ESSAYS IN ECONOMICS

by

Vitor Baisi Hadad

Submitted to the Faculty of the Department of Economics
in partial fulfillment of the requirements for the
degree of Doctor of Philosophy

Boston College
Morrissey College of Arts and Sciences
Graduate school

Abstract

Acknowledgements

Table of Contents

List of Figures

List of Tables

List of Abbreviations

Part I

Heterogeneous

Chapter 1

Heterogenous Production Functions, Panel Data, and Productivity Dispersion

1 Introduction

[Primary motivations of panel data: controlling for unobserved heterogeneity]

[Correlated random coefficients]

[] Random coefficients Consumer panels with heterogeneous consumers Production function parameters (A,B) vary across firms in same industry

Random coefficients (A,B) can be correlated with inputs (X1,X2) Estimator allows arbitrary correlations of production function parameters (A1,B1) with inputs (X1,X2) Timing assumptions with random coefficients (A,B) Input decisions made in period t 1 with knowledge of t 1 production function Estimation, consistency: censoring for unconditional means India: log TFP dispersion increases in random coefficients vs OLS

2 Literature review

In its most general form, the basic linear panel model used in econometrics can be described as

$$Y_{it} = A_{it} + \sum_{k=1}^{K} X_{it}^{(k)} B_{it}^{(k)} \tag{1.1}$$

where $Y_{it}, X_{it}, \{A_{it}, B_{it}\}_{k=1}^{K}$ are random variables associated with individual i at period t. The parameters $B_{it}^{(k)}$ represent the causal effect of an increase in their associated regressor, while A_{it} is a disturbance term whose interpretation depends on the specific application. When they have a nondegerate probability distribution, they receive various names in the literature, such as "*unobserved component*, *latent variable*, or *unobserved heterogeneity*"(Wooldridge, 2010, p.251).

but here we call them *random coefficients*.

Without further restrictions, nothing can be said about the objects A_{it} and B_{it}, or even its moments. In the linear panel data literature, common identifying assumptions include the following three.

1. $A_{it} = a + \epsilon_{it}$ and $B_{it} = b$, where a, b are constants and $E[\epsilon_{it}|X_{it}] = 0 \forall t$.
2. $A_{it} = A_i + \epsilon_{it}$ and $B_{it} = b$, where b is a constant and $E[\epsilon_{it}|X_{is}] = 0 \forall t, s$.
3. $A_{it} = A_i + \epsilon_{it}$ and $B_{it} = b$, where b is a constant and $E[\epsilon_{it}|X_{is}] \neq 0 \forall t, s$.

The first one is the standard linear model that can be consistently estimated by simply running a standard linear regression of Y_{it} on X_{it}. The second and third are called *random* and *fixed* panel data models, and have received a great deal of attention in the literature (see [???] for a review). Figure 1 illustrates the difference between them.

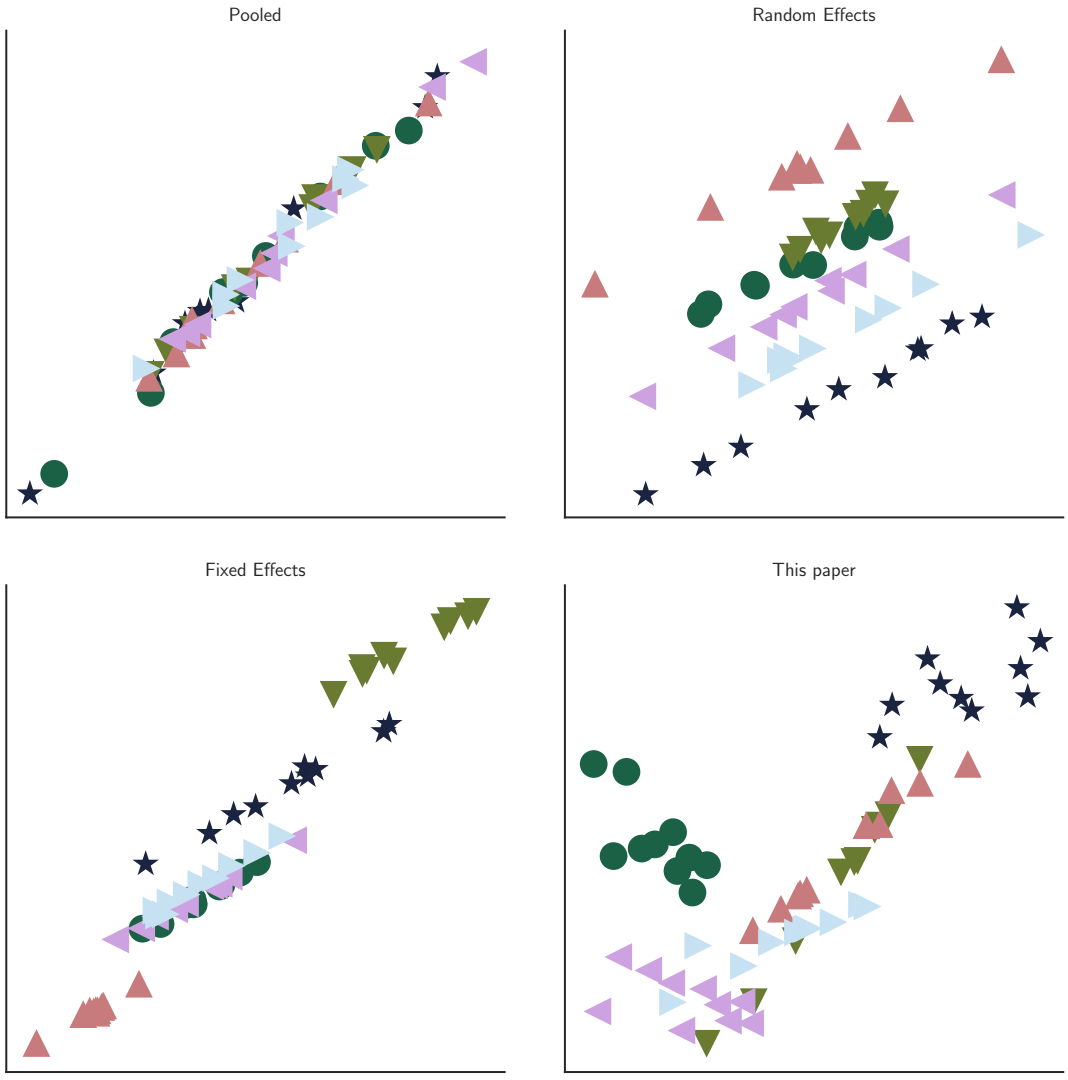


Figure 1: **Linear panel data models**
Different assumptions leading to different panel data models. Each different marker type and color indicates a different individual. *Pooled*: all observations share same intercept and slope; *Random effects*: each observation possesses their own time-invariant slope, but all share the same slope; *Fixed effects*: same as random effects, but intercept and regressors can be correlated (figure shows positive correlation); *FHHPS*: our paper generalizes the model by 1) allowing for intercept and slope to be correlated with regressors, and 2) allowing both intercept and slope to drift according to a specific Markovian process.

In this paper we focus on the more general case where we allow for more than one random coefficient, and also allow them to be correlated with the regressor. The literature dealing with this more general case is substantially smaller. Hoderlein et al. (2010) analyzed a for any number of random coefficients, but required the stronger assumption that the coefficients are independent from the regressors.

¿ [Chamberlain (1982), Arellano and Bonhomme (2011), Graham and Powell (2012), Evdokimov (2011)]

¿ Recent papers on heterogeneous production functions Kasahara, Shrimpf and Suzuki (2016), Balat, Brambilla and Sazaki (2016) Extend proxy variable approaches but require variables to be chosen in period t 1 with only static considerations, as in FOCs. Ackerberg and Hahn (2016) Scalar unobservable enters into nonparametric production function

Cobb-Douglas Yi,t =Ai,t +BKKi,t +BLLi,t Productivity dispersion: standard deviation of Ai,t across plants or firms in same industry Syverson (2011) surveys empirical findings on dispersion Typical finding: some plants produce more than twice as much output for same inputs Assuming BK,BL is the same across plants in a 4- or 2-digit industry classification may be too restrictive If BK,BL varybyplanti andtimet andabove i,t i,t specification is estimated, much of this heterogeneity enters into Ai,t

Estimate BK as expenditure share on capital input i,t Based on static profit maximizing FOC for Cobb-Douglas Treats input as having no adjustment costs, opposite of our approach Based on strong conduct assumption: static profit maximization We do not impose profit maximization, important for studying unproductive firms We allow unobservables outside production function to affect input choice Firm-specific input prices Adjustment costs Product demand

3 Model and assumptions

In order to reduce clutter we will drop the i subscript. For $1 \leq t \leq T$, define

$$Y_t = A_t + B_t X_t$$

where X_t is a scalar, and A_t, B_t are scalar random coefficients that evolve according to an AR(1) Markov process.

$$A_t = A_{t-1} + U_t \tag{1.2}$$

$$B_t = B_{t-1} + V_t \tag{1.3}$$

We will make the following sufficient assumptions for identifiability. Some of these are known to be stronger than necessary, and we will mention how to relax them later on. For ease of notation, let us denote $W_t = [X_t, X_{t-1}]^T$

Assumption 3.1. *Conditional on recent covariates W_t, contemporaneous shocks are independent from past shocks.*

$$\textit{For } 2 \leq t \leq T, \quad (U_t, V_t) \perp \{U_s, V_s\}_{s=0}^{t-1} \mid W_t \tag{1.4}$$

Assumption 3.2. *Shocks are sequentially independent from all covariates*

$$\forall t, \qquad U_t, V_t \perp (X_1, \cdots, X_T) \tag{1.5}$$

Assumption 3.3. *The covariate associated with the random coefficient is continuously distributed and is supported on the entire real line for all periods. Furthermore,*

their joint distribution is non-degenerate.

$$supp(X_1) = \cdots = supp(X_T) = \mathbb{R} \qquad and \qquad \forall s \neq t, \quad P(X_s = X_t) = 0 \tag{1.6}$$

Assumption 3.4. *Nonsingularity of the covariate matrix.*

$$For\ 2 \leq t \leq T, \quad E[W_t W_t^T] is\ nonsingular \tag{1.7}$$

We should note that assumptions 3.1 and 4.3 allow for future regressors to be correlated with past shocks. In the context of firm profit maximization, this means that a firm is allowed to choose future inputs based on their previous productivity. In a different context, they would also allow for a dynamic model with $X_t = Y_{t-1}$. Crucially, we have not made any assumptions about contemporaneous independence between random coefficients and regressors, thus allowing for a unobserved heterogeneity of a general type.

Assumption 3.3 is a simplifying "no-stayer" assumption that will facilitate some of the asymptotic derivations below. Assumption 3.4 is a common technical assumption often used for generic identification in linear models.

4 Identifying random coefficient moments

In this section, we show how to identify the r^{th} moments of the random coefficients (A_t, B_t). Broadly, our identification strategy will follow the next steps.

1. Identify shock moments $E[U_t^\ell V_t^{r-\ell}]$
2. Express the conditional moments of the regressand $E[Y_1^\ell Y_2^{r-\ell}|W_{2i}]$ as a function of conditional moments of random coefficients $E[A_t^\ell B_t^{r-\ell}|W_2]$ and shocks $E[U_t^\ell V_t^{r-\ell}]$

3. Invert the above so as to represent conditional moments of random coefficients as a function of conditional moments of the regressand and shocks $E[U_t^\ell V_t^{r-\ell}|W_2]$

4. Integrate out to get unconditional moments $E[A_t^\ell B_t^{r-\ell}]$

4.1 First period moments

Let us start with the first moment ($r = 1$). In this case, we are required to have $T \geq 2$ consecutive waves of panel data.

We begin by simply reminding ourselves of the system of equations for these two periods.

$$\begin{cases} Y_1 = A_1 + B_1 X_1 \\ Y_2 = A_2 + B_2 X_2 \end{cases} \tag{1.8}$$

Let's expand the coefficients according to our model equations **??**.

$$\begin{cases} Y_1 = A_1 + B_1 X_1 \\ Y_2 = A_1 + U_2 + B_1 X_2 + V_2 X_t \end{cases} \tag{1.9}$$

Step 1 In order to identify shocks, we consider the expectation of the difference between the two periods, conditioning on some point $W_2 = w_2 = (x, x)$.

$$E[Y_2 - Y_1 \mid W_2 = w_2] = E[U_2] + E[V_2]x \tag{1.10}$$

The expression above contains two caveats. First, that thanks our full support assumption 3.3, we can choose any point on $\mathbb{R}^2$ to condition on, including a point where $X_1 = X_2 = x$ as we just did. Second, that Assumption 3.1 allowed us to drop the conditioning for the shocks.

Now, note that the moments $E[U_2]$ and $E[V_2]$ on the right-hand side are solutions to this minimization problem

$$\min_{\theta_u,\theta_v} E[(Y_2 - Y_1 - \theta_u - x\theta_v)^2 \mid W_2 = w_2] \tag{1.11}$$

and that the solution is unique, by virtue of the strict convexity of the quadratic function and the full rank assumption **??**. It follows that the shock moments $E[U_2]$ and $E[V_2]$ are generically identified, and so are β_1 and β_2.

Step 2 For this step, we simply have to take expectations of 1.9, now conditioning on $W_2 = w_2 = (x_1, x_2)$. (Note we have dropped the constraint $X_1 = X_2$).

$$\begin{cases} E[Y_1|W_2] = E[A_1|W_2] + E[B_1|W_2]x_1 \\ E[Y_2|W_2] = E[A_1|W_2] + E[U_2] + E[B_1 \mid W_2]x_2 + E[V_2]x_2 \end{cases} \tag{1.12}$$

Step 3 Writing the previous equation in matrices after a little rearranging, we get:

$$\begin{bmatrix} E[Y_1|W_2] \\ E[Y_2|W_2] - E[U_2] - E[V_2]x_t \end{bmatrix} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \end{bmatrix} \begin{bmatrix} E[A_1|W_2] \\ E[B_1|W_2] \end{bmatrix} \tag{1.13}$$

As long as $x_1 \neq x_2$, we can invert the first matrix on the right-hand side.

$$\begin{bmatrix} E[A_1|W_2] \\ E[B_1|W_2] \end{bmatrix} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \end{bmatrix}^{-1} \begin{bmatrix} E[Y_1|W_2] \\ E[Y_2|W_2] - E[U_2] - E[V_2]x_2 \end{bmatrix} \tag{1.14}$$

since all the quantities on the right-hand side are known, we have identified all the conditional first moment of our random coefficients outside the line $X_1 = X_2$. Since our "no-stayer" assumption 3.3 guarantees that a randomly drawn point from the joint distribution of X_1 and X_2 will not be on the diagonal, we have generic identification.

Conditional second-period moments are also automatically identified, since due to

our 4.3 assumption we have that shocks are mean-independent from contemporaneous regressors, which allows us to write

$$E[A_2|W_2 = w_2] = E[A_1|W_2 = w_2] + E[U_2] \tag{1.15}$$

$$E[B_2|W_2 = w_2] = E[B_1|W_2 = w_2] + E[V_2] \tag{1.16}$$

$$\tag{1.17}$$

Step 4 The final step is simply to integrate out over the distribution of W_2 to get unconditional moments.

4.2 Second period moments

The identification procedure is analogous, but this time we use the square of our model equations.

Step 1 Identify shock second moments on the diagonal $W_2 = w_2 = (x, x)$ using the following equation

$$E[(Y_2 - Y_1)^2 \mid W_2 = w_2] = E[U_2^2] + E[V_2^2]x^2 + 2E[U_2V_2]x \tag{1.18}$$

The analogous minimization procedure is

$$E[U_2^2], E[V_2^2], E[U_2V_2] = \min_{\theta_{uu},\theta_{vv},\theta_{uv}} E[(Y_2 - Y_1 - \theta_{uu} - x^2\theta_{vv} - 2x\theta_{uv})^2 \mid W_2 = w_2] \tag{1.19}$$

point identification is once more possible because the solution to this minimization problem is unique.

Steps 2-3 Again drop the constraint $X_1 = X_2$, and conditioning on $W_2 = w_2 = (x_1, x_2)$ consider the expectation of the square of our model equations.

$$\begin{bmatrix} E[A_1^2|x_1,x_2] \\ E[B_1^2|x_1,x_2] \\ E[A_1B_1|x_1,x_2] \end{bmatrix} = \begin{bmatrix} 1 & x_1^2 & 2x_1 \\ 1 & x_2^2 & 2x_2 \\ 1 & x_1x_2 & x_1+x_2 \end{bmatrix}^{-1} \begin{bmatrix} E[Y_1^2|x_1,x_2] \\ E[Y_2^2|x_1,x_2] - C_{1i} \\ E[Y_1Y_2|x_1,x_2] - C_{2i} \end{bmatrix} \tag{1.20}$$

where C_{1i} and C_{2i} involve only quantities that were already estimated in previous steps.

$$C_{1i} = E[U^2] + 2E[U_2V_2]x_2 + E[V_2^2]x_2^2 \tag{1.21}$$

$$- 2\{E[A_1|x_1,x_2] + E[B_1|x_1,x_2]x_2\}\{E[U_2] + E[V_2]x_2\} \tag{1.22}$$

$$C_{2i} = \{E[A_1|x_1,x_2] + E[B_1|x_1,x_2]x_1\}\{E[U_2] + E[V_2]x_2\} \tag{1.23}$$

Note that the relevant matrix above is invertible whenever $x_1 \neq x_2$.

Step 4 Analogous to step 4 in the first moment case.

4.3 Identifying further moments

The argument above generalizes for arbitrary moments. In fact, if we are only after moments a finite number of moments of the distribution, then assumption 4.3 as follows. Let r be a positive integer and for $t \geq 2$ define $W_t = (X_{t-1}, X_t)$. We will say that (U_t, V_t) are r^{th}-*moment independent* of W_t if

$$\text{For } 1 \leq j \leq r, \quad E[U_t^{r-j}V_t^j|W_t] = E[U_t^{r-j}V_t^j] \tag{1.24}$$

We can replace assumption 4.3 by the following weaker assumption.

Assumption 3.1' Shocks are r$^{\text{th}}$-moment independent from all covariates

$$\text{For } 1 \leq j \leq r, \quad E[U_t^{r-j} V_t^j | W_t] = E[U_t^{r-j} V_t^j] \tag{1.25}$$

4.4 Extension: fixed coefficients

Our model can be augmented with an arbitrary number of regressors associated with coefficients that are individual-invariant – but not necessarily time-invariant. The model equations become

$$Y_t = A_t + B_t X_t + Z_t^T \beta_t \qquad \beta_t : \text{fixed}$$

Identification of the fixed coefficients happens on Step 1, when the minimization problem becomes

$$E[U_2], E[V_2], \beta_1, \beta_2 = \min_{\theta_u, \theta_v, \theta_{b_1}, \theta_{b_2}} E[(Y_2 - Y_1 - a - bx_2 + z_2^T \theta_{b_1} - z_1^T \theta_{b_2})^2 | W_2 = w_2] \tag{1.26}$$

and the argument follows for the same reasons that were previously explained. Once we have identified β_1, β_2, we remove them from future computations using the modified regressand $\tilde{Y}_t := Y_t - Z_t \beta_t$, and the rest of the process delineated above goes through without further change.

4.5 Identifying the marginal distribution of random coefficients

In this section we will prove that we can identify the marginal density of random coefficients $f_{A_t B_t}$. We begin by strengthening the

5 Estimation

Our estimation procedure follows three steps:

1. Use local polynomial regression to estimate shock moments and individual-invariant coefficients

2. Use any nonparametric estimator to retrieve estimates of random coefficient moments conditional on observables

3. Average out conditional random moments by averaging out conditional moments, taking care to avoid observations for which $X_1 \approx X_2$.

Let us consider each one of these steps in detail.

Estimating shocks

First moments Regress ΔY_2 on X_2 locally at the diagonal $X_1 \approx X_2$:

$$(\widehat{E}[U_2], \widehat{E}[V_2],\ \cdot\ ) = \arg\min_{\theta_U, \theta_V, \gamma} \sum_i K_h(\Delta X_{2i}) \cdot (\Delta Y_{2i} - \theta_U - X_{2i}\theta_V - g_1(X_{2i}, \Delta X_{2i}; \gamma)^2 \tag{1.27}$$

where $K_h(\cdot)$ is a standard kernel with asymptotically zero bandwidth:

$$h_{shocks}(n) = c_{shocks} n^{\alpha_{shocks}} \qquad \text{where } \lim_{n\to\infty} h_n = 0 \tag{1.28}$$

and g_1 is a K-order polynomial in X_2 and ΔX_2 with coefficients γ. In theory its inclusion is optional, but we have observed that its presence substantially improves the quality of the estimates.

If our model included time-invariant coefficients, they would also be estimated in this step.

Second moments We estimate the uncentered moments by proceeding similarly to above:

$$(\widehat{E}[U_2^2], \widehat{E}[U_2V_2]\widehat{E}[V_2^2],\ \cdot\ ) = \tag{1.29}$$

$$\arg \min_{\theta_{U_2}, \theta_{UV}, \theta_{V_2}, \gamma} \sum_i K_h(\Delta X_{2i}) \cdot \left(\Delta Y_{2i}^2 - \theta_{U_2} - 2X_{2i}\theta_{UV} - X_{2i}^2\theta_{V_2} - g_2(X_{2i}, \Delta X_{2i}; \gamma)\right)^2 \tag{1.30}$$

Naturally, once we have uncentered second moments and first moments, we can compute estimates of centered moments using the usual formulas for centered moments, e.g. $\widehat{Var}[U_2] = \widehat{E}[U_2^2] - \widehat{E}[U_2]^2$.

Conditional random coefficient moments

Conditional moments of Y_1 and Y_2 Here we begin by estimating $E[Y_1^m Y_2^n | X_1 = x_1, X_2 = x_2]$ for $m, n \in \{(1,0), (0,1), (1,1), (2,0), (0,2)\}$. This can be done using any nonparametric estimator, but we choose the Nadaraya-Watson for simplicity.

$$\widehat{E}[Y_1 | X_1 = X_1, X_2 = X_2] = \frac{\sum_i Y_{1i} K_h(X_{1i} - x_1) K_h(X_{2i} - x_2)}{\sum_i K_h(X_{1i} - x_1) K_h(X_{2i} - x_2)} \tag{1.31}$$

where K_h is again a standard kernel endowed with asymptotically vanishing bandwidth.

$$h_{nw}(n) = c_{nw} \hat{\sigma} n^{-\alpha_{nw}} \tag{1.32}$$

where $\hat{\sigma}$ is an estimate of $Std(X_1) = Std(X_2)$. Note that for the theoretical MISE-minimizing bandwidth, one should set $\alpha_{nw} = \frac{1}{6}$.

Solving for first moments Once in possession of all the previous estimates, we can solve for estimates of conditional random coefficient moments using the empirical

analogs of equations ?? and ??.

Unconditional random coefficient moments

In this final step, we must exclude observations near the diagonal X_1X_2, and then average the conditional moments associated with the remaining observations. According to our asymptotic results, included observations must satisfy the following conditions:

$$|\Delta X_{2i}| > c_{cens1}\hat{\sigma}_{X_2}n^{-\alpha_{cens1}} \quad \text{for first moments} \tag{1.33}$$

$$|\Delta X_{2i}| > c_{cens1}\hat{\sigma}_{X_2}n^{-\alpha_{cens2}} \quad \text{for second moments} \tag{1.34}$$

As shown in section ??, the optimal thresholds can be shown to be associated with exponents $\alpha_{cens1} = \frac{1}{4}$ and $\alpha_{cens2} = \frac{1}{8}$. However, theory gives us no guidance regarding the choice of constants c_{cens1} and c_{cens2}, so we resort to experimentation in the next section.

6 Simulations

The purpose of this sections is to provide a simulation study in a setting that is similar to our empirical application. We begin by generating the first period random coefficients (A_1, B_1), their second-period shocks (U_2, V_2), and regressors for both periods (X_1, X_2). They are drawn from a jointly Normal distribution:

$$\begin{bmatrix} A_1 \\ B_1 \\ X_1 \\ X_2 \\ U_2 \\ V_2 \end{bmatrix} \sim \mathcal{N} \left(\begin{bmatrix} 2 \\ 1 \\ 0 \\ 0 \\ .3 \\ .1 \end{bmatrix}, \begin{bmatrix} 9 & 0.95 & 1.5 & 1.5 & 0 & 0 \\ 0.95 & 0.4 & 0.32 & 0.32 & 0 & 0 \\ 1.5 & 0.32 & 1 & 0.5 & 0 & 0 \\ 1.5 & 0.32 & 0.50 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0.16 \\ 0 & 0 & 0 & 0 & 0.16 & 0.1 \end{bmatrix} \right) \tag{1.35}$$

These number were chosen to roughly reflect the characteristics that we expect them to have in a real data application. The implied correlation matrix has a very simple structure.

$$\begin{bmatrix} 1 & 0.5 & 0.5 & 0.5 & 0 & 0 \\ 0.5 & 1 & 0.5 & 0.5 & 0 & 0 \\ 0.5 & 0.5 & 1 & 0.5 & 0 & 0 \\ 0.5 & 0.5 & 0.5 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0.5 \\ 0 & 0 & 0 & 0 & 0.5 & 1 \end{bmatrix} \tag{1.36}$$

The remaining variables were created from these in the obvious manner. In order to understand how our estimator behaves as the number of observation increases, we used $n \in \{500, 2000, 5000, 10000, 20000\}$.

As seen in Section 5, our method requires the careful tuning of many different parameters. Since our methods are not amenable to cross-validation, we simulated the model in section 1 over a large grid of parameter combinations, which were then compared according to their RMSE against the true value. The RMSE-minimizing parameter configuration for each number of observations is shown on Table 1.

n	c_{shocks}	α_{shocks}	c_{nw}	α_{nw}
500	2	$\frac{1}{5}$	$\frac{1}{2}$	$\frac{1}{6}$
2000	3	$\frac{1}{5}$	$\frac{1}{2}$	$\frac{1}{6}$
5000	4	$\frac{1}{5}$	$\frac{1}{2}$	$\frac{1}{6}$
10000	4	$\frac{1}{5}$	$\frac{1}{2}$	$\frac{1}{6}$
20000	3	$\frac{1}{5}$	$\frac{1}{2}$	$\frac{1}{3}$

Table 1: RMSE-minimizing parameter configurations for our DGP.

Note that the best parameters remain fairly stable as we increase the number of observations. The most notable variation is in α_{nw}: intuitively, when the number observations is large enough, a much smaller bandwidth is preferred, and that is translated into a more aggressive choice of exponent. We will use these numbers to guide our choice of tuning parameters in our empirical application in the next section.

The performance of our estimators is illustrated on Figures **??** and **??** and on the accompanying tables on the Appendix. We observe that indeed the distribution tends to collapse to its true value as the number of observations increases.

6.1 Bootstrap coverage

Tables 2-4 show the coverage of bootstrap confidence intervals for all estimated parameters. To produce these tables, we generated 2000 datasets using different seeds, computed 500 bootstrap estimates in each of these datasets, and calculated their 0.025 and 0.975 quantiles. The numbers in each cell are the proportion of times that quantiles straddled the true value.

Table 2: Bootstrap coverage for shock moments

	$E[U_t]$	$E[V_t]$	$Std[U_t]$	$Std[V_t]$	$Cov[U_t, V_t]$	$Corr[U_t, V_t]$
t=1	0.948	0.942	0.952	0.968	0.942	0.950
t=2	0.951	0.941	0.945	0.964	0.941	0.954

Table 3: Bootstrap coverage for random coefficient conditional moments

(x_1, x_2)	(-1, 0)	(1, 0)	(0, 1)	(0, -1)
$E[A_1\|X]$	0.960	0.920	0.957	0.919
$E[B_1\|X]$	0.936	0.929	0.923	0.953
$Std[A_1\|X]$	0.920	0.950	0.945	0.932
$Std[B_1\|X]$	0.952	0.929	0.932	0.953
$Cov[A_1, B_1\|X]$	0.940	0.958	0.928	0.936
$Corr[A_1, B_1\|X]$	0.932	0.966	0.932	0.940

Table 4: Bootstrap coverage for random coefficient unconditional moments

	$E[A_t]$	$E[B_t]$	$Std[A_t]$	$Std[B_t]$	$Cov[A_t, B_t]$	$Corr[A_t, B_t]$
t=1	0.945	0.958	0.951	0.928	0.948	0.945
t=2	0.935	0.954	0.951	0.932	0.945	0.941

7 Empirical application

Now we provide an illustration of our methods by estimating a Cobb-Douglas production function with random elasticity for a large number of Indian plants. We will be using the Annual Survey of Industries (ASI) dataset, released by the Central Statistical Organization of India for the years 2008 and 2009. This yearly survey collects data from sampled Indian economic units of production (individual factories, workshops, and establishments, hereafter called "plants") that employ more than 10 regular workers. From this data we extract four variables: *gross sale value* (value of

the products sold by the plant, as purchased by their clients) S_t; *capital* (fixed assets with a productive life of more than one year) K_t; *wages* W_t; and production *materials* M_t. All variables are in 2005 rupees. We keep only observations that we present in both years of analysis, and we are left with 13298 observations. Summary statistics are in subsection 8.1 in the Appendix.

In addition to the variables above, we generate our model variables:

$$Y_t = \log\left(\frac{S_t - M_t}{W_t}\right) \qquad X_t = \log\left(\frac{K_t}{W_t}\right) \tag{1.37}$$

where Y_t represents production value-added after sales normalized by wages, and X_t represents normalized capital.

[[Interpretation]]

Table 5: Empirical application: Shock Estimates

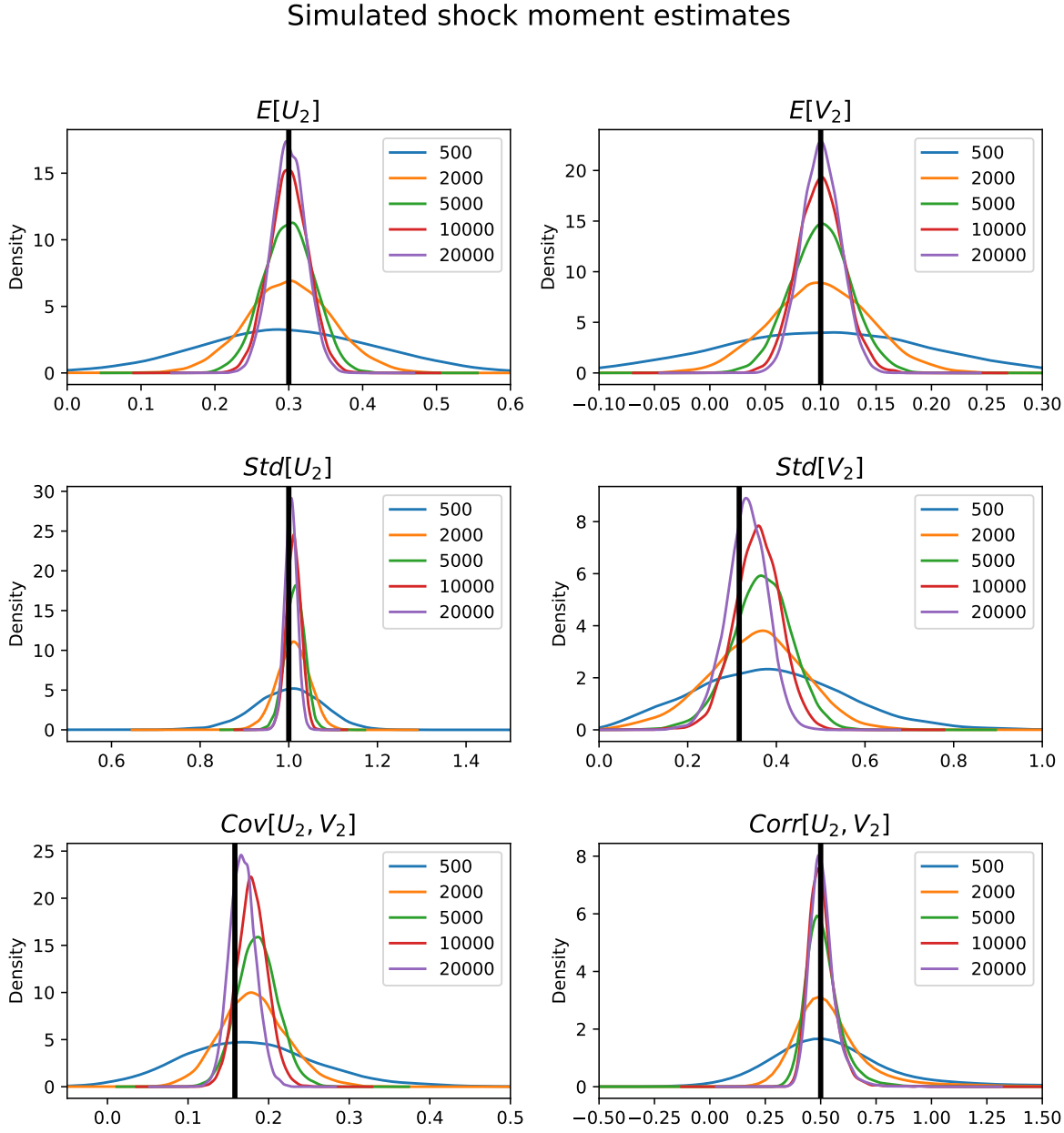
	$E[U_2]$	$E[V_2]$	$Std[U_2]$	$Std[V_2]$	$Cov[U_2, V_2]$	$Corr[U_2, V_2]$
Estimate	0.011	-0.002	0.704	0.050	0.032	0.91
Std. Error	0.010	0.005	0.016	0.018	0.005	0.539
Min	-0.014	-0.013	0.664	0.008	0.018	0.463
Lower CI (2.5%)	-0.010	-0.012	0.67	0.014	0.023	0.572
Median	0.011	-0.001	0.702	0.052	0.032	0.872
Upper CI (97.5%)	0.028	0.007	0.73	0.086	0.044	2.612
Max	0.0303	0.010	0.735	0.095	0.047	3.696

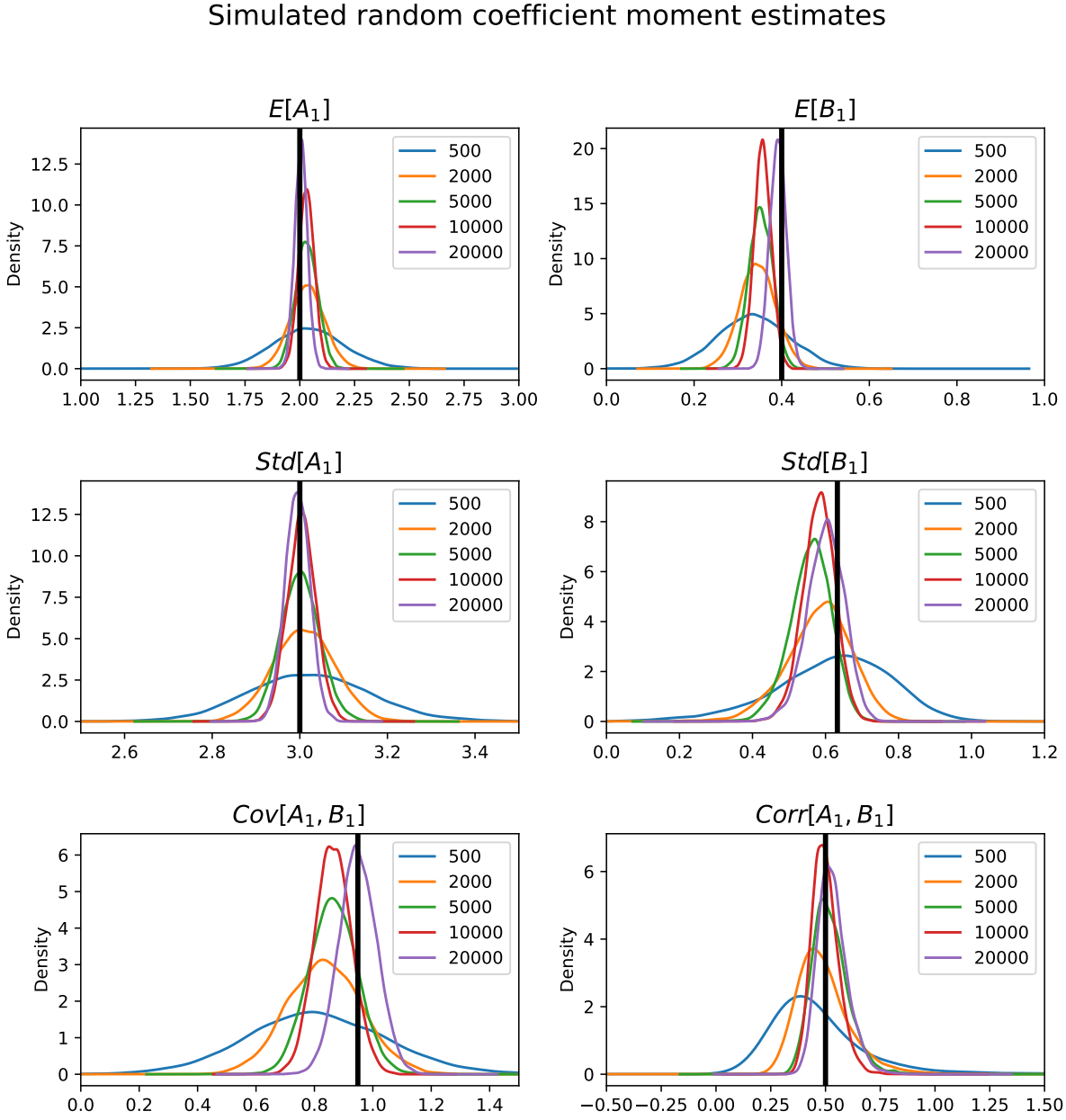
Table 6: Empirical application: Random Coefficient Estimates (first period)

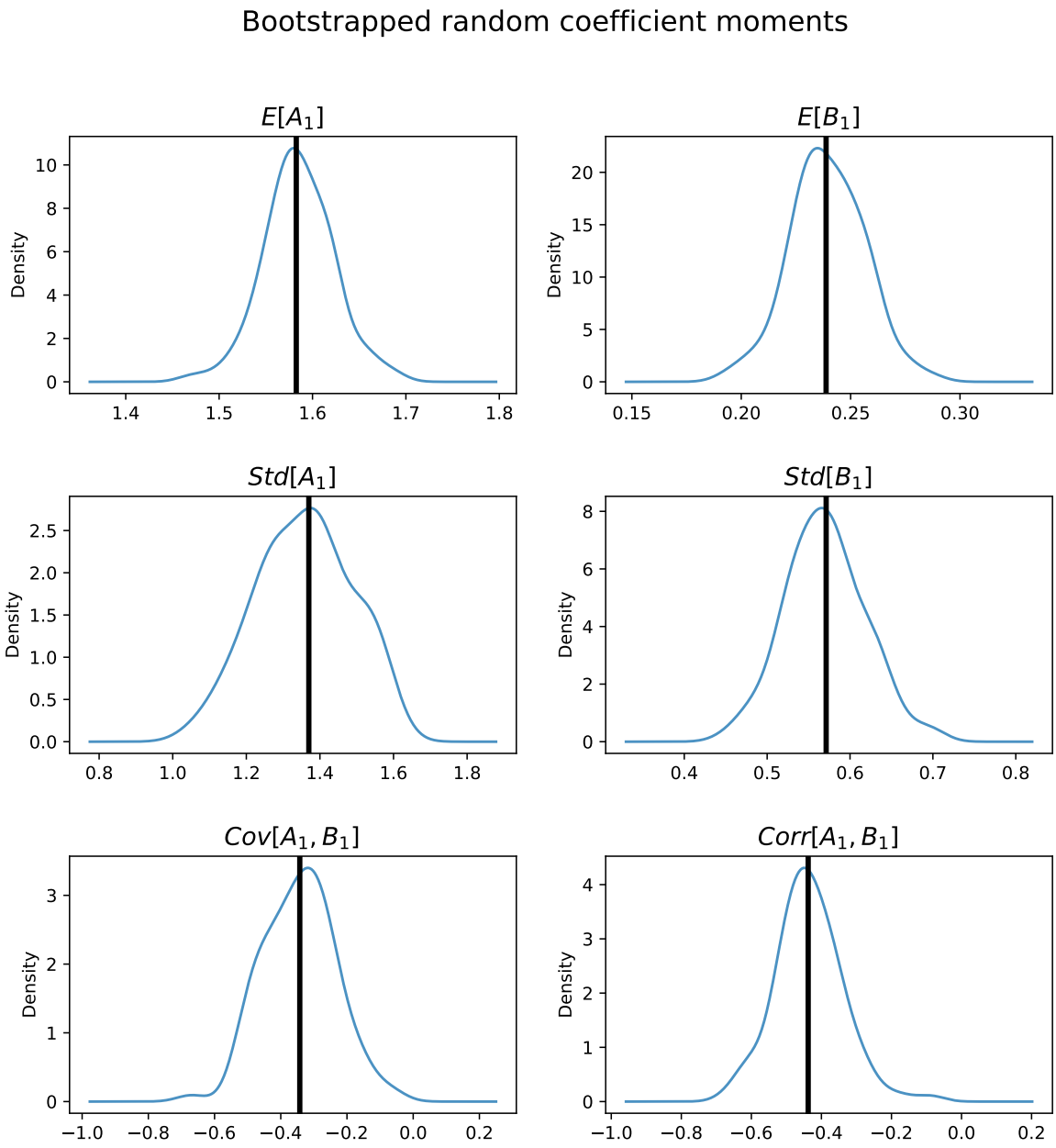
	$E[A_1]$	$E[B_1]$	$Std[A_1]$	$Std[B_1]$	$Cov[A_1, B_1]$	$Corr[A_1, B_1]$
Estimate	1.582	0.238	1.369	0.571	-0.342	-0.437
Std. Error	0.037	0.016	0.127	0.047	0.108	0.096
Min	1.470	0.193	1.050	0.452	-0.669	-0.668
Lower CI (2.5%)	1.513	0.205	1.121	0.477	-0.518	-0.615
Median	1.583	0.237	1.369	0.566	-0.339	-0.440
Upper CI (97.5%)	1.657	0.273	1.562	0.663	-0.142	-0.242
Max	1.687	0.286	1.602	0.697	-0.056	-0.088

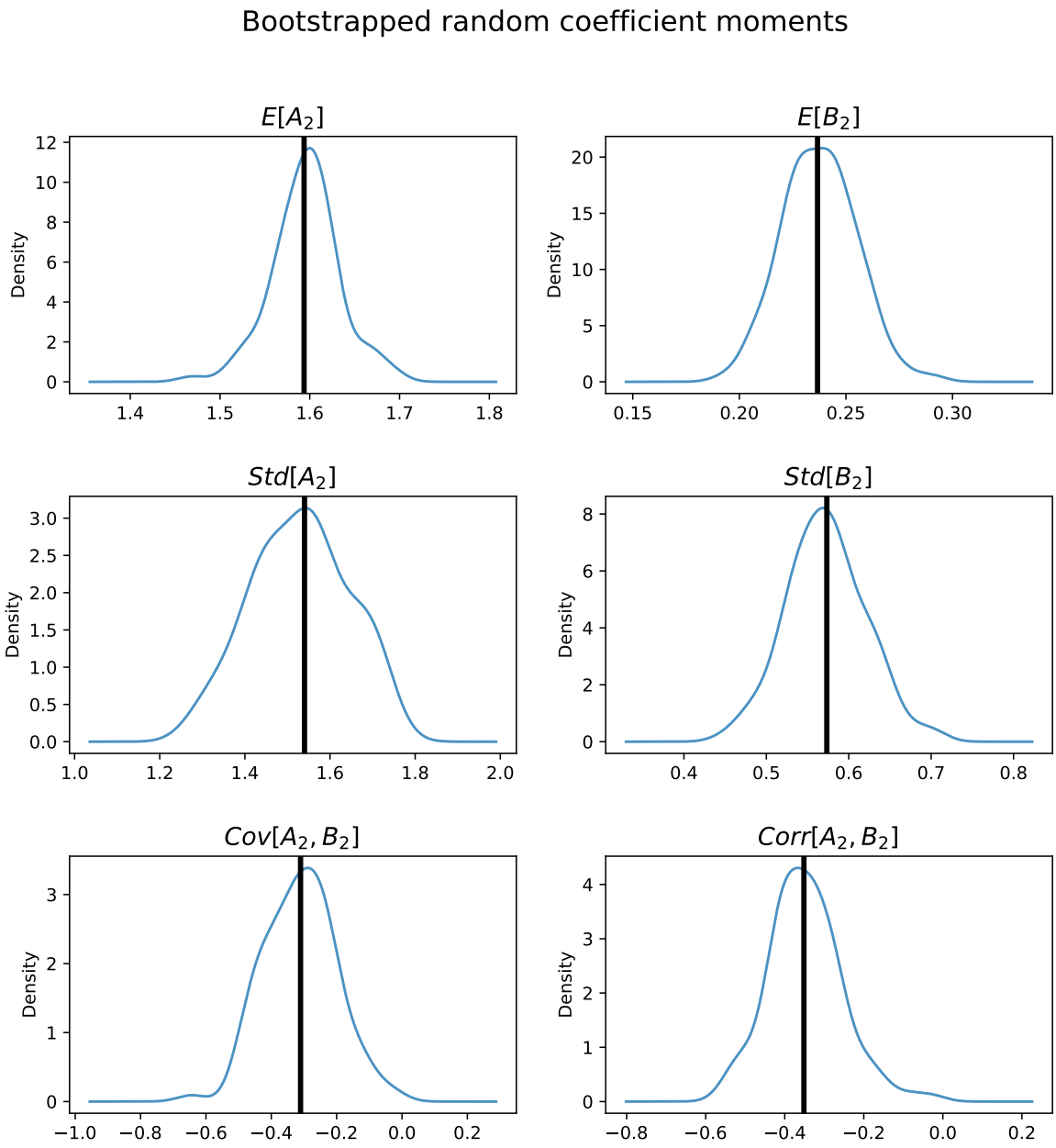
Table 7: Empirical application: Random Coefficient Estimates (second period)

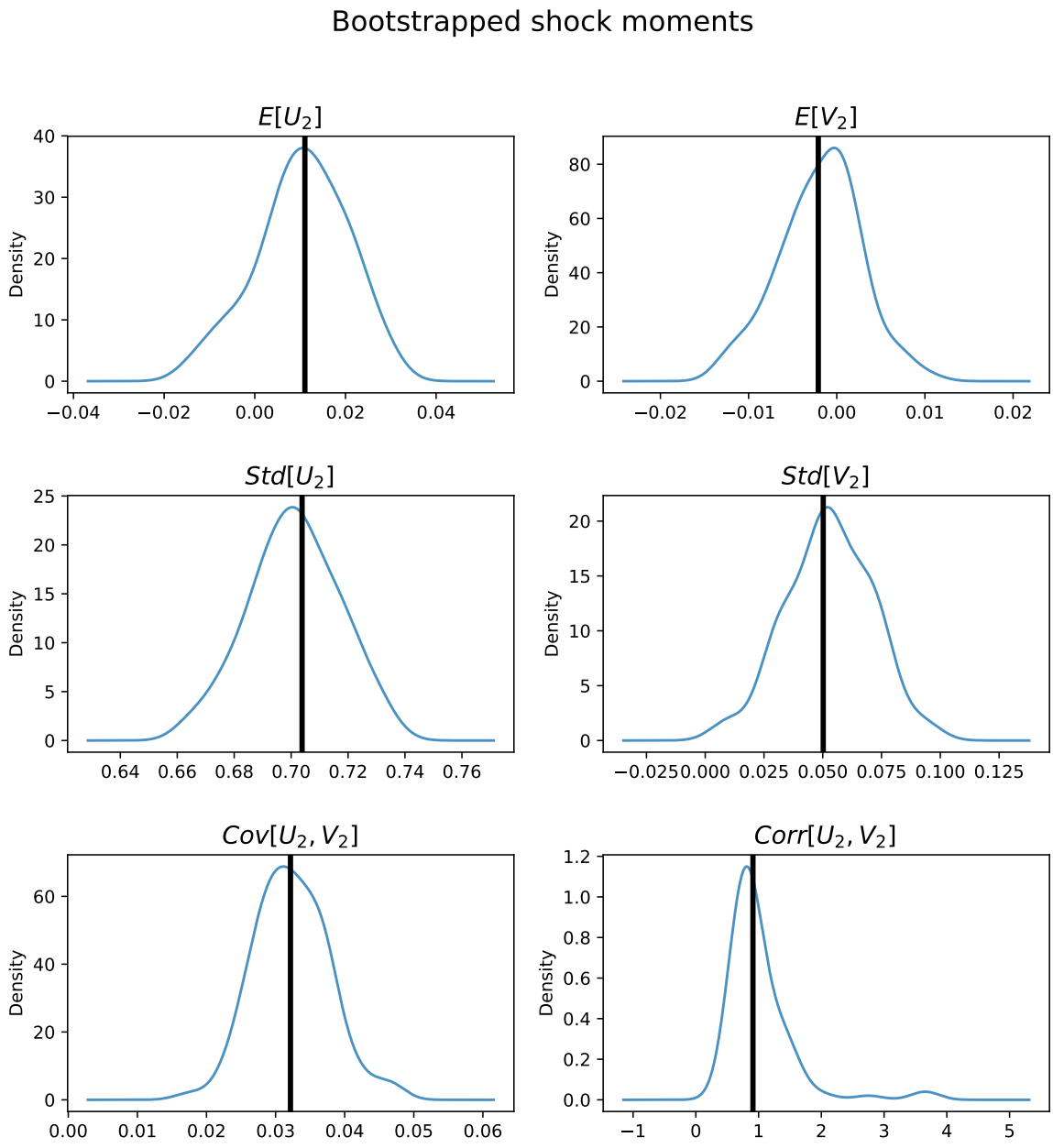
	$E[A_2]$	$E[B_2]$	$Std[A_2]$	$Std[B_2]$	$Cov[A_2, B_2]$	$Corr[A_2, B_2]$
Estimate	1.593	0.236	1.540	0.573	-0.310	-0.351
Std. Error	0.037	0.017	0.113	0.047	0.109	0.092
Min	1.468	0.194	1.274	0.452	-0.644	-0.544
Lower CI (2.5%)	1.521	0.206	1.318	0.479	-0.485	-0.514
Median	1.598	0.238	1.538	0.570	-0.308	-0.348
Upper CI (97.5%)	1.670	0.271	1.715	0.662	-0.112	-0.155
Max	1.694	0.289	1.751	0.699	-0.023	-0.001


Simulated shock moment estimates
$E[U_2]$
$E[V_2]$
$Std[U_2]$
$Std[V_2]$
$Cov[U_2, V_2]$
$Corr[U_2, V_2]$
Density
500
2000
5000
10000
20000


Simulated random coefficient moment estimates
$E[A_1]$
$E[B_1]$
$Std[A_1]$
$Std[B_1]$
$Cov[A_1, B_1]$
$Corr[A_1, B_1]$
Density
500
2000
5000
10000
20000


Bootstrapped random coefficient moments
$E[A_1]$
$E[B_1]$
$Std[A_1]$
$Std[B_1]$
$Cov[A_1, B_1]$
$Corr[A_1, B_1]$
Density


Bootstrapped random coefficient moments
$E[A_2]$
$E[B_2]$
$Std[A_2]$
$Std[B_2]$
$Cov[A_2, B_2]$
$Corr[A_2, B_2]$
Density


Bootstrapped shock moments
$E[U_2]$
$E[V_2]$
$Std[U_2]$
$Std[V_2]$
$Cov[U_2, V_2]$
$Corr[U_2, V_2]$
Density

8 Appendix

8.1 Summary statistics

Table 8: ASI survey data: summary statistics

	K_{2008}	M_{2008}	W_{2008}	S_{2008}	K_{2009}	M_{2009}	W_{2009}	S_{2009}
mean	16.8	17.9	15.6	18.4	16.8	17.9	15.6	18.4
std	2.63	2.29	1.97	2.29	2.67	2.31	1.99	2.32
min	-0.137	8.3	8.14	8.1	-0.207	7.96	7.85	5.83
25%	15.0	16.3	14.1	16.7	15.0	16.3	14.1	16.8
50%	17.0	18.1	15.8	18.6	17.0	18.2	15.8	18.7
75%	18.7	19.5	17.1	20.0	18.7	19.6	17.1	20.1
max	26.3	27.1	23.3	27.4	26.2	27.2	23.2	27.4

Variables are capital K, materials M, wages W and sales value S. Variable units are 2005 rupees (in logs).

Table 9: ASI survey data: correlations

	K_{2008}	M_{2008}	W_{2008}	S_{2008}	K_{2009}	M_{2009}	W_{2009}	S_{2009}
K_{2008}	1.0	0.825	0.808	0.827	0.98	0.822	0.812	0.829
M_{2008}		1.0	0.843	0.976	0.823	0.963	0.834	0.948
W_{2008}			1.0	0.861	0.802	0.817	0.971	0.839
Y_{2008}				1.0	0.824	0.944	0.851	0.96
K_{2009}					1.0	0.828	0.814	0.834
M_{2009}						1.0	0.837	0.973
W_{2009}							1.0	0.859
Y_{2009}								1.0

Correlations between capital K, materials M, wages W and sales value S. Variable units are 2005 rupees (in logs).

Table 10: Transformed variables: summary statistics

	X_1	X_2	Y_1	Y_2
mean	1.05	1.09	1.83	1.82
std	1.51	1.51	1.09	1.11
min	-14.9	-15.9	-5.93	-7.12
25%	0.188	0.237	1.22	1.18
50%	1.15	1.18	1.83	1.82
75%	2.02	2.06	2.47	2.47
max	7.96	7.96	6.5	9.75

$X = \log\left(\frac{K}{W}\right)$ and $Y = \log\left(\frac{S-M}{W}\right)$

Table 11: Transformed variables: correlations

	X_1	X_2	Y_1	Y_2
X_1	1.0	0.911	0.43	0.401
X_2		1.0	0.394	0.433
Y_1			1.0	0.717
Y_2				1.0

$X = \log\left(\frac{K}{W}\right)$ and $Y = \log\left(\frac{S-M}{W}\right)$

8.2 Simulation results

Here we show how our estimates of shock and random coefficient moments change as we increase the number of observations. Throughout, we used the RMSE-minimizing parameters shown in table 1. To produce the tables below, we generated 8000 datasets for each number different number of observations, produced the relevant estimates and computed the performance measures.

Shock moments

Table 12: Performance measures for estimates of E[U_2]

n	bias	rel_bias	abs_bias	rmse
500	-0.0013	-0.0043	0.0979	0.1230
2000	0.0010	0.0032	0.0458	0.0576
5000	0.0000	0.0001	0.0275	0.0342
10000	-0.0001	-0.0002	0.0208	0.0262
20000	-0.0001	-0.0003	0.0181	0.0227

Table 13: Performance measures for estimates of E[V_2]

n	bias	rel_bias	abs_bias	rmse
500	0.0002	0.0023	0.0776	0.0972
2000	-0.0001	-0.0014	0.0356	0.0446
5000	0.0001	0.0006	0.0215	0.0270
10000	-0.0002	-0.0017	0.0163	0.0205
20000	0.0003	0.0029	0.0139	0.0173

Table 14: Performance measures for estimates of Std[U_2]

n	bias	rel_bias	abs_bias	rmse
500	-0.0021	-0.0021	0.0614	0.0780
2000	0.0096	0.0096	0.0302	0.0379
5000	0.0135	0.0135	0.0208	0.0256
10000	0.0109	0.0109	0.0159	0.0197
20000	0.0048	0.0048	0.0116	0.0145

Table 15: Performance measures for estimates of Std[B_2]

n	bias	rel_bias	abs_bias	rmse
500	-0.0120	-0.0189	0.1242	0.1586
2000	-0.0515	-0.0815	0.0793	0.1034
5000	-0.0733	-0.1159	0.0771	0.0923
10000	-0.0512	-0.0810	0.0559	0.0677
20000	-0.0324	-0.0512	0.0474	0.0601

Table 16: Performance measures for estimates of Cov[U_2, V_2]

n	bias	rel_bias	abs_bias	rmse
500	0.0186	0.1175	0.0681	0.0873
2000	0.0237	0.1499	0.0372	0.0473
5000	0.0282	0.1786	0.0312	0.0376
10000	0.0217	0.1375	0.0237	0.0285
20000	0.0097	0.0614	0.0147	0.0186

Table 17: Performance measures for estimates of Corr[U_2, V_2]

n	bias	rel_bias	abs_bias	rmse
500	0.1198	0.2397	0.2586	0.6747
2000	0.0736	0.1473	0.1387	0.3689
5000	0.0145	0.0290	0.0613	0.1130
10000	0.0050	0.0100	0.0456	0.0630
20000	0.0077	0.0154	0.0437	0.0601

Random coefficient moments

Table 18: Performance measures for estimates of $E[A_1]$

n	bias	rel_bias	abs_bias	rmse
500	0.0411	0.0205	0.1323	0.1669
2000	0.0344	0.0172	0.0683	0.0856
5000	0.0313	0.0157	0.0476	0.0591
10000	0.0300	0.0150	0.0381	0.0467
20000	0.0057	0.0029	0.0234	0.0293

Table 19: Performance measures for estimates of $E[B_1]$

n	bias	rel_bias	abs_bias	rmse
500	-0.0637	-0.1593	0.0843	0.1035
2000	-0.0540	-0.1349	0.0578	0.0677
5000	-0.0479	-0.1197	0.0486	0.0547
10000	-0.0433	-0.1083	0.0435	0.0474
20000	-0.0085	-0.0212	0.0164	0.0205

Table 20: Performance measures for estimates of $\mathrm{Std}[A_1]$

n	bias	rel_bias	abs_bias	rmse
500	0.0229	0.0076	0.1119	0.1411
2000	0.0094	0.0031	0.0561	0.0702
5000	0.0013	0.0004	0.0352	0.0443
10000	0.0041	0.0014	0.0257	0.0323
20000	-0.0054	-0.0018	0.0226	0.0282

Table 21: Performance measures for estimates of Std[B_1]

n	bias	rel_bias	abs_bias	rmse
500	-0.0120	-0.0189	0.1242	0.1586
2000	-0.0515	-0.0815	0.0793	0.1034
5000	-0.0733	-0.1159	0.0771	0.0923
10000	-0.0512	-0.0810	0.0559	0.0677
20000	-0.0324	-0.0512	0.0474	0.0601

Table 22: Performance measures for estimates of Cov[A_1, B_1]

n	bias	rel_bias	abs_bias	rmse
500	-0.1401	-0.1477	0.2252	0.2790
2000	-0.1130	-0.1191	0.1396	0.1701
5000	-0.0840	-0.0885	0.0972	0.1174
10000	-0.0840	-0.0885	0.0892	0.1042
20000	0.0006	0.0006	0.0513	0.0643

Table 23: Performance measures for estimates of Corr[A_1, B_1]

n	bias	rel_bias	abs_bias	rmse
500	-0.0125	-0.0250	0.1913	0.3675
2000	-0.0042	-0.0085	0.0994	0.1554
5000	0.0220	0.0440	0.0652	0.0862
10000	-0.0010	-0.0019	0.0476	0.0607
20000	0.0333	0.0666	0.0576	0.0769

8.3 Python module `FHHPS`

In order to facilitate the adoption of the methods described in this paper by other researchers, I have developed the Python module `FHHPS`. Instructions for installations and usage can be found in the github repository[1]. Its API is inspired by the celebrated

[1]Url: https://github.com/halflearned/FHHPS

`scikit-learn` libraryBuitinck et al. (2013), and it should be familiar to statisticians and machine learning practitioners that use Python. The webpage linked above also contains one-line instructions for reproducing all of our figures and tables.

Table 24: Panel regressions

	$Y := \log(\frac{Output-Materials}{Wages})$ and $X := \frac{Capital}{Wages}$								
	Y ~X			Y ~X + factor(panel_group)		Y ~X + factor(state)		Y ~X + factor(nic1987)	
	(F)	(P)	(R)	(P)	(R)	(P)	(R)	(P)	(R)
Intercept		1.489	1.495	1.361	1.360	1.669	1.673	1.318	1.318
		(0.007)	(0.009)	(0.131)	(0.169)	(0.049)	(0.063)	(0.145)	(0.185)
X	0.277	0.315	0.309	0.272	0.272	0.302	0.298	0.295	0.292
	(0.011)	(0.004)	(0.005)	(0.004)	(0.005)	(0.004)	(0.005)	(0.004)	(0.005)
R^2	0.045	0.186	0.135	0.271	0.196	0.205	0.148	0.276	0.202
Adjusted R^2	−0.909	0.186	0.135	0.262	0.186	0.204	0.147	0.275	0.201

Note: F: Fixed effects, R: Random effects, P: Pooling. All coefficients significant with $p < 0.01$. N=13298, T=2.

Part II

Kidney Exchange

Chapter 2

Two novel algorithms for dynamic kidney exchange

1 Introduction

Patients suffering from end-stage renal disease have two available treatments: dialysis and renal transplant. While transplant is associated with "lower mortality rates and improved quality of life compared to chronic dialysis treatment" (Tonelli et al., 2011), severe kidney shortage prevents tens of thousands of patients every year from receiving a transplant. In the US alone, there are currently over 90,000 patients in the kidney transplant waiting list. These patient may wait several years before finding an available donor, and many will die – often of co-morbidities – before receiving a transplant.

In order to overcome this challenge, *kidney paired donation* (KPD) has emerged as an alternative option for patients who have an incompatible but otherwise willing living donor. This approach, conceived about three decades ago by Rapaport (1986), involves pooling patient-donor pairs and swapping or exchanging donors so as to

increase the overall number of matched patients.

[[[History – Benefits]]]

Nevertheless, even today the amount of kidney exchanges remains smaller than its potential. As we will argue in this paper, one source of inefficiency in kidney exchange is that most transplant centers do not take into consideration the evolution of their kidney pool over time, which can lead to large losses in terms of the overall number of pairs matched over time.

To this end, this paper contributes to the *dynamic* kidney matching literature by introducing two novel algorithms for kidney exchange. They are explained next.

Main idea This paper proposes two novel approaches for increasing the cardinality of matched pairs in a discrete-time dynamic model of kidney exchange: the *direction prediction* and the *multi-armed-bandit* methods. We empirically evaluate these methods using simulations under a variety of data-generating processes that we call *environments*.

Let's see how these methods work, and some of the challenges they overcome.

First, our *direct prediction* method recasts the dynamic kidney exchange problem as a binary classification task: for each pair in the pool, we predict a binary label representing whether they should be matched today (one), or left for the future (zero), given their observable characteristics. This prediction is produced by an estimator trained in a large a number of simulations that were solved by an offline solver.

The main challenge with this approach is how to represent the data: while covariates such as ABO bloody type and current waiting times can be naturally represented by real numbers, it is not immediately clear how to represent their relationship to other nodes in the graph. We attempt to overcome this issue by augmenting the data set with graph-theoretic notions such as measures of node centrality, size of graph, average degree, and so on.

Our second method, named the *multi-armed bandit* (MAB) method, uses simulations to answer the question: "if we commit to matching one particular set of pairs today, how likely are we to decrease the *overall* number of matched pairs between now and a horizon h?". This probability can in principle be estimated via simulations. However, the main challenge here is that simulations are so computationally costly, and the number of available actions so vast, that we are cannot realistically produce accurate estimates of this probability for all points in the action space. To overcome this issue, we employ multi-armed bandit algorithms. Their role is to manage a relatively small computational budget and determine which actions are likely attractive (and should be explored further), and which are not (and thus not worth exploring much).

Main results Our methods are evaluated entirely via simulations. The simulation setups, which we call *environments*, are inspired by models used the kidney exchange literature, and differ by their rules regarding blood- and tissue-type compatibility.

We compare our method against two benchmarks: an algorithm that clears the maximal matching at each period, called MYOPIC; an infeasible offline optimal algorithm that we call OPT. The former roughly represents what most organ clearinghouses are doing today, while the latter represents the maximal payoff we could have achieved, had we been able to know the future. Our metric for comparison is the *per-period average number of matched pairs* over a long period of time.

In all environments, we observe that our implementation of the *direct estimation* method improves upon MYOPIC, but only under very restricted conditions. However, the *multi-armed bandit* (MAB) method is able to improve uniformly and substantially upon the MYOPIC benchmark. In particular, we observe that our MAB does particularly well in environments of moderate sparsity.

1.1 Related literature

This paper pulls ideas from microeconomics, matching theory, and operations research, and uses methods drawn from computer science and machine learning. Let us take a brief look at these fields in turn.

1.1.1 Kidney Exchange

Early years In microeconomics, the literature on kidney exchange began with Roth et al. (2004), who provided a kidney exchange mechanism inspired by the *housing* problem studied in the literature of mechanism design as in Shapley and Scarf (1974) and later Abdulkadiroğlu and Sönmez (1999). While their mechanism had significant virtues as being strategy-proof and Pareto-efficient, it also relied on large number of exchanges being conducted simultaneously, and drew criticism for making assumptions about heterogeneous preferences over kidneys. Subsequent work by Roth et al. (2005) addressed some of the criticisms by focusing on logistically simple mechanisms that used only 2-way exchanges, and assumed also that patients were indifferent between all compatible kidneys. In a later paper, Roth et al. (2007) demonstrated via simulations that allowing for 3-way exchanges in addition to 2-way exchanges could increase the cardinality of matched pairs by a great deal, but larger cycles would only bring about modest improvements.

[[[More here]]]

Recent advances in static exchange In the last ten years, both in academia and in medical practice the focus began to shift from exchanges via cycles to exchanges emanating from *non-directed donors* (NDD) – altruistic donors who are willing to donate their organs to anyone. Such exchanges yield a *chain* of transplants that begins with the NDD and may either terminate with the last pair donating to a waitlist recipient in a kidney registry, or not terminate at all and have the last pair's

donor await an opportunity to become a future living donor. In the former kind of exchange, called *domino paired donation* by Montgomery et al. (2006), all transplants occur simultaneously. In the second kind, transplants may be spread over several months. For this reason, the latter kind of transplant is called *non-simultaneous, extended, altruistic donor chain* by Rees et al. (2009).[1]

In recent years, a significant amount of attention has been devoted to these NEAD chains, including notably Ashlagi et al. (2011) and Ashlagi et al. (2012), whose results show that NEAD chains benefit highly sensitized patients in sparse pools of moderate size, because they decrease the need for simultaneous double-coincidence of wants in the exchange market. In fact, Anderson et al. (2015) reports that "NEAD chains are responsible for the majority of successful kidney transplants conducted via kidney exchange at both the [Allied for Paired Donation] and within other major exchange programs".

Dynamic kidney exchange The literature concerning the *dynamic kidney exchange* problem, where we take into account the evolution of the kidney exchange pool over time, is much smaller. It began with a seminal paper by Ünver (2010), who derives a dynamic mechanism that produces optimal n-way cyclic exchanges in the steady state of a continuous-time model with Poisson arrivals. The results in that paper rely on three simplifying assumptions. First, that the waiting cost is constant for all pairs; second, that pairs do not leave the pool unless they are matched; finally, that pairs within the pool are only blood-type incompatible. It can be shown that, under these assumptions, all pairs are rendered homogeneous. This dramatically decreases the dimension of the state space, and allows for an easily interpretable solution.

Ashlagi et al. (2013) note that the graphs of real-life pools tend to be sparse and filled with pairs that are either easy or extremely hard to match. In order to model

[1] On occasion, NEAD chains have also been called *Never-ending altruistic donor chains*, as relayed by (Roth, 2015, p. 235-6).

this particular feature, they postulate a sparse heterogeneous random graph model containing only two such types, and propose a greedy algorithm that waits until the pool contains a certain number of pairs of each type, and then matches as many as pairs as possible. In a different vein, Akbarpour et al. (2017) studies the relationship between "market thickness" (the number of available pairs in the exchange pool) and matching time in a simple model of stochastic arrival and departure. They show that, in their model, greedy algorithms that cleverly exploit the time to match can can perform close to infeasibly optimal benchmarks, even if these algorithms are ignorant about the global structure of the graph, and have no information about agents' departure times.

In the computer science and operations research literature, Abraham et al. (2007) and more recently Anderson et al. (2015), Dickerson et al. (2016) and Dickerson et al. (2017) have mostly focused on producing scalable combinatorial programming algorithms that can take on static matching problems with large graphs and allowing for both chains and cycles of moderately large length. However, a series of papers starting with Awasthi and Sandholm (2009) and followed by Dickerson et al. (2012) and Dickerson and Sandholm (2015) have dealt with the dynamic kidney problem by *weighted myopia*. Their idea is to prevent wasteful matchings (e.g., an O-donor to an AB-patient) by artificially introducing negative weights to graph components containing them. In Dickerson and Sandholm (2015), these optimal weights are computed from simulations involving historical data.

Computer science and machine learning We were inspired by several computational alternatives to dynamic kidney exchange that have been proposed in the past few years. In particular, the *multi-armed bandit method* described here is reminiscent Algorithm 1 in Awasthi and Sandholm (2009), inasmuch it also selects the best actions today by repeatedly simulating the future, solving an offline problem, com-

puting a "score" for each cycle, and selecting actions with maximum score. However, both our simulation and "scoring" methods differ importantly because we leverage multi-armed bandit algorithms to decrease the computational burden of producing multiple simulations.

Dickerson et al. (2012); Dickerson and Sandholm (2015) propose a related method that they call *weighted myopia*. The idea is to use simulations for learning "potentials" of specific elements of the graph. Much like the "scores" in Awasthi and Sandholm (2009), the set of "potentials" associated with a specific exchange encodes how desirable each exchange is in terms of its future value. The authors then use these pre-learned numbers to revise the weights in a myopic algorithm, forcing it to select more desirable exchanges where it would be otherwise indifferent. Our *direct prediction method* works similarly, with three crucial differences. First, we do not directly predict how "desirable" a matching will be, but instead we predict whether or not an offline, optimal algorithm would choose the node or not. Second, our method uses a more aggressive thresholding mechanism to select which nodes should be matched today, and which should be left for later. Third, we optionally make use of information about how the node fits inside the compatibility graph, so that the information about the node (e.g., the blood type and HLA profile of patient and donor) is augmented with graph-theoretic notions (e.g., measures of node centrality, degree, etc).

Sequential decision problems Zooming out of our application, our problem lies within a larger class of sequential decision problems whose central feature is a concern with the *exploration-exploitation trade-off*. That is, problems that can be described as the one faced by an agent who must spend a limited computational budget to *explore* a certain action space so as to find and *exploit* actions that will maximize her expected rewards. Crucially, the agent sequentially observes rewards for actions that she has taken, but does not observe rewards for other actions. This class also

encompasses, for example, *Markov decision process* (MDP) and its variants.

A *multi-armed bandit*[2] (MAB) problem is a particular version of an MDP where time is discrete, and at every period the agent faces a finite and fixed number of actions K that produce stochastic rewards. The objective is usually to minimize the amount of *regret* that accumulates over time, where *regret* is loosely defined as the difference between the agent's actual accumulated rewards under her own strategy and an the average reward that the agent would have gotten had she chosen the optimal action at each period since the beginning. (We will make this definition more rigorous in a later section). Crucially, at every period the agent only observer rewards for actions that she has chosen, and not for other actions.

Multi-armed bandit problems were studied sporadically in the last century, with the earliest reference going as far back in time as Thompson (1933). Interest was rekindled after a seminal paper by Lai and Robbins (1985), who developed a strategy that provably attains an asymptotic lower bound of regret. Later, their analysis was simplified by Agrawal (1995) who also developed a finite-time analysis of regret. In recent years, other alternative algorithms have been proposed. In particular, in our paper we use the *Thompson sampling* algorithm studied by Agrawal and Goyal (2012) and Kaufmann et al. (2012), and the *upper-confidence bound* (UCB1) algorithm developed by Agrawal (1995) and Auer et al. (2002). For an approachable review of multi-armed bandits, optimal strategies, and their variations, we refer the reader to the recent book by Lattimore and Szepesvari (2018).

[2]The terminology comes from a turn-of-the-20th-century United States colloquialism, when slot machines were called "one-armed bandits". While the name apparently suggests a loss in the long run, the "bandit" problems studied in academic literature do not necessarily have negative expected payoff.

2 Background and definitions

Medical background We say that a patient and a donor are *compatible* if the donor's organ cells do not present *antigens* that are capable of inducing an aggressive response by the patient's immune system. A successful transplant usually requires patients to be *blood-type compatible* and *tissue-type compatible* (also known as *histo-compatible*).

Blood-type compatibility refers to compatibility with respect to major ABO blood groups. For example, O-type patients can only receive from O-type donors, AB patients may receive from any blood group, etc).

A patient is tissue-type compatible with a donor if the donor does not present alleles of gene complex called the *human leukocyte antigens* (HLA) that are deemed unacceptable by the patient's immune system.

Technical definitions We model the dynamic kidney exchange problem in a manner similar to Ünver (2010) and Akbarpour et al. (2017). A *kidney exchange pool* is a directed random graph process $G_t = (V_t, E_t, X_t), t \in \mathbb{N}$ whose vertices $v \in V_t$ represent (patient, donor)-*pairs*[3]. A directed edge $e \in E_t$ between vertex v and v' means that the donor in pair v can donate to the patient in pair v'. When such an edge exits, we say that v *is compatible with* v'.

Each pair is endowed with certain characteristics $x \in X_t$[4] such as patient and donor blood type (other characteristics will be discussed below).

We say that a pair *enters* the kidney exchange pool when it first becomes available for exchange. After a certain number of periods, the pair *leaves* the pool once and for all, or *dies*. The difference between entry and death is called a pair's *sojourn*.

An *exchange* is an ordered tuple of nodes $m = (v_{i_1}, \cdots, v_{i_k})$ where each pair in

[3]Throughout, we will use the terms *pair*, *node* and *vertex* interchangeably.

[4]In an slight abuse of notation, we will also denote by X_t the matrix of pairs' observable characteristics.

the tuple is compatible with the next pair in the sequence. A *matching* is a set of exchanges where no pair appears in more than one exchange.

An *environment* is, informally speaking, a collection of rules governing which pairs are deemed compatible, the entry and death processes that govern the evolution of the pool, and which vertex characteristics are observable in and relevant to the problem. More formally, it can be defined as the conditional probability distribution between two kidney exchange pools, given the current pool G_t and G'_t, given a matching M_t. We will expand on this below. Finally, a *dynamic matching algorithm* is a procedure that selects matchings M_t at every period. Once a matching M_t is selected, all vertices and their edges are removed from the kidney exchange pool.

2.1 Environments

Our paper presents three environments inspired by previous works on dynamic kidney exchange. All three have the following assumptions in common.

First, the number of new incoming pairs in each period is drawn from the $Poisson(r)$ distribution, where $r \in \mathbb{N}$ denotes the *entry rate*, and also equals the expected number of entrants per period. Second, each pair independently draws the length of their sojourn from the $Geometric(d)$ distribution. The parameter $d \in \mathbb{R}$ is the *death rate*, and its reciprocal $\frac{1}{d}$ is the expected sojourn length. We note that, due to the memoryless property[5] of the Geometric distribution, the amount of time a pair has waited in the pool gives us no information about how much time they have until their death. Third, we assume that patients do not discriminate between compatible kidneys. This last assumption could also be understood as patients having binary preferences over kidneys (i.e., they receive utility 1 if they receive a transplant, and 0 otherwise). This assumption is consistent with other works in the literature, notably Roth et al. (2005).

[5]If $X \sim Geometric(p)$, then $P(X > t + s|X > s) = P(X > t)$

In what follows we will explain the differences between each environment.

2.2 ABO Environment

In the *ABO environment*, compatibility between two distinct pairs is based only on blood-type compatibility.

Blood types are drawn independently for patients and donors from the corresponding probability distribution in the US population[6], but are adjusted by the following assumption previously used in Ünver (2010): we allow for incompatibility between a donor and their own patient. The reason for this additional assumption is that, if there were truly no tissue type compatibilities, we would never observe pairs of type $(AB, \cdot)$, $(\cdot, O)$, or (A, A), (O, O), (B, B), and (AB, AB)[7]., since their donors would be automatically compatible with their patients and they would never participate in an exchange. Following Zenios et al. (2001), we assume that for such patients the probability that a donor and their patient are incompatible is $p_c = 0.11$. Arrival rates are then adjusted accordingly, e.g., the arrival rate of (A, O) pair is proportional to $0.48 \times 0.36 \times 0.11 \approx 0.019$. The exact probability distribution for all types can be found in the appendix.

2.3 RSU Environment

The *RSU* environment is named after a classic simulation model in Roth et al. (2007) and Saidman et al. (2006). Each pair is characterized by patient and donor ABO blood types, current waiting time, and a *calculated panel reactive antibody (cPRA)* level that represents the probability of a crossmatch with a random donor.[8] The lower

[6]Roughly O:49%, A:36%, B:11%, AB:4%

[7]For shorthand, we will sometimes write "An (X, Z) pair" to mean "any pairs where the patient has blood type X, and the donor has blood type Z".

[8]The original Roth et al. (2007) paper called this simply *PRA*, and in real life there is an important distinction between the two measures. However, for the purposes of a simulation model this distinction is immaterial. We keep the name *cPRA* for consistency with OPTN environment

the cPRA, the higher the number of potentially compatible pairs.

The simulation process is as follows. First, we draw a pair in the same manner as in the ABO environment. Next, we draw if the patient is a female (with probability around 41%), and if so we also draw whether her donor is her husband (spouses comprise about 49% of donors). Finally, we draw a cPRA level for the patient (Low: 70.1%, Medium: 20%, High: 9.9%). This cPRA level determines the probability that they can receive a kidney from any donor, including their own: patients with low cPRA have a 5% probability of positive crossmatch with a random donor; patients with medium cPRA have a 45% chance, and patients with high cPRA have a 90% chance of a crossmatch. If a patient is bloody or tissue-type incompatible with their own donor, they enter the pool. In addition, if the patient is female and her husband is the donor, the probability of positive crossmatch for low, medium and high cPRA patients goes up to 28.75%, 58.75% and 92.25%. This last adjustment reflects the fact that women tend to produce antibodies against their husbands' antigens during pregnancy.

Once in the pool, the pair immediately forms directed edges with the existing pairs, again following the patient cPRA distribution. The resulting random graph is akin to a Erdös-Rényi $G(n, p)$ random graph where the probability of forming edges is heterogeneous across different pair types.

2.4 OPTN Environment

In the *OPTN environment*, we use historical data collected by the United Network for Organ Sharing (UNOS) data provided in the Standard Research and Analysis (STAR) dataset. The STAR dataset contains information from all patients that were ever registered to the kidney waiting list in the United States for the past three

later.

decades, as well as from all living donors that actually participated in an transplant[9]. From this original dataset, we excluded entries associated with the following:

- Patients that were registered for more than one organ (including those who were simultaneously waiting for kidney and pancreas)
- Patients that were not waiting for their first kidney transplant
- Donors and patients with incomplete tissue-type profile information.

The resulting dataset contained 117813 patients and 9337 living donors. We call this the "historical dataset".

Patient and donor cPRA We added two additional variables to the original dataset: a *patient cPRA* and a *donor cPRA*. While the patient cPRA is a measure of patient tissue-type incompatibility with a random donor Cecka (2010), our donor cPRA is a novel measure of the opposite direction – how frequently a donor is tissue-type incompatible with a random patient.[10] Both were computed empirically: for each patient our dataset, we checked the how many donors exhibited antigens that are unacceptable for the patient in any of the A, B, Bw, C, DR, DPB, DQ, and DQA loci, and assigned this positive crossmatch probability as their patient cPRA[11]; for the donors, we worked in the opposite direction by calculating the frequency of patients who exhibited antibodies against their donor's antigens, and that became their donor cPRA. Figure 2 shows the distribution across the entire population.[12]

[9] To our knowledge, there is no centralized dataset containing information about registered donors that never went to transplant.

[10] We thank Itai Ashlagi for the suggestion of a donor cPRA.

[11] A genetic *locus* is a specific position on a chromossome. The loci above encodes a donor tissue type.

[12] We should remark that we found a very different patient cPRA distribution than the one in the OPTN dataset. This may have been because: in real life cPRA is computed using deceased donor data, while we used living donor data; we may have used different HLA equivalence tables; OPTN uses a more sophisticated model based on population genetics.Organ Procurement and Transplantation Network (2013)

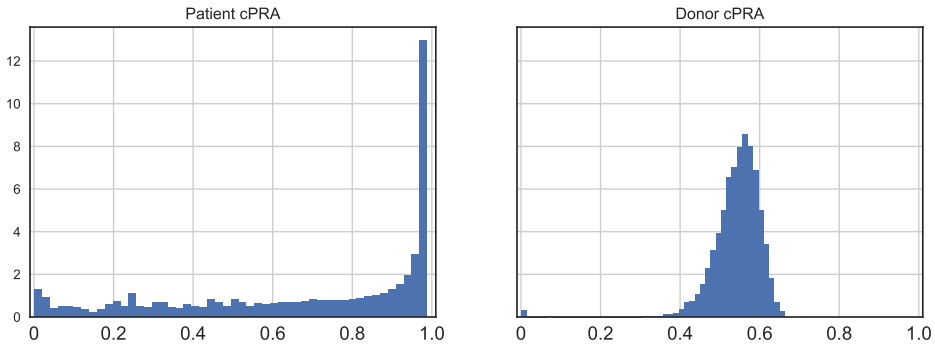


Figure 2: **Patient and donor cPRA**
Computed from historical data. Our patient cPRA is the frequency of living donors that exhibit antigens that are unacceptable to the patient. We also define a donor cPRA by calculating the frequency of patients that have antibodies against the donors antigens.

Artificial dataset We created an artificial dataset by randomly drawing patients and living donors from the historical dataset and checking for blood-type compatibility, and tissue-type compatibility as explained above. Compatible pairs were discarded. We iterated in this manner to construct a dataset of about one million incompatible pairs. At every simulation period, a random number of pairs is drawn from this dataset.

3 Methods

3.1 Objective and benchmarks

Objective In this work, we focus on maximizing the undiscounted cardinality of matched pairs over T periods. To make comparison to our benchmarks easier, we formulate the problem equivalently as maximizing the per-period average matching size over T periods.

Benchmarks Let $\mathcal{M}$ a set of available matchings at period t.

$$\max_{M \in \mathcal{M}} \sum_{m \in \mathcal{M}} w_m x_m \tag{2.1}$$

$$\text{s.t.} \qquad \sum_{v \in m} x_m \leq 1 \qquad \forall v \in V \tag{2.2}$$

$$x_m \in \{0, 1\} \qquad \forall m \in \mathcal{M} \tag{2.3}$$

where $c \subset V$ is an exchange, w_c is the cardinality of the exchange, and x_c is a binary variable indicating whether or not the pair was selected. Constraint 2.3 ensures that each vertex is selected only once.

By an *static matching problem at period t*, we mean a version the problem above where the only available matchings involve pairs $v \in V_t$. By an *offline matching problem between t and s*, we mean a version of the same problem where involving vertices $v \in \cup_{k=0}^{s} V_{t+k}$, and any matching $m = (v_1, \cdots, v_k)$ has the property that the sojourns of the donating pair overlaps with the sojourn of the receiving pair.

We are interested in how our new methods perform relative to the following two benchmarks.

Myopic At every period, the MYOPIC algorithm solves a static matching problem, finds the maximal matching and clears it immediately. In doing so, it disregards all observable characteristics of each pair, and in particular it ignores that some pairs might be useful to keep certain pairs may be easier or harder to match. Therefore, it may forgo the opportunity of matching a hard-to-match patient today, or postpone an easy-to-match pair for later. In essence, this is an approximation to what kidney exchanges currently do.

Optimal (OPT) This infeasible algorithm (henceforth OPT) solves the offline matching problem encompassing all periods between 1 and T. This completely does away with the uncertainty arising from the temporal structure of the problem, hence we know that this is the maximum achievable utility

Remark *How much is there to be improved upon?* In Figure 3, we compare Myopic and OPT for a grid of different entry and death rates. These results show that the performance gap between the two can be fairly large, in particular in sparser environments like *RSU* and *OPTN*. We also note that the gap is narrower when: the death rate is high, because if most pairs will die soon, dynamic considerations play a smaller role; and when the entry rate is very large, because in a thicker market pairs are able to encounter a suitable match more easily.

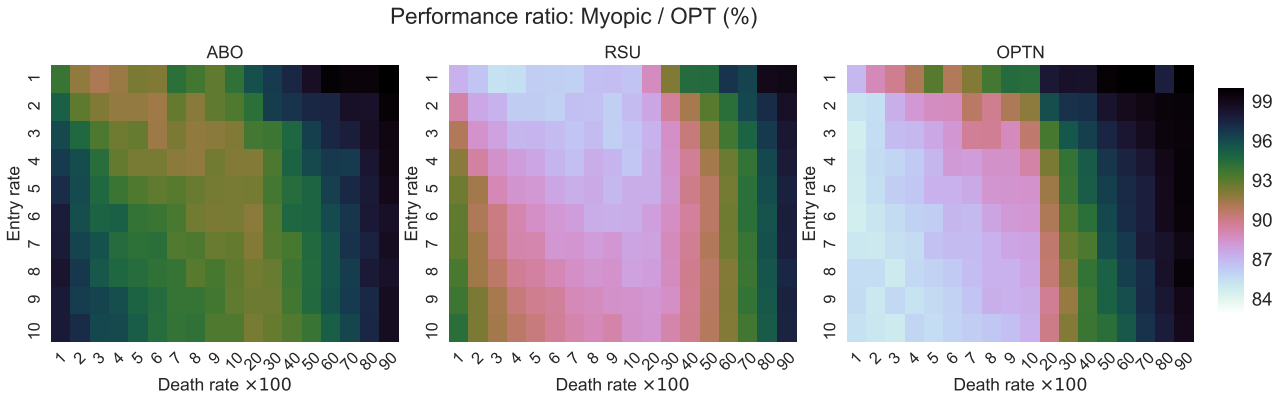


Figure 3: **Comparing Myopic and OPT**
Ratio of average per-period matched pairs for different entry and death rates, over 3000 periods (darker hues are better). `Myopic` has better chances of achieving performances similar to `OPT` when the death rate is high (moving rightwards on the graphs), or when entry rate is high (moving downwards on the graphs).

4 Algorithms

We now present two novel methods to determine which patients should be matched.

4.1 Direct prediction

We can break down the dynamic kidney problem into two parts. The first is determining *which* pairs should be matched today, and the second is deciding *how* the selected pairs should be matched among themselves. Note also, that the second part of the problem can be solved immediately as an integer programming problem.

This is the insight that our *direct prediction* method exploits. Essentially, we can reduce the problem to a classification task: at each period, we aim to produce a binary label for each node indicating whether is should be matched in this period (1) or left for later (0). Selected nodes are then passed to a static solver that finds the maximal matching among them. Once these nodes are cleared, time evolves to the next period.

The procedure is formalized in Algorithm 1, and also illustrated in Figure 4.

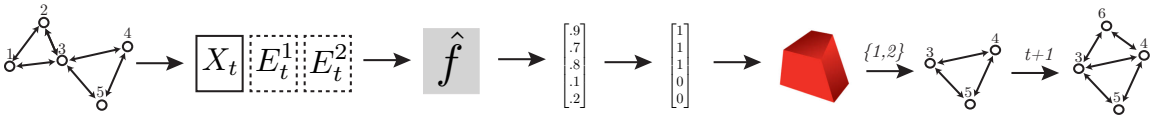


Figure 4: **Direct prediction method**
One period of simulation using direct prediction methods to choose cycles. See Algorithm 1 for details.

ALGORITHM 1: DIRECT_PREDICTION Method

Data: Environment simulator object ENV;
Integer programming solver object SOLVER;
Statistical method CLASSIFIER
Threshold *thres*;
Function `DIRECT_PREDICTION`(ENV, SOLVER, CLASSIFIER, *thres*)**:**

```
// Retrieve node (and potentially also graph) data from current environment
X, E_1, E_2 ← ENV.GET_DATA()
// Classifier predicts matching probability for each pair using data
prob ← CLASSIFIER(X, E_1, E_2)
// Get index of pairs whose probability is higher than threshold
index ← WHICH(prob > thres)
// Find maximal matching restricted to this subset
chosen_cycles ← SOLVER.SOLVE(ENV, SUBSET=index)
// Return chosen cycles to be cleared
return chosen_cycles
```

Training the classifier In order to produce data to feed into our "classifier", we repeatedly created 1000-period simulation runs and solved them offline using OPT. Then, for every period t in simulation k, we stored: a matrix X_t^k whose rows represent each pairs observable characteristics; an additional matrices E_t^{1k} containing additional graph-theoretical information such as several centrality measures (betweenness, in-degree, out-degree, harmonic, closeness), in- and out-degrees, and average neighbor degree; a conforming matrix E_t^{2k} containing the entry and death rates used in that simulation; a binary vector y_t^k indicating which nodes OPT chose to match in period t. Finally, we vertically concatenated each one of these matrices to create the final data set.

We repeated this procedure for each environment, and produced three artificial data sets, each containing approximately 2 million observations. These data sets were then fed to a series of predictive algorithms: penalized logistic regression (Wu et al., 2009), random forest classifiers (Breiman, 2001), and gradient tree boosting classifiers (Friedman, 2001).

4.2 Multi-armed bandit methods

Since we are assuming access to the exact data-generating process, we can use simulations to choose the best action at each period t: choose a certain action; simulate the evolution of the kidney exchange pool until some time horizon $t+h$; solve the offline problem; evaluate the performance of the chosen action under some criterion; repeat until[[[[]]]]

However, the approach outlined above turns out to be naive because simulations are computationally expensive, and in practice we cannot repeat them enough times get reliable estimates of the average performance for each cycle choice, especially in large graphs. It is also incomplete because it does not specify exactly what is the best information we should extract from `OPT` results. In order to solve the first problem, we leverage theory and algorithms from the multi-armed bandit (MAB) literature. For the second, we propose a secondary objective based on what we call *pseudo-rewards*.

Our procedure is illustrated in Figures 11 and **??**. At the beginning of the period t, the agent receives a set of cycles C that are available to be cleared. If C is empty, nothing happens and we move to the next period. Otherwise, the agent then picks a cycle $c \in C$ and simulates the future, including new entries and deaths, up to a horizon h. Next, `OPT` is run twice, once normally, and once with the additional constraint that c be removed today. The size of the resulting matching in these two scenarios is compared. Naturally, the constrained version of `OPT` cannot achieve anything better than its unconstrained counterpart, but it might get to be equal. If it is, the agent receives a *pseudo-reward* of one, otherwise it receives zero. This process is repeated: at each iteration ℓ, a cycle c_ℓ is chosen and its pseudo-reward $r_{c,\ell}$ is revealed. When a preset computational budget of $L(|C|)$ iterations is hit, the agent then analyses the whole history of

cycle choices and pseudo-rewards $H_t = \{(c_\ell, r_{c\ell})\}_{\ell=1}^{L}$, and decides whether to match one of the cycles or move on to the next period. If a cycle c is chosen it is immediately cleared, however the environment does not evolve to the next period yet. Instead, the history H_t is discarded the procedure is repeated again with a reduced set of actions $C' \subset C$ that produces a new history H'_t and so on, until either there are no more available choices or the agent decides to allow the environment to move on to time $t+1$. When that at last happens, entries and deaths are revealed, the agent receives a new set of cycles, and the process begin anew. All past information is ignored.

Two important details were left out of the explanation above. First, how does the agent chooses the next cycle to test? Second, how does it decide which cycle to choose (or no cycle at all)?

Let c^* be a cycle that maximizes expected pseudo-rewards during one round of the algorithm:

$$c^*_\ell \in \arg\max_c E[r_{c,\ell}]$$

Also, let *regret* be defined as the difference between expected reward of the optimal choice c^* and its own selected choice c_ℓ.

$$\Delta_\ell := E[r_{c^*,\ell}] - E[r_{c,\ell}]$$

A *multi-armed bandit* (MAB) algorithm is an adaptive exploration procedure that seeks to minimize the cumulative regret over L rounds. A good MAB algorithm will act so as to balance exploration (trying out different choices to get high-quality estimates of their rewards) and exploitation (using out better choices more often to increase total rewards), and produce regret that grows at an asymptotically slow rate. Here, we experiment with two common bandits algorithms, namely *UCB1*, and *Thompson sampling*. The literature on bandit

algorithms is extensive and an in-depth explanation is outside the score of this paper. However, for context in the next paragraphs we provide some intuition for how and why they work.

UCB1 The *Upper Confidence Bound 1* prescribes that at each period of repeat the following two steps.

1. Construct a certain confidence interval around the average reward estimate for each arm.
2. Choose the action with the highest confidence upper bound.

This heuristic is commonly named *optimism in the face of uncertainty*Kaelbling et al. (1996), because at every period we are choosing the action with the "largest plausible"(Lattimore and Szepesvari, 2018, Ch. 3) average reward estimate. The intuition is the following: if the agent is correct in choosing the optimistic action, then they receive zero regret; if the agent is incorrect, then they will thereafter review their estimates so as to decrease the upper bound for that action and avoid it in the future.

Auer et al. (2002) proved that, in the absence of further information about the distribution of rewards, by appropriately constructing the confidence interval around each reward average we are able to attain an optimal logarithmic cumulative regret rate. Their proposed confidence interval for action c in round ℓ has the form

$$UCB[\ell, c] = \hat{\mu}_{c,\ell-1} + \sqrt{\frac{2\log(\ell)}{n_{c,\ell-1}}} \tag{2.4}$$

where ℓ is the current round of the bandit algorithm, $\hat{\mu}_{c,\ell-1}$ is the running reward estimate for action c, and $n_{c,\ell-1}$ is the number of times that the action c has been selected before round ℓ. It is instructive to remark that the expression

above is increasing in $\hat{\mu}_{c,\ell}$ – so that actions with higher expected reward will be naturally chosen more often, leading to exploitation – but it is also decreasing in $n_{c,\ell-1}$ – making other actions more likely to be chosen in the future, leading to more exploration.

Thompson sampling Also known as *posterior sampling*, this algorithm reformulates the bandit problem in a Bayesian framework. The agent begins with a prior distribution $P(\mu_c)$ for each action c. This initial prior is updated as the history of actions and rewards accumulates. Like UCB1, the entire algorithm is explained in few steps:

1. Compute and sample from the posterior distribution of average rewards

$$\tilde{\mu}_c \sim P(\mu_c|H_{\ell-1}), \forall c$$

2. Choose the action with maximal reward among the samples.

$$c_\ell = \arg\max_c \mu_c$$

This simple algorithm can be shown to be optimal in terms of attaining a asymptotically logarithmic lower bound on Bayesian regretsli (2011). However, more surprisingly, Kaufmann et al. (2012) showed that it also enjoys good frequentist properties, and under certain conditions performs no worse than UCB1 in terms of *frequentist* regret. Moreover, Thompson sampling algorithm can be generalized in a number of ways (some of which we discuss in section 6).

What are pseudo-rewards? As we match and clear out a cycle c today, we forgo the opportunity of using any future cycles involving the nodes in c. However, because there may be multiple optimal matchings, sometimes the

ALGORITHM 2: MULTI_ARMED_BANDIT Method

Data: Environment simulator object ENV;
Integer programming solver object SOLVER;
Threshold *thres*; Horizon h;
Function `MULTI_ARMED_BANDIT`(ENV, SOLVER, BANDIT, *thres*, h)**:**

```
done ← Falsewhilenot at end of this documentdo
    // Select a cycle or a null token
    c ← CHOOSE_CYCLE(ENV,h,thres)
    // Check if a cycle was indeed chosen
    if c is not NULL then
        // Remove cycle and continue search
        ENV.REMOVE(c)
    else
        // Just terminate search
        done ← Trueend
    end
    // Return environment with removed cycles
    return ENV
```

ALGORITHM 3: Function GET_PSEUDO_REWARD

Data: Cycle c; Horizon h;
Lists pseudo-reward statistics *Avg*, *Std*;
Environment simulator object ENV;
Oracle solver object OPT;
Function `GET_PSEUDO_REWARD`(ENV, c, *Avg*, *Std*, h)**:**

```
// Simulate up to horizon h and find optimal matching
ENV.SIMULATE(h)
r1 ← OPT.SOLVE(ENV)
// Remove cycle c, find constrained optimal matching
ENV.REMOVE(c)
r2 ← OPT.SOLVE(ENV)
// Return 1 if rewards are equal, 0 otherwise
return r1 == r2
```

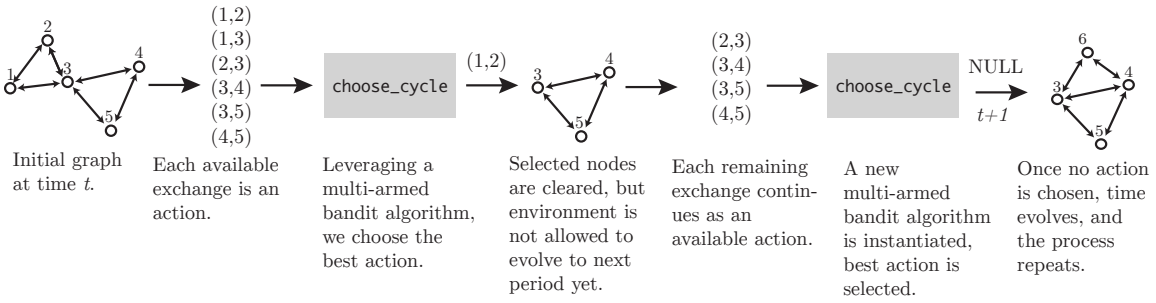


Figure 5: **Multi-armed bandit methods**
One period of simulation using multi-armed bandit methods to choose cycles.

ALGORITHM 4: Function `choose_cycle`

Data: Horizon h; Number of iterations L; Threshold *thres*;
Environment simulator object Env; Multi-armed bandit algorithm object MAB;
Function `choose_cycle(`Env, *h*, *thres*`):`

```
// Initialize lists of current pseudo-reward statistics
C ← ENV.GET_AVAILABLE_CYCLES()
Avg ← ZEROS(LENGTH(C))
Std ← ZEROS(LENGTH(C))
// Begin iterations
for i ← 0 to L do
    // Bandit algorithm chooses next cycle to test given statistics
    c ← MAB.PULL(C, AVG, STD)
    // Compute pseudo reward for this cycle and update statistics
    r ← GET_PSEUDO_REWARD(ENV, c, Avg, Std, h)
    Avg ← UPDATE_RUNNING_AVERAGE(Avg, r)
    Std ← UPDATE_RUNNING_STD(Std, r)
end
// Bandit algorithm chooses best cycle given statistics
c_best ← MAB.CHOOSE(C, AVG, STD)
// Return best cycle, unless none of the pseudo-reward averages are above a certain threshold
if ALL(Avg ≤ thres) then
    return NULL
else
    return c
end
```

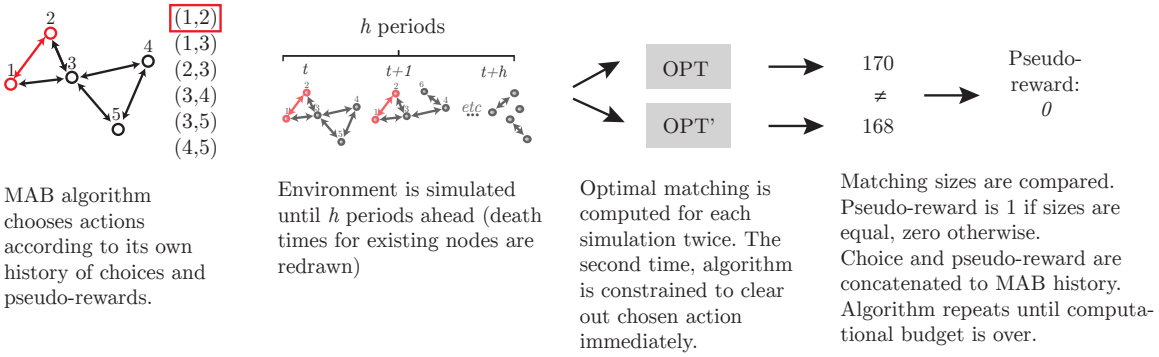


Figure 6: **Function** GET_PSEUDO_REWARD
Multi-armed bandits evaluate if a cycle should be cleared by checking if there is a high chance that the cycle will be used in the future. Such cycles get a *lower* reward, and are left for later.

cycles that become unavailable due to the removal of the nodes in c do not matter, and we can still find a matching of the same cardinality without them. In other words, we pay no price for removing these future cycles.

The pseudo-reward associated with cycle c is a random variable whose expectation is the probability that removing a cycle today will *not* negatively impact the optimal matching size between t and $t + h$. The higher this number, the more confident we are that clearing c out today will not give us trouble in the future. A concrete example of this idea in shown in Figure 7.

Pseudo-rewards are a natural way for us to control which patients should be matched today. Pairs that are easy to match will likely belong to many cycles, so by removing them we will be incurring a large cost in terms of future cycles that will become unavailable. But that means that the pseudo-reward associated with cycles that involve them will be lower, making them less attractive. On the other hand, patients that are harder to match will not participate many future exchange, so the price we pay for matching them today is low, and their average pseudo-reward is high.

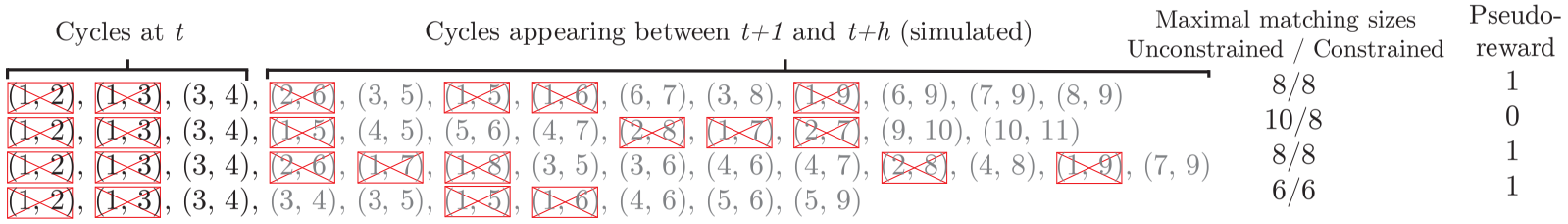


Figure 7: **Intuition for pseudo-rewards**
In the example above, at time t we have three available cycles to choose from: $(1, 2)$, $(1, 3)$ and $(3, 4)$, and we are contemplating choosing cycle $(1, 2)$ at t. If we constrain ourselves to do so, several other future cycles become unavailable (in red). Nevertheless, in three out of our four simulations we were able to clear the same number of cycles in the constrained scenario, so the running pseudo-reward average is $\frac{3}{4}$.

5 Results

5.1 Direct prediction methods

Table 26 shows the performance results for direct prediction methods as estimators, i.e., how accurately they are able to predict whether a node should be matched or not. We see that while overall accuracy can be relatively high at 70-80%, precision (defined as the ratio between true positives and both true and false positives) is low even for simpler environments like ABO, indicating that the models are over-predicting matchings. Also, augmenting the data with information about the graph has little discernible effect under any of the performance criteria. This suggests that there might be gains from using other algorithms that make better use of information about the compatibility graph.

In order to empirically evaluate the performance of our proposed direct prediction method, we proceeded as follow. First, we simulated data using our environments, and applied Algorithm 1 at each step, following the outline on Figure 4. Next, using the *same* random seed, we simulated each environment and solved it again using MYOPIC in lieu of our method. Finally, we computed the average number of matched patients between periods 250 and 1000 (the first 250 periods were used as burn-in) for each algorithm and compared the two.

This process was repeated several times for: each classifier (logistic regression, gradient tree boosting, random forests), each environment (ABO, RSU, $OPTN$) and nine entry and death rate combinations. The threshold variable was fixed at $thres = \frac{1}{3}$ everywhere.

The result is shown on Figures **??**, where we show the average performance ratio computed as above for each combination of environment, algorithm and entry/death rate configuration. **??**

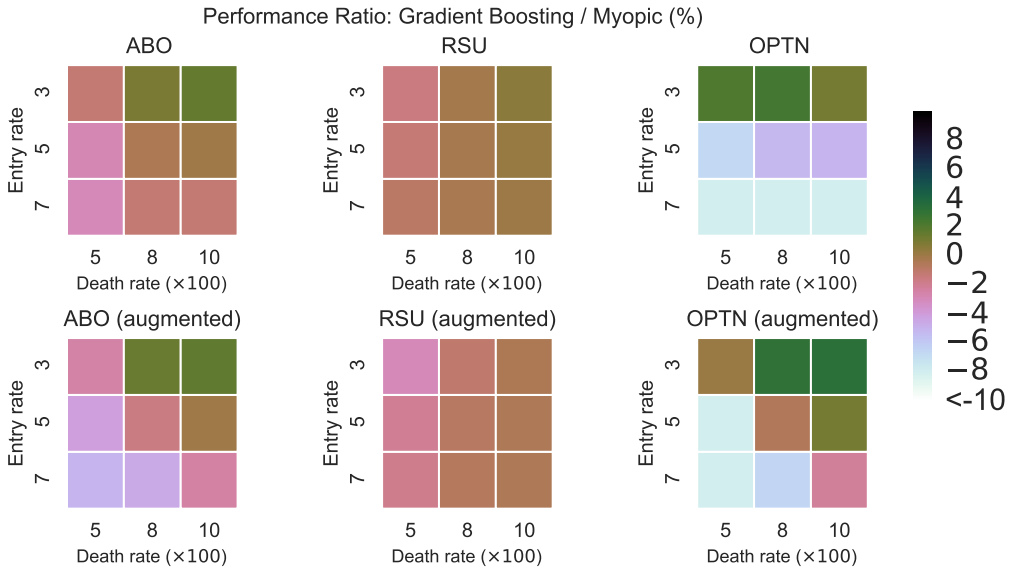


Figure 8: **Performance of gradient boosting in direct prediction method** Ratio of average matched patients against MYOPIC. Simulations ran over 750 periods (after a 250-period burn-in sequence). Lower row is when data was augmented with information about graph.

Unfortunately, as we predicted in the beginning of the section, the poor predictive performance of the algorithms translates into poor performance as the classifier in our direct prediction method. Random forests, in particular, have the lowest rate of accuracy (Table 26), and also exhibit the lowest performance in the direct prediction method.

bb

5.2 Multi-armed bandit methods

We evaluated the performance of our *multi-armed bandit* in an analogous manner to the direct prediction method described in the previous section.

The average ratio between our method and MYOPIC is shown on Figure 11 and elaborated upon on Tables 33-??.[13] The results suggest that, for the entry and

[13]For reasons of space, in the main paper we present only Table 33. The remaining tables will be made available in a supplementary only appendix.

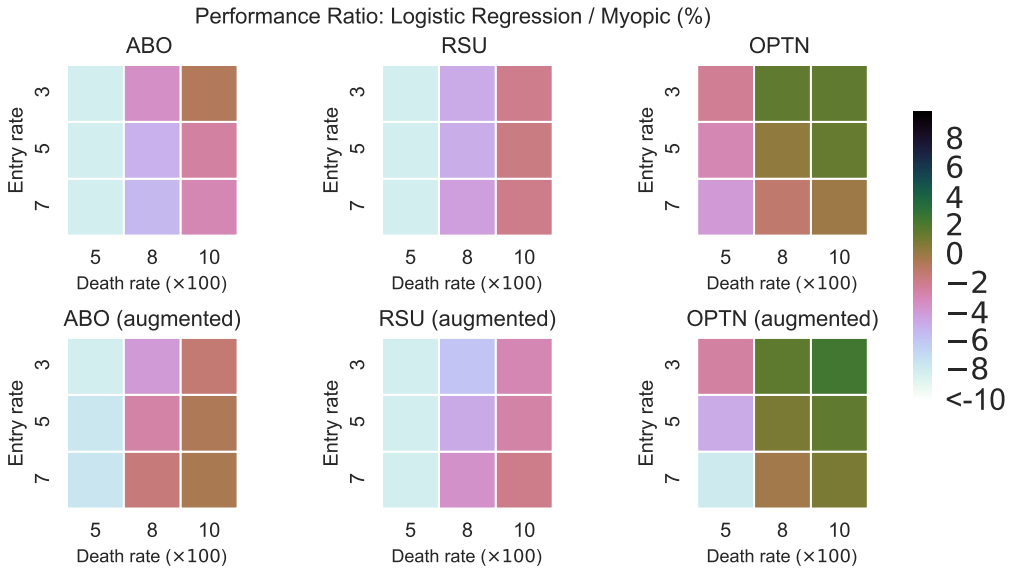


Figure 9: **Performance of logistic regression in direct prediction method** Similar to gradient boosting shown in Figure 8, logistic regression is only able to perform better than MYOPIC when the death rate is large.

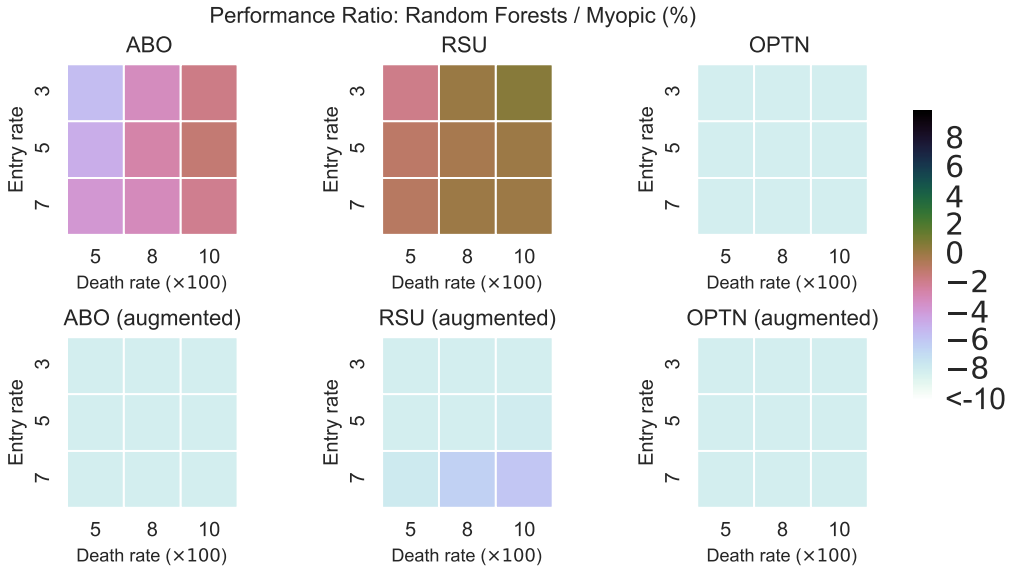


Figure 10: **Performance of random forests in direct prediction method** Random Forests's poor performance is likely due to its low accuracy, as we show on Table **??**.

death rate combinations we experimented on, the *multi-armed bandit* method uniformly dominates MYOPIC in terms of average number of matched patients per period.

In sparser environments (RSU and $OPTN$), for particular entry and death rate combinations, our method improve upon MYOPIC by up to about 4%. This reflects an earlier observation we did when analyzing Figure 3: gains from taking dynamic consideration into account are larger in situations of moderate sparsity, because that is where MYOPIC forces the pool to be too thin, and drives the performance away from optimality.

6 Extensions and future work

Our methods suggest a variety of extensions.

Improving the direct prediction method First, the relatively weak performance of our *direct prediction* method could be improved in two ways. First, by improving our ability to use information about graph structure when predicting if a node should be matched or not. This could be done using new statistical methods that are applicable to non-euclidean spaces(Shuman et al., 2013). In particular, recent years have seen the emergence of interest in generalizing neural networks to model with structured datasets such as graphs. A promising approach is the one taken by Kipf and Welling (2016), who build on earlier work by Defferrard et al. (2016) to create a very simple recurrence relation for graph embedding:

$$H^{\ell+1} = \sigma((A_t + I)H^{\ell}W^{\ell}) \qquad \text{with} \quad H^{\ell} = X_t \tag{2.5}$$

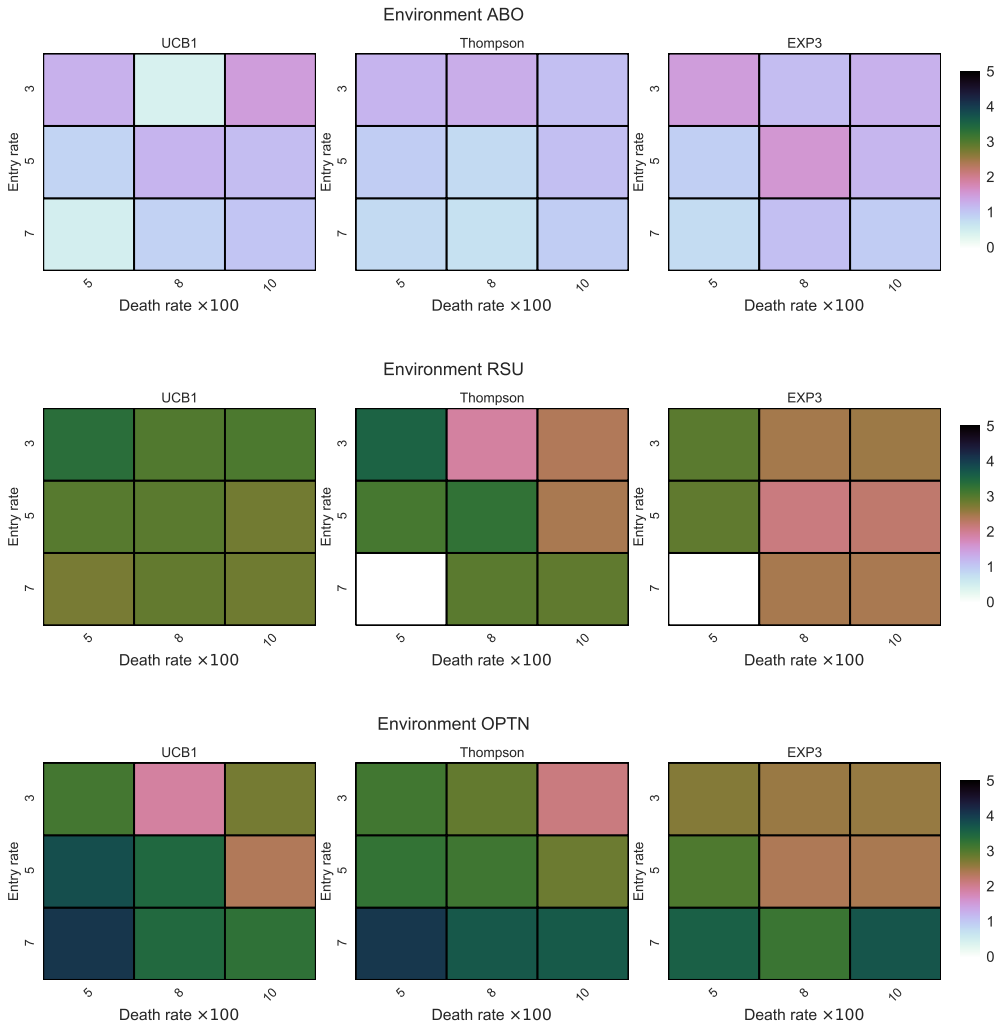


Figure 11: **Performance multi-armed bandit methods across environments** Average percent improvement by the *multi-armed bandit* method over MYOPIC. Averages were taken over 750 periods, after a 250-period burn-in sequence. See also Tables 33-**??**.

Note: In *RSU* row, white entries are missing (they are not zero).

where X_t is the matrix of observable characteristics, W^ℓ is a matrix of coefficients, A_t is the adjacency matrix associated with the kidney exchange pool at t, I is a comformable identity matrix, and σ is a nonlinear function such as the inverse logistic CDF or ReLU[14]. The authors prove that this can be interpreted as a differentiable version of the Weisfeiler-Lehman algorithm for graph isomorphism tests. For us, it is an especially convenient method because the input graphs need not be the same size throughout.

Second, the *direct prediction* method produces a probability for every pair, but it does not consider correlations between multiple pairs in the same kidney exchange pool. This is unrealistic, as it should be often the case that pairs in the same exchange should have roughly equally high or low probability of being matched. In order to rectify this shortcoming of our approach, we would like to train our statistical estimator using the sequence-to-sequence models of recurrent neural networks of

(LeCun et al., 2015). We will be turning our attention to these methods next.

Second, while we see that the performance of our *multi-armed bandit* methods was already satisfactory, we note that it uses only simulations and no data to decide which exchange to clear at each period. In that sense, it worked in a diametrically opposite way to the *direct prediction* method that used only data and no simulations. In fact, an intermediate approach, using *contextual bandits* (Lattimore and Szepesvari, 2018) could be used, blending information about data and simulations.

Thirdly, we would like to experiment with different objective functions, since, in reality, different exchanges may have different levels of desirability or priority. For example, it is common for pediatric patients and previous organ donors receive higher priority, as do *ABDR0* exchanges involving "perfectly matched"

[14] $ReLU(x) = \max(x, 0)$

patients and donors.

Lastly, we remark that there still remains a moderately large gap between what could be optimally achieved if we could perfectly predict the evolution of the pool. It is conceivable that as better predictive methods arise, this gap will become smaller.

7 Conclusion

In this paper, we examined two novel algorithms for dynamic matching in a discrete-time model of kidney exchange: a *direct prediction method* that tries to predict which pair should be matched at each period; and a *multi-armed bandit method*, that uses simulations to score available exchanges in terms of their desirability. We evaluate these methods them using simulations under a variety of different settings, and compared them in terms of average number of matched pairs per period to a MYOPIC algorithm that finds and immediately clears the maximal matching at each period.

We find that our *multi-armed bandit* method is able to uniformly dominate MYOPIC in all our simulation settings, including one where we draw pairs from the historical list of patients and donors that have undergone transplants in the United States.

Disclaimer

This work was supported in part by Health Resources and Services Administration contract 234-2005-37011C. The content is the responsibility of the authors alone and does not necessarily reflect the views or policies of the Department

of Health and Human Services, nor does mention of trade names, commercial products, or organizations imply endorsement by the U.S. Government.

8 Tables

Table 25: Blood type probabilities in the ABO environment

Pair blood type	Probability
(O, O)	0.058689
(O, A)	0.373803
(O, B)	0.158257
(O, AB)	0.042669
(A, O)	0.041119
(A, A)	0.028809
(A, B)	0.110888
(A, AB)	0.029899
(B, O)	0.017410
(B, A)	0.110888
(B, B)	0.005160
(B, AB)	0.012660
(AB, O)	0.004690
(AB, A)	0.003290
(AB, B)	0.001390
(AB, AB)	0.000380

Environment	Algorithm	Additional	Recall (TPR)	Specificity (TNR)	Precision	Accur
ABO	Gradient boosting	Both	0.834	0.818	0.225	0.
ABO	Gradient boosting	Embedding	0.841	0.826	0.234	0.
ABO	Gradient boosting	Graph stats	0.832	0.818	0.225	0.
ABO	Gradient boosting	None	0.838	0.827	0.237	0.
ABO	Logistic reg.	Both	0.802	0.787	0.194	0.
ABO	Logistic reg.	Embedding	0.735	0.797	0.188	0.
ABO	Logistic reg.	Graph stats	0.800	0.787	0.193	0.
ABO	Logistic reg.	None	0.734	0.798	0.188	0.
ABO	Random forests	Both	0.359	0.983	0.571	0.
ABO	Random forests	Embedding	0.423	0.973	0.496	0.
ABO	Random forests	Graph stats	0.360	0.983	0.577	0.
ABO	Random forests	None	0.745	0.854	0.244	0.
OPTN	Gradient boosting	Both	0.592	0.830	0.436	0.
OPTN	Gradient boosting	Embedding	0.555	0.882	0.512	0.
OPTN	Gradient boosting	Graph stats	0.597	0.828	0.435	0.
OPTN	Gradient boosting	None	0.556	0.881	0.510	0.
OPTN	Logistic reg.	Both	0.732	0.552	0.267	0.
OPTN	Logistic reg.	Embedding	0.760	0.552	0.274	0.
OPTN	Logistic reg.	Graph stats	0.730	0.553	0.266	0.
OPTN	Logistic reg.	None	0.763	0.551	0.273	0.
OPTN	Random forests	Both	0.442	0.923	0.562	0.
OPTN	Random forests	Embedding	0.450	0.918	0.550	0.
OPTN	Random forests	Graph stats	0.443	0.924	0.562	0.
OPTN	Random forests	None	0.448	0.913	0.534	0.
RSU	Gradient boosting	Both	0.744	0.793	0.441	0.
RSU	Gradient boosting	Embedding	0.732	0.810	0.457	0.
RSU	Gradient boosting	Graph stats	0.745	0.793	0.441	0.
RSU	Gradient boosting	None	0.732	0.812	0.460	0.
RSU	Logistic reg.	Both	0.779	0.634	0.316	0.
RSU	Logistic reg.	Embedding	0.778	0.629	0.314	0.
RSU	Logistic reg.	Graph stats	0.780	0.634	0.318	0.
RSU	Logistic reg.	None	0.777	0.629	0.313	0.
RSU	Random forests	Both	0.507	0.932	0.621	0.
RSU	Random forests	Embedding	0.500	0.930	0.609	0.
RSU	Random forests	Graph stats	0.504	0.933	0.621	0.
RSU	Random forests	None	0.732	0.797	0.441	0.

Table 26: **Performance of statistical methods as classifiers**
We experiment with three variations on the *direct prediction* method by implementing it using different statistical. Here we see their performance as classifiers, that is, how well they are able to predict that the node was chosen to be matched by an offline algorithm.

			Boosting		Myopic		Difference	p-value	Ratio (%)	N
			Mean	Std Error	Mean	Std Error				
Environ.	Entry	Death								
ABO	3	5	1.791	0.019	1.812	0.020	-0.021	0.036	-1.173	13
		8	2.324	0.018	2.301	0.020	0.022	0.027	0.958	15
		10	2.386	0.016	2.349	0.018	0.037	0.000	1.572	16
	5	5	2.549	0.024	2.621	0.026	-0.072	0.000	-2.746	10
		8	2.870	0.018	2.880	0.020	-0.010	0.196	-0.356	16
		10	2.948	0.018	2.948	0.020	0.000	0.976	0.010	15
	7	5	3.230	0.026	3.327	0.027	-0.097	0.000	-2.903	11
		8	3.531	0.026	3.575	0.028	-0.045	0.001	-1.248	9
		10	3.512	0.026	3.555	0.028	-0.043	0.000	-1.209	9
OPTN	3	5	2.236	0.023	2.189	0.024	0.048	0.000	2.179	10
		8	2.314	0.016	2.259	0.017	0.055	0.000	2.431	19
		10	2.348	0.017	2.322	0.018	0.026	0.007	1.112	18
	5	5	2.347	0.026	2.515	0.027	-0.168	0.000	-6.679	9
		8	2.369	0.018	2.500	0.018	-0.130	0.000	-5.210	19
		10	2.393	0.017	2.519	0.018	-0.126	0.000	-5.008	21
	7	5	3.043	0.028	3.373	0.030	-0.330	0.000	-9.791	10
		8	2.900	0.030	3.192	0.032	-0.292	0.000	-9.148	8
		10	2.860	0.030	3.141	0.032	-0.281	0.000	-8.957	8
RSU	3	5	2.380	0.025	2.417	0.025	-0.038	0.000	-1.553	10
		8	2.499	0.019	2.503	0.019	-0.004	0.552	-0.155	18
		10	2.640	0.020	2.625	0.021	0.015	0.034	0.570	16
	5	5	4.193	0.031	4.247	0.031	-0.054	0.000	-1.281	12
		8	4.134	0.027	4.140	0.027	-0.007	0.252	-0.163	15
		10	4.209	0.025	4.197	0.025	0.013	0.067	0.299	17
	7	5	5.987	0.039	6.037	0.040	-0.050	0.000	-0.836	10
		8	5.794	0.037	5.809	0.037	-0.015	0.068	-0.260	11
		10	5.924	0.043	5.918	0.043	0.006	0.334	0.097	8

Table 27: **Gradient Boosting (no graph information)**

			Boosting		Myopic		Difference	p-value	Ratio (%)	N
			Mean	Std Error	Mean	Std Error				
Environ.	Entry	Death								
ABO	3	5	1.827	0.023	1.874	0.024	-0.047	0.004	-2.489	9
		8	2.211	0.016	2.179	0.017	0.032	0.001	1.474	19
		10	2.292	0.016	2.254	0.017	0.039	0.000	1.713	18
	5	5	2.643	0.026	2.757	0.029	-0.114	0.000	-4.131	8
		8	2.823	0.018	2.870	0.020	-0.046	0.000	-1.610	16
		10	2.915	0.020	2.914	0.021	0.000	0.937	0.011	13
	7	5	3.356	0.030	3.534	0.033	-0.178	0.000	-5.039	7
		8	3.377	0.029	3.537	0.031	-0.160	0.000	-4.537	8
		10	3.432	0.028	3.516	0.030	-0.084	0.000	-2.398	8
OPTN	3	5	2.131	0.023	2.127	0.024	0.004	0.695	0.192	10
		8	2.455	0.016	2.380	0.017	0.076	0.000	3.175	17
		10	2.464	0.015	2.384	0.017	0.080	0.000	3.345	17
	5	5	2.188	0.023	2.416	0.025	-0.228	0.000	-9.437	10
		8	2.812	0.017	2.825	0.019	-0.014	0.085	-0.485	19
		10	2.881	0.017	2.848	0.018	0.033	0.001	1.143	18
	7	5	2.930	0.026	3.370	0.029	-0.440	0.000	-13.054	11
		8	3.139	0.025	3.358	0.028	-0.220	0.000	-6.543	11
		10	3.394	0.025	3.469	0.027	-0.075	0.000	-2.162	11
RSU	3	5	2.380	0.025	2.451	0.025	-0.071	0.000	-2.913	10
		8	2.547	0.022	2.574	0.022	-0.027	0.000	-1.044	14
		10	2.625	0.020	2.634	0.020	-0.009	0.148	-0.348	17
	5	5	4.183	0.032	4.269	0.033	-0.086	0.000	-2.021	10
		8	4.179	0.026	4.211	0.027	-0.032	0.000	-0.760	15
		10	4.164	0.025	4.183	0.025	-0.019	0.001	-0.449	16
	7	5	5.992	0.038	6.103	0.038	-0.111	0.000	-1.813	11
		8	5.832	0.036	5.870	0.036	-0.039	0.000	-0.660	12
		10	5.914	0.037	5.939	0.037	-0.024	0.005	-0.406	11

Table 28: **Gradient Boosting (augmented with graph information)**

			Logistic		Myopic		Difference	p-value	Ratio (%)	N
			Mean	Std Error	Mean	Std Error				
Environ.	Entry	Death								
ABO	3	5	1.035	0.016	1.185	0.018	-0.150	0.000	-12.660	10
		8	1.479	0.015	1.530	0.015	-0.051	0.000	-3.325	19
		10	1.798	0.018	1.809	0.018	-0.011	0.158	-0.594	16
	5	5	1.774	0.024	2.029	0.026	-0.255	0.000	-12.556	8
		8	1.970	0.027	2.071	0.028	-0.101	0.000	-4.873	7
		10	2.275	0.021	2.328	0.021	-0.053	0.000	-2.284	13
	7	5	2.536	0.033	2.908	0.036	-0.372	0.000	-12.792	6
		8	2.716	0.025	2.865	0.025	-0.150	0.000	-5.219	11
		10	2.859	0.029	2.942	0.030	-0.083	0.000	-2.808	8
OPTN	3	5	1.933	0.022	1.972	0.022	-0.039	0.004	-1.978	11
		8	2.270	0.017	2.232	0.019	0.038	0.000	1.712	17
		10	2.369	0.016	2.330	0.017	0.039	0.000	1.678	17
	5	5	2.717	0.027	2.795	0.029	-0.078	0.000	-2.779	9
		8	2.893	0.019	2.881	0.019	0.013	0.027	0.442	18
		10	2.922	0.019	2.879	0.020	0.043	0.000	1.490	15
	7	5	3.470	0.028	3.608	0.029	-0.138	0.000	-3.832	11
		8	3.602	0.025	3.640	0.026	-0.038	0.000	-1.049	12
		10	3.611	0.026	3.606	0.027	0.005	0.470	0.148	11
RSU	3	5	2.037	0.025	2.434	0.027	-0.397	0.000	-16.302	9
		8	2.221	0.034	2.331	0.035	-0.109	0.000	-4.688	5
		10	2.355	0.019	2.399	0.019	-0.044	0.000	-1.825	17
	5	5	3.636	0.030	4.249	0.033	-0.613	0.000	-14.429	10
		8	3.886	0.028	4.080	0.028	-0.194	0.000	-4.752	14
		10	3.966	0.026	4.031	0.025	-0.066	0.000	-1.634	16
	7	5	5.164	0.039	6.040	0.042	-0.877	0.000	-14.512	9
		8	5.635	0.038	5.875	0.039	-0.240	0.000	-4.080	10
		10	5.606	0.038	5.706	0.038	-0.100	0.000	-1.751	10

Table 29: **Logistic Regression (no graph information)**

			Logistic		Myopic		Difference	p-value	Ratio (%)	N
			Mean	Std Error	Mean	Std Error				
Environ.	Entry	Death								
ABO	3	5	1.063	0.018	1.196	0.019	-0.133	0.000	-11.114	9
		8	1.350	0.017	1.405	0.017	-0.055	0.000	-3.891	14
		10	1.638	0.019	1.658	0.019	-0.020	0.001	-1.186	14
	5	5	1.894	0.023	2.047	0.024	-0.153	0.000	-7.452	9
		8	2.170	0.019	2.224	0.019	-0.054	0.000	-2.445	15
		10	2.430	0.019	2.439	0.019	-0.010	0.109	-0.404	17
	7	5	2.745	0.034	2.963	0.035	-0.218	0.000	-7.370	6
		8	2.991	0.032	3.033	0.032	-0.043	0.001	-1.403	7
		10	3.257	0.028	3.265	0.029	-0.008	0.290	-0.235	9
OPTN	3	5	1.803	0.021	1.846	0.022	-0.043	0.025	-2.313	11
		8	2.302	0.018	2.263	0.019	0.039	0.000	1.733	16
		10	2.362	0.016	2.304	0.018	0.058	0.000	2.519	16
	5	5	2.519	0.027	2.642	0.028	-0.123	0.000	-4.664	9
		8	2.925	0.019	2.896	0.020	0.028	0.003	0.984	16
		10	2.956	0.018	2.908	0.019	0.048	0.000	1.643	18
	7	5	3.192	0.027	3.459	0.029	-0.267	0.000	-7.716	11
		8	3.604	0.028	3.604	0.029	-0.000	0.940	-0.009	10
		10	3.654	0.024	3.617	0.025	0.037	0.000	1.014	13
RSU	3	5	2.062	0.023	2.424	0.024	-0.362	0.000	-14.916	11
		8	2.164	0.021	2.296	0.021	-0.132	0.000	-5.760	14
		10	2.284	0.021	2.348	0.021	-0.065	0.000	-2.762	15
	5	5	3.660	0.035	4.215	0.037	-0.555	0.000	-13.169	8
		8	3.876	0.028	4.061	0.027	-0.186	0.000	-4.572	15
		10	3.883	0.026	3.979	0.025	-0.096	0.000	-2.424	17
	7	5	5.177	0.054	5.992	0.062	-0.815	0.000	-13.596	4
		8	5.638	0.046	5.835	0.044	-0.197	0.000	-3.376	8
		10	5.688	0.043	5.787	0.041	-0.100	0.000	-1.727	9

Table 30: **Logistic Regression (augmented with graph information)**

Environ.	Entry	Death	Forest Mean	Std Error	Myopic Mean	Std Error	Difference	p-value	Ratio (%)	N
ABO	3	5	1.173	0.019	1.240	0.020	-0.067	0.000	-5.417	8
		8	1.331	0.016	1.373	0.017	-0.042	0.000	-3.082	14
		10	1.513	0.017	1.539	0.017	-0.026	0.002	-1.672	15
	5	5	1.952	0.025	2.049	0.026	-0.097	0.000	-4.733	8
		8	2.223	0.019	2.281	0.020	-0.059	0.000	-2.573	15
		10	2.366	0.018	2.396	0.019	-0.030	0.000	-1.240	17
	7	5	2.805	0.030	2.913	0.031	-0.108	0.000	-3.715	8
		8	2.888	0.029	2.978	0.030	-0.089	0.000	-3.002	8
		10	2.990	0.025	3.049	0.026	-0.059	0.000	-1.936	11
OPTN	3	5	0.962	0.018	1.256	0.020	-0.293	0.000	-23.349	8
		8	0.815	0.011	1.169	0.014	-0.354	0.000	-30.298	19
		10	0.775	0.012	1.168	0.014	-0.394	0.000	-33.693	17
	5	5	1.566	0.025	2.185	0.030	-0.619	0.000	-28.324	7
		8	1.269	0.014	1.969	0.018	-0.701	0.000	-35.585	17
		10	1.160	0.014	1.895	0.018	-0.735	0.000	-38.766	16
	7	5	2.322	0.025	3.275	0.029	-0.953	0.000	-29.110	10
		8	1.924	0.024	2.993	0.029	-1.069	0.000	-35.731	9
		10	1.736	0.024	2.807	0.030	-1.071	0.000	-38.162	8
RSU	3	5	2.423	0.026	2.465	0.027	-0.043	0.000	-1.728	9
		8	2.670	0.022	2.664	0.022	0.006	0.368	0.213	14
		10	2.752	0.020	2.733	0.020	0.019	0.002	0.685	17
	5	5	4.175	0.037	4.214	0.037	-0.039	0.008	-0.927	8
		8	4.163	0.025	4.172	0.026	-0.009	0.145	-0.213	16
		10	4.288	0.027	4.283	0.027	0.005	0.499	0.125	15
	7	5	5.996	0.052	6.042	0.052	-0.047	0.013	-0.772	6
		8	5.915	0.041	5.909	0.041	0.006	0.455	0.102	9
		10	5.827	0.036	5.820	0.036	0.007	0.481	0.121	11

Table 31: **Random Forests (no graph information)**

Environ.	Entry	Death	Forest Mean	Std Error	Myopic Mean	Std Error	Difference	p-value	Ratio (%)	N
ABO	3	5	0.953	0.017	1.181	0.018	-0.228	0.000	-19.284	9
		8	0.934	0.013	1.152	0.014	-0.217	0.000	-18.874	15
		10	0.916	0.013	1.118	0.014	-0.202	0.000	-18.084	15
	5	5	1.719	0.025	2.066	0.027	-0.347	0.000	-16.805	7
		8	1.645	0.017	1.938	0.018	-0.293	0.000	-15.105	16
		10	1.673	0.018	1.940	0.019	-0.268	0.000	-13.796	14
	7	5	2.385	0.059	2.883	0.061	-0.497	0.049	-17.253	2
		8	2.405	0.031	2.811	0.032	-0.406	0.000	-14.460	7
		10	2.424	0.027	2.819	0.028	-0.394	0.000	-13.983	9
OPTN	3	5	0.517	0.012	1.180	0.018	-0.663	0.000	-56.214	10
		8	0.552	0.010	1.087	0.014	-0.535	0.000	-49.181	17
		10	0.619	0.010	1.111	0.013	-0.491	0.000	-44.245	18
	5	5	0.946	0.018	2.161	0.027	-1.215	0.000	-56.221	8
		8	0.909	0.012	1.962	0.018	-1.053	0.000	-53.661	17
		10	0.963	0.012	1.873	0.017	-0.910	0.000	-48.590	17
	7	5	2.173	0.028	3.313	0.032	-1.141	0.000	-34.424	9
		8	1.314	0.017	2.993	0.026	-1.679	0.000	-56.088	12
		10	1.344	0.017	2.822	0.025	-1.479	0.000	-52.392	12
RSU	3	5	2.102	0.025	2.372	0.027	-0.270	0.000	-11.369	9
		8	2.074	0.019	2.304	0.020	-0.229	0.000	-9.957	15
		10	2.087	0.018	2.301	0.019	-0.215	0.000	-9.322	16
	5	5	3.859	0.032	4.243	0.033	-0.385	0.000	-9.064	10
		8	3.744	0.025	4.070	0.026	-0.325	0.000	-7.998	15
		10	3.666	0.020	3.976	0.021	-0.311	0.000	-7.815	23
	7	5	5.564	0.052	6.027	0.051	-0.464	0.000	-7.693	6
		8	5.455	0.041	5.821	0.043	-0.365	0.000	-6.277	8
		10	5.427	0.042	5.762	0.043	-0.335	0.000	-5.816	8

Table 32: **Random Forests (augmented with graph information)**

Environ.	Entry	Death	UCB1 Mean	Std Error	Myopic Mean	Std Error	Difference	p-value	Ratio (%)	N
ABO	3.0	5.0	1.156	0.021	1.142	0.021	0.014	0.013	1.263	5
		8.0	1.121	0.026	1.116	0.026	0.005	0.369	0.418	3
		10.0	1.095	0.020	1.079	0.020	0.016	0.003	1.484	5
	5.0	5.0	2.035	0.009	2.017	0.009	0.018	0.000	0.875	63
		8.0	2.021	0.031	1.996	0.031	0.025	0.004	1.237	4
		10.0	1.967	0.025	1.945	0.025	0.022	0.001	1.133	6
	7.0	5.0	2.908	0.050	2.894	0.050	0.014	0.048	0.497	8
		8.0	2.831	0.027	2.805	0.026	0.026	0.000	0.911	12
		10.0	2.801	0.033	2.771	0.033	0.030	0.005	1.088	5
OPTN	3.0	5.0	1.196	0.020	1.160	0.020	0.036	0.000	3.107	6
		8.0	1.063	0.027	1.043	0.027	0.020	0.023	1.919	3
		10.0	0.998	0.020	0.971	0.019	0.027	0.002	2.763	5
	5.0	5.0	2.260	0.039	2.175	0.038	0.085	0.001	3.917	4
		8.0	1.981	0.025	1.914	0.025	0.066	0.000	3.469	6
		10.0	1.868	0.030	1.825	0.030	0.043	0.012	2.361	4
	7.0	5.0	3.398	0.020	3.264	0.020	0.134	0.000	4.106	34
		8.0	3.071	0.026	2.969	0.026	0.102	0.000	3.447	9
		10.0	2.880	0.037	2.790	0.037	0.090	0.000	3.225	5
RSU	3.0	5.0	2.477	0.026	2.398	0.026	0.078	0.000	3.268	8
		8.0	2.380	0.039	2.310	0.038	0.069	0.005	3.004	3
		10.0	2.268	0.029	2.201	0.029	0.067	0.000	3.057	5
	5.0	5.0	4.330	0.025	4.201	0.025	0.128	0.000	3.056	40
		8.0	4.190	0.040	4.072	0.040	0.118	0.000	2.900	8
		10.0	4.084	0.011	3.973	0.011	0.112	0.000	2.815	64
	7.0	5.0	5.519	0.370	5.370	0.399	0.148	nan	2.759	1
		8.0	6.041	0.044	5.870	0.044	0.171	0.000	2.906	19
		10.0	5.904	0.023	5.742	0.023	0.162	0.000	2.824	53

Table 33: **Multi-armed bandit algorithm UCB1**
Environ is the environment used for simulation. Entry and Death are the Poisson entry rate of entry and the the Geometric rate of departure (times 100), respectively. Mean and Std refers to the average number of matched patients over 750 periods (after 250 periods of burn-in). Difference and Ratio compare the average improvement between the algorithm and Myopic. N is the number of 1000-period simulations that used that particular configuration. Tables for other bandits are similar, and can be found in the online supplement that will be available online.

			Thompson		Myopic		Difference	p-value	Ratio (%)	N
			Mean	Std Error	Mean	Std Error				
Environ.	Entry	Death								
ABO	3.0	5.0	1.184	0.016	1.169	0.016	0.014	0.000	1.224	10
		8.0	1.138	0.027	1.123	0.027	0.015	0.037	1.308	3
		10.0	1.114	0.017	1.102	0.017	0.012	0.019	1.116	7
	5.0	5.0	2.065	0.025	2.045	0.025	0.021	0.001	1.005	12
		8.0	1.999	0.017	1.982	0.017	0.017	0.000	0.835	13
		10.0	1.938	0.012	1.916	0.011	0.022	0.000	1.134	28
	7.0	5.0	2.861	0.058	2.837	0.058	0.024	0.026	0.836	6
		8.0	2.844	0.020	2.823	0.020	0.020	0.000	0.724	20
		10.0	2.814	0.022	2.789	0.022	0.025	0.000	0.889	12
OPTN	3.0	5.0	1.197	0.020	1.161	0.020	0.036	0.000	3.133	6
		8.0	1.058	0.019	1.028	0.018	0.030	0.001	2.922	6
		10.0	1.000	0.025	0.980	0.025	0.021	0.031	2.112	3
	5.0	5.0	2.259	0.043	2.190	0.043	0.069	0.013	3.159	3
		8.0	2.023	0.022	1.962	0.021	0.061	0.000	3.098	9
		10.0	1.921	0.025	1.868	0.024	0.053	0.000	2.841	6
	7.0	5.0	3.393	0.019	3.263	0.019	0.131	0.000	4.009	40
		8.0	3.115	0.020	3.003	0.020	0.111	0.000	3.710	16
		10.0	2.910	0.032	2.806	0.031	0.103	0.000	3.684	6
RSU	3.0	5.0	2.447	0.025	2.364	0.025	0.084	0.000	3.542	8
		8.0	2.338	0.067	2.294	0.068	0.044	nan	1.920	1
		10.0	2.250	0.038	2.198	0.038	0.052	0.009	2.369	3
	5.0	5.0	4.310	0.028	4.182	0.028	0.128	0.000	3.058	32
		8.0	4.145	0.035	4.013	0.035	0.133	0.000	3.305	10
		10.0	4.076	0.029	3.975	0.029	0.100	0.000	2.527	10
	7.0	8.0	5.971	0.044	5.801	0.044	0.170	0.000	2.926	19
		10.0	5.891	0.021	5.725	0.022	0.166	0.000	2.903	59

Table 34: **Multi-armed bandit algorithm Thompson Sampling**
Environ is the environment used for simulation. Entry and Death are the Poisson entry rate of entry and the the Geometric rate of departure (times 100), respectively. Mean and Std refers to the average number of matched patients over 750 periods (after 250 periods of burn-in). Difference and Ratio compare the average improvement between the algorithm and Myopic. N is the number of 1000-period simulations that used that particular configuration.

Bibliography

(2011). In *Advances in neural information processing systems*, pages 2249–2257.

Abdulkadiroğlu, A. and Sönmez, T. (1999). House allocation with existing tenants. *Journal of Economic Theory*, 88(2):233–260.

Abraham, D. J., Blum, A., and Sandholm, T. (2007). Clearing algorithms for barter exchange markets: Enabling nationwide kidney exchanges. In *Proceedings of the 8th ACM conference on Electronic commerce*, pages 295–304. ACM.

Agrawal, R. (1995). Sample mean based index policies by o (log n) regret for the multi-armed bandit problem. *Advances in Applied Probability*, 27(4):1054–1078.

Agrawal, S. and Goyal, N. (2012). Analysis of thompson sampling for the multi-armed bandit problem. In *Conference on Learning Theory*, pages 39–1.

Akbarpour, M., Li, S., and Oveis Gharan, S. (2017). Thickness and information in dynamic matching markets.

Anderson, R., Ashlagi, I., Gamarnik, D., and Roth, A. E. (2015). Finding long chains in kidney exchange using the traveling salesman problem. *Proceedings of the National Academy of Sciences*, 112(3):663–668.

Ashlagi, I., Gamarnik, D., Rees, M. A., and Roth, A. E. (2012). The need for (long) chains in kidney exchange. Technical report, National Bureau of Economic Research.

Ashlagi, I., Gilchrist, D. S., Roth, A. E., and Rees, M. A. (2011). Nonsimultaneous chains and dominos in kidney-paired donationrevisited. *American Journal of transplantation*, 11(5):984–994.

Ashlagi, I., Jaillet, P., and Manshadi, V. H. (2013). Kidney exchange in dynamic sparse heterogenous pools. *arXiv preprint arXiv:1301.3509*.

Auer, P., Cesa-Bianchi, N., and Fischer, P. (2002). Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256.

Awasthi, P. and Sandholm, T. (2009). Online stochastic optimization in the large: Application to kidney exchange. In *IJCAI*, volume 9, pages 405–411.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B., and Varoquaux, G. (2013). API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122.

Cecka, J. (2010). Calculated pra (cpra): the new measure of sensitization for transplant candidates. *American journal of transplantation*, 10(1):26–29.

Defferrard, M., Bresson, X., and Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, pages 3844–3852.

Dickerson, J. P., Kazachkov, A. M., Procaccia, A. D., and Sandholm, T. (2017). Small representations of big kidney exchange graphs. In *AAAI*, pages 487–493.

Dickerson, J. P., Manlove, D. F., Plaut, B., Sandholm, T., and Trimble, J. (2016). Position-indexed formulations for kidney exchange. In *Proceedings of the 2016 ACM Conference on Economics and Computation*, pages 25–42. ACM.

Dickerson, J. P., Procaccia, A. D., and Sandholm, T. (2012). Dynamic matching via weighted myopia with application to kidney exchange. In *AAAI*.

Dickerson, J. P. and Sandholm, T. (2015). Futurematch: Combining human value judgments and machine learning to match in dynamic environments. In *AAAI*, pages 622–628.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232.

Hoderlein, S., Klemelä, J., and Mammen, E. (2010). Analyzing the random coefficient model nonparametrically. *Econometric Theory*, 26(3):804–837.

Kaelbling, L. P., Littman, M. L., and Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285.

Kaufmann, E., Korda, N., and Munos, R. (2012). Thompson sampling: An asymptotically optimal finite-time analysis. In *International Conference on Algorithmic Learning Theory*, pages 199–213. Springer.

Kipf, T. N. and Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Lai, T. L. and Robbins, H. (1985). Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22.

Lattimore, T. and Szepesvari, C. (2018). Bandit algorithms.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436.

Montgomery, R. A., Gentry, S. E., Marks, W. H., Warren, D. S., Hiller, J., Houp, J., Zachary, A. A., Melancon, J. K., Maley, W. R., Rabb, H., et al. (2006). Domino paired kidney donation: a strategy to make best use of live non-directed donation. *The Lancet*, 368(9533):419–421.

Organ Procurement and Transplantation Network (2013). Current cpra calculation, updated 2013.

Rapaport, F. T. (1986). The case for a living emotionally related international kidney donor exchange registry. In *Transplantation proceedings*, volume 18, pages 5–9.

Rees, M. A., Kopke, J. E., Pelletier, R. P., Segev, D. L., Rutter, M. E., Fabrega, A. J., Rogers, J., Pankewycz, O. G., Hiller, J., Roth, A. E., et al. (2009). A nonsimultaneous, extended, altruistic-donor chain. *New England Journal of Medicine*, 360(11):1096–1101.

Roth, A. E. (2015). *Who Gets Whatand Why: The New Economics of Matchmaking and Market Design*. Houghton Mifflin Harcourt.

Roth, A. E., Sönmez, T., and Ünver, M. U. (2004). Kidney exchange. *The Quarterly Journal of Economics*, 119(2):457–488.

Roth, A. E., Sönmez, T., and Ünver, M. U. (2005). Pairwise kidney exchange. *Journal of Economic theory*, 125(2):151–188.

Roth, A. E., Sönmez, T., and Ünver, M. U. (2007). Efficient kidney exchange: Coincidence of wants in markets with compatibility-based preferences. *American Economic Review*, 97(3):828–851.

Saidman, S. L., Roth, A. E., Sönmez, T., Ünver, M. U., and Delmonico, F. L. (2006). Increasing the opportunity of live kidney donation by matching for two-and three-way exchanges. *Transplantation*, 81(5):773–782.

Shapley, L. and Scarf, H. (1974). On cores and indivisibility. *Journal of mathematical economics*, 1(1):23–37.

Shuman, D. I., Narang, S. K., Frossard, P., Ortega, A., and Vandergheynst, P. (2013). The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine*, 30(3):83–98.

Thompson, W. R. (1933). On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294.

Tonelli, M., Wiebe, N., Knoll, G., Bello, A., Browne, S., Jadhav, D., Klarenbach, S., and Gill, J. (2011). Systematic review: kidney transplantation compared with dialysis in clinically relevant outcomes. *American journal of transplantation*, 11(10):2093–2109.

Ünver, M. U. (2010). Dynamic kidney exchange. *The Review of Economic Studies*, 77(1):372–414.

Wooldridge, J. M. (2010). *Econometric analysis of cross section and panel data*. MIT press.

Wu, T. T., Chen, Y. F., Hastie, T., Sobel, E., and Lange, K. (2009). Genome-wide association analysis by lasso penalized logistic regression. *Bioinformatics*, 25(6):714–721.

Zenios, S., Woodle, E. S., and Ross, L. F. (2001). Primum non nocere: avoiding increased waiting times for individual racial and blood-type subsets of kidney wait list candidates in a living donor/cadaveric donor exchange program. *Transplantation*, 72(4):648–654.