Novel algorithms for dynamic kidney exchange

SUBMISSION 42

In recent years, kidney paired donation (KPD) has emerged as an alternative for end-stage renal disease patients with incompatible living donors. However, we argue that a larger number of patients may be reached if kidney transplant centers take into consideration how their pool of patients and donors will evolve over time. To that end, we contribute to the matching literature in dynamic settings by proposing two novel kidney exchange methods, and empirically evaluate these methods using simulations. Our objective function is the average number of matched patients per period. We restrict our experiments to pairwise kidney exchange, but the methods described here are easily extensible, computational constraints permitting. We find that at least one of our methods, based on multi-armed bandit algorithms, is able to outperform the myopic method that is used by kidney transplants in practice, sometimes by as much as 4%.

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, and patients often must wait several years before finding an available donor.

In order to overcome this challenge, in recent years *kidney paired donation* (KPD) has emerged as an alternative option for patients who have an incompatible but otherwise willing living donor. The approach involves pooling patient-donor pairs and swapping or exchanging donors so as to increase the overall number of matched patients.

Kidney paired donation was proposed about three decades ago by [Rapaport, 1986], but its implementation in the United States began only in the early 2000s. At the same time, [Roth et al., 2004] helped elevate KPD as an object of academic study, and since then it has evolved and expanded rapidly both in academic and outside of it. [Sönmez and Ünver, 2013].

Nevertheless, even today the amount of kidney exchanges remains smaller than its potential, due to several reasons. 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 and, as we will demonstrate shortly via simulations, this can lead to large losses in terms of the overall number of pairs matched over time.

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

Main idea. We propose 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 also empirically evaluated 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. The estimator used here was trained in data extracted from simulations under a variety of setups.

The main challenge with this approach is how to represent the data: while covariates such as ABO bloody type and current waiting time 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* method, uses simulations to answer the question: "if we take a particular action today, how likely are we to decrease the number of matched pairs between now and a horizon h?". Naturally, actions for which this probability is large should be left for the future, while actions for which this probability is low should be prioritized. 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 (MAB) 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 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 have to benchmarks against which we compare our methods. First, a MYOPIC algorithm that clears the maximal matching at each period. This benchmarks represents roughly what most organ clearinghouses are doing today. Second, in order to know what is the maximal payoff we could have achieve, had we been able to know the future, we also compare to an infeasible offline optimal algorithm OPT. 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 improve upon MYOPIC, but only under restricted conditions. However, the *multi-armed bandit* method is able to improve uniformly and substantially upon the MYOPIC benchmark, in particular 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.

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. Other papers that influence ours are Akbarpour et al. [2017] and Ashlagi et al. [2013].

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 the offline problem, producing a "score" for each cycle, and selecting actions with maximum score. However, our simulation system and "scoring" differ importantly in their details 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).

2 ENVIRONMENTS

Informally speaking, an *environment* is a simulation setup. It includes the rules for which pairs are deemed compatible, the entry and death processes that govern the evolution of the pool, which characteristics are observable to the algorithm at each period, and what utility the agent gets as it matches the pairs.

In our paper we have three environments inspired by previous works on dynamic kidney exchange. In what follows we will explain how they differ.

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 assumption was previously used by 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

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

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 $0.48 \times 0.36 \times 0.11 \approx 0.019$.

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 represents 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 tissue-type 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[4]; for the donors, we worked in the opposite direction by calculating the frequency of patients who exhibited antibodies against

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

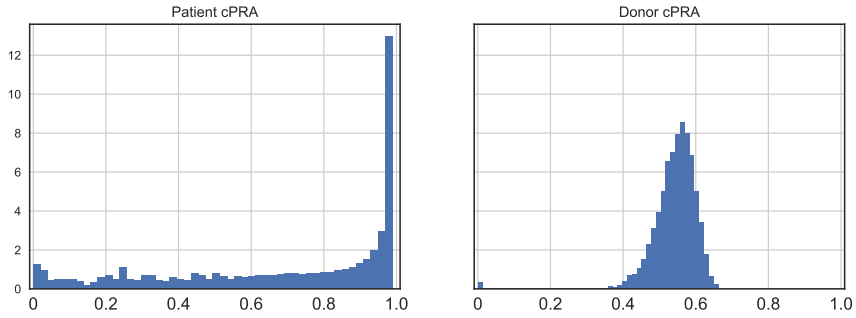

Fig. 1. **Patient and donor cPRA**
Computed from historical data. Our patient cPRA is the frequency of living donors that exhibit antigens that are unacceptable to the patient. We also define a donor cPRA by calculating the frequency of patients that have antibodies against the donors antigens.

their donor's antigens, and that became their donor cPRA. Figure 1 shows the distribution across the entire population.[5]

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

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.

[5]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]

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 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.

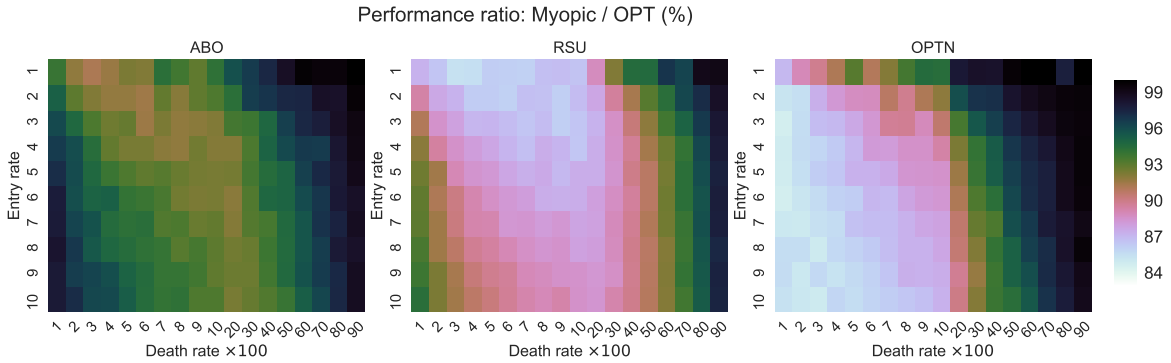

Fig. 2. **Comparing Myopic and OPT**
Ratio of average per-period matched pairs for different entry and death rates, over 3000 periods (darker hues are better). `Myopic` has better chances of achieving performances similar to `OPT` when the death rate is high (moving rightwards on the graphs), or when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

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

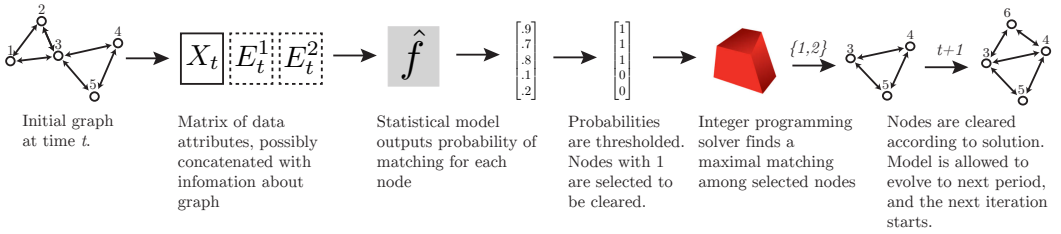

Fig. 3. **Direct prediction method**
One period of simulation using direct prediction methods to choose cycles. See Algorithm 1 for details.

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, E1, E2 ← ENV.GET_DATA()
// Classifier predicts matching probability for each pair using data
prob ← CLASSIFIER(X, E1, E2)
// 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
```

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

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 10 and 5. 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), and produce regret that grows asymptotically logarithmically with the horizon. Here, we will be experimenting with three common bandits algorithms, named *UCB1*, *EXP3* and *Thompson sampling*.

The literature on bandit algorithms is extensive and spans at least seventy years of research, and an in-depth review is outside the score of this paper. We refer the reader to [Lattimore and Szepesvari, 2018] for a survey.

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)
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
```

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 ← 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
```

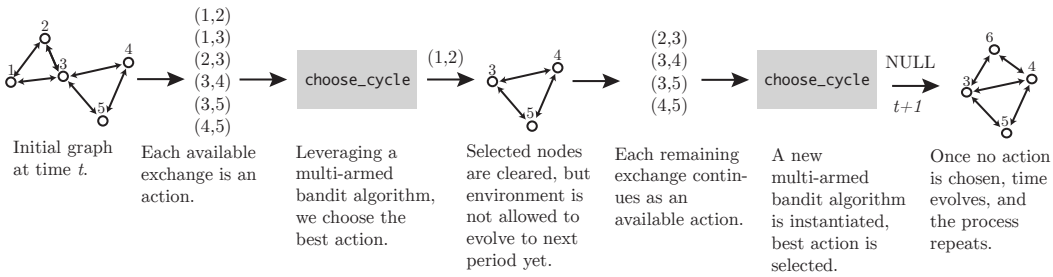

Fig. 4. **Multi-armed bandit methods**
One period of simulation using multi-armed bandit methods to choose cycles.

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.

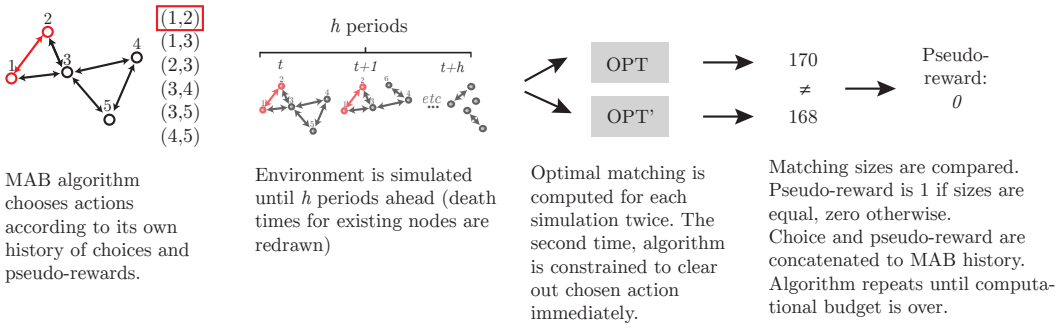

Fig. 5. **Function** GET_PSEUDO_REWARD
Multi-armed bandits evaluate if a cycle should be cleared by checking if there is a high chance that the cycle will be used in the future. Such cycles get a *lower* reward, and are left for later.

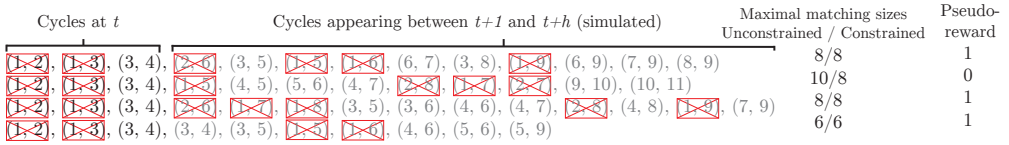

Fig. 6. **Intuition for pseudo-rewards**
In the example above, at time t we have three available cycles to choose from: $(1, 2)$, $(1, 3)$ and $(3, 4)$, and we are contemplating choosing cycle $(1, 2)$ at t. If we constrain ourselves to do so, several other future cycles become unavailable (in red). 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 $\frac{3}{4}$.

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.

5 RESULTS

5.1 Direct prediction methods

Table 1 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 3. 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

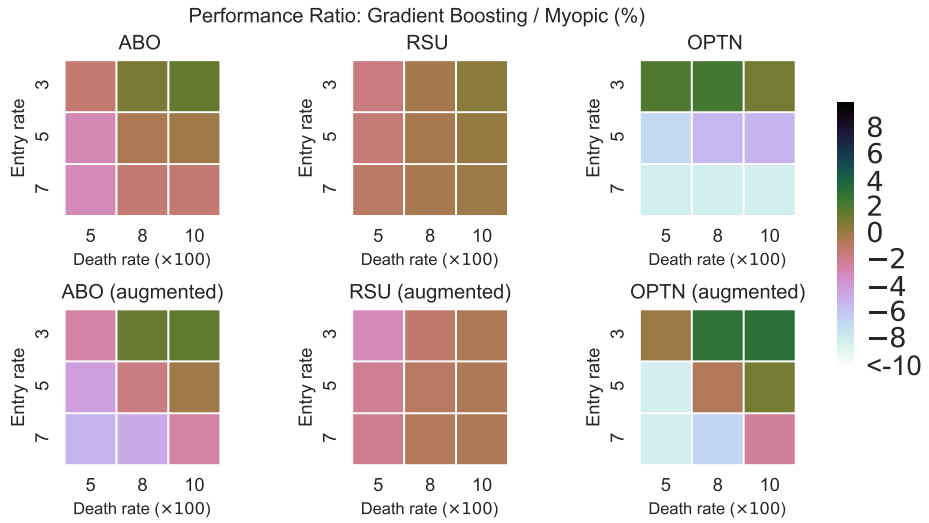

Fig. 7. **Performance of gradient boosting in direct prediction method**
Ratio of average matched patients against Myopic. Simulations ran over 750 periods (after a 250-period burn-in sequence). Lower row is when data was augmented with information about graph.

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 $thres = 3$ everywhere.

The result is shown on Figures 7-9, where we show the average performance ratio computed as above for each combination of environment, algorithm and entry/death rate configuration. [6]

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 1), and also exhibit the lowest performance in the direct prediction method.

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, except that we used Algorithms 3 and 2 when selecting nodes to be matched at each period.

The average ratio between our method and Myopic is shown on Figure 10 and elaborated upon on Table 2 in the appendix.[7] The results suggest that, for the entry and 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 2: gains from taking dynamic consideration into account are larger in situations

[6]Traditional tables are also available as part of the Supplementary Materials. They will be made available online later.
[7]For reasons of space, in the main paper we present only Table 2. The remaining tables will be made available in a supplementary online appendix.

Fig. 8. **Performance of logistic regression in direct prediction method**
Similar to gradient boosting shown in Figure 7, logistic regression is only able to perform better than Myopic when the death rate is large.

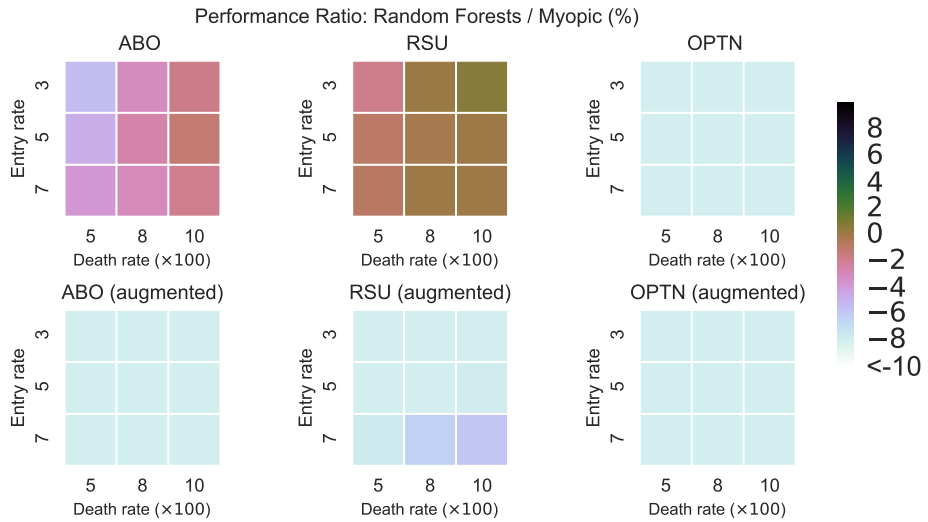

Fig. 9. **Performance of random forests in direct prediction method**
Random Forests's poor performance is likely due to its low accuracy (shown on table in online supplement)

of moderate sparsity, because that is where Myopic forces the pool to be too thin, and drives the performance away from optimality.

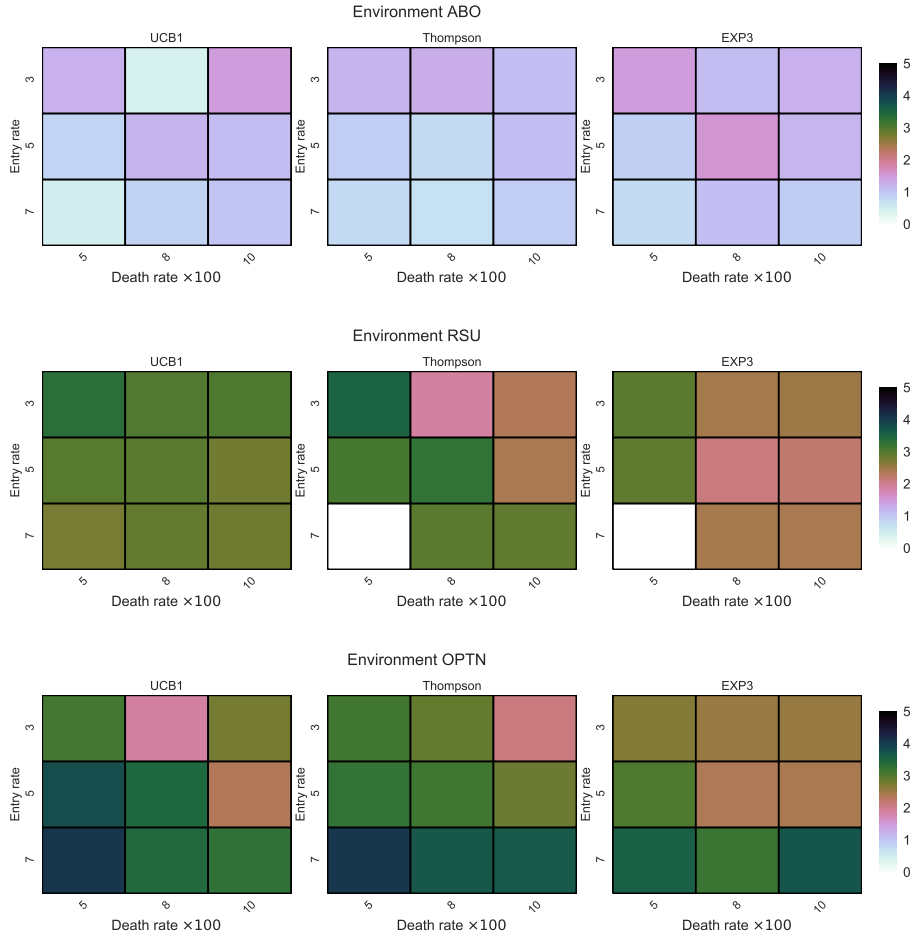

Fig. 10. **Performance multi-armed bandit methods across environments**
Average percent improvement by the *multi-armed bandit* method over Myopic. Averages were taken over 750 periods, after a 250-period burn-in sequence. See also Table 2 and additional tables in online supplement.
Note: In *RSU* row, white entries are missing (they are not zero).

6 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.

Our methods suggest a variety of extensions. First, the relatively weak performance of our *direct prediction* method could be improved in two ways: by improving our ability to use information about graph structure when predicting if a node should be matched or not; and, by making sure that what we predict for one pair should be dependent on what we predict for the other pairs in the same graph. The former could be done using new statistical methods on non-euclidean spaces [Shuman et al., 2013], and the latter using algorithms that output joint distributions of arbitrary length such as recurrent neural networks [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" 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.

ACKNOWLEDGMENTS

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

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.

John P Dickerson, Ariel D Procaccia, and Tuomas Sandholm. 2012. Dynamic Matching via Weighted Myopia with Application to Kidney Exchange.. In *AAAI*.

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.

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

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature* 521, 7553 (2015), 436.

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

Felix T Rapaport. 1986. The case for a living emotionally related international kidney donor exchange registry.. In *Transplantation proceedings*, Vol. 18. 5–9.

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

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.

David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. 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 (2013), 83–98.

Tayfun Sönmez and M Utku Ünver. 2013. Market design for kidney exchange. *The Handbook of Market Design* (2013), 93–137.

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

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.

A TABLES

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. **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.

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 2. **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.