New strategies for dynamic kidney exchange

SUBMISSION 42

[[Kidney exchange is great, provides a way to increase number of lives saved]] [[Most algorithms static, don't think about future]] [[There might be ways to improve, but literature still small]] [[This work adds two contributions to the literature]] [[Direct prediction method, MAB-based methods]] [[The latter can improve the average number of kidneys exchange by 3% over the myopic algorithm]]

1 INTRODUCTION

[[Kidney exchange]]

[[Will interchangeably use the words 'node', 'vertex' and 'pair']]

Main idea. We propose and empirically evaluate two novel approaches for increasing the cardinality of matched pairs in a discrete time dynamic model of kidney exchange. Our two methods are named the *direction estimation* and *multi-armed-bandit* methods. The former 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. The latter uses simulations to select exchanges that are less likely to decrease the size of an optimal future matching. We restrict our analyses to pairwise exchanges (i.e., only two-cycles are allowed), but the ideas introduced here can be easily extended to more complex exchanges, computational constraints permitting.

Each of our two approaches faces their own challenges. In the *direct estimation* method, one challenge is that an important part of our data – the compatibility graph – lives in a non-Euclidean space, and it is not obvious how to use it as an input to a classification algorithm. We attempt to apply a variety of new methods from the graphical signal processing literature to tackle this problem. [[[An additional challenge is that traditional classification algorithms are not equipped to produce dependent labels, but in our setting whether or not we decide that a certain pair is to be matched today ought to be related to our decisions about everyone else in the pool.]]]]

In the *multi-armed bandit* method, we would like to predict how likely it is that a certain pair should be reserved for a future exchange, and match those less frequently. However, computationally costly simulations and the enormous branching factor of our problem precludes us from accurately estimating of this probability. In order to overcome this challenge, we employ multi-armed bandit (MAB) algorithms that optimally choose which exchange to collect statistics about under a relatively small computational budget.

Main results. Our methods are evaluated 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 apply our methods to each simulation setup and compare them to a MYOPIC algorithm that clears the maximal matching at each period, and to an infeasible OPT algorithm that is able to observe future states of the world and act optimally. 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 the *direct estimation* unfortunately does equally well or worse than the myopic algorithm. However, the *multi-armed bandit* method is able to improve substantially upon the MYOPIC benchmark, [[[suggesting that we could save more lives etc.]]]]

1.1 Related literature

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

Matching theory. Kidney exchange as an economic problem was first studied in a series of papers by Roth et al. [2004, 2005, 2007], but the fist paper on dynamic kidney exchange was written a few years later by Ünver [2010], who derived an optimal matching mechanism in a simple continuous time model. Two of the simulation setups in our work follows a modified, discrete-time version of the models in these seminal papers.

In the computer science literature, Awasthi and Sandholm [2009] and later Dickerson and Sandholm [2015] wrote computational alternatives to simple myopic kidney exchange. Our methods are similar to the *weighted myopia* scheme seen in these papers. Other papers that influence ours are Akbarpour et al. [2017] and Ashlagi et al. [2013].

Computer science and machine learning.

2 SIMULATION ENVIRONMENTS

Basic definitions and notation. An *agent* is an algorithm that decides who should be matched.

2.1 Common features across environments

Poisson entry, Geometric death. At each period, the number of new incoming pairs is drawn from the $Poisson(\lambda)$ distribution, where $\lambda \in \mathbb{N}$ denotes the *entry rate*, and equals the expected number of entrants per period. Upon entrance, 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[1] 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.

Binary preferences. In this work we abstract from these concerns and consider every exchange to be equally desirable, so long as it is available. This is the same assumption as used in [Roth et al., 2005].

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 patient and donor from the US population (roughly O:49%, A:36%, B:11%, AB:4%). In addition, we make an assumption previously used in Ünver [2010] to 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), 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 adjusted accordingly, e.g., the arrival rate of (A, O) pair is proportional to $0.48 \times 0.36 \times 0.11 \approx 0.019$.

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

2.3 RSU Environment

The *RSU* environment is named after the simulation model in Roth, Sönmez, and Ünver [2007]. Each pair is characterized by patient and donor ABO blood types, current waiting time, and a *calculated panel reactive antibody (cPRA)* level that governs the probability of a crossmatch with a random donor.[2] The lower 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 decades, as well as from living donors that participated in an transplant. From this original dataset, we excluded patients that were registered for more than one organ (including kidney+pancreas), patients that were not waiting for their first kidney transplant, and patients and donors with incomplete HLA profile information. The resulting dataset contained 117813 patients and 9337 living donors.

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.[3] 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; 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 1 shows the distribution across the entire population.[4]

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

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

[4]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

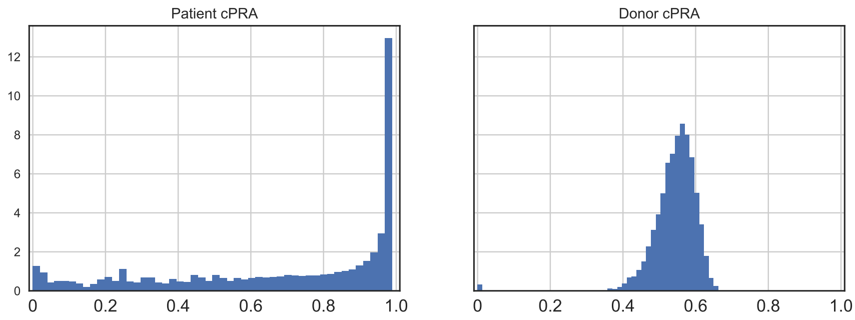

Fig. 1. **Patient and donor cPRA**
Computed from historical data. Our patient cPRA is the frequency of living donors that exhibit antigens that are unacceptable to the patient. We also define a donor cPRA by calculating the frequency of patients that have antibodies against the donors antigens.

Artificial dataset. Next, 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. Its columns are a set of dummies representing patient and donor blood times, a set of dummies for each allele in each HLA locus, and
At every simulation period, a random number of pairs are drawn from this dataset.
[[[[Algorhtm can still see original alleles. Patient and donor cPRA offer "long-run" measure of compatibility]]]]

3 METHODS

3.1 Objective and benchmarks

In this work, we focus on maximizing the average number of matched pairs over T periods, where T is a large number relative to the rates of patient entry and death. We are interested in how our new methods perform relative to the following two benchmarks.

Myopic. This algorithm finds the maximal matchings at every period, and clears it immediately. It disregards all observable characteristics of each pair, and in particular it disregards that some pairs might be useful to keep certain pairs may be easier or harder to match. In doing so, 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) does away with the uncertainty arising from the temporal structure of the problem: it knows exactly which pairs will come into the pool, as well as their arrival, and the duration of their sojourns. This algorithm is essentially a large static kidney matching problem, except that the set of available exchanges has the additional constraint that a cycle cannot exist between two vertices if their sojourns fail to overlap.

Remark*. How much is there to be improved upon?* In Figure 2, we compare Myopic and OPT for a grid of different entry and death rates. These results show that the performance gap between

have used different HLA equivalence tables; OPTN uses a more sophisticated model based on population genetics.[Organ Procurement and Transplantation Network, 2013] 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.

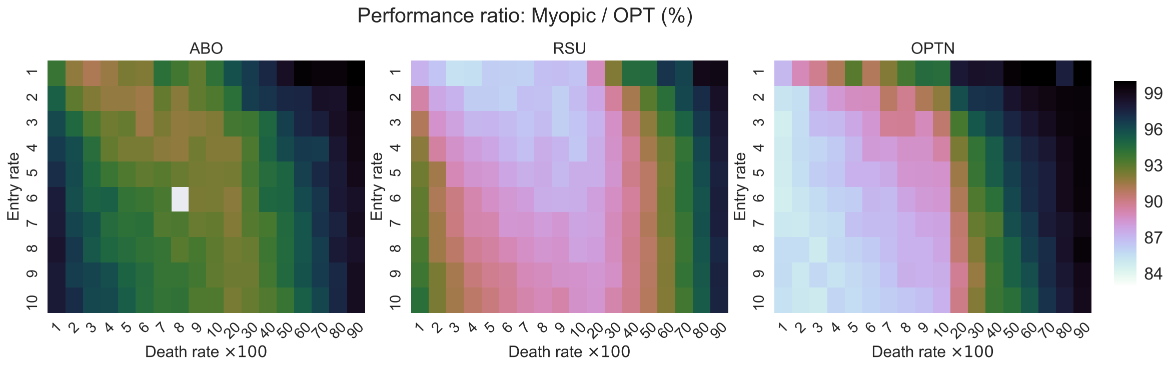

Fig. 2. **Comparing Myopic and OPT**
Ratio of average per-period matched pairs for different entry and death rates, over 3000 periods (darker hues are better). `Myopic` has better chances of achieving performances similar to `OPT` when the death rate is high (moving rightwards on the graphs), or when 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

Dynamic and static algorithms can work in tandem: the former can determine which pairs should be matched today, while the latter decides how they should be matched among themselves. The *direct prediction*, exploits this insight, essentially reducing 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 3.

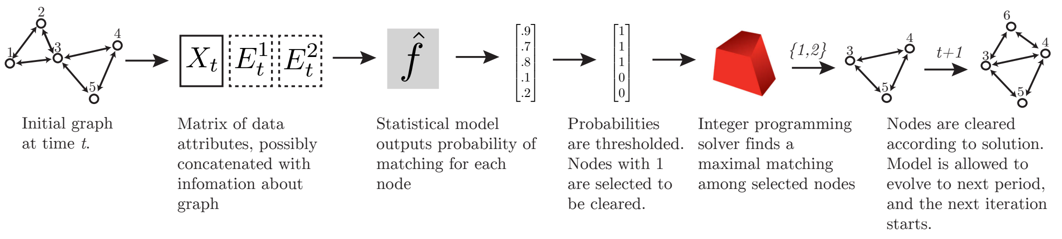
1
2
3
4
5
X_t
E^1_t
E^2_t
$\hat{f}$
.9
.7
.8
.1
.2
1
1
1
0
0
{1,2}
t+1
6
Initial graph at time *t*.
Matrix of data attributes, possibly concatenated with infomation about graph
Statistical model outputs probability of matching for each node
Probabilities are thresholded. Nodes with 1 are selected to be cleared.
Integer programming solver finds a maximal matching among selected nodes
Nodes are cleared according to solution. Model is allowed to evolve to next period, and the next iteration starts.

Fig. 3. **Direct prediction method**
One period of simulation using direct prediction methods to choose cycles. See Algorithm 1 for details.

ALGORITHM 1: Function DIRECT_PREDICTION

```
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
    $X, E_1, E_2 \leftarrow$ ENV.GET_DATA()
    // Classifier predicts matching probability for each pair using data
    $prob \leftarrow$ CLASSIFIER(X, E$_1$, $E_2$)
    // Get index of pairs whose probability is higher than threshold
    $index \leftarrow$ WHICH($prob > thres$)
    // Find maximal matching restricted to this subset
    chosen_cycles $\leftarrow$ SOLVER.SOLVE(ENV, SUBSET=$index$)
    // Return chosen cycles to be cleared
    return chosen_cycles
```

4.2 Direct prediction methods

In order to produce data to feed into our “classifier”, we repeatedly created 1000-period simulation runs and solved using them OPT using with each each node’s observable characteristics as regressors, and a binary variable indicating whether they were matched or not as the regressand. The resulting artificial dataset of approximately 2 million observations was then fed to a series of predictive algorithms, including penalized logistic regression [Wu et al., 2009], support vector machines with radial basis function kernels [Cortes and Vapnik, 1995], random forest classifiers [Breiman, 2001], gradient tree boosting classifiers [Friedman, 2001] and deep feedforward networks [Goodfellow et al., 2016].

Moreover, in order to represent information about the graph and how the nodes fit within it, we augment the original dataset using one or both of the following. First, with vertex properties for each node such as the number of in and out-edges, average neighbor degree, some centrality measures (betweenness, in-degree, out-degree, harmonic, closeness) and more exotic indices such as the core number and pagerank of each node[Newman, 2010]. Second, we augment the data using the *node2vec* graph embedding algorithm of [Grover and Leskovec, 2016], a continuous feature representation algorithm inspired by the skip-gram model.[[[More on this?]]]. These algorithms will produce auxiliary matrices E_t^1 and E_t^2 that are concatenated to the original data regressor matrix X_t, and the entire augmented data set can again be fed to the predictive algorithms above. Their performance is compared on the bottom half of Table **??**.

4.3 Multi-armed bandit methods

We have access to the data-generating process (*environments*) and an algorithm that is able to find the best matching for any data in hindsight (OPT). Therefore, in principle we could estimate the best choice of cycle to clear by simulating several periods ahead in the future, running OPT, and measuring the performance of our choice under any desired criterion.

The approach outlined above turns out to be naive and incomplete, but it essentially contains the insight under which we will be working in this section. It is 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 4 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 a procedure 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).

The literature on bandit algorithms is extensive and spans at least seventy years of research [Lattimore and Szepesvari, 2018]. Here we will focus on binary

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 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 6.

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

ALGORITHM 2: 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($h$)
    $r_1 \leftarrow$ OPT.solve(Env)
    // Remove cycle $c$, find constrained optimal matching
    Env.remove($c$)
    $r_2 \leftarrow$ OPT.solve(Env)
    // Return 1 if rewards are equal, 0 otherwise
    return $r_1 == r_2$
```

ALGORITHM 3: 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):
    // Initialize lists of current pseudo-reward statistics
    $C \leftarrow$ Env.get_available_cycles()
    $Avg \leftarrow$ zeros(length(C))
    $Std \leftarrow$ zeros(length(C))
    // Begin iterations
    for $i \leftarrow 0$ to $L$ do
        // Bandit algorithm chooses next cycle to test given statistics
        $c \leftarrow$ MAB.pull($C$, Avg, Std)
        // Compute pseudo reward for this cycle and update statistics
        $r \leftarrow$ get_pseudo_reward(Env, c, Avg, Std, h)
        $Avg \leftarrow$ update_running_average($Avg$, $r$)
        $Std \leftarrow$ update_running_std($Std$, $r$)
    end
    // Bandit algorithm chooses best cycle given statistics
    $c\_best \leftarrow$ MAB.choose($C$, Avg, Std)
    // Return best cycle, unless none of the pseudo-reward averages are above a certain threshold
    if All($Avg \leq thres$) then
        return NULL
    else
        return $c$
    end
```

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.

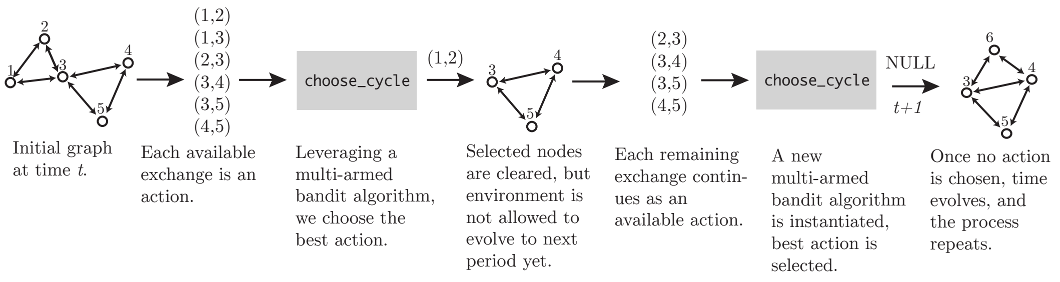
Initial graph at time *t*.
(1,2) (1,3) (2,3) (3,4) (3,5) (4,5)
Each available exchange is an action.
choose_cycle
Leveraging a multi-armed bandit algorithm, we choose the best action.
(1,2)
Selected nodes are cleared, but environment is not allowed to evolve to next period yet.
(2,3) (3,4) (3,5) (4,5)
Each remaining exchange continues as an available action.
choose_cycle
A new multi-armed bandit algorithm is instantiated, best action is selected.
NULL
t+1
Once no action is chosen, time evolves, and the process repeats.

Fig. 4. **Multi-armed bandit methods**
One period of simulation using multi-armed bandit methods to choose cycles.

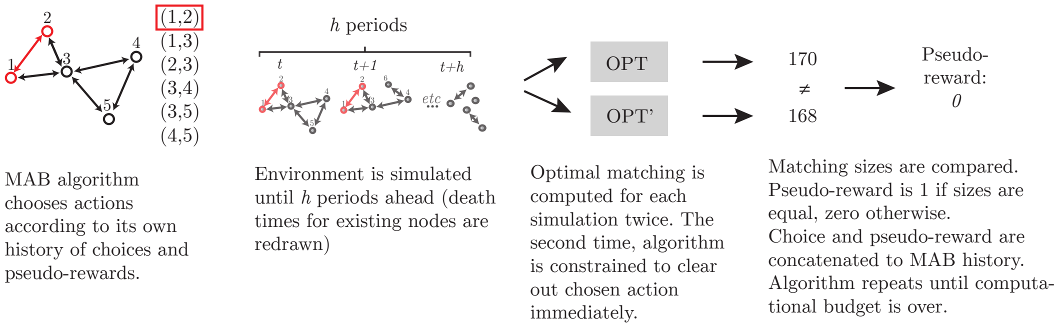
(1,2) (1,3) (2,3) (3,4) (3,5) (4,5)
MAB algorithm chooses actions according to its own history of choices and pseudo-rewards.
h periods
t
t+1
t+h
etc
Environment is simulated until *h* periods ahead (death times for existing nodes are redrawn)
OPT
OPT'
Optimal matching is computed for each simulation twice. The second time, algorithm is constrained to clear out chosen action immediately.
170
≠
168
Pseudo-reward:
0
Matching sizes are compared. Pseudo-reward is 1 if sizes are equal, zero otherwise.
Choice and pseudo-reward are concatenated to MAB history.
Algorithm repeats until computational budget is over.

Fig. 5. **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 at t	Cycles appearing between $t+1$ and $t+h$ (simulated)	Maximal matching sizes Unconstrained / Constrained	Pseudo-reward
~~(1, 2)~~, ~~(1, 3)~~, (3, 4),	~~(2, 6)~~, (3, 5), ~~(1, 5)~~, ~~(1, 6)~~, (6, 7), (3, 8), ~~(1, 9)~~, (6, 9), (7, 9), (8, 9)	8/8	1
~~(1, 2)~~, ~~(1, 3)~~, (3, 4),	~~(1, 5)~~, (4, 5), (5, 6), (4, 7), ~~(2, 8)~~, ~~(1, 7)~~, ~~(2, 7)~~, (9, 10), (10, 11)	10/8	0
~~(1, 2)~~, ~~(1, 3)~~, (3, 4),	~~(2, 6)~~, ~~(1, 7)~~, ~~(1, 8)~~, (3, 5), (3, 6), (4, 6), (4, 7), ~~(2, 8)~~, (4, 8), ~~(1, 9)~~, (7, 9)	8/8	1
~~(1, 2)~~, ~~(1, 3)~~, (3, 4),	(3, 4), (3, 5), ~~(1, 5)~~, ~~(1, 6)~~, (4, 6), (5, 6), (5, 9)	6/6	1

Fig. 6. **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 Direction prediction methods

Table ?? shows the performance results for direct prediction methods as classification methods, i.e., how accurately they are able to predict whether a node should be matched or not. Results imply the following.

First, 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. Second, augmenting the data with information about the graph has no 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.

Unfortunately, the algorithms' substandard performance as classifiers translates into poor performance as policy as well.

5.2 Multi-armed bandit methods

6 CONCLUSION

Summary.

Extensions. Another source of error is the fact that we are predicting a label for each node independently from the labels we have predicted for other nodes.

In reality, some exchanges are more or less desirable than others, either for ethical concerns or because of predicted health benefit. For example, it is common for pediatric patients and previous organ donors receive higher priority, as do exchanges involving patients with no HLA mismatch.

7 CONCLUSIONS

A SECTIONS

B SUPPLEMENTARY MATERIALS

ACKNOWLEDGMENTS

The author would like to thank Itai Ashlagi for the idea of including a donor cPRA in the OPTN environment.

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.

REFERENCES

Mohammad Akbarpour, Shengwu Li, and Shayan Oveis Gharan. 2017. Thickness and information in dynamic matching markets. (2017).

Itai Ashlagi, Patrick Jaillet, and Vahideh H Manshadi. 2013. Kidney exchange in dynamic sparse heterogenous pools. *arXiv preprint arXiv:1301.3509* (2013).

Pranjal Awasthi and Tuomas Sandholm. 2009. Online Stochastic Optimization in the Large: Application to Kidney Exchange.. In *IJCAI*, Vol. 9. 405–411.

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

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

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning* 20, 3 (1995), 273–297.

John P Dickerson and Tuomas Sandholm. 2015. FutureMatch: Combining Human Value Judgments and Machine Learning to Match in Dynamic Environments.. In *AAAI*. 622–628.

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

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep learning*. MIT press.

Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 855–864.

Tor Lattimore and Csaba Szepesvari. 2018. Bandit Algorithms. (2018). http://banditalgs.com/

Mark Newman. 2010. *Networks: an introduction*. Oxford university press.

Organ Procurement and Transplantation Network. 2013. Current CPRA calculation, Updated 2013. (2013). http://transplantpro.org/wp-content/uploads/Current-CPRA-Calculation-2.ppt

Alvin E Roth, Tayfun Sönmez, and M Utku Ünver. 2004. Kidney exchange. *The Quarterly Journal of Economics* 119, 2 (2004), 457–488.

Environment	Algorithm	Additional	Recall (TPR)	Specificity (TNR)	Precision	Accuracy
ABO	Gradient boosting	Both	0.834	0.818	0.225	0.819
ABO	Gradient boosting	Embedding	0.841	0.826	0.234	0.826
ABO	Gradient boosting	Graph stats	0.832	0.818	0.225	0.819
ABO	Gradient boosting	None	0.838	0.827	0.237	0.828
ABO	Logistic reg.	Both	0.802	0.787	0.194	0.788
ABO	Logistic reg.	Embedding	0.735	0.797	0.188	0.793
ABO	Logistic reg.	Graph stats	0.800	0.787	0.193	0.787
ABO	Logistic reg.	None	0.734	0.798	0.188	0.794
ABO	Random forests	Both	0.359	0.983	0.571	0.945
ABO	Random forests	Embedding	0.423	0.973	0.496	0.940
ABO	Random forests	Graph stats	0.360	0.983	0.577	0.946
ABO	Random forests	None	0.745	0.854	0.244	0.847
OPTN	Gradient boosting	Both	0.592	0.830	0.436	0.787
OPTN	Gradient boosting	Embedding	0.555	0.882	0.512	0.822
OPTN	Gradient boosting	Graph stats	0.597	0.828	0.435	0.786
OPTN	Gradient boosting	None	0.556	0.881	0.510	0.822
OPTN	Logistic reg.	Both	0.732	0.552	0.267	0.585
OPTN	Logistic reg.	Embedding	0.760	0.552	0.274	0.590
OPTN	Logistic reg.	Graph stats	0.730	0.553	0.266	0.585
OPTN	Logistic reg.	None	0.763	0.551	0.273	0.589
OPTN	Random forests	Both	0.442	0.923	0.562	0.835
OPTN	Random forests	Embedding	0.450	0.918	0.550	0.833
OPTN	Random forests	Graph stats	0.443	0.924	0.562	0.836
OPTN	Random forests	None	0.448	0.913	0.534	0.828
RSU	Gradient boosting	Both	0.744	0.793	0.441	0.784
RSU	Gradient boosting	Embedding	0.732	0.810	0.457	0.796
RSU	Gradient boosting	Graph stats	0.745	0.793	0.441	0.785
RSU	Gradient boosting	None	0.732	0.812	0.460	0.798
RSU	Logistic reg.	Both	0.779	0.634	0.316	0.660
RSU	Logistic reg.	Embedding	0.778	0.629	0.314	0.656
RSU	Logistic reg.	Graph stats	0.780	0.634	0.318	0.660
RSU	Logistic reg.	None	0.777	0.629	0.313	0.656
RSU	Random forests	Both	0.507	0.932	0.621	0.856
RSU	Random forests	Embedding	0.500	0.930	0.609	0.853
RSU	Random forests	Graph stats	0.504	0.933	0.621	0.856
RSU	Random forests	None	0.732	0.797	0.441	0.786

Table 1. **Traditional methods**
Performance of traditional econometric and classification methods when predicting whether a node will be matched (1) or not (0). *Additional* refers to extra information about graph structure passed to the classifier: *Graph Stats* are graph-theoretical measures of node characteristics such as centrality; *Embedding* is a 10-dimensional real embedding of the node provided by node2vec [Grover and Leskovec, 2016]. Recall is true positive rate $\frac{tp}{tp+fn}$; Specificity is true negative rate $\frac{tp}{tn+fp}$; Precision is $\frac{tp}{tp+fp}$; Accuracy is $\frac{tp+tn}{tp+tn+fp+fn}$. See section **??** for methodology and section **??** for discussion of results.

Alvin E Roth, Tayfun Sönmez, and M Utku Ünver. 2005. Pairwise kidney exchange. *Journal of Economic theory* 125, 2 (2005), 151–188.

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

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

Tong Tong Wu, Yi Fang Chen, Trevor Hastie, Eric Sobel, and Kenneth Lange. 2009. Genome-wide association analysis by lasso penalized logistic regression. *Bioinformatics* 25, 6 (2009), 714–721.

Stefanos Zenios, E Steve Woodle, and Lainie Friedman Ross. 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 (2001), 648–654.