Driving Markov Chains to Desired Equilibria via Linear Programming

Harish S. Bhat
Department of Applied Mathematics
University of California, Merced
Merced, CA, USA
hbhat@ucmerced.edu

Li-Hsuan Huang
Department of Applied Mathematics
University of California, Merced
Merced, CA, USA
lhuang33@ucmerced.edu

Sebastian Rodriguez
Department of Statistics
Northwestern University
Evanston, IL, USA
SebastianRodriguez2022@u.northwestern.edu

I. Introduction

Categorical time series arise frequently in both biomedical and sports analytics settings. For instance, [1] considers EEG measurements of infants' sleep states, where each state is an element of {"quiet sleep", "indeterminate sleep", "active sleep", "awake"}. In our previous work [2], we have modeled substitutions of players in NBA basketball games—here each state corresponds to a group of five players on the court. We consider modeling such systems using Markov chains. We must then estimate the entries of either the transition matrix, for discrete-time Markov chains (DTMCs), or the transition rate matrix, for continuous-time Markov chains (CTMCs). In both cases, we have access to closed-form maximum likelihood estimators (MLEs). If we decide to use a DTMC model, to estimate the transition probability from state i to state j, we simply count the number of transitions from i to j present in the data, and divide by the total number of transitions from i to any state. This is the MLE. The MLE is consistent: assume the data is generated by an irreducible, recurrent DTMC with transition probabilities p_{ij}. Then, given time series of length N, as $N \to \infty$, the MLE estimates $\hat{p}_{ij}$ converge to p_{ij} with probability 1. We have stated these results for DTMCs; for CTMCs, the story is analogous.

In practical settings, we have finite-length time series, which may or may not explore thoroughly the state space of the system. This can cause problems for the MLE. For instance, if a particular transition $i \to j$ is never observed, the MLE estimate $\widehat{p}_{ij}$ will be zero; depending on the application, this may lead to poor predictive power. More drastically, suppose we have time series in which we observe the system transitioning into a particular state j^* while never observing a transition from j^* to any other state. Then, for the MLE Markov chain, j^* will be an absorbing state; the equilibrium distribution of the estimated chain will assign zero probability to all states other than j^*. In this case, it is almost certain that the empirical fraction of time spent in each state will be a poor match to the long-term predictions of the estimated model.

In a different but related practical setting, we may have perfectly reasonable observations of a Markov chain whose equilibrium distribution we find suboptimal. How can we drive the Markov chain to a desired equilibrium while only perturbing as few entries of the Markov chain's transition (or transition rate) matrix as possible? The main contribution of this paper is a a pair of algorithms to solve this problem for both DTMC and CTMC models. In both cases, the model is specified by an $M \times M$ matrix $\widehat{p}$, where M is the number of states. Starting from the MLE $\widehat{p}$, we ask: what is the most sparse perturbation ε that yields a Markov chain $\widehat{p} + \varepsilon$ whose equilibrium vector matches a prescribed equilibrium vector w? We formulate this question as an optimization problem and show how to solve it efficiently using linear programming.

While we have not found published work that solves the problems outlined above, we can certainly identify subsets of the literature that help contextualize the present work. Perhaps the most relevant subset is that dealing with inverse eigenvector problems for nonnegative matrices—see [3]–[5]. Here we find techniques to construct nonnegative matrices with prescribed eigenvalues and eigenvectors. Once such a nonnegative matrix has been constructed, it can be transformed into a stochastic matrix, i.e., a transition matrix for a DTMC. It is entirely possible to use these methods to construct a DTMC whose equilibrium vector matches data, i.e., matches the empirical fraction of time spent in each state. Special algorithms to deal with this particular case of the inverse eigenvector problem have been developed [6], [7]. However, there is no link between these techniques and short-term trends in the data—for each $i \neq j$, the (i, j) entry of the resulting stochastic matrix does not necessarily have anything to do with the empirical frequency of transitions from state i to state j. In contrast, our method *constrains* the estimated model to be close to these frequencies, i.e., the MLE estimates $\widehat{p}_{i,j}$.

Note that inverse eigenvector methods can be used to construct Markov Chain Monte Carlo (MCMC) methods with optimal properties [8]. We plan to explore in future work whether the methods from the present paper can be used in the MCMC context. Note also that the inverse eigenvector problem—in which a desired eigenvalue-eigenvector pair is prescribed—is different from inverse problems for eigenvalues only [9]–[11].

The next subset of papers deals with PageRank, a fundamental algorithm used to rank web pages [12]. Consider a collection $\mathcal{C}$ of M interlinked web pages and treat each web page as a Markov chain state. If there are n links on a particular web page, we assign a probability of $1/n$ to transition from that particular web page to each of the linked pages. This gives us a Markov chain corresponding to a random walk on the graph corresponding to $\mathcal{C}$. Also consider the $M \times M$ stochastic matrix in which each entry is $1/M$—this corresponds to a Markov chain in which all states communicate uniformly with all other states. The PageRank DTMC model consists of a linear combination of the random walk and uniform transition matrices; if the weights in the linear combination sum to 1, the resulting matrix is itself a valid Markov chain. The equilibrium vector of the PageRank DTMC model can then be used to rank web pages.

There has been significant development in estimating/analyzing how this equilibrium vector changes when links between web pages are changed [13]–[15], and also in optimizing the PageRank vector according to various criteria and methods [16]–[18]. Given a desired PageRank equilibrium vector w and a PageRank DTMC model $\widehat{p}$ that has already been estimated, our algorithm computes the most sparse perturbation ε that yields a DTMC model $\widehat{p}+\varepsilon$ with equilibrium w. Suppose we treat the web graph as weighted, as in weighted PageRank [19] or certain link recommendation models [20]. Then the ε developed by our algorithm tells us how to adjust weights to achieve a desired ranking of pages.

II. Background

For the purposes of this paper, we describe Markov chains via the methods we use to generate sample trajectories. For more mathematical definitions, consult [21].

A DTMC with M states is determined by an $M \times M$ transition matrix p. Suppose the DTMC is currently in state i. We examine p_i, row i of the matrix p. To be a valid transition matrix, each p_i must be a probability mass function over the state space. To generate the next sample in the trajectory, we sample from the distribution p_i; this yields a state $i' \in \{1, 2, \dots, M\}$.

To continue, we repeat the above procedure, starting in state i'. In this way, we can generate trajectories of arbitrary length, consisting of sequences of states.

A CTMC with M states is determined by an $M \times M$ transition rate matrix p. Suppose the CTMC is currently in state i. We examine p_i, row i of the matrix p. The entries of p_i off the diagonal must be nonnegative—we treat these entries as parameters of $M-1$ independent, exponentially distributed random variables. We draw one sample from each exponential random variable. The minimum of these samples represents how long we spend in state i before transitioning. The argmin of these samples represents the new state i' to which we transition.

To continue, we repeat the above procedure, starting in state i'. In this way, we can generate trajectories of arbitrary length, consisting of sequences of states and corresponding transition times.

III. Mathematical Methods

A. Discrete-Time Optimization Problem/Solution

Suppose we have an $M \times M$ DTMC transition matrix $\widehat{p}$. We seek a perturbation ε such that $\widehat{p}+\varepsilon$ has a desired equilibrium. We still want $\widehat{p}+\varepsilon$ to be a valid Markov transition matrix, so we require that

$$\sum_j (\widehat{p}_{i,j} + \varepsilon_{i,j}) = 1. \tag{1}$$

Since $\sum_j \widehat{p}_{i,j} = 1$, we have $\sum_j \varepsilon_{i,j} = 0$. So, we only need to solve for the off-diagonal part of ε; then

$$\varepsilon_{i,i} = -\sum_{j \neq i} \varepsilon_{i,j}. \tag{2}$$

We constrain the equilibrium vector to be w: $w^T(\widehat{p}+\varepsilon) = w^T$. In coordinates, this reduces to

$$\sum_{i \neq j} w_i \varepsilon_{i,j} - \sum_{i \neq j} w_j \varepsilon_{j,i} = w_j - \sum_i w_i \widehat{p}_{i,j}.$$

The entries of the perturbed matrix must also be valid probabilities: $0 \leq \widehat{p}_{i,j} + \varepsilon_{i,j} \leq 1$ for all i,j. For $i \neq j$, we can enforce this constraint verbatim. For $i = j$, we use (2) to rewrite the constraint as:

$$0 \leq \widehat{p}_{i,i} - \sum_{j \neq i} \varepsilon_{i,j} \leq 1.$$

For each i, we enforce the lower bound of 0:

$$\sum_{j \neq i} \varepsilon_{i,j} \leq \widehat{p}_{i,i}.$$

However, for the upper bound, we replace 1 by w_i and require, for each i,

$$-\sum_{j \neq i} \varepsilon_{i,j} + \widehat{p}_{i,i} \leq w_i. \tag{3}$$

In this paper, we will take w to be a candidate equilibrium distribution with no absorbing states, i.e., $0 < w_i < 1$ for each i. Hence (3) and (2) guarantee that $\varepsilon_{ii} + \widehat{p}_{ii} \leq \max_i w_i < 1$. In short, the diagonal entries of the perturbed transition matrix $\widehat{p}+\varepsilon$ are bounded away from 1, implying that the perturbed system cannot have any absorbing states.

We also enforce (3) because we know that

$$\widehat{p}+\varepsilon = \begin{bmatrix} \text{—}w\text{—} \\ \vdots \\ \text{—}w\text{—} \end{bmatrix} \tag{4}$$

automatically has equilibrium distribution w. This observation proves that our optimization problem's feasible set is nonempty.

We now consider the choice of objective function. We seek a transition matrix $\widehat{p}+\varepsilon$ that retains as many of the original MLE entries $\widehat{p}$ as possible. In other words, we want to maximize the number of zero entries of ε. Consequently, we set our objective function equal to the sparsity-promoting 1-norm of the off-diagonal elements of ε:

$$J(\varepsilon) = \sum_{i,j:i \neq j} |\varepsilon_{i,j}|.$$

Putting the objective and constraints together, we are led to

$$\begin{aligned} \min_{\varepsilon} \quad & J(\varepsilon) \\ \text{s.t.} \quad & 0 \le \widehat{p}_{i,j} + \varepsilon_{i,j} \le 1, \ \forall i \neq j \\ & 0 \le -\sum_{j \neq i} \varepsilon_{i,j} + \widehat{p}_{i,i} \le w_i, \ \forall i \\ & \sum_{i \neq j} w_i \varepsilon_{i,j} - \sum_{i \neq j} w_j \varepsilon_{j,i} = w_j - \sum_i w_i \widehat{p}_{i,j}, \ \forall j \end{aligned} \tag{5}$$

To handle the 1-norm, we employ the standard technique of introducing additional decision variables $t_{i,j}$ and inequality constraints, resulting in a problem equivalent to (5):

$$\begin{aligned} \min_{\varepsilon, t} \quad & \sum_{i \neq j} t_{i,j} \\ \text{s.t.} \quad & -t_{i,j} \le \varepsilon_{i,j} \le t_{i,j}, \ \forall i \neq j \\ & t_{i,j} \ge 0, \ \forall i \neq j \\ & 0 \le \widehat{p}_{i,j} + \varepsilon_{i,j} \le 1, \ \forall i \neq j \\ & 0 \le -\sum_{j \neq i} \varepsilon_{i,j} + \widehat{p}_{i,i} \le w_i, \ \forall i \\ & \sum_{i \neq j} w_i \varepsilon_{i,j} - \sum_{i \neq j} w_j \varepsilon_{j,i} = w_j - \sum_i w_i \widehat{p}_{i,j}, \ \forall j. \end{aligned} \tag{6}$$

This is a linear program [22], which we solve using CVXOPT [23] and Mosek [24]. We are committed to making our Python source code available—see https://github.com/hbhat4000/MCOresults.

B. Continuous-Time Optimization Problem/Solution

Let us now consider the problem for CTMC models. Suppose we have an $M \times M$ transition rate matrix $\widehat{\alpha}$ estimated using MLE. Suppose that $\widehat{\alpha}$ has absorbing states.

We seek a perturbation ε such that $\widehat{\alpha}$ has no absorbing states. For the continuous-time Markov chain, this means that for each i, we must have

$$\sum_{j \neq i} (\widehat{\alpha} + \varepsilon)_{i,j} > 0.$$

If this sum were to be zero, then state i would be an absorbing state for the continuous-time Markov chain. Note, however, that numerical optimizers do not distinguish between strict and non-strict inequalities. To guarantee positivity, we instead use the constraint

$$\sum_{j \neq i} (\widehat{\alpha} + \varepsilon)_{i,j} \ge \delta > 0$$

for some tolerance δ. Note that for a continuous-time Markov chain, we always choose the diagonal elements of the transition rate matrix to guarantee that the total row sum equals zero. For this reason, we only need to solve for the off-diagonal elements of ε, just as in the discrete-time case.

Assume that w is a desired equilibrium vector, again with $0 < w_i < 1$ for all i. For the perturbed system to have w as its equilibrium, we must have

$$w^T (\widehat{\alpha} + \varepsilon) = 0.$$

Let $\mathcal{Y} = \{y_1, \ldots, y_M\}$ denote all row sums—not including the diagonal element—of the estimated transition rate matrix $\widehat{\alpha}$. We set $\delta_1 = \min \mathcal{Y} \cap \{r \ : \ r > 0\}$, the minimum positive entry from $\mathcal{Y}$. We also set $\delta_2 = \min_{1 \le i \le M} (1 - w_i)$. Finally, we set $\delta = \min\{\delta_1, \delta_2\}$. By setting δ in this way, we ensure that a system that has *no* absorbing states and that already achieves the desired equilibrium vector w will result in a solution of $\varepsilon = 0$. We also ensure that there exists a feasible solution—with desired equilibrium w—given by

$$\widehat{\alpha} + \varepsilon = \begin{bmatrix} \text{—}w\text{—} \\ \vdots \\ \text{—}w\text{—} \end{bmatrix} - I, \tag{7}$$

where I is the $M \times M$ identity matrix. Additionally, we require that the perturbed transition rates are nonnegative: for all $i \neq j$,

$$\widehat{\alpha}_{i,j} + \varepsilon_{i,j} \ge 0.$$

Because we are interested in retaining as many of the MLE entries of $\widehat{\alpha}$ as possible, we again seek a perturbation ε that is as sparse as possible. Therefore, we use the 1-norm objective function J defined above. Putting the objective and constraints together, we are led to the following optimization problem:

$$\begin{aligned} \min_{\varepsilon} \quad & J(\varepsilon) \\ \text{s.t.} \quad & \widehat{\alpha}_{i,j} + \varepsilon_{i,j} \ge 0, \ \forall i \neq j \\ & \sum_{j \neq i} (\widehat{\alpha} + \varepsilon)_{i,j} \ge \delta > 0, \ \forall i \\ & -w_j \sum_{i \neq j} (\widehat{\alpha} + \varepsilon)_{j,i} + \sum_{i \neq j} w_i (\widehat{\alpha} + \varepsilon)_{i,j} = 0, \ \forall j. \end{aligned} \tag{8}$$

We again employ the standard technique of introducing decision variables $t_{i,j}$ and additional constraints. The resulting optimization problem, equivalent to the one above, is:

$$\begin{aligned} \min_{\varepsilon, t} \quad & \sum_{i \neq j} t_{i,j} \\ \text{s.t.} \quad & -t_{i,j} \le \varepsilon_{i,j} \le t_{i,j}, \ \forall i \neq j \\ & t_{i,j} \ge 0, \ \forall i \neq j \\ & \widehat{\alpha}_{i,j} + \varepsilon_{i,j} \ge 0, \ \forall i \neq j \\ & \sum_{j \neq i} (\widehat{\alpha} + \varepsilon)_{i,j} \ge \delta > 0, \ \forall i \\ & -w_j \sum_{i \neq j} (\widehat{\alpha} + \varepsilon)_{j,i} + \sum_{i \neq j} w_i (\widehat{\alpha} + \varepsilon)_{i,j} = 0, \ \forall j. \end{aligned} \tag{9}$$

Again, this is a linear program [22]; we solve it using the same software described above.

IV. Simulated Data Tests

For each simulated data test, we generate data from a known DTMC. When we apply MLE to this data, we obtain a DTMC $\widehat{p}$, which we refer to as the *naive* model. We then apply the optimization method from Section III, resulting in a DTMC $\widehat{p} + \varepsilon$, which we refer to as the *fixed* model.

A. Metrics

There are two metrics in which we compare naive and fixed models. The first, which we call *long-term error*, measures the ability of the model to predict well in a distributional sense. Given a time series, let π^{emp} denote the empirical fraction of time spent in each state. Given a DTMC model with transition matrix P, we compute π^{mod} by solving $\pi^{\text{mod}} P = \pi^{\text{mod}}$; for a CTMC model with transition rate matrix P, we instead solve $\pi^{\text{mod}} P = 0$. The long-term error is the L^1 distance between the empirical and model distributions:

$$\mathcal{E}_{\text{LT}} = \|\pi^{\text{mod}} - \pi^{\text{emp}}\|_1 = \sum_i \left|\pi_i^{\text{mod}} - \pi_i^{\text{emp}}\right|. \tag{10}$$

The second metric, *short-term error*, measures the model's ability to predict the very next element of a time series. Suppose we have a DTMC and data consisting of a sequence of states $\{s_0, s_1, s_2, \ldots, s_L\}$. For each $i \in \{0, 1, \ldots, L-1\}$, we use the Markov chain to compute a one-step prediction $\widehat{s}_{i+1}$ given the current state s_i. The mean short-term error is

$$\mathcal{E}_{\text{ST}} = \frac{1}{L} \sum_{i=0}^{L-1} 1_{\widehat{s}_{i+1} \neq s_{i+1}}. \tag{11}$$

Here 1_X is the indicator function that equals 1 if condition X is true and 0 otherwise. For both the DTMC and CTMC models, we compute the prediction $\widehat{s}_{i+1}$ by sampling from the Markov chain conditional on currently being in state s_i. In both cases, this amounts to examining row s_i of the transition (or transition rate) matrix P and applying the appropriate sampling procedure from Section II.

Test Series Length	5×10^2	10^3	10^4	10^5	10^6
DTMC States					
4	0.019	0.040	0.009	0.008	0.016
8	0.069	0.090	0.034	0.015	0.024
16	0.086	0.086	0.050	0.026	0.028
32	0.185	0.179	0.053	0.048	0.061
64	0.286	0.214	0.080	0.051	0.061
128	0.447	0.285	0.130	0.099	0.086
256	0.585	0.422	0.172	0.136	0.135
CTMC States					
4	0.051	0.058	0.006	0.035	0.004
8	0.123	0.068	0.029	0.022	0.023
16	0.177	0.137	0.053	0.058	0.032
32	0.279	0.160	0.091	0.065	0.069
64	0.383	0.289	0.103	0.096	0.079
128	0.526	0.400	0.180	0.136	0.122
256	0.748	0.622	0.258	0.189	0.170

TABLE I

LONG-TERM TEST ERRORS FOR BOTH FIXED DTMC MODELS (FIRST 6 ROWS) AND FIXED CTMC MODELS (LAST 6 ROWS). CORRESPONDING LONG-TERM TEST ERRORS FOR NAIVE MODELS ARE ALL ≈ 2.

B. Long-Term Error Test

To show that our method reduces long-term prediction error, we carry out the following test. We fix the number of states M. We randomly generate an $(M-1)$-state Markov chain that we then sample to create both a training time series of length 10^4 and a test time series of length $L \in \{5 \times 10^2, 10^3, 10^4, 10^5, 10^6\}$. For the training time series, we tack on a final transition into state M.

Simulations	50	250	500	1000
DTMC States				
4	1.57e-02	1.91e-02	1.89e-02	2.05e-02
8	4.65e-03	3.15e-03	3.70e-03	3.62e-03
16	1.73e-03	6.74e-04	8.84e-04	8.94e-04
32	2.27e-04	2.60e-04	1.76e-04	8.70e-05
64	6.00e-05	2.41e-04	1.81e-04	1.38e-04
128	4.33e-04	3.06e-04	2.18e-04	8.30e-05
256	2.53e-04	1.95e-04	9.00e-05	7.80e-05
CTMC States				
4	5.60e-04	9.83e-04	1.03e-03	1.14e-03
8	1.81e-03	3.20e-04	6.38e-04	5.85e-04
16	6.80e-04	6.30e-05	1.40e-05	1.04e-04
32	1.32e-03	1.40e-04	1.84e-04	3.42e-04
64	6.33e-04	4.80e-05	6.00e-06	6.20e-05
128	3.30e-05	7.40e-05	5.80e-05	3.80e-05
256	2.00e-05	1.15e-04	1.13e-04	1.95e-04

TABLE II

ABSOLUTE DIFFERENCES BETWEEN FIXED AND NAIVE SHORT-TERM TEST ERRORS FOR DTMC (FIRST 7 ROWS) AND CTMC (LAST 7 ROWS) MODELS, SHOWING ONLY A MINIMAL CHANGE IN SHORT-TERM PREDICTIVE POWER.

We use the training set to estimate both $\widehat{p}$, the naive MLE model, and π^{emp}, the empirical fraction of time spent in each state. We then produce a fixed model by applying the optimization procedure from Section III to $\widehat{p}$, setting the desired equilibrium vector w equal to π^{emp}. We carry out all of this for both discrete- and continuous-time models.

Using both training and test time series, we compute the long-term error $\mathcal{E}_{\text{LT}}$—see (10)—for all fixed models. In Table I, we display long-term test errors for, respectively, fixed DTMC and fixed CTMC models. *We omit training errors, which are all on the order of 10^{-16}, indicating that the optimizer succeeds in creating Markov models with desired equilibria.*

We see from Table I that, for each fixed number of states M, as the length of the test set increases, the long-term test error decreases. As more time passes, we observe an increasingly better match between the fixed Markov chain's equilibrium and the empirical fraction of time spent in each state.

If we were to create a table analogous to Table I for the corresponding naive DTMC models, every entry would be close to 2.0, the maximum possible value of $\mathcal{E}_{LT}$. This is because state M is the unique absorbing state for the naive model; therefore, $\pi^{\text{mod}} = (0, \ldots, 0, 1)$. Meanwhile, the M-th entry of the π^{emp} vector is either nearly zero (for the training set) or identically zero (for the test set).

C. Short-Term Error Test

In the quest to reduce long-term prediction error, we do not want to discard the short-term predictive accuracy of the MLE. Because we have used the sparsity promoting 1-norm objective function for ε, we expect that most entries of ε will be zero. If this is the case, the fixed model will retain many of the MLE entries from the naive model, and therefore the fixed model's short-term predictive power should not differ greatly from that of the naive model.

	Dim	**NNZ**	**CV**	**Dim**	**NNZ**	**CV**
Team						
Atl	72092	240	2.167e-18	112560	274	4.445e-17
Bkn	67340	164	1.484e-18	105950	272	4.888e-18
Bos	90902	237	8.345e-19	114582	277	5.117e-18
Cha	35532	185	1.775e-18	68382	263	2.957e-18
Chi	54522	223	1.842e-18	151710	286	1.045e-17
Cle	72630	243	3.305e-18	119370	275	4.916e-18
Dal	110556	208	1.935e-18	213906	364	3.374e-18
Den	69432	144	8.410e-19	144020	344	2.740e-18
Det	13806	114	2.252e-18	35910	178	1.107e-18
GS	70490	170	2.479e-18	120756	286	8.207e-18
Hou	73170	263	2.690e-18	141000	2040	1.440e-17
Ind	58806	171	2.730e-18	118680	280	3.404e-17
LAC	55932	156	3.320e-18	97032	258	4.264e-18
LAL	43056	132	2.267e-18	68382	216	2.339e-18
Mem	59292	238	2.149e-18	181902	302	4.993e-18
Mia	63252	1256	6.223e-17	114582	1866	6.285e-17
Mil	73170	1241	2.534e-17	104652	237	3.334e-18
Min	57360	202	2.542e-18	93942	235	3.524e-18
NO	93942	193	2.229e-18	172640	2373	5.705e-17
NY	56882	183	3.909e-18	99540	267	4.578e-18
OKC	48180	190	1.604e-18	90300	234	1.641e-18
Orl	57840	235	1.326e-18	134322	306	8.248e-18
Phi	175980	280	9.187e-18	270920	389	4.804e-18
Pho	93330	171	2.320e-18	234740	328	3.836e-18
Por	34040	1046	7.235e-17	45582	236	1.085e-18
SA	69432	243	5.111e-18	141000	316	1.709e-17
Sac	66306	243	1.382e-18	102720	286	4.837e-18
Tor	30102	165	4.034e-18	46010	204	3.328e-18
Uta	113232	288	1.172e-17	199362	384	8.124e-18
Was	87912	259	2.558e-18	146306	317	4.125e-18

TABLE III
FOR EACH TEAM, WE COMPUTE FIXED CTMC MODELS USING TRAINING SETS OF EITHER THE FIRST 40 (LEFT OF BAR) OR 60 (RIGHT OF BAR) NON-OVERTIME, REGULAR SEASON GAMES. FOR EACH FIXED MODEL, WE REPORT DIM, THE DIMENSION OF ε, EQUAL TO $M(M-1)$ WHERE M IS THE NUMBER OF STATES OR UNIQUE 5-PERSON UNITS IN THAT TEAM'S TRAINING SET. WE REPORT NNZ, THE NUMBER OF NONZERO ENTRIES IN THE COMPUTED ε—THE SMALL VALUES OF NNZ RELATIVE TO DIM SHOW THAT THE COMPUTED SOLUTIONS ARE HIGHLY SPARSE. FINALLY, WE RECORD CV, THE MAXIMUM CONSTRAINT VIOLATION REPORTED BY THE OPTIMIZER—ALL VALUES ARE CLOSE TO ZERO.

To test this expectation, we run the following simulation procedure: for fixed M, we randomly generate an $(M-1)$-state Markov chain that we then sample to create both a training time series of length 10^3 and a test time series of length 3×10^3. For the training time series, we append a final transition into state M. We then estimate naive and fixed models using the training set only. For both naive and fixed models, we then use the test set to compute short-term errors $\mathcal{E}_{\text{ST}}$—see (11). We record the absolute difference between the short-term error of the naive and fixed models. Again, we carry out all of the above for both discrete- and continuous-time models.

We repeat the above simulation procedure up to 1000 times. The results, displayed in Table II, show that each fixed model's short-term predictive power does not differ greatly from that of the corresponding naive model. As we proceed down the table, we see smaller differences between the short-term errors of the naive and fixed models. This is due to larger state spaces yielding more sparse solutions for ε. For the largest state space ($M = 256$), the dimension of ε is $M(M-1) = 65280$; for systems of this size, it is typical that less than one percent of the entries returned by the optimizer are nonzero.

V. Real Data Tests

In this section, we apply the optimization procedures from Section III to Markov models estimated from real data sets in the areas of basketball analytics and the life sciences.

A. NBA Data

We begin with a description of the raw data itself. For each regular-season game from the 2015-16 NBA season, we obtained play-by-play files (in HTML form) from public web sites. These files contain time-stamped textual markers that we mined to determine who was playing on the court at all times for all games. Each team plays 82 games per season. Starting with all 1230 regular-season games, we omitted games that went to overtime. Hence all games in the data set considered here lasted 48 minutes (2880 seconds). For each team, we assigned a state number to each unique 5-person unit that played at any time for that team. Using these state numbers and the time stamps at which substitutions occur, we then extracted from each game a trajectory $\{(t_i, s_i)\}_{i=0}^{N}$, where $t_0 = 0$ and $t_N = 2880$. At each time t_i, the system transitions from state s_{i-1} to state s_i, corresponding to a substitution of one or more players on the court for one team only.

Our goal here is to use this data to build CTMC models that predict how long each 5-person unit plays on the court. Once we fix the sizes of the training and test sets, we build one naive CTMC model for each team. We find that all naive models—including those trained on all non-overtime games—features at least one absorbing state; in many cases, several absorbing states exist. Using the empirical fraction of time spent in each state, computed using the corresponding training set, we applied the optimization procedure from Section III to generate fixed CTMC models for each team.

In Figure 1, we plot training set result for the naive (left) and fixed (right) CTMC models. For these results only, the training set consists of all non-overtime games from the entire regular season. Each point on each plot corresponds to one 5-person unit for one team. We see that the optimization procedure results in a fixed model in which the naive model's training error has been reduced to nearly zero.

For the remainder of this section, we consider naive and fixed CTMC models trained on proper subsets of the regular season. We build naive and fixed CTMC models using a training set of either the first 40 or first 60 non-overtime games played by each team. The corresponding test sets consist of the remaining non-overtime games in the season, less than or equal to 42 or 22 games, respectively.

In Table III, we examine the performance of our optimization procedure applied to CTMC models trained on, respectively, the first 40 and 60 non-overtime games. The tables show that the optimizer succeeds in finding highly sparse solutions that satisfy all constraints—removing absorbing states and achieving the desired equilibrium. The sparsity is indicated by the small number of nonzero entries of the computed ε compared to its dimension, which equals $M(M-1)$. Note also that M, the

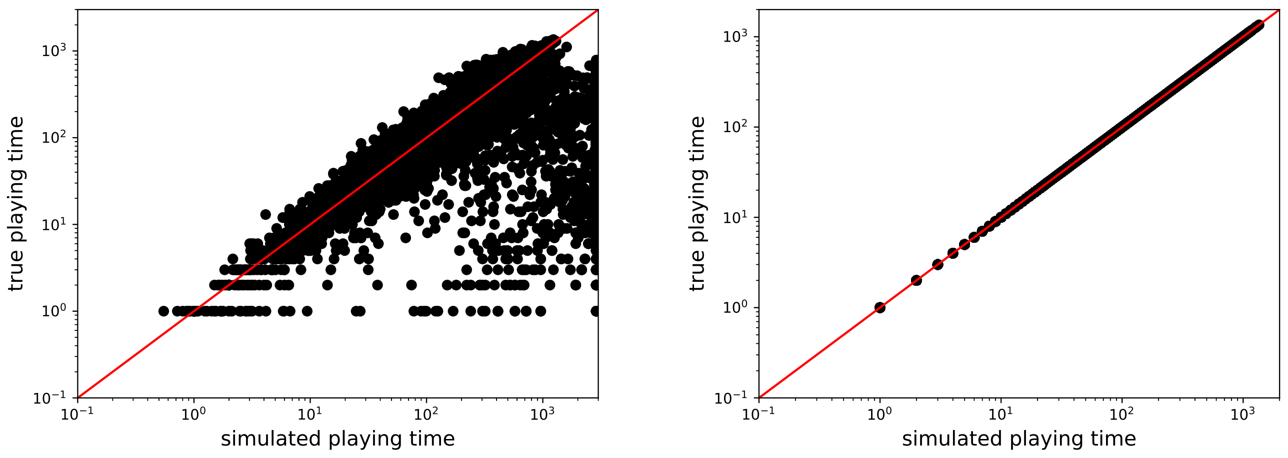

Fig. 1. We plot simulated versus true (training set) playing times for naive (left) and fixed (right) CTMC models. Each point on each plot corresponds to one 5-person unit for one team. We see that the fixed model features practically zero training error, dramatically reducing the training error from the naive model.

total number of unique 5-person units that appear for each team in each training set, is always in the hundreds.

In Figure 2 (40-game) and Figure 3 (60-game), we plot test set results for naive (left) and fixed (right) CTMC models with differing training set sizes. Each point on each plot corresponds to a 5-person unit for a given team. For each team, the predicted playing times for each 5-person unit correspond to the entries in the equilibrium vector for the CTMC model for that team, scaled by the number of seconds in the test set for that team. We plot the real time played by each 5-person unit versus the predicted playing time. Note that there is a massive concentration of points near $(0, 0)$; we have decided against log-scaled axes in order to give a sense of the range of the predictions and true times in natural units. For the naive 40-game model, the RMSE (root mean-squared error) between all predicted and real times is 2.979. For the fixed 40-game model, the RMSE is 1.178, approximately 60.4% less than the naive model. This RMSE is measured in thousands of seconds across roughly 40 games; in minutes per game, the 40-game fixed model's average error is 0.49.

The naive 60-game model's RMSE is 1.584, while the fixed 60-game model's RMSE is 0.602, approximately 61.9% less. This RMSE is measured in thousands of seconds across roughly 20 games; in minutes per game, the 60-game fixed model's average error is 0.50.

Hence each CTMC predicts each 5-person unit's per-game playing time to within half a minute, on average. We consider this to be an excellent result.

B. *Biomedical data*

Holson and preproglucacon are discrete-time data sets from the R package `markovchain` [25]. The Holson data set contains life history trajectories for 1000 unique patients, each measured at 11 points in time. The measurement at each time has value 1, 2, or 3. We split the data into training and test sets of size 500 each. We fit a 3-state DTMC to this data and compare the performance of naive and fixed models.

Preproglucacon data is the DNA sequence for the gene that encodes the protein preproglucacon. This data consists of 1572 observations with bases A, T, C, G coded numerically as 1, 4, 2, 3. We split this data into a training set of size 500 and a test set of size 1072. Using this data, we fit a 4-state DTMC and compare the performance of naive and fixed models.

For the sake of comparison, we also fit a hidden Markov model (HMM) to both data sets. The HMM is more sophisticated than the DTMC and requires much more computational effort to train [26]. When we trained HMM models, we explored hyperparameters such as the number of internal states and the random initialization of the model. We report results for the best (smallest long-term test error) hyperparameter choices we were able to find.

Table IV shows long-term training and test errors for naive DTMC, fixed DTMC, and HMM models. The fixed DTMC models feature greatly reduced test set errors as compared to the naive DTMC models. Note also that for the Holson data set, the long-term test errors for the fixed DTMC and HMM models are comparable. For the preproglucacon data, the HMM's long-term test error is about 4 times less than that of the fixed DTMC. While the HMM is able to achieve better test set results in some cases, we see that this achievement comes at some computational expense. Including time spent solving linear programs, our method requires ≈ 50x less time to train compared to the HMM.

VI. Conclusion

Both CTMC and DTMC models containing absorbing states do not capture well the observed fraction of time spent in each state. We remove absorbing states by finding a sparse perturbation to the transition (or transition rate) matrix such that the new matrix achieves a desired equilibrium distribution. We formulated this problem as a linear programming problem.

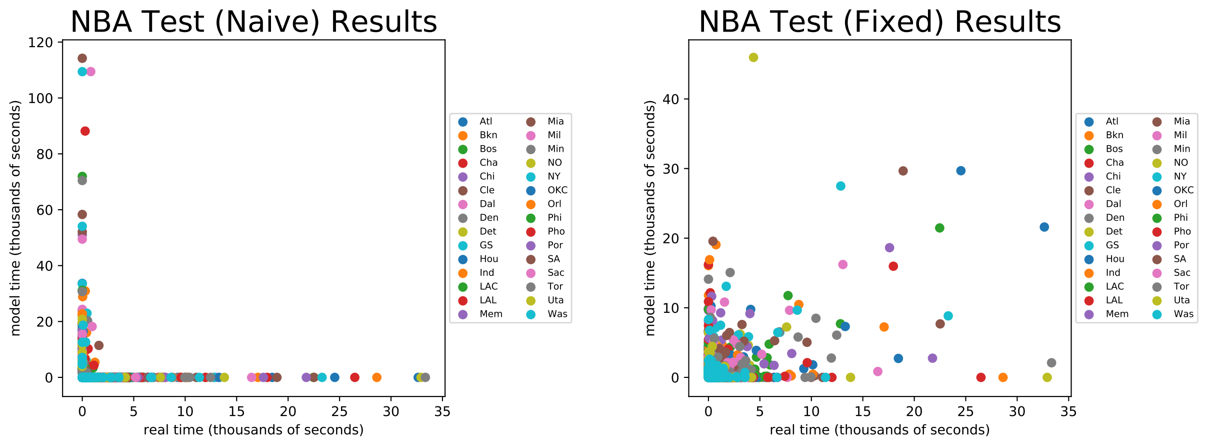

Fig. 2. We plot naive (left) and fixed (right) CTMC test results using 40-game training sets. The fixed model decreases RMSE error by $\approx 60.4\%$. For further details, see Section V-A.

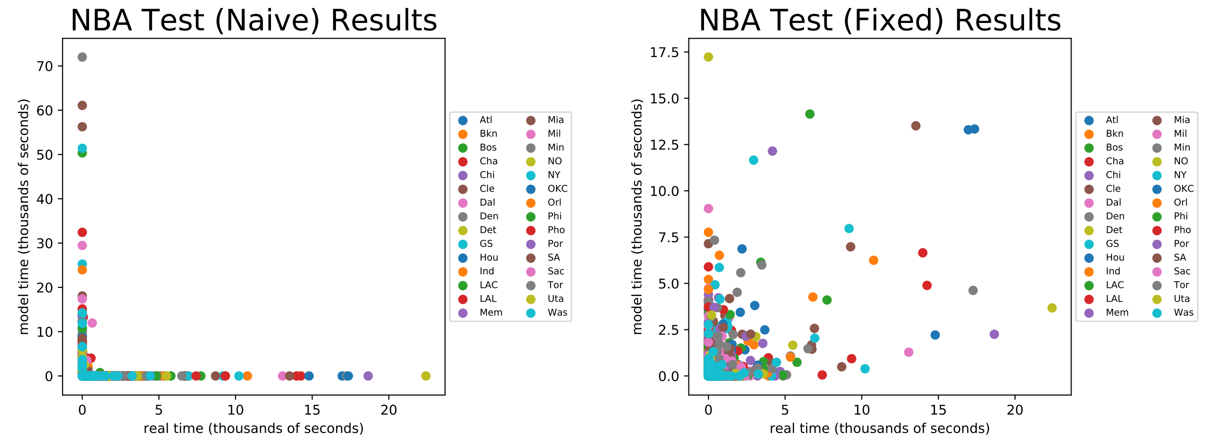

Fig. 3. We plot naive (left) and fixed (right) CTMC test results using 60-game training sets. The fixed model decreases RMSE error by $\approx 61.9\%$. For further details, see Section V-A.

Holson			
	Naive	Fixed	HMM
Training Time		0.71	54.68
LT Training Error	0.160545	1.665335e-16	0.000835
LT Test Error	0.235091	7.454545e-02	0.075380
Preproglucacon			
	Naive	Fixed	HMM
Training Time		0.05	28.40
LT Training Error	0.003426	1.942890e-16	0.000509
LT Test Error	0.113921	1.154179e-01	0.036629

TABLE IV
LONG-TERM ERRORS WITH TRAINING SIZE 500 ON HOLSON AND PREPROGLUCACON DATA. WE ALSO REPORT THE TRAINING TIME FOR FIXED AND HMM MODELS, IN SECONDS.

Through extensive tests with simulated and real data, we have shown that our method improves long-term predictions without sacrificing much short-term accuracy. Furthermore, our method requires far less time to train than HMM methods.

APPENDIX

We start by deriving the DTMC MLE. Consider data $\{s_0, s_1, s_2, \dots, s_N\}$. Define $p_{i,j}$ to be the probability of transitioning from i to j. Then the likelihood function is:

$$L = p_{s_0,s_1} p_{s_1,s_2} \cdots p_{s_{N-1},s_N} = \prod_{i,j} p_{i,j}^{N(i,j)}$$

where $N(i,j)$ is the number of times that the pattern (i,j) occurs. Note that $p_{i,i} = 1 - \sum_{j \neq i} p_{i,j}$. Therefore,

$$\log L = \sum_{i,j:i\neq j} N(i,j) \log p_{i,j} + \sum_i N(i,i) \log\big(1 - \sum_{j\neq i} p_{i,j}\big).$$

Fix states i', j' and maximize the log likelihood function over the parameter $p_{i',j'}$:

$$\frac{\partial \log L}{\partial p_{i',j'}} = \frac{N(i',j')}{p_{i',j'}} - \frac{N(i',i')}{1 - \sum_{j\neq i'} p_{i',j}} = 0.$$

Suppose there are $M-1$ states in total. Then for fixed i', the above equation gives us $M-1$ equations in $M-1$ unknowns.

Let $\vec{1}$ denote the $(M-1) \times 1$ vector of all ones. Then the $M-1$ equations can be summarized by

$$\left(N(i',i')I + \vec{N}\vec{1}^T\right)\vec{p} = \vec{N}$$
$$\vec{N} = (N(i',1),\ldots,N(i',i'-1),N(i',i'+1),\ldots,N(i',M))$$
$$\vec{p} = (p_{i',1},\ldots,p_{i',i'-1},p_{i',i'+1},\ldots,p_{i',M}).$$

Note that $\vec{N}\vec{1}^T$ is a rank-1 matrix; the matrix multiplying $\vec{p}$ is therefore a rank-1 perturbation of a multiple of the identity. We can then solve for $\vec{p}$ using the Sherman-Morrison-Woodbury matrix inversion formula:

$$\left(N(i',i')I + \vec{N}\vec{1}^T\right)^{-1} = \frac{1}{N(i',i')}I - \frac{1}{N(i',i')}I\vec{N}\left(1 + \frac{\vec{1}^T\vec{N}}{N(i',i')}\right)^{-1}\vec{1}^T\frac{1}{N(i',i')}I$$

Hence $\vec{p} = \vec{N}/\sum_{j=1}^{M} N(i',j)$. Since i' and j' were arbitrary, we see that the MLE for the (i,j)-th entry of the transition matrix is $\widehat{p}_{i,j} = N(i,j)/\sum_{k=1}^{M} N(i,k)$. We have derived this formula for $i \neq j$. Because $\widehat{p}_{i,i} = 1 - \sum_{j\neq i}\widehat{p}_{i,j}$, it is valid for $i = j$ as well.

Next we derive the CTMC MLE. Consider data consisting of times $\{0 = t_0, t_1, t_2, \ldots, t_N\}$ and states $\{s_0, s_1, s_2, \ldots, s_N\}$. Define $\alpha(x,y)$ as the rate that state x jumps to state y. Then the transition rate out of state x is $\alpha(x) = \sum_{y\neq x}\alpha(x,y)$. For $i = 0,\ldots,N-1$, define $T_i = t_{i+1} - t_i$. Then the likelihood function is [27]:

$$\begin{aligned} L &= \alpha(s_0)e^{-\alpha(s_0)T_0}\frac{\alpha(s_0,s_1)}{\alpha(s_0)}\ \alpha(s_1)e^{-\alpha(s_1)T_1}\frac{\alpha(s_1,s_2)}{\alpha(s_1)}\ \cdots \\ &= e^{-\alpha(s_0)T_0}\alpha(s_0,s_1)\ e^{-\alpha(s_1)T_1}\alpha(s_1,s_2)\ \cdots \\ &= e^{-\sum_x \alpha(x)W(x)}\prod_{x,y:x\neq y}\alpha(x,y)^{N(x,y)}, \end{aligned}$$

where $W(x) = \sum_{i=0}^{N-1} T_i \cdot \mathbf{I}(s_i = x)$ and $N(x,y) = \sum_{i=0}^{N-1}\mathbf{I}(s_i = x, s_{i+1} = y)$. In words, $W(x)$ is the total time spent in state x, and $N(x,y)$ is the total number of times that the pattern (x,y) is observed. Then

$$\log L = \sum_{x,y:x\neq y}[-W(x)\alpha(x,y) + N(x,y)\log\alpha(x,y)].$$

Fix states x', y' and maximize the log likelihood function over the parameter $\alpha(x',y')$:

$$\frac{\partial \log L}{\partial \alpha(x',y')} = -W(x') + \frac{N(x',y')}{\alpha(x',y')} = 0.$$

We obtain the MLE $\widehat{\alpha}(x',y') = N(x',y')/W(x')$. This holds for all $x' \neq y'$.

H. S. Bhat acknowledges support from NSF award DMS-1723272. Both authors acknowledge use of the MERCED computational cluster, funded by NSF award ACI-1429783.

References

[1] K. Fokianos and B. Kedem, "Regression theory for categorical time series," *Statistical Science*, vol. 18, no. 3, pp. 357–376, 2003.

[2] H. S. Bhat, L.-H. Huang, and S. Rodriguez, "Learning stochastic models for basketball substitutions from play-by-play data," in *MLSA15, Workshop at ECML/PKDD 2015*, 2015, pp. 55–64. [Online]. Available: http://ceur-ws.org/Vol-1970/paper-08.pdf

[3] M. T. Chu and Q. Guo, "A numerical method for the inverse stochastic spectrum problem," *SIAM J. Matrix Anal. Appl.*, vol. 19, no. 4, pp. 1027–1039, 1998.

[4] M. Chu and G. H. Golub, *Inverse Eigenvalue Problems: Theory, Algorithms, and Applications*. Oxford University Press, 2005, vol. 13.

[5] Z.-J. Bai, S. Serra-Capizzano, and Z. Zhao, "Nonnegative inverse eigenvalue problems with partial eigendata," *Numerische Mathematik*, vol. 120, no. 3, pp. 387–431, 2012.

[6] R. Kumar, A. Tomkins, S. Vassilvitskii, and E. Vee, "Inverting a steady-state," in *Proc. 8th ACM WSDM*, 2015, pp. 359–368.

[7] L. Maystre and M. Grossglauser, "Fast and accurate inference of Plackett–Luce models," in *Advances in Neural Information Processing Systems 28*, 2015, pp. 172–180.

[8] S.-J. Wu and M. T. Chu, "Constructing optimal transition matrix for Markov chain Monte Carlo," *Linear Algebra and its Applications*, vol. 487, pp. 184–202, 2015.

[9] D. P. Laurie, "Solving the inverse eigenvalue problem via the eigenvector matrix," *J. Comp. Appl. Math.*, vol. 35, pp. 277–289, 1991.

[10] X. Chen and D. Liu, "Isospectral flow method for nonnegative inverse eigenvalue problem with prescribed structure," *Journal of Computational and Applied Mathematics*, vol. 235, no. 14, pp. 3990–4002, 2011.

[11] T.-T. Yao, Z.-J. Bai, Z. Zhao, and W.-K. Ching, "A Riemannian Fletcher–Reeves conjugate gradient method for doubly stochastic inverse eigenvalue problems," *SIAM J. Matrix Anal. Appl.*, vol. 37, no. 1, pp. 215–234, 2016.

[12] A. N. Langville and C. D. Meyer, *Google's PageRank and Beyond: the Science of Search Engine Rankings*. Princeton, NJ: Princeton University Press, 2006.

[13] A. Y. Ng, A. X. Zheng, and M. I. Jordan, "Link analysis, eigenvectors and stability," in *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, IJCAI 2001, Seattle, Washington, USA, August 4-10, 2001*, 2001, pp. 903–910.

[14] A. N. Langville and C. D. Meyer, "Updating Markov chains with an eye on Google's PageRank," *SIAM J. Matrix Anal. Appl.*, vol. 27, no. 4, pp. 968–987, 2006.

[15] T. P. Chartier, E. Kreutzer, A. N. Langville, and K. E. Pedings, "Sensitivity and stability of ranking vectors," *SIAM J. Sci. Comput.*, vol. 33, no. 3, pp. 1077–1102, 2011.

[16] O. Fercoq, M. Akian, M. Bouhtou, and S. Gaubert, "Ergodic control and polyhedral approaches to PageRank optimization," *IEEE Transactions on Automatic Control*, vol. 58, no. 1, pp. 134–148, 2013.

[17] O. Fercoq, "Perron vector optimization applied to search engines," *Applied Numerical Mathematics*, vol. 75, pp. 77–99, 2014.

[18] B. C. Csáji, R. M. Jungers, and V. D. Blondel, "PageRank optimization by edge selection," *Disc. Appl. Math.*, vol. 169, pp. 73–87, 2014.

[19] D. F. Gleich, "PageRank Beyond the Web," *SIAM Review*, vol. 57, no. 3, pp. 321–363, 2015.

[20] L. Backstrom and J. Leskovec, "Supervised random walks: predicting and recommending links in social networks," in *Proc. 4th ACM WSDM*, 2011, pp. 635–644.

[21] G. F. Lawler, *Introduction to Stochastic Processes*. Boca Raton: Chapman & Hall/CRC, 2006.

[22] J. Nocedal and S. J. Wright, *Numerical Optimization*. New York: Springer, 2006.

[23] M. S. Andersen, J. Dahl, and L. Vandenberghe, "CVXOPT: A Python package for convex optimization, version 1.2.0," http://cvxopt.org, 2018.

[24] M. ApS, *MOSEK Optimizer API for Python 8.1.0.80*, 2018. [Online]. Available: https://docs.mosek.com/8.1/pythonapi/index.html

[25] G. A. Spedicato, T. S. Kang, S. B. Yalamanchi, and D. Yadav, *The markovchain Package*, 2017. [Online]. Available: https://cran.r-project.org/web/packages/markovchain/vignettes/an_introduction_to_markovchain_package.pdf

[26] A. M. Fraser, *Hidden Markov Models and Dynamical Systems*. Philadelphia: SIAM, 2008.

[27] P. Guttorp, *Stochastic Modeling of Scientific Data*. Springer Science+Business Media, Dordrecht, 1995.