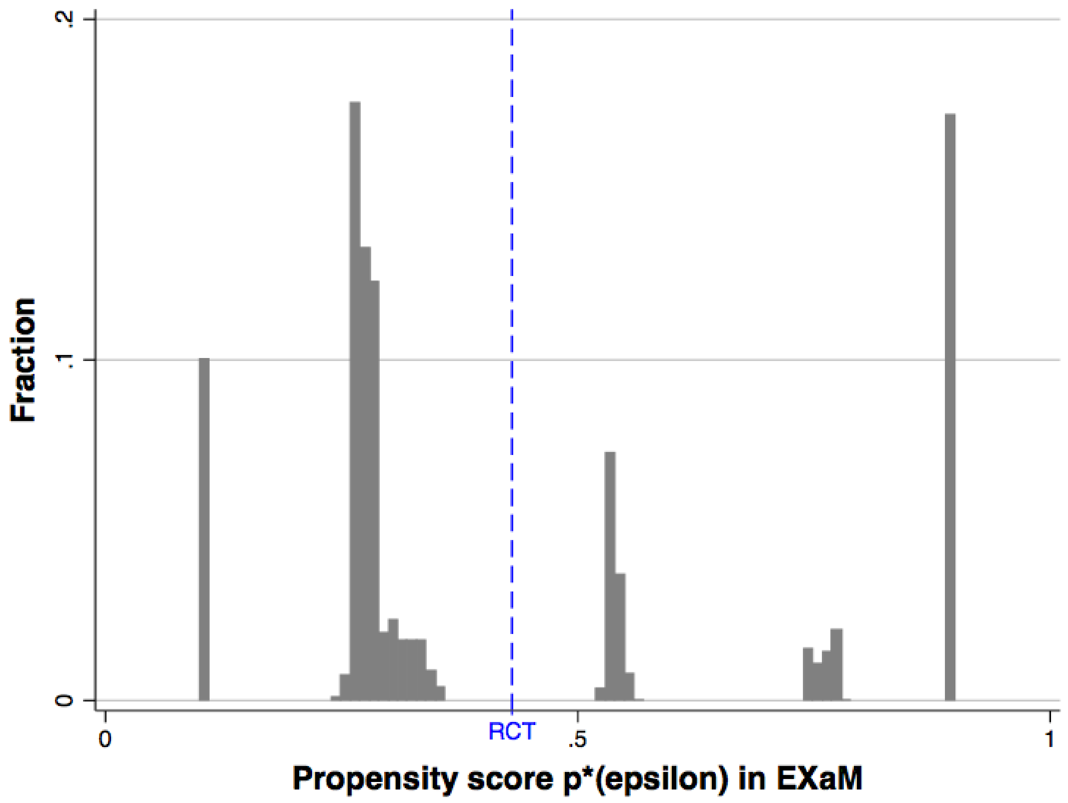


Figure 3: This figure shows the distribution of EXaM's treatment assignment probabilities p_{it}^{EXaM} over 1000 bootstrap simulations and households – the output of the algorithm on real experimental data. The vertical dashed line is for comparison against the vanilla RCT's constant assignment probability p_{it}^{RCT}.

5.4 Welfare

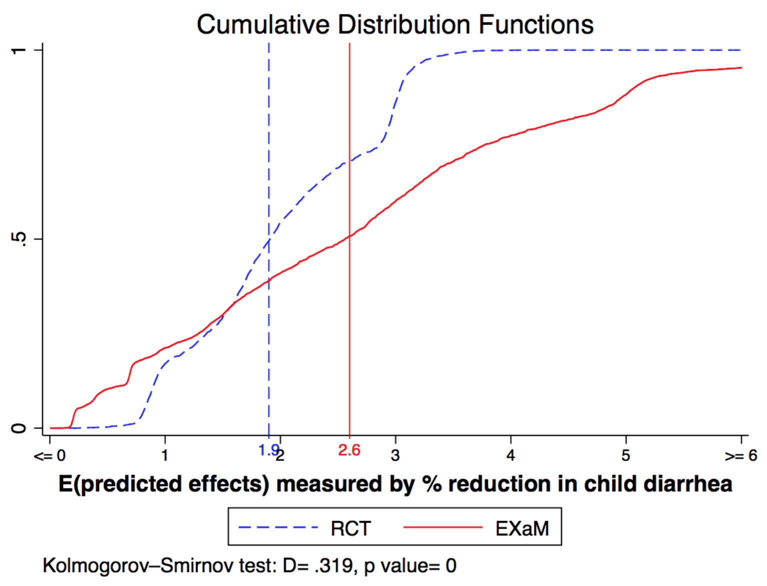
Having shown that EXaM and RCT lead to different assignment of subjects to treatment, the natural next question is to compare the two designs in terms of subject welfare. The primary selling point of EXaM is that it incorporates subject welfare into the experimental design, so if the algorithm works correctly, we would expect the EXaM treatment assignments to lead to a higher welfare measure.

We calculate two welfare measures for each household i:

$$w_i^* \equiv \Sigma_t p_{it}^*(\epsilon) w_{it} \text{ and } e_i^* \equiv \Sigma_t p_{it}^*(\epsilon) e_{ti}.$$

Here w_i^* and e_i^* are two ex ante welfare measure in terms of WTP and PTE. We can similarly simulate the w_i^{RCT} and e_i^{RCT} values for the RCT, except that the RCT has a fixed probability of assignment to the treatment group $p_{it_1}^{RCT} \equiv c_t/n = 663/1540 = 0.43$.

We can see that EXaM improves on the RCT in terms of welfare measures w_i^* and e_i^*. In Figure 5, I draw the distribution of w_i^* and e_i^* over households and 1000 bootstrapped sample runs of the algorithm. The median of average WTP w_i^* for assigned treatments increases by about 60% or 5.7 workday-equivalent utilities under EXaM over vanilla RCT. Similarly, the median average PTE e_i^* improves by a 0.6% absolute reduction or 30% reduction relative to RCT. This predicted treatment effect benefit is about 13% of the average treatment effect of the spring protection found by Kremer et al. (2011). Thus by applying the algorithm using real experimental data, we can clearly see that EXaM leads to better welfare outcomes in comparison to the RCT.



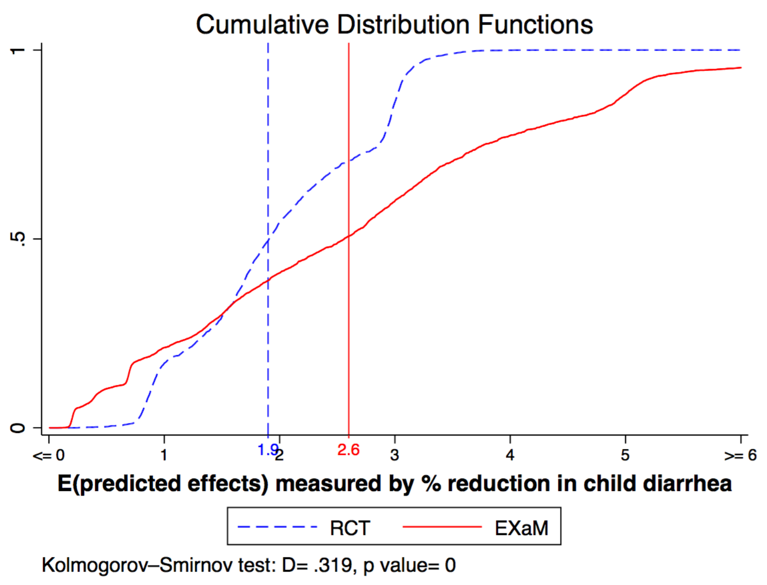


Figure 4: Here we show the distribution of average subject welfare over 1000 bootstrap simulations, to compare the welfare properties of EXaM and RCT. Welfare is measure for WTP in the first figure, and PTE in the second. The vertical lines show the median, dotted for RCT and solid for EXaM.

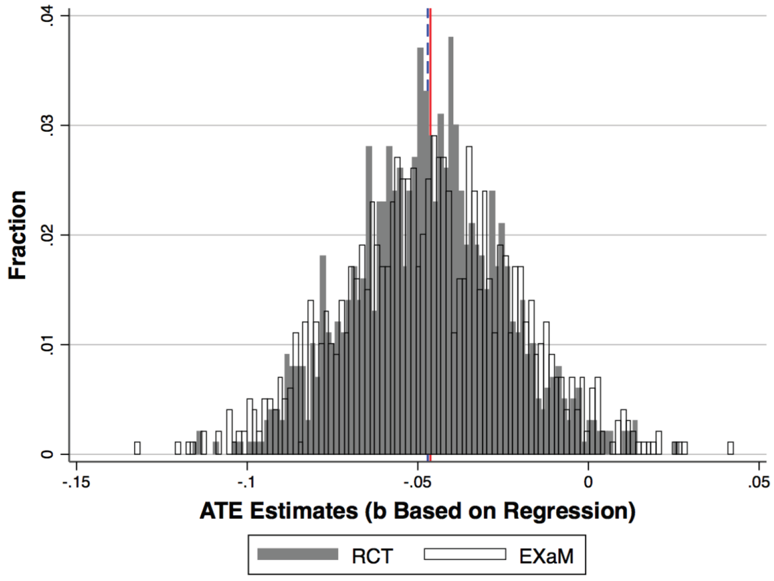
5.5 Information

We now compare EXaM and RCT on the econometric conclusions each design presents for the same experiment. In addition to the benefit of incorporating subject welfare, EXaM is expected to extract as much information from the experiment as a RCT. To get the treatment effect estimates under EXaM, we run the following procedure multiple times:

1. Estimate (w_{it}, e_{ti}) based on experimental data, run the EXaM algorithm using these estimates to get the treatment assignment probabilities $p^*_{it}(\epsilon)$. Use $p^*_{it}(\epsilon)$ to draw a final deterministic treatment assignment $D_i \equiv 1\{i \text{ is ex-post assigned to } t_1\}$
2. Simulate the predicted outcome Y_i under D_i based on the experimental scheme in Kremer et al. (2011)
3. Use the simulated Y_i and D_i to estimate the treatment effects with $\hat{b}_{OLS}$ from the regression
$$Y_i = a + bD_i + cp^*_{it}(\epsilon) + e_i$$
where we control for the propensity score $p^*_{it}(\epsilon)$ to make the treatment assignment D_i conditionally random, as suggested by the EXaM mechanism.

When calculating the treatment effect estimates under RCT, the procedure is analogous except that the treatment assignment probability is fixed at p^{RCT}_{it}.

The treatment effect estimates with EXaM turn out to be very similar to those from the RCT.



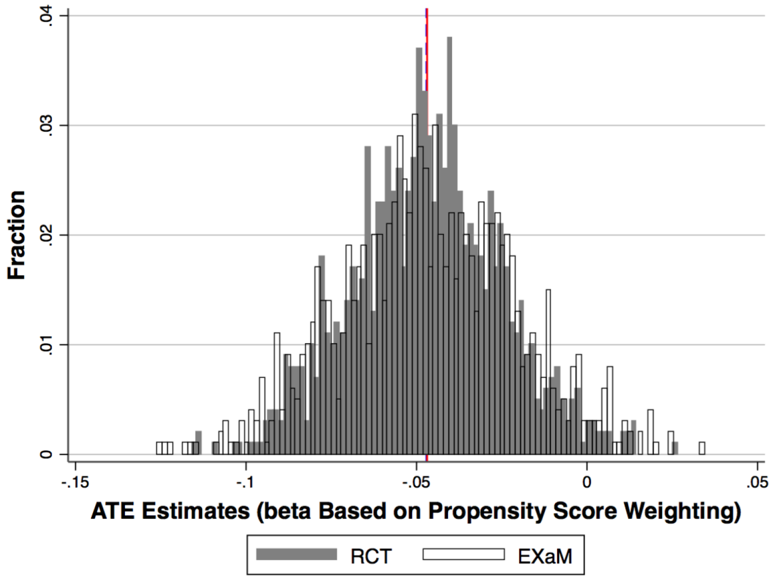


Figure 5: These graphs compare the performance of EXaM and RCT by plotting the distribution of average treatment effect estimates under each experimental design. The grey bins are average treatment effect estimates for RCT, transparent for EXaM. The lines represent the median values, dashed for RCT and solid for EXaM. We see effectively the same estimates from both designs.

The distribution of the treatment effect estimates $\hat{b^*}$ and $\hat{\beta^*}$ over 1000 simulations is plotted in figure 5. We can see that the median values for both $\hat{b^*}$ and $\hat{\beta^*}$ are indistinguishably similar for EXaM and RCT, and replicate the average treatment effect in the original paper by Kremer et al. (2011).

6 Literature Review

General equilibrium theory, first pioneered by Irving Fisher and Leon Walras, deals with the complex interaction of many agents in a market, where each is willing to trade goods he owns for goods that he desires (Walras (1954), Brainard et al. (2000)). To determine the demand for a good, each good is assigned a price and buyers can sell their endowments at these prices to buy the optimal bundle of goods that they can afford. A utility function is used to express how desirable each bundle of goods is to an agent. A market equilibrium corresponds to a situation when the demand and supply of each good are exactly balanced – this occurs at the market equilibrium prices. Arrow and Debreu (1954) proved the existence of equilibrium prices in this model of a market in their seminal work "Existence of Equilibrium for a Competitive Economy". Scarf (1982) called this proof of the existence of a solution for the neoclassical model of economic equilibrium as the major triumph of economics in his time.

However, this proof is non constructive and relies on fixed point theorems. Arrow and Debreu (1954) used the Kakutani fixed point theorem, a generalization of the Brouwer fixed point theorem. The non constructive proof leads to a natural question – is there an efficient computation process which can solve for the market equilibrium?

The ultimate goal of equilibrium theory can only be achieved if we can actually find an equilibrium, not just know that one exists. This problem of computation of market equilibrium has a long history, and presents an interesting intersection of economic theory and computer science. In this review, I will draw on literature from both mathematical economics and algorithm theory.

The question of computation has been crucial to the general equilibrium model from the very beginning. Irving Fisher's Ph.D. thesis at Yale University in 1891 contained a fully articulated general equilibrium model, similar to Walras and his successors (Brainard et al. (2000)). Perhaps even more interesting is that Fisher also presented a hydraulic apparatus for calculating the equilibrium prices and the resulting distribution of endowments among the agents in the economy. While the central ideas of equilibrium theory behind general equilibrium were independently discovered at several locations, no other economist of Fisher's time has suggested the exploration of equilibrium analysis by constructing specific numerical models with a moderately large number of commodities and consumers, and finding the prices that would equate supply and demand in all markets. Fisher developed a complex and ingenious hydraulic machine that correctly solved for equilibrium prices in a three person, three commodity exchange economy in which each consumer has additive, monotonic and concave utility functions and a specified money income, where market supplies of each good are exogenously given.

The next big step in the literature around this problem occurred in the 1960s, when the intimate connection between the ideas of fixed-point and market equilibrium was exploited for computational goals by Herbert Scarf and some coauthors, who employed path-following techniques to compute an approximate equilibrium price (Scarf and Hansen (1973), Scarf (1982), Eaves and Scarf (1976)). In their simplest form, these methods are based upon a decomposition of the price simplex into a large number of small regions and on the use of information about the problem instance to construct a path that can be shown to terminate close to a fixed point. While the appropriate termination is guaranteed by the fixed point theorems, the worst case running time of these algorithms turns out to be exponential.

The problem of computing economic equilibrium has also attracted some attention in theoretical computer science research and algorithms theory. While algorithmists have primarily focused on the related problem of computing Nash equilibria, work by Christos Papadimitriou and others has developed some efficient algorithms for this problem (including polynomial time algorithms for restricted problems), and created a complexity class to understand computational limits of the problem of solving for a market equilibrium (Papadimitriou (1994), Papadimitriou (2014), Devanur et al. (2002)).

Algorithms theory studies a foundational question of computer science first posed by Alan Turing – 'What are the limits of computation?' Turing (1937). The golden standard for efficient computation in algorithms theory is the criteria of polynomial time (P). The total number of operations a polynomial time algorithm performs is always bounded by a polynomial function of the size of the input (as opposed to an exponential function, for example). This is the class of all search problems that can be solved efficiently, such as finding the shortest path. In contrast to P, a different class of complexity covers search problems which are related to Brouwer's theorem – polynomial parity arguments for directed graphs or $PPAD$. By Brouwer's theorem, every continuous function from a convex compact set to itself has a fixed point.

The question then is, if given such a function, how can one find this fixed point? Papadimitriou (1994) showed this problem to be $PPAD$-complete. The problem of finding equilibrium prices in an economy is also $PPAD$-complete, even for an exchange economy with concave constant elasticity of substitution consumer utilities (a class of simple, well behaved utility functions).

In recent economics literature, Budish et al. (2016) solve a related problem of combinatorial allocation in their paper on 'Course Match: A Large-Scale Implementation of Approximate Competitive Equilibrium from Equal Incomes for Combinatorial Allocation'. Budish et al. (2016) propose a new mechanism for allocating a bundle of course schedules to students at Wharton, and develop a new course allocation mechanism (CourseMatch) that uses an algorithm to practically implement the mechanism with some modifications. This algorithm uses tabu search to perform a parallel heuristic search solving mixed-integer problems to find an approximate competitive equilibrium. Budish et al. (2016) draw on economic theory and computer science to implement a new mechanism design, in a manner similar to this project.

7 Conclusion

The Experiment-as-Market (EXaM) mechanism presents an alternative design for a Randomized Control Trial (RCT), which is the gold standard of causal inference. RCTs enroll hundreds of millions of subjects and are used to randomize high-stakes treatment, such as antiretroviral therapy to prevent HIV and unconditional cash transfers. EXaM is motivated by the large impact such interventions have on subjects' live, and seeks to incorporate subject welfare into the design of RCTs. By using information of willingness to pay and predicted treatment effects, the EXaM mechanism maximizes subject welfare while producing as much information as RCT. In EXaM, subjects are randomly assigned to treatment through an artificial, centralized market, building on competitive market equilibrium from equal incomes.

For the EXaM mechanism to be applied in practice, it is crucial to be able to compute the correct assignment of subjects to treatment arms. This paper presents the algorithm to match subjects to treatment that implements the EXaM mechanism. The algorithm efficiently computes a market equilibrium using iterative random local search to find a market clearing price that maximizes subject utility while satisfying capacity and feasibility constraints. I describe the functioning of the algorithm, and use this practical implementation of the EXaM mechanism to empirically verify its properties.

Using real experimental data from a RCT by Kremer et al. (2011) studying water source protection in Kenya, I run the algorithm to find a treatment assignment under EXaM that is different from that under RCT. The simulated counterfactual EXaM study leads to an increase in subject welfare, while replicating the treatment effects of the original RCT. The results demonstrate that the algorithm correctly and efficiently computes the market equilibrium, provide a practical implementation of the EXaM mechanism, and verify the properties of EXaM in an empirical setting.

As a part of the larger research project around EXaM detailed in Narita (2017), this paper integrates economic theory and computer science to develop a new algorithm to compute market equilibrium, and provides an open-source implementation of the EXaM mechanism to make it viable for practical use by researchers.

Bibliography

Arrow, Kenneth J and Gerard Debreu, "Existence of an equilibrium for a competitive economy," *Econometrica: Journal of the Econometric Society*, 1954, pp. 265–290.

Athey, Susan and Guido W Imbens, "The econometrics of randomized experiments," *Handbook of Economic Field Experiments*, 2017, *1*, 73–140.

Brainard, William C, Herbert E Scarf et al., *How to compute equilibrium prices in 1891*, Cowles Foundation for Research in Economics, 2000.

Budish, Eric, Gérard P Cachon, Judd B Kessler, and Abraham Othman, "Course match: A large-scale implementation of approximate competitive equilibrium from equal incomes for combinatorial allocation," *Operations Research*, 2016, *65* (2), 314–336.

Chan, Tat Y and Barton H Hamilton, "Learning, private information, and the economic evaluation of randomized experiments," *Journal of Political Economy*, 2006, *114* (6), 997–1040.

Cohen, Myron S, Ying Q Chen, Marybeth McCauley, Theresa Gamble, Mina C Hosseinipour, Nagalingeswaran Kumarasamy, James G Hakim, Johnstone Kumwenda, Beatriz Grinsztejn, Jose HS Pilotto et al., "Prevention of HIV-1 infection with early antiretroviral therapy," *New England journal of medicine*, 2011, *365* (6), 493–505.

Devanur, Nikhil R, Christos H Papadimitriou, Amin Saberi, and Vijay V Vazirani, "Market equilibrium via a primal-dual-type algorithm," in "Foundations of Computer Science, 2002. Proceedings. The 43rd Annual IEEE Symposium on" IEEE 2002, pp. 389–395.

Eaves, B Curtis and Herbert Scarf, "The solution of systems of piecewise linear equations," *Mathematics of Operations Research*, 1976, *1* (1), 1–27.

Glover, Fred, "Future paths for integer programming and links to artificial intelligence," *Computers & operations research*, 1986, *13* (5), 533–549.

Haushofer, Johannes and Jeremy Shapiro, "The short-term impact of unconditional cash transfers to the poor: Experimental Evidence from Kenya," *The Quarterly Journal of Economics*, 2016, *131* (4), 1973–2042.

Kremer, Michael, Jessica Leino, Edward Miguel, and Alix Peterson Zwane, "Spring cleaning: Rural water impacts, valuation, and property rights institutions," *The Quarterly Journal of Economics*, 2011, *126* (1), 145–205.

Narita, Yusuke, "Roll of the Dice with a Human Touch: Injecting Ethics into Randomized Experiments," *Work in Progress*, Nov 2017.

Papadimitriou, Christos, "Algorithms, complexity, and the sciences," *Proceedings of the National Academy of Sciences*, 2014, *111* (45), 15881–15887.

Papadimitriou, Christos H, "On the complexity of the parity argument and other inefficient proofs of existence," *Journal of Computer and system Sciences*, 1994, *48* (3), 498–532.

Scarf, Herbert E, "The computation of equilibrium prices: an exposition," *Handbook of mathematical economics*, 1982, *2*, 1007–1061.

_ **and Terje Hansen**, *The computation of economic equilibria* number 24, Yale University Press, 1973.

Turing, Alan Mathison, "On computable numbers, with an application to the Entscheidungsproblem," *Proceedings of the London mathematical society*, 1937, *2* (1), 230–265.

Walras, Leon, *Elements of Pure Economics Or The Theory of Social Wealth:(A Transl. of the Éd. Définitive, 1926, of the Eléments D'économie Politique Pure, Annotated and Collated with the Previous Editions)*, George Allen & Unwin Limited, 1954.

Appendix

A Python Implementation of Algorithm

Below I have attached a complete, working implementation of the algorithm using python that runs in a Jupyter Notebook. The code is also freely available on GitHub at `https://github.com/devanshtandon/EXaM_algorithm`. Additional code used to interface with the data pipeline is available on request.

In [ ]:
```
# import libraries
from scipy.optimize import linprog
from math import sqrt
from collections import defaultdict
import numpy as np
import random
import copy
import math
import matplotlib.pyplot as plt
import timeit
import pandas as pd
from decimal import Decimal
from scipy import stats
import time
random.seed(42)
```

In [ ]:
```
# initialize constants

# hardcoded constants for # of subjects, treatments, capacity, budget
num_subjects = 1540 # i
num_treatments = 2 # t
capacity_matrix = [663, 877]
budget = 100
epsilon = 0.1 # has to be less than 0.5
rct_treatment_probabilities = [capacity_matrix[0]*1.0/num_subjects, capa
city_matrix[1]*1.0/num_subjects]
budget_matrix = [budget] * num_subjects

# Scaling factor for alpha, beta to set new prices -- hyperparameters
beta_scaling_factor = budget/50
```

In [ ]:
```
# Initialize alpha, beta assumed to be positive
def init_alpha():
    alpha = np.asarray([random.randint(-budget, 0) for i in range(num_tr
eatments)])
    return alpha
def init_beta():
    beta = np.asarray([random.randint(-budget, budget) for i in range(nu
m_treatments)])
    return beta
```

In [ ]:
```
# Price vector pi(i,t) = alpha(t) * pte(i,t) + beta(t). Dimensions num_s
ubjects * num_treatments
def get_price_matrix(alpha, beta):
    price_matrix = [[(alpha[index] * pte_t + beta[index]) for index, pte
_t in enumerate(pte)] for pte in pte_matrix]
    price_matrix = np.asarray(price_matrix)
    return price_matrix
```

```
In [ ]: # Demand p*(i,t) matrix. Solve LP to get values. Dimensions num_subjects
         * num_treatments
        def get_demand_matrix(price_matrix):
            x0_bounds = (0,1)
            x1_bounds = (0,1)

            # dummy first row necessary for linprog package
            demand_matrix = np.ndarray((num_subjects,num_treatments), float)
            for i in range(num_subjects):
                # Constraints:
                # 1. <p*(i), pi(i)> <= b(i) for every subject i
                # 2. sum of all p*(t) = prob_threshold for every subject i
                coefficients = price_matrix[i]
                thresholds = budget_matrix[i]

                result = linprog(c=-wtp_matrix[i],
                                 A_ub = [[coefficients[0], coefficients[1]]],
                                 b_ub = thresholds,
                                 A_eq = [[1,1]],
                                 b_eq = 1,
                                 bounds = (x0_bounds, x1_bounds))
                demand_matrix[i] = result.x

            # change the type to matrix
            demand_matrix = np.asmatrix(demand_matrix)
            # compute the bounded probability (as in slide page 7)
            min_prob0 = float(min(demand_matrix[:,0]))
            max_prob0 = float(max(demand_matrix[:,0]))
            q1 = (epsilon - min_prob0) / (rct_treatment_probabilities[0] - min_p
        rob0)
            q2 = (1 - epsilon - max_prob0) / (rct_treatment_probabilities[0] - m
        ax_prob0)
            min_prob1 = float(min(demand_matrix[:,1]))
            max_prob1 = float(max(demand_matrix[:,1]))
            q3 = (epsilon - min_prob1) / (rct_treatment_probabilities[1] - min_p
        rob1)
            q4 = (1 - epsilon - max_prob1) / (rct_treatment_probabilities[1] - m
        ax_prob1)
            q = max(q1,q2,q3,q4)
            demand_matrix[:,0] = (1-q) * demand_matrix[:,0] + q * rct_treatment_
        probabilities[0]
            demand_matrix[:,1] = (1-q) * demand_matrix[:,1] + q * rct_treatment_
        probabilities[1]
            demand_matrix = np.asarray(demand_matrix)

            return demand_matrix
```

```
In [ ]: # Treatment_demand(t) = sum of demand(t) across all i. Dimensions 1 * nu
        m_treatments
        def get_treatment_demand_matrix(demand_matrix):
            treatment_demand_matrix = np.zeros(num_treatments)
            for subject in range(num_subjects):
                for treatment in range(num_treatments):
                    treatment_demand_matrix[treatment] += demand_matrix[subject,
         treatment]
            return treatment_demand_matrix
```

```
In [ ]: # Excess_demand(t) = treatment_demand(t) - capacity(t). Dimensions 1 * n
        um_treatments
        def get_excess_demand_matrix(treatment_demand_matrix):
            excess_demand_matrix = treatment_demand_matrix - capacity_matrix
            return excess_demand_matrix

        # Clearing error in market = sqrt(sum of excess_demand(t)^2 for every tr
        eatment t)
        def get_clearing_error(excess_demand_matrix):
            # If demand is satisfied everywhere and total capacity > number of s
        ubjects, no clearing error
            if all(excess <= 0 for excess in excess_demand_matrix):
                print "get_clearing_error: Market clear, no clearing error!"
                return 0
            else:
                clearing_error = sqrt(sum([excess**2 for excess in excess_demand
        _matrix]))
                clearing_error = clearing_error / sum(capacity_matrix)
                print "get_clearing_error: Clearing error:", clearing_error
                return clearing_error
```

```
In [ ]: # Recalibrate alpha, beta values to set new prices
        def get_alpha_new(alpha, excess_demand_matrix):
            alpha_new = alpha
            return alpha_new

        def get_beta_new(beta, excess_demand_matrix):
            beta_new = beta + excess_demand_matrix * beta_scaling_factor
            return beta_new
```

In [ ]:
```
# Find market clearing price vector. The objective is to change alpha an
d beta values so that we reduce clearing error
def clear_market():

    # Initialize market prices and demand
    alpha = init_alpha()
    beta = init_beta()
    price_matrix = get_price_matrix(alpha, beta)
    demand_matrix = get_demand_matrix(price_matrix)
    excess_demand_matrix = get_excess_demand_matrix(get_treatment_demand
_matrix(demand_matrix))
    clearing_error = get_clearing_error(excess_demand_matrix)

    # clearing error is percentage of total capacity so we want the mark
et to clear at 1%
    clearing_error_threshold = 0.01
    threshold_iterations = 10
    iterations = 0
    minimum_clearing_error = clearing_error
    alpha_star = 0
    beta_star = 0

    # Set new prices to clear market
    while True:
        if iterations > threshold_iterations:
            # new search start
            alpha = init_alpha()
            beta = init_beta()
            iterations = 0
            print "new search start"
        else:
            # continue down current search
            alpha = get_alpha_new(alpha, excess_demand_matrix)
            beta = get_beta_new(beta, excess_demand_matrix)

        price_matrix = get_price_matrix(alpha, beta)
        demand_matrix = get_demand_matrix(price_matrix)
        excess_demand_matrix = get_excess_demand_matrix(get_treatment_de
mand_matrix(demand_matrix))
        clearing_error = get_clearing_error(excess_demand_matrix)

        # Store parameter values for minimum clearing error
        if clearing_error < minimum_clearing_error:
            minimum_clearing_error = clearing_error
            alpha_star = alpha.copy()
            beta_star = beta.copy()
        # cleared the market!
        if minimum_clearing_error < clearing_error_threshold:
            break
        iterations += 1

    print "Minimum clearing error:", minimum_clearing_error
    print "Alpha_star:", alpha_star
    print "Beta star:", beta_star
    return (minimum_clearing_error, alpha_star, beta_star)
```